Definition and Computation of Similarity Operations between Web-specific Lexical Items

Luís Sarmento

Definition and Computation of Similarity Operations between Web-specific Lexical Items

Universidade do Porto

Faculdade de Engenharia

FEUP

Supervisor: Prof. Dr. Eugénio de Oliveira

Faculdade de Engenharia
Universidade do Porto
Rua Dr. Roberto Frias, s/n
4200-465 Porto, Portugal

This work was supported by Fundação para a Ciência e Tecnologia (FCT) through grant SFRH/BD/23590/2005.

This thesis is dedicated to my Parents,

Maria Helena and Luís Carlos,

whose love, support and faith in me

have always been unconditional.

Acknowledgments

This work was only possible thanks to the contribution of many people, who helped in several different ways during almost five years. It is hard to acknowledge all these people, whose true impact on the final work I may not even realize at this moment.

First of all, I would like to thank my Parents, Maria Helena and Luís Carlos, and my Family. I had many dark moments during the preparation of this thesis but they were always there giving love and support and helping to cast some light over the shadows.

Secondly, I would like to express my gratitude to my supervisor Prof. Eugénio de Oliveira, who always supported this thesis at various levels and provided all the necessary conditions and protection for this project to grow and to flourish. Thank you, Professor Eugénio de Oliveira, for standing by my side, and for helping me navigate through the eye of the storm.

I also had the fortune of having had the help of other excellent Professors from outside the University of Porto. All of them taught me things that were fundamental in complementing what I learned in my University. I thank Professor Maarten de Rijke, from the University of Amsterdam, for his bright insights and ability to see clearly what for me was very fuzzy. I thank Professor Valentin Jijkoun, also from the University of Amsterdam, for his method and objectivity. I thank Professor Lyle Ungar, from the University of Pennsylvania, for his ability to patiently teach me and motivate me during most of the period of this thesis. And, finally, I thank Professor Mário Silva, from the University of Lisbon, for his surgical criticisms, for involving me in his projects and, last but not least, for his keen sense of humour.

This thesis also gave me the opportunity to meet and work with other young researchers who had either recently finished or were still developing work for their PhD's. I would like to thank Fabien Gouyon, Paula Carvalho, Sérgio Nunes and Rui Sousa Silva. All of them helped me to develop my work in many ways and to explore new ideas. I have learned a lot from all of them, and more importantly, they all contributed to making me feel part of a community of promising young researchers based in Portugal.

A great deal of the work presented in this thesis was started and partially developed in collaboration with industry leaders. In 2007, I had the chance to spend a six-month internship at the New York offices of Google. I learned a great deal during this internship, not only from a technical point of view, but also about the organics of team work. I thank all the Google team in NY and especially Alexander Kehlenbeck, Casey Whitelaw and Nemanja Petrovic. I have also had the opportunity to collaborate with SAPO.pt in several projects. The entire SAPO.pt team has always been very enthusiastic about this collaboration, and this has resulted in good ongoing relationship. I would like to thank especially Benjamin Junior, João Pedro Goncalves, Paulo Trezentos and also Celso Martinho.

Additionally, I would like to thank all my colleagues at LIACC / FEUP for their support, for the productive discussions and exchange of ideas, and also for the friendly environment that they helped to create. I also wish to thank all the staff from the Secretary of the Department of Informatics, and specially Sandra Reis, for helping me to fight the never-ending battle against bureaucracy.

I would also like to thank Professor Belinda Maia, from the University of Porto, with whom I have worked on projects and events for more than eight years, and who has always been patient and supportive in issues more important and much more complex than science.

I am also deeply indebted to the Fundação para a Ciência e Tecnologia (FCT) who supported my work financially during 4 years. I would never have been able to develop the work required for this thesis if I had not been allowed to dedicate myself full time to the task, and that was only possible thanks to the PhD grant provided by the FCT.

I would like to end by thanking Jorge Teixeira, who, midway through this thesis, decided to join me in tilting at windmills. His enthusiasm, dedication and mental strength have been invaluable. I am happy to see that he is now getting ready to tilt at his own windmills. Good luck Jorge!

Resumo

Nesta tese iremos estudar vários tipos de relações de similaridade que podem ser estabelecidas entre diversas classes de items lexicais *típicos da Web*, nomeadamente (i) *palavras (ou expressões multi-palavra) do léxico comum*, (ii) *nomes próprios de uma alargada gama de entidades*, e (iii) outras estruturas lexicais específicas das Web como *etiquetas de classificação*, *tags Web 2.0* e *palavras-chave geradas espontaneamente pelos utilizadores*. Iremos focar particularmente em três tipos de relações de similaridade que se podem encontrar em vários cenários práticos de processamento linguagem: (i) *similaridade de conteúdo*, (ii) *similaridade de tipo* e (iii) *similaridade de função*.

Começaremos esta tese por formalizar cada um dos três conceitos de similaridade referidos e demonstraremos como é possível unificar os três conceitos numa estrutura formal mais abrangente. Depois, discutiremos assuntos relacionados com o *Modelo de Espaço Vectorial*, a principal ferramenta computacional que iremos empregar para a implementação de funções de similaridade entre items lexicais. Em seguida, abordaremos diversas questões relacionadas com a avaliação dos métodos de computação da similaridade, incluindo uma discussão acerca das *estratégias* de avaliação possíveis, das *medidas* de avaliação apropriadas e dos recursos existentes para uso como *referência padrão*.

Prosseguiremos com a apresentação de *sete* casos de estudo, todos eles motivados por questões práticas encontradas durante o desenvolvimento de diferentes projectos de processamento automático de linguagem. Cada um destes casos práticos permitirá ilustrar uma situação em que um determinado conceito de similaridade associado a um dos tipos de items lexicais em causa é a chave para a resolução de um problema concreto de processamento de linguagem. Através destes sete casos de estudo iremos validar, em diversos cenários, a formalização e os métodos de computação de similaridade propostos nesta tese.

Terminaremos apresentando as conclusões finais, enumerando as principais contribuições desta tese e apontando possíveis linhas de investigação futuras.

Abstract

In this thesis we study several types of similarity relations that can be established between different classes of lexical items that are typical of the Web, namely (i) words (or multi-word expressions) in the general lexicon, (ii) names belonging to a wide range of entities, and (iii) other lexical structures that are specific to the Web, such as classification labels, Web 2.0 tags and keywords spontaneously generated by users. We focus more specifically on three types of similarity relations that can be found in several practical language processing scenarios: (i) content similarity , (ii) type similarity and (iii) functional similarity.

We begin this thesis by formalizing each of the three previously mentioned similarity concepts, and we show how it is possible to unify them into a more comprehensive formal structure. Then, we discuss issues related to the Vector Space Model, the main computational tool used for supporting the implementation of similarity functions between lexical items. Next, we address several issues related to the evaluation of the proposed methods for computing similarity, including a discussion about the different possible evaluation strategies, the most appropriate evaluation measures, and the resources available for serving as a gold standard in our evaluation procedures.

We proceed by presenting seven case studies, all of them motivated by practical questions found during the development of several automatic language processing projects. Each of these case studies illustrates situations in which a given similarity concept is the key for solving a practical language processing problem. These seven case studies allow us to validate, in various settings, the formalization and methods for computing similarity that we propose in this thesis.

We end by presenting the final conclusions, by enumerating the main contributions of this thesis and by presenting lines for future research work.

Contents

Acknowledgments **v**

Resumo **vii**

Abstract **ix**

1 Introduction **1**

1.1 Research Questions 4
1.2 Organization of this Thesis 7
1.3 Foundations 8
1.4 Outline of the Main Contributions 9

I Background **11**

2 Similarity and Ambiguity **15**

2.1 Why is similarity important? 15
2.2 Lexical Items, Referents and Dynamic Lexicons 17
2.3 Defining Similarity 19
2.3.1 Content Similarity 19
2.3.2 Type Similarity 20
2.3.3 Functional Similarity 22
2.4 Semantic Ambiguity 24
2.5 Conclusion 26

3 The Vector Space Model **27**

3.1 Introduction 27
3.2 VSM Parameters and Distance Metrics 28
3.3 Feature Context 30
3.3.1 Directly Observable Feature Contexts 30

3.3.2 Indirectly Observable Feature Contexts 31
3.3.3 Partially Observable Feature Contexts 32
3.4 Feature Weighting Functions 32
3.4.1 Observations, Contingency Table and Estimations 33
3.4.2 Significance Tests as Feature Weighting Functions 34
3.4.3 Weighting Functions based on Information Theory 42
3.4.4 tf-idf as a Feature Weighting Function 44
3.5 Vector Distance Measures 45
3.5.1 Distance Measures over Real-Valued Vector Spaces 46
3.5.2 Distance Measures over Binary Vector Spaces 49
3.6 Conclusion 51

4 Evaluation **53**
4.1 Direct vs. Indirect Strategies 53
4.2 Direct Evaluation Strategies 55
4.2.1 Evaluating Pairings 55
4.2.2 Evaluating Ranked Lists 56
4.2.3 Evaluating Clusters 58
4.2.4 Gold-Standard Resources for Direct Evaluation 62
4.3 Indirect Evaluation Strategies 67
4.4 Conclusion 68

II Common Lexicon **69**

5 Identification of Verb Synonyms **73**
5.1 Introduction 74
5.2 Related Work 75
5.3 VSM for Verb Synonyms 78
5.4 Evaluating Verb Synonyms 79
5.5 Experimental Setup 81
5.5.1 Feature Information 82
5.6 Results and Analysis 82
5.7 Conclusions 86

6 Paraphrasing Job Titles **87**
6.1 Introduction 88
6.2 Related Work 89
6.3 Method Description 91
6.3.1 Finding Potential Short Paraphrases 92
6.3.2 Inferring Substitution Rules 93
6.4 Experimental Setup 94
6.4.1 Evaluation 97

6.5 Results and Analysis . . . 98
6.5.1 Substitution Rules . . . 98
6.5.2 Evaluating new Job Titles . . . 101
6.6 Conclusions . . . 102

III Names and Entities 105

7 Finding Type Similar Entities 109
7.1 Introduction . . . 110
7.2 Related Work . . . 110
7.3 Expansion using Membership Functions . . . 111
7.3.1 Implementing a Class Membership Function . . . 112
7.3.2 Identification of Lists . . . 112
7.3.3 Building Feature Vectors . . . 113
7.3.4 Calculating membership function . . . 114
7.4 Evaluation Using Wikipedia . . . 114
7.4.1 Test sets and performance measures . . . 115
7.4.2 Test Set Construction using Wikipedia . . . 115
7.4.3 A Baseline using Wikipedia Categories . . . 118
7.5 Experimental Setup . . . 119
7.5.1 Extracting entities and features . . . 119
7.5.2 Evaluation setup . . . 121
7.6 Results . . . 122
7.6.1 Results on least frequent elements . . . 124
7.7 Conclusions . . . 125

8 Web-Scale Entity Disambiguation 127
8.1 Introduction . . . 128
8.2 Related Work . . . 130
8.2.1 Approaches based on clustering . . . 130
8.2.2 Approaches using external knowledge . . . 135
8.3 A Clustering Approach to NED . . . 139
8.3.1 Feature Vector Generation . . . 139
8.3.2 Clustering Procedure Overview . . . 140
8.4 Clustering-based Approaches to Web-Scale NED: Challenges . . . 141
8.5 Efficient Web Clustering by Finding Connected Components . . . 142
8.6 Additional Scalability Issues . . . 144
8.7 Evaluation Framework . . . 146
8.7.1 Preparing the Gold Standard . . . 147
8.8 Experimental Setup . . . 147
8.9 Results and Analysis . . . 148
8.10 Conclusion . . . 152

IV Labels, Web 2.0 Tags and other User-Generated Keywords 153

9 Conflating News Topic Labels 159

9.1 Introduction 160
9.2 Related Work 161
9.3 Propagating Topic Labels 162
9.4 Experimental Set-Up 163
9.5 Results and Analysis 164
9.6 Conclusions 166

10 Expanding User-Defined Tag Sets 167

10.1 Introduction 168
10.2 Related Work 170
10.3 Tag Propagation Method 170
10.4 Data 171
10.5 Experimental Setup 172
10.5.1 Evaluation 174
10.6 Results and analysis 175
10.6.1 Automatic vs manual evaluations 176
10.6.2 Particular case of Long Tail artists 176
10.6.3 Further error analysis 177
10.7 Conclusions 178

11 The Suggestion of Keywords for Ads 181

11.1 Introduction 182
11.2 Related Work 183
11.3 Method Description 186
11.3.1 Computing Keyword Synonymy 187
11.3.2 Keyword Suggestion and Ranking 188
11.3.3 Overview of the Suggestion Procedure 189
11.4 Evaluation: an On-line Method 190
11.4.1 Evaluating Method Usefulness 192
11.4.2 Impact on Revenue 193
11.5 Experimental Set-Up 194
11.6 Results and Analysis 195
11.6.1 Method Usefulness 196
11.6.2 Impact on Revenue 197
11.7 Conclusion 199

V Conclusions 201

12 Conclusions 203

12.1 Overview . . . 203

12.2 Answer to Research Questions . . . 204

12.3 Main Contributions . . . 208

12.4 Future Directions . . . 209

12.4.1 High-Level Improvements in Previous Work . . . 209

12.4.2 New Lines of Research . . . 211

A Demonstrations for Chapter 8 215

Bibliography 217

Chapter 1
Introduction

This thesis is built around the notion of *semantic similarity*, and, more specifically, around the question of how to decide *in context* if two *lexical items* represent "similar" referents or not, according to a pre-defined *notion* of semantic similarity. This question is central to natural language processing, since many "real-world" language processing applications depend, directly or indirectly, on such knowledge about similarity, especially when some form of robust semantic analysis is required. For example, in Information Retrieval (IR) or Question-Answering (QA) applications, information about word synonymy is important for matching query terms with documents that do *not* contain those specific query terms but only their synonyms.

For many years, lexical-semantic resources, such as (hand-made) machine-readable dictionaries (MRD) and thesauri, have provided the information about word similarity that was required by most language processing applications. Traditionally, such applications were designed to process specific document collections, usually of a restricted *domain* and *genre*, and, therefore, were supposed to face rather controlled text processing environments, with stable lexicons and concept ontologies. And, since text production and dissemination were essentially in the hands of experts and editors, text correctness, both from a lexical and grammatical point of view, was taken for granted by application developers.

However, the environment for "real-world" text processing technologies has changed dramatically over the last decade, and the Web has brought totally new and different ways of *producing*, *disseminating* and *consuming* text contents. We can find many distinct text formats on the Web today and while some of them are comparable to traditional formats, others are totally new. For example, the Web contains:

- industrial media (e.g. on-line newspapers);
- revised contents (e.g. scientific papers, on-line books);
- user-generated contents (e.g. blogs, mailing-list archives);

- a wide variety of semi-structured contents (e.g. information tables, bibtex entries) and media-specific contents (e.g. bookmark lists).

These formats – and the interconnections that have been organically established between them – have decisively reshaped the essential *nature of text*. The building blocks of such an amalgam of contents are no longer "words" belonging to a relatively stable and well-established lexicon, but "text symbols" which are part of an ever-expanding universe of *words*, entity *names*, *tags*, *domain-specific vocabularies*, *neologisms*, and other *idiosyncratic* sequences such as *spelling mistakes* or *emoticons*.

In such a scenario, traditional lexical-semantic resources are becoming insufficient for supporting "real-world" applications. First of all, what we understand by "lexicon" has changed dramatically, and we are yet to find the best linguistic treatment for certain lexical items found on the web, such as for example *Web 2.0 tags*. Second, even if we had such a linguistic framework, the speed at which new lexical items are added to the universe of the Web (e.g. names) is so high that it would be impossible to update dictionaries. Third, it would also be impossible to differentiate all senses of the lexicon used on the Web (i.e. by providing standard definitions for all words), since the number of contexts is huge and grows in an uncontrolled and distributed fashion. This harsh environment calls for a new approach to similarity, to its computation and to the generation of related resources.

A Uniform Approach to Similarity: Challenges

We need a framework capable of uniformly accommodating the various perspectives of similarity that emerge from practical applications. This demands a notion of similarity that covers several related, yet different, sub-concepts. For instance, *word synonymy* and *paraphrase* are traditional similarity relations. They are instances of *content similarity*, since synonyms and paraphrases convey equivalent *content* using different *forms*. But, a definition of similarity that is motivated by practical concerns needs to embrace other (less traditional) concepts related to similarity, such as *co-hyponymy* (i.e *type similarity*), *name equivalence* (e.g. the relationship between names and nicknames), or more generally, *functional equivalence* (e.g. items can be interchanged without significantly affecting the end result of a given task). Ideally, these concepts would be seamlessly integrated in a common framework, which, as the Web evolves, it should be possible to update with the integration of new concepts.

Moreover, the decentralized nature of the Web imposes several additional challenges. Since there is no global supervision mechanism that ensures a common ontology between all content producers, the web is essentially populated by "local" semantics: there is no guarantee that semantic relations, and thus item similarities, that apply in a given context can be ported to another context, even when there is overlap between concepts. In fact, since most of these "local" semantics systems are open to edition by the community, they tend to evolve in an organic and possibly incoherent way, merging the contributions of users who may even share different visions about the subjects at stake. This is the case, for example, of tagging systems, which are now a standard feature in Web 2.0 sites. The question that arises is: how can a framework for similarity cope with such "local" semantics?

For all these reasons, the current generation of applications cannot simply rely on *static* pre-computed resources to decide about the similarity of two referents: they must possess, at least, the ability to adapt such information to context. Ideally, applications would use contextual information to infer the semantic knowledge as they proceed. The focus, then, should be on developing efficient *methods* for inferring similarity relations *dynamically* and in a *context-dependent* way. These adaptation capabilities are required not only for the sake of keeping up with the dynamics of the *lexicon*. In fact, because Web contents are always in mutation, the relations of similarity *themselves* change over time and require constant updating. The work presented in this thesis is a contribution to developing methods capable of coping with such changes.

Semantic Ambiguity

Another concept closely related to similarity is *semantic ambiguity*. Rather than deciding if two distinct lexical representations are "similar" (e.g. content similar such as synonyms or spelling variations), resolving *semantic ambiguity* involves deciding if the *same* lexical representation found in two different contexts is pointing to the same *referent* or not. There is a strong connection between these two concepts, since one can formulate the problem of lexical ambiguity as a question regarding the similarity between two lexical items: do these *equal* lexical representations that occur in *different* documents, or distinct parts of the same document, relate to the surrounding context in "similar" ways so that they can be considered representatives of the same referent?

While approaches to ambiguity resolution in relatively controlled text environments are not new, as one can conclude by examining the large body of work that has been produced in the field of Word-Sense Disambiguation (WSD) and Named-Entity Disambiguation (NED), new approaches are required to tackle the problem of lexical ambiguity on the Web. The Web accommodates references to many more concepts and entities than any other environment. Just for the purpose of illustration, the English version of the Wikipedia (i.e. an extremely small part of the web) contains more than 90 articles for entities named "Paris", ranging from several possible *locations* to several possible *songs*. This is a much higher level of ambiguity than would be found in most domain-specific collections. However, the number of distinct entities named "Paris" that can be found on the Web is definitely several orders of magnitude higher.

Hence, the number of distinct concepts or entities whose lexical representations may collide on the Web is much higher than is found in smaller and topic-restricted collections, such as those traditional approaches to NED try to address. In this thesis, we also hope to contribute to solving some of the aforementioned issues.

1.1 Research Questions

The work presented partially results from a set of experiments motivated by practical problems found while developing language processing resources and applications. Table 1.1 presents those projects and some questions related to the computation of "similarity" that have naturally emerged from them.

From these practical questions, we formulated the following higher-level research questions regarding similarity:

1. Can we provide a unified conceptualization that embraces all sub-concepts of similarity found between text elements, and, at the same time, provides solutions to practical questions imposed by the Web?

This is the top question of this thesis. We aim at presenting a formulation of similarity accommodating several specific similarity relations, such as synonymy, co-hyponymy (i.e. type similarity), and various forms of functional equivalence. The formulation should be applicable both to traditional words and to Web-specific lexical units (e.g. Web 2.0 tags). Additionally, it should allow a straightforward computational implementation to be of practical use in "real-world" language applications.

2. How does one deal with web-specific lexical items, such as tags or topic labels? What notions of similarity can be applied to them?

As the lexical universe of the web becomes more complex and varied by the inclusion of new lexical items that are different from existing words, it is important to understand whether we can extend traditional similarity relations to such new items, especially when the semantics of such items are subject to locality. For example, do certain similarity relations, such as synonymy or antonymy, apply to a given set of user-defined tags of a particular Web 2.0 site in the same way as they do to traditional words? What restrictions should be applied, if any?

3. Which information sources can be used for grounding the computation of similarity?

Although several techniques for computing word-similarity based on pre-existing lexical-semantic resources have been proposed previously (e.g. [RD00]), for the

Table 1.1: Projects developed and various questions related to similarity that emerged from them.

Project / Application	Questions emerged
Semantic Lexicons for Information Extraction	Can two specific words be considered synonyms (within a certain context)? Should two specific words be grouped under the same semantic class?
Named-Entity Recognition (NER) systems	Is a given entity similar enough to one found in a reference list for it to be tagged with the same semantic tag?
Named-Entity Disambiguation (NED) systems	How can we measure the "similarity" between two mentions of the same (or different) name? What features can be used to know if two mentions of the same name refer to the same entity or not?
Automatic Question-Answering (QA) systems	Is synonym-based query expansion useful in finding answers?
Keyword or Tag Suggestion systems	Given a set of seed keywords can we expand it with "similar" or equally relevant keywords? How can we find "local" synonyms (i.e. domain-specific synonyms)? Can we use external information sources to mine previously "unseen" relevant keywords?
Quotation Extraction and Topic Classification systems	How can we robustly find mentions to relevant entities in news when they are made not only by name but also by *job title*, and these can be expressed in so many different equivalent ways? Are two *topic labels* found on RSS feeds published by two different on-line newspapers "equivalent" in a certain context, despite using different lexical representations?

reasons stated above, our goal is different. We need to find an adaptive method for computing similarity for various types of lexical items, using information derived from the Web and with the least human intervention possible. Therefore, we wish to understand which information sources can be used to infer such similarity. Are purely statistical approaches based on item co-occurrence information enough? Or do we need to obtain additional information from text sources using some form of extraction patterns? Can we benefit from community-edited resources, such as Wikipedia, and their corresponding folksonomies or bottom-up classification structures? Can we extract valuable similarity information from domain-specific social networks?

4. How can we evaluate the results of procedures for computing similarity?

Ideally, evaluation of similarity computation procedures would involve comparing the results of such procedures against gold-standard resources. However, as we have argued before, one of the motivations for this thesis is precisely the lack of up-to-date lexical-semantic resources containing information about similarity, particularly for some Web-specific contexts. While manual evaluation is always possible, it may require too much effort and it is inherently not reproducible. One alternative involves extracting (partially complete) gold-standard information from the Web. Another alternative is to perform an application-oriented evaluation, since we are approaching similarity computation from the point of view of applications. In this thesis, we discuss the possibilities and limits of such alternatives.

Theoretical Instruments

One of the mathematical tools that has allowed us to approach the different facets of similarity in a more unified fashion is the Vector Space Model (VSM). With the VSM, items to be processed are converted into feature vectors, i.e. vector representations that populate a *n-dimensional hyperspace*. The axes of such hyperspace are derived from the set of features that are used to describe the items being processed. Using the VSM, item comparison becomes a matter of comparing the corresponding vectors in the hyperspace, an operation that can be performed using any of the several vector similarity (or distance) metrics available. The crux of successful representation under the VSM lies precisely in the set of features used to describe the items (i.e. in the choice of the axes of the hyperspace). The selection of such features determines the semantic information that is projected to the hyperspace, and thus, the nature of what is actually compared. Therefore, a fundamental part of this thesis is related to discussing the selection, and subsequent transformations, of the feature information, as well as evaluating the impact of such choices on the values of similarity obtained.

1.2 Organization of this Thesis

For the purpose of making a more focused discussion of the vast number of interconnected questions surrounding similarity, this thesis is divided into *five* parts. Despite this division, it should be understood as a *single* unit of work, which tries to tackle a large problem from different angles. To make the connections between each of the five parts more obvious, we begin each part with a summary of the main issues being addressed.

In part I we present the background information required for the remainder of the thesis, and we discuss several methodological issues. We begin by providing additional motivation for our study in chapter 2. We present several possible perspectives related to similarity, motivated by questions raised during development of language processing applications. Then, in chapter 3, we present the fundamental notions regarding the Vector Space Model. Finally, in chapter 4, we address methodological issues regarding evaluation of several tasks related to the computation of similarity.

Parts II to IV address similarity issues of different types of lexical items and referents:

Part II - Common Lexicon: We begin by studying the possibilities that exist for automatically detecting similarities between "traditional" words and phrases. We focus on *synonymy* (chapter 5) and *paraphrase* (chapter 6) relations. The main questions here are related to the selection of the best features for grounding the computation of similarity, which is then performed either by transferring semantic information to the vector space, or by using algorithms for finding semantic associations.

Part III - Names and Entities: Part III focuses on questions related to entities and their lexical representations (i.e. names and nicknames). More specifically, we focus on the co-hyponymy (i.e. *type* similarity) relation which has direct application in named-entity recognition applications (chapter 7). We then move to a related question on the disambiguation of entity names (chapter 8).

Part IV - Labels, Web 2.0 Tags and other User-Generated Keywords: In part IV we address issues related to some web-specific lexical items namely (i) (manually assigned) news *topic labels* (chapter 9), (ii) *user-assigned tags* describing artists in a Web 2.0 radio (chapter 10), and (iii) *trigger keywords* provided by advertisers to their web ads (chapter 11). In all cases, the motivation is to obtain suggestions of lexical items that could be used to expand an initial set of examples. Suggestions are required to be synonyms, co-hyponyms or functional equivalents of the seed examples in the application context at stake. The main questions addressed are related to choosing the appropriate sources for grounding the computation of similarity for these web-specific items.

To maintain a more focused discussion, for each chapter of parts II to IV we present specific sections regarding related work. Also, in each of those chapters, we describe some specific contributions to which we add the more general contributions that arise from this thesis (summarized in section 1.4).

Finally, in part V (chapter 12), we begin by summarizing the work presented in this thesis. Then we explicitly state the answers to research questions posed in this chapter, and outline the main contributions of this thesis. We end by suggesting lines for future work.

1.3 Foundations

The work presented in this thesis is the outcome of several projects developed at LIACC's "Distributed AI & Robotics Group" (NIAD&R) in which I participated during the period of this thesis. Some of these projects were the result of successful cooperation between LIAAC and academia researchers such as Professor Marteen de Rijke and Professor Valentin Jijkoun (University of Amsterdam) Professor Lyle Ungar (University of Pennsylvania), Doctor Fabien Gouyon (Inescporto) and Professor Mário Silva and Doctor Paula Carvalho (University of Lisbon). Others have been established with leading industry practitioners such as Google NY[1], and Sapo.pt[2]. Other have been started with my colleague, Sérgio Nunes. Undoubtedly, without such collaborations this thesis would not have been possible.

This thesis contains some work that has already been published, namely in the following papers (ordered by date of publication):

- "More like these": growing entity classes from seeds [SJdRO07]
- Music Artist Tag Propagation with Wikipedia abstracts [SGO09]
- Inferring Local Synonyms for Improving Keyword Suggestion in an On-line Advertisement System [STGO09]
- An Approach to Web-scale Named-Entity Disambiguation [SKOU09a]
- Efficient clustering of web-derived data sets [SKOU09b]
- Exploring the Vector Space Model for Finding Verb Synonyms in Portuguese [SCO09]

[1]Special thanks to all the Google Team in NY and, in particular, to Alexander Kehlenbeck, Casey Whitelaw and Nemanja Petrovic

[2]Special thanks to Paulo Trezentos, João Pedro Gonçalves, Celso Martinho and Benjamin Junior

- Propagating Fine-Grained Topic Labels in News Snippets [SNTO09]

While working on the research presented in this thesis, other lines of research were explored. These lines were definitely useful for acquiring valuable background knowledge and intuition, and for preparing future research. Nevertheless, for the sake of brevity and consistency, *not* all such work was integrated in this thesis. It can, however, be found in the following publications (ordered by date):

- SIEMÊS - a Named-Entity Recognizer for Portuguese Relying on Similarity Rules [Sar06c]
- REPENTINO - A Wide-Scope Gazetteer for Entity Recognition in Portuguese [SPC06]
- BACO - A large database of text and co-occurrences [Sar06a]
- A first step to address biography generation as an iterative QA task [Sar07]
- Assessing the Impact of Thesaurus-Based Expansion Techniques in QA-centric IR [STO08]
- Visualizing Networks of Music Artists with RAMA [SGCO09]
- The Design of OPTIMISM, an Opinion Mining System for Portuguese Politics [SCS$^+$09]
- Automatic Creation of a Reference Corpus for Political Opinion Mining in User-Generated Content [SCSdO09]
- Clues for Detecting Irony in User-Generated Contents: Oh...!! It's "so easy" ;-) [CSSdO09]
- Comparing Sentence-Level Features for Authorship Analysis in Portuguese [SSSG$^+$10]
- Comparing Verb Synonym Resources for Portuguese [TSO10]

1.4 Outline of the Main Contributions

This thesis provides three main contributions:

1. An original *formal framework* for *uniform* treatment of *several* similarity-related concepts (including *disambiguation*), applicable *both* to traditional and to Web-based text environments;
2. A study on the *information sources* and *types of features* that can be used for supporting the computation of similarity-related functions, as well as some of their intrinsic *limitations*;

3. The proposal of several *methods* and *metrics* and *strategies* for obtaining *gold-standard information* for evaluating similarity-related functions in a variety of *different scenarios*.

Part I

Background

The goal of part I is mainly to provide background information for the remainder of this thesis. It contains all the common ground required for parts II to IV, thus avoiding the need for repeating the discussion about generic methodological issues in each of the subsequent parts.

In chapter 2, we introduce and formalize the main concepts related to similarity, which are explored in more detail later in parts II to IV. We begin by presenting further motivation for studying similarity, in the context of *traditional* and *non-traditional* text formats. Then, after introducing some basic concepts and terminology, we provide formal definitions for *Semantic Similarity* (including some more specific concepts such as *Content Similarity*, *Type Similarity* and *Functional Similarity*) and *Semantic Ambiguity*.

Then, in chapter 3, we focus on the main mathematical tool used in this thesis, the Vector Space Model (VSM), which is used in most of the experiments here presented. We address several issues that are fundamental for a successful utilization of the VSM, namely issues related to *extracting* information for building feature vectors, *weighting* such feature information and *comparing* the resulting features vectors.

Finally, in chapter 4, we present methods, measures and possible gold standard resources for evaluating the performance of similarity-based functions. This chapter first includes a panoramic view about *evaluation*, and then it addresses more specific issues that are fundamental to understanding the evaluation procedures used in our experiments.

Chapter 2

Similarity and Ambiguity

We begin this chapter by providing a concrete example of why the concept of *semantic similarity* is important for practical Natural Language Processing (NLP) tasks. We then introduce the essential concepts for defining Semantic Similarity in an encompassing way. We proceed by providing definitions for the several specific notions of Semantic Similarity which are addressed in the thesis, namely *Content Similarity*, *Type Similarity* and *Functional Similarity*. Next, we provide a definition for Semantic Ambiguity, a strongly related concept. Finally, we present a measure of the degree of ambiguity, and formalize *disambiguation* as a similarity computation problem.

2.1 Why is similarity important?

Similarity is central to many language processing tasks, especially those involving classification or analysis. In fact, many classification or analysis operations can be formulated as a similarity-based procedure, such as:

based on the example annotations made on these snippets of texts, perform equivalent annotation (or classification) in all "similar" situations found in other texts

Let us consider, for instance, the task of Name-Entity Recognition (NER). Given an example *annotation*, A_0:

A_0**:** "... the [composer]$_{ERGO}$ [Richard Wagner]$_{PER}$ ..."

where ERGO represents an *ergonym* or *job title* and PER represents a person entity, we would like to have a NER system capable of annotating the following *instances*, I_1 and I_2:

I_1**:** "... the composer Johann Sebastian Bach ..."

I_2**:** "... the pianist Richard Wagner ..."

Such a NER system should be able to detect that I_1 and I_2 are structurally and semantically *similar* to the example A_0, and that they should thus be annotated likewise, i.e.:

A_1**:** "... the [composer]$_{ERGO}$ [Johann Sebastian Bach]$_{PER}$ ..."

A_2**:** "... the [pianist]$_{ERGO}$ [Richard Wagner]$_{PER}$ ..."

Such ideal NER system should also be able to conclude that the following instances:

I_3**:** "... the German Ludwig van Beethoven ..."

I_4**:** "... the famous Wolfgang Amadeus Mozart ..."

are not as similar to the example A_0 as the two previous instances (I_1 and I_2), but they are similar enough to be possible to annotate them *partially* and, at the same time, infer that there are two new "classes" of words (that could be learned later):

A_3**:** "... the [German]$_{?\#1}$ [Ludwig van Beethoven]$_{PER}$ ..."

A_4**:** "... the [famous]$_{?\#2}$ [Wolfgang Amadeus Mozart]$_{PER}$ ..."

Finally, the NER system should conclude that the following instance:

I_5**:** "... the opera Richard Wagner ..."

is not similar enough to the initial example and, thus, it should not be annotated in the same way.

The core of such an ideal system is the ability to detect the similarities between instances. More specifically, the system would be capable of recognizing similar *structures* and similar *components* within these instances. Let us assume that we have a function $\mathcal{S}$ that is capable of computing the value of the similarity between the referents of two lexical items given as input. For simplicity, we assume that:

- *lexical items* are either common nouns or entity names.
- $\mathcal{S}$ has only two parameters, i.e. the lexical items to be compared (i.e. all information about context is, for now, disregarded).
- the values produced by the function range from 0 (no similarity) to 1 (full similarity)

Then, given two items to be compared, i_i and i_j, the similarity of their referents is obtained by function $\mathcal{S}(i_i, i_j)$:

$$\mathcal{S}(i_i, i_j) \longrightarrow [0, 1] \tag{2.1}$$

For now, the mechanics of $\mathcal{S}$ need not to be explained, and $\mathcal{S}$ can be said to work as an *oracle*. Using such oracle, our ideal NER system is able to know that:

1. $\mathcal{S}$(“composer”, “pianist”) $> s_{min}$
2. $\mathcal{S}$(“Richard Wagner”, “Johann Sebastian Bach”) $> s_{min}$

with s_{min} being the minimum value of similarity for two items to be actually considered similar. Such knowledge is enough to support annotations A_1 and A_2 since the remainder components of instances I_1 and I_2 are the same as I_0.

Using the oracle, it is also possible to know that some components of instances I_3 and I_4 are very similar to the corresponding component of the annotated example I_0:

3. $\mathcal{S}$(“Richard Wagner”, “Ludwig van Beethoven”) $> s_{min}$
4. $\mathcal{S}$(“Richard Wagner”, “Wolfgang Amadeus Mozart”) $> s_{min}$

However, the oracle should be able to tell that the other components of instances I_3 and I_4 do not share enough similarity with the corresponding component in I_0:

5. $s_{excl} < \mathcal{S}$(“composer”, “German”) $< s_{min}$
6. $s_{excl} < \mathcal{S}$(“composer”, “famous”) $< s_{min}$

Nevertheless, because the similarities computed in 5) and 6) are still higher than a minimum threshold s_{excl}, it is possible to *partially* annotate I_3 and I_4 as shown in A_3 and A_4.

Finally, the oracle should be able to identify that the similarity between “composer” and “opera” is lower than the threshold s_{excl}:

7. $\mathcal{S}$(“composer”, “opera”) $< s_{excl}$

which excludes any chance of similarity between instance I_5 and the annotated example instance, and thus no annotation is performed.

Although this illustration is based on the entity recognition task, other examples of how such an ideal similarity function could be used in several language processing tasks (specially at the level of lexicon acquisition and semantic tagging) could be easily formulated. This simplified example, however, hides many of the subtleties of such an ideal similarity function. In the next sections, we expose them and provide a formalization for similarity.

2.2 Lexical Items, Referents and Dynamic Lexicons

The notion of similarity that was informally introduced in the previous example is *type similarity*, which is related with *class membership*: the words “composer” and "pianist" refer to music-related ergonyms, while the proper names "Richard Wagner", "Ludwig van Beethoven", "Wolfgang Amadeus Mozart" and "Johann Sebastian Bach" refer to people that can be said to belong to the class "composers" (among several other possible classes).

This, however, is not the only similarity paradigm we wish to address in this work. For example, *synonyms* do not fit in this type of paradigm. But, before defining similarity in a more encompassing way, we need to clarify some closely related concepts and terminology. We start by defining two core concepts: *lexical item* and *referent*.

Briefly, we can say that a *lexical item* is a *text symbol* (i.e. a word or a multi-word expression) that is used to represent a certain *referent*. A referent, which we denote by ρ, is any concrete or abstract entity or concept (including states and events) that belongs to an *universe of referents*, denoted by $\mathcal{R}$.

Standard definitions for these concepts are based on rather conservative assumptions, which do not hold when the text environment at stake is the Web. For example, the traditional definition of lexical item implicitly assumes the pre-existence of a stable lexicon (one that is represented in a standard dictionary), which defines the set of *valid* items (i.e. words) that can be used in text production.

However, it is clear that the set of symbols used on the Web, which goes far beyond traditional "words" (e.g. tags), is constantly growing as a direct consequence of the continuous expansion of the universe of *referents*. Let $\#\mathcal{R}_{www}(t)$ be the (unknown) number of referents that exist on the Web in a given time instant. We assume that such universe is growing *monotonically*, since new information is constantly being added to the Web:

$$\#\mathcal{R}_{www}(t+t') \geq \#\mathcal{R}_{www}(t), \forall t' > 0 \tag{2.2}$$

Additionally, because there is no practical way of knowing all the information that exists on the Web at a given time, the universe of referents $\mathcal{R}_{www}(t)$ can only be *partially known*. We are now able to introduce the concept of *Dynamic Lexicon* of the Web, denoted by $\mathcal{L}_{www}(t)$:

2.2.1. Definition. A Dynamic Lexicon of the Web, $\mathcal{L}_{www}(t)$, is an open set of symbols used in the Web in a given time instant t. The Dynamic Lexicon of the Web includes common lexicon (traditional words), names, specific-domain lexicon (technical terminology), acronyms, Web 2.0 tags, automatically generated labels, "emoticons", spelling mistakes and all other more or less idiosyncratic symbols and expressions used on the Web.

The number of elements in $\mathcal{L}_{www}(t)$, i.e. $\#\mathcal{L}_{www}(t)$, is also supposed to grow monotonically:

$$\#\mathcal{L}_{www}(t+t') \geq \#\mathcal{L}_{www}(t), \forall t' > 0 \tag{2.3}$$

Again, $\mathcal{L}_{www}(t)$ can only be partially known. For these reasons, it becomes virtually impossible to have a complete and up to date dictionary of the lexicon of the Web, $\mathcal{L}_{www}(t)$. Any definition of *lexical item* that is intended to be of practical use in the Web environment cannot depend on the existence of a pre-compiled dictionary.

Thus, we propose the following generic definition for Lexical Item:

2.2.2. Definition. A *Lexical Item*, denoted by $l_{\sigma,\rho}$ is a symbol σ belonging to Lexicon $\mathcal{L}_{www}(t)$ that is used to represent a certain *referent*, $\rho_r \in \mathcal{R}_{www}(t)$.

This definition of Lexical item is *relational* because it relates a symbol σ with a referent ρ. Such relation may only be verifiable in a certain *semantic context*, such as a specific *domain* or a specific *subset of the Web*. We denote such contexts by $\mathcal{C}$. Also, we are assuming that the set of *valid* symbols is dynamic, and cannot be completely known in advance. This imposes two important requirements for *robust* language processing procedures:

1. ability to adapt to different semantic contexts;
2. ability to process *unknown* lexical items.

2.3 Defining Similarity

We can now provide a generic definition of Similarity taking into account the previously defined concepts. Let $\mathcal{S}$ be a function that computes the value of similarity between the referents of two lexical items, taking into account a given context, $\mathcal{C}$. For convenience reasons, and without loss of generality, let us assume that the value of similarity ranges from 0 (no similarity) to 1 (full similarity):

$$\mathcal{S}(l_{\sigma_i,\rho_i}, l_{\sigma_j,\rho_j}, \mathcal{C}) \longrightarrow [0,1] \tag{2.4}$$

We can specialize this function taking into account different types of similarity. The examples given in the previous section illustrate *just one* case of similarity - *Type Similarity* - but we wish to have a definition applicable to other cases, namely *Content Similarity* and *Functional Similarity*.

2.3.1 Content Similarity

Content Similarity occurs when two lexical items have the same referent, despite using different symbols. It embraces two other more specialized – and traditional – semantic relations:

- *synonymy*, when symbols are part of the common lexicon (names, verbs, adjectives, etc.); and

- *paraphrase*, when at least one of the symbols is a multi-word expression, such as a *technical term* or an *idiomatic expression*.

Content Similarity also includes the relation of *name equivalence* that occurs when symbols are *name variations* or *nicknames* of the same entity.

We propose the following definition for Content Similarity:

2.3.1. DEFINITION. Two different lexical items, l_{σ_i,ρ_i} and l_{σ_j,ρ_j} with $\sigma_i \neq \sigma_j$, are said to be *content similar* in a certain context $\mathcal{C}$, if the two referents ρ_i and ρ_j can be said to be the same $\rho_i = \rho_j$ (or equivalent, $\rho_i \simeq \rho_j$).

Let us now define *Content Similarity Function*, $\mathcal{S}_{cnt}$, as a function that measures the degree of equivalence ϵ between the referents of two lexical items:

$$\mathcal{S}_{cnt}(l_{\sigma_i,\rho_i}, l_{\sigma_j,\rho_j}, \mathcal{C}) = \epsilon_{cnt}(\rho_i, \rho_j, \mathcal{C}) \tag{2.5}$$

with ϵ_{cnt} being a boolean function based on the equivalence of the referents:

$$\epsilon_{cnt}(\rho_i, \rho_j, \mathcal{C}) = \begin{cases} 1 & \text{if} \rho_i = \rho_j \\ 0 & \text{if} \rho_i \neq \rho_j \end{cases} \tag{2.6}$$

In practice, it might not possible to have a straightforward binary equivalence function, so an alternative might be using a continuous equivalence function, such as:

$$\epsilon_{cnt}^{\simeq}(\rho_i, \rho_j, \mathcal{C}) = \begin{cases} 1 & \text{if } \rho_i = \rho_j \\ ]0,1[& \text{if } \rho_i \simeq \rho_j \\ 0 & \text{if } \rho_i \neq \rho_j \end{cases} \tag{2.7}$$

whose result might then be binarized according to a given decision threshold.

2.3.2 Type Similarity

Type similarity is the relation between two lexical items whose referents belong to the some more or less specific *class*. When considering common lexicon (nouns, adjectives, etc), this relation is usually known as *co-hyponymy*. For example, "orange" and "lemon" are *co-hyponyms* because they share a common hypernym: "fruit". When considering *names of entities*, there is no specific term in literature so we use the term *class relatives*. For example, "Coldplay" and "Travis" are *class relatives* since they both belong to the class "britpop band".

Generically, we can propose the following definition regarding *Type Similarity*:

2.3.2. DEFINITION. Two different lexical items, l_{σ_i,ρ_i} and l_{σ_j,ρ_j} with $\sigma_i \neq \sigma_j$, are said to be *type similar* in a certain context $\mathcal{C}_c$, if the two corresponding referents ρ_i and ρ_j are elements of at least one common class.

Let us denote $\mathcal{S}_{type}$ as the *Type Similarity Function*, which computes the degree of type similarity between the referents of two lexical items. The *Type Similarity Function* can be expressed by:

$$\mathcal{S}_{type}(l_{\sigma_i,\rho_i}, l_{\sigma_j,\rho_j}, \mathcal{C}) = \epsilon_{type}(\rho_i, \rho_j, \mathcal{C}) \tag{2.8}$$

For defining $\mathcal{S}_{type}(l_{\sigma_i,\rho_i}, l_{\sigma_j,\rho_j}, \mathcal{C})$ we have to consider that the relation between items and classes is complex: not only an item can belong to many classes, but also the degree to which it belongs to any of these classes is not a strictly binary function. Consider, for example the items "Michelangelo" and "Leonardo da Vinci". In principle, they both can be assigned to any of the following classes c_1: "Renaissance artists", c_2: "Italian artists" or c_3: "Italian people". Binary membership cannot deal with such descriptive vagueness straightforwardly.

Let us start by defining a *class membership function*[1] $\mu(l_{\sigma_i,\rho_i}, c_k)$ that computes the degree of membership of the referent of l_{σ_i,ρ_i} in class c_k (a value between 0 and 1):

$$0 \leq \mu(l_{\sigma_i,\rho_i}, c_k) \leq 1, \forall i, k \tag{2.9}$$

For the example above, we expect μ("Jan van Eyck", c_1) $\simeq 1$, while μ("Jan van Eyck", c_2) $\simeq 0$. Again, to account for the influence of the context, $\mathcal{C}$, in the level of membership, we can introduce an additional parameter to Equation 2.9:

$$0 \leq \mu(l_{\sigma_i,\rho_i}, c_k, \mathcal{C}) \leq 1, \forall i, k, \mathcal{C} \tag{2.10}$$

We can further simplify this function in order to take into account only the referent ρ_i:

$$0 \leq \mu(\rho_i, c_k, \mathcal{C}) \leq 1, \forall i, k, \mathcal{C} \tag{2.11}$$

Now, since lexical items may belong to many different classes, the task of ranking items by similarity is not as trivial as it looks, and in most cases it is highly dependent on the context. For example, "Lisbon" and "Porto" are entities part of the class "*Portuguese cities*", while "Lisbon" and "Madrid" are part of the class "*Iberian Capitals*". Therefore, if context C_1 focus on *National (Portuguese) Affairs*, then we might have:

$$\mathcal{S}_{type}(\text{``}Lisbon\text{''}, \text{``}Porto\text{''}, \mathcal{C}_1) > \mathcal{S}_{type}(\text{``}Lisbon\text{''}, \text{``}Madrid\text{''}, \mathcal{C}_1) \tag{2.12}$$

But, if context C_2 is related with *International Politics*, then possibly:

$$\mathcal{S}_{type}(\text{``}Lisbon\text{''}, \text{``}Porto\text{''}, \mathcal{C}_2) < \mathcal{S}_{type}(\text{``}Lisbon\text{''}, \text{``}Madrid\text{''}, \mathcal{C}_2) \tag{2.13}$$

Scope plays also an important role in defining the degree of type similarity. For example, "orange", "lemon" and "strawberry" are all co-hyponyns of "fruit" but only "orange", "lemon" are co-hyponyms of "citrine". So, one might say that

[1]We borrow the terminology from Fuzzy Set Theory.

"orange" and "lemon" are more *type* similar than "orange" and "strawberry" or "lemon" and "strawberry".

In order to provide a more complete description of the Type Similarity Function, $\mathcal{S}_{type}$, we need to include additional factors in the formalization. Let $M(\rho_i)$ be the set of classes of which the referent ρ_i can be considered member of:

$$M(\rho_i) = \{c_1, c_2, ...c_n\} \tag{2.14}$$

$M(\rho_i)$ can be transformed into a *vector of weights* using the values provided by the class membership function $\mu(\rho_i, c_k, \mathcal{C})$ (see Equation 2.11), which describe the degree of membership of ρ_i to each of the classes in $M(\rho_i)$:

$$\overline{C(\rho_i)} = [(c_1, \mu(\rho_i, c_1, \mathcal{C})), (c_2, \mu(\rho_i, c_2, \mathcal{C})), ...(c_n, \mu(\rho_i, c_n, \mathcal{C}))] \tag{2.15}$$

Now, let Ω be the *Class Overlap Function* that measures the overlap between the classes to which two referents ρ_i and ρ_j belong, taking into account context $\mathcal{C}$.

$$\Omega(\overline{M(\rho_i)}, \overline{M(\rho_j)}, \mathcal{C}) \longrightarrow [0, 1] \tag{2.16}$$

We can now redefine the generic *Type Similarity Function* first introduced in Equation (2.8) in order to take into account both (i) the various classes to which each referent can belong, (ii) the different degrees of membership of the referents to each of the corresponding classes, and (iii) the context $\mathcal{C}$ in which we are performing the computation of similarities:

$$\mathcal{S}_{type}(l_{\sigma_i,\rho_i}, l_{\sigma_j,\rho_j}, \mathcal{C}) = \Omega(\overline{M(\rho_i)}, \overline{M(\rho_j)}, \mathcal{C}) \tag{2.17}$$

In section 3.5 we present some instantiations of Ω, although many other are possible. Generically, $\Omega(\overline{M(\rho_i)}, \overline{M(\rho_j)}, \mathcal{C})$ function is to be defined in a context-dependent fashion.

2.3.3 Functional Similarity

Briefly, *Functional Similarity* relates lexical items that can be alternatively used as *inputs* to a given natural language processing function and still lead to the production of the *same*, or *equivalent* result.

Let us assume the existence of a *low-level* language processing operation, $\mathcal{O}$, that takes as input one lexical item, l_{σ_i,ρ_j} and, based on it, operates over a given text object, $\mathcal{T}$, to produce the *intermediate output* o_{inter}:

$$o_{inter} = \mathcal{O}(\mathcal{T}, l_{\sigma_i,\rho_j}) \tag{2.18}$$

Let us further assume that such output is intended to be used by a *higher-level* function $\mathcal{F}$, to produce the *final result* $\mathcal{R}_{final}$:

$$\mathcal{R}_{final} = \mathcal{F}(o_{inter}) \tag{2.19}$$

Though abstractly defined, the *low-level* operation $\mathcal{O}$ and the *high-level* function $\mathcal{F}$ can be related to many concrete NLP tasks. For example, $\mathcal{O}$ can be the operation of (automatic or manual) tag assignment to a blog page (i.e. assigning $l^{tag}_{\sigma_i,\rho_j}$ to $\mathcal{T}^{blog}$) to improve retrieval efficiency by retrieval function $\mathcal{F}^{blog}_{IR}$. Or, $\mathcal{O}$ can be the operation of automatically selecting a term, $l^{exp}_{\sigma_i,\rho_j}$, for expanding a query (i.e. $\mathcal{T}^{query}$) for improving the recall of a document retrieval function, $\mathcal{F}^{doc}_{IR}$.

In this setting it becomes straightforward to define Functional Similarity.

2.3.3. Definition. Two different lexical items, l_{σ_i,ρ_i} and l_{σ_j,ρ_j} with $\sigma_i \neq \sigma_j$, are said to be *functional similar* in a certain context $\mathcal{C}_c$ if, given a *low-level text operation*, $\mathcal{O}(\mathcal{T}, l_{\sigma_k,\rho_k})$, which produces as intermediate result, o_{inter}, and $\mathcal{F}$, a *high-level function* that consumes o_{inter} in order to produce a final result, $\mathcal{R}$, we obtain the same or equivalent results from function $\mathcal{F}$ no matter whether we use l_{σ_i,ρ_i} or l_{σ_j,ρ_j} to obtain the intermediate result required by $\mathcal{F}$:

$$\mathcal{F}(o^i_{inter}) = \mathcal{F}(o^j_{inter}) \Longleftrightarrow \mathcal{R}_i = \mathcal{R}_j \tag{2.20}$$

with

- $o^i_{inter} = \mathcal{O}(\mathcal{T}, l_{\sigma_i,\rho_i})$
- $o^j_{inter} = \mathcal{O}(\mathcal{T}, l_{\sigma_j,\rho_j})$

Or, more compactly, by exposing the *functional pipeline*, $\mathcal{F} \odot \mathcal{O}$:

$$\mathcal{F}(\mathcal{O}(\mathcal{T}, l_{\sigma_i,\rho_i})) = \mathcal{F}(\mathcal{O}(\mathcal{T}, l_{\sigma_j,\rho_j})) \Longleftrightarrow \mathcal{F} \odot \mathcal{O}(\mathcal{T}, l_{\sigma_i,\rho_i}) = \mathcal{F} \odot \mathcal{O}(\mathcal{T}, l_{\sigma_j,\rho_j}) \tag{2.21}$$

We define the *Functional Similarity Function*, $\mathcal{S}_{fnc}$ as the functions that measures the functional equivalence of two lexical items. The function is computed by calculating the degree of equivalence of the results produced by the pipeline $\mathcal{F} \odot \mathcal{O}$ for each one of the two lexical items (given as input):

$$\mathcal{S}_{fnc}(l_{\sigma_i,\rho_i}, l_{\sigma_j,\rho_j}, \mathcal{C}) = \epsilon_{fnc}(\mathcal{R}_i, \mathcal{R}_j, \mathcal{C}) \tag{2.22}$$

with:

- $\mathcal{R}_i = \mathcal{F} \odot \mathcal{O}(\mathcal{T}, l_{\sigma_i,\rho_i})$
- $\mathcal{R}_j = \mathcal{F} \odot \mathcal{O}(\mathcal{T}, l_{\sigma_j,\rho_j})$

and ϵ_{fnc} being a binary equivalence function:

$$\epsilon_{fnc}(\mathcal{R}_i, \mathcal{R}_j, \mathcal{C}) = \begin{cases} 1 & \text{if } \mathcal{R}_i = \mathcal{R}_j \\ 0 & \text{if } \mathcal{R}_i \neq \mathcal{R}_j \end{cases} \tag{2.23}$$

Again, in practice it might be important to have a continuous equivalence function, such as:

$$\epsilon_{fnc}^{\approx}(\mathcal{R}_i, \mathcal{R}_j, \mathcal{C}) = \begin{cases} 1 & \text{if } \mathcal{R}_i = \mathcal{R}_j \\ ]0,1[& \text{if } \mathcal{R}_i \simeq \mathcal{R}_j \\ 0 & \text{if } \mathcal{R}_i \neq \mathcal{R}_j \end{cases} \tag{2.24}$$

so that:

$$\mathcal{S}_{fnc}(l_{\sigma_i,\rho_i}, l_{\sigma_j,\rho_j}, \mathcal{C}) = \epsilon_{fnc}^{\approx}(\mathcal{R}_i, \mathcal{R}_j, \mathcal{C}) \longrightarrow [0,1] \tag{2.25}$$

In many scenarios, Functional Similarity subsumes Content Similarity and Type Similarity. For example, synonyms (and co-hyponyms) can certainly be valid expansions for a given search keyword in most IR settings, so they are *both* Content (or Type) Similar and Functionally Similar. The key point is that while Content Similarity and Type Similarity presuppose a conceptual framework from which relations are derived (even if such framework is not totally accessible, is abstract, or is context dependent), Functional Similarity depends only on the impact that lexical items have on the results of a given NLP function pipeline, $\mathcal{F} \odot \mathcal{O}$.

2.4 Semantic Ambiguity

Semantic ambiguity occurs whenever two (or more) lexical items have the *same* lexical representation but mention *different* referents. Semantic Ambiguity is also known as *homography*. Using notation defined in the previous sections we can provide a formulation for semantic ambiguity:

2.4.1. DEFINITION. Two lexical items, l_{σ_i,ρ_i} and l_{σ_j,ρ_j} are said to be lexically ambiguous if their lexical representation, σ_i and σ_j, are the same, $\sigma_i = \sigma_j$, but the two corresponding referents, ρ_i and ρ_j, are *not* the same, $\rho_i \neq \rho_j$.

In this definition we do not frame the event of lexical ambiguity within "certain context $\mathcal{C}_c$" as we do when defining the several different notions of similarity. In fact, all lexical items are potentially semantically ambiguous in the sense that, in practice, there is no way of *ensuring* that there exists only one referent for a given same lexical representation in *all* possible contexts.

Interestingly, when restricting ourselves to smaller contexts, semantic ambiguity may not exist, or may not be *relevant*. For example, when focusing on news about politics, name ambiguity is not as severe as one would initially think, since the most popular politicians tend become known by names that do not collide with the names of other popular politicians. So, in the context of "politic news" name ambiguity might almost never occur, at least for the most frequently mentioned politicians. There are, however, a few good counterexamples in American politics due to the existence of political "clans", such as the Bush or the Clinton, but even these tend not to co-occur within the same time span.

Quantifying Ambiguity

To quantify the ambiguity that exists between several lexical items in a certain context, we can compute the *entropy* of the corresponding symbol in relation to the set of possible referents. Let $\mathcal{C}_c$ be a context in which it is possible to find n semantically ambiguous lexical items, i.e. having the same lexical representation but different referents: l_{σ,ρ_1}, l_{σ,ρ_2} ... l_{σ,ρ_n}. Let p_1, p_2 ... p_n be the probability of the a specific occurrence of σ in such context referring to ρ_1, ρ_2, ... ρ_n, respectively. Then:

$$E_{amb} = -\sum_{i}^{n} p_i \log_2 p_i \qquad (2.26)$$

quantifies the degree of ambiguity that there is *in practice*. If all referents are equally probable ($p_i = k$, $\forall\ i$) then E_{amb} reaches a maximum, indicating that there is in fact great uncertainty when guessing the referent of symbol σ. If, on the other hand, the probability of one referent is significantly larger than all others ($p_j \gg p_i$, $\forall\ i \neq j$), then E_{amb} will be close to zero, meaning that almost all occurrences of σ found in context $\mathcal{C}_c$ mention one specific referent. In such case, the ambiguity of the lexical items should not be a particularly relevant problem *in practice*.

Disambiguation and Similarity

Disambiguating two ambiguous lexical items, i.e. determining if they mention the *same* referent or *not*, can be formulated as a question of measuring how *similar* the corresponding referents are within a given context. Let l_{σ,ρ_1} and l_{σ,ρ_2} be two potentially ambiguous lexical items in context $\mathcal{C}_c$. For instance, these two lexical items may have been found in two different documents that belong to the same document collection. Let $\mathcal{S}_{cnt} = \epsilon_{cnt}^{\simeq}(\rho_i, \rho_j, \mathcal{C})$ be the content similarity function as defined by Equations 2.5 and 2.7 (see section 2.3.1). A decision regarding the disambiguation of l_{σ,ρ_1} and l_{σ,ρ_2} can be made according the following rules:

1. if $\epsilon_{cnt}(\rho_1, \rho_2, \mathcal{C}_c) \geq \epsilon_{min}$ then lexical items l_{σ,ρ_1} and l_{σ,ρ_2} mention the *same* referent

2. if $\epsilon_{cnt}(\rho_1, \rho_2, \mathcal{C}_c) < \epsilon_{min}$ then lexical items l_{σ,ρ_1} and l_{σ,ρ_2} mention *different* referents

Hence, in this setting disambiguating lexical items becomes a matter of choosing the appropriate content similarity function for comparing the referents (within the context at stake).

2.5 Conclusion

In this chapter we formalized the fundamental concepts about similarity which are the core of our research. We began by defining:

- Lexical Item;
- Symbol and Referent; and
- Dynamic Lexicon.

We then formalized three cases of similarity between lexical items:

1. Content Similarity;
2. Type Similarity; and
3. Functional Similarity.

We showed how each case can be supported by other lower-level functions, such as *content equivalence functions*, *class overlap functions* and *functional equivalence functions*, and how they can thus be grouped in order to form a uniform concept of similarity. Finally, we presented a compatible formalization for Semantic Ambiguity, and showed how disambiguation can be formulated as a Content Similarity problem. Although the formalization we provide is, at this stage, rather abstract, in parts II to IV we illustrate how these concepts instantiate in practical situations.

Chapter 3
The Vector Space Model

In this chapter, we focus on one of the fundamental tools used in this thesis: the Vector Space Model (VSM). We explain the principles underlying the use of the VSM as the basis for computing similarity. We describe the main issues related to the representation of text information in the VSM, and we discuss specific questions regarding the selection of the *feature space* and the usage of *feature weighting functions*. Then, because the computation of similarity between lexical items and the computation of vector proximity are *isomorphic* operations, we present several functions for computing the proximity (or distance) between vectors on real-valued and binary-valued vector spaces.

3.1 Introduction

In the Vector Space Model (VSM), each lexical item l_{σ_i,ρ_i} is represented by a *vector of features* in a multidimensional *feature space*, $\overline{l_{\sigma_i,\rho_i}}$. Such vector representations allow us to perform several language processing operations by isomorphically computing geometric operations over vectors (e.g. distance between two vectors). We can, thus, take advantage of strong algebraic frameworks to compute complex semantic functions, which would otherwise be very difficult to compute due to lack of *simple* conceptual and operational formalizations.

For instance, the Vector Space Model provides a very convenient framework for computing content similarity between lexical items because it allows to geometrically express a strong intuition regarding semantic similarity: the *Distributional Hypothesis* [Har54]. According to the Distributional Hypothesis, words[1] that occur in the *same contexts* tend to have *"similar" meanings* (i.e. should be synonyms). If words are transformed into the appropriate vector representations (which we discuss later), then the vectors of two synonymous words should be

[1]In this paragraph we are using the term "word" instead our own "lexical item" in order to maintain the generally accepted formulation of the Distributional Hypothesis

"close" to each other in the vectors space. Alternatively, vectors that are relatively close in the vector space should correspond to sets of synonymous words. There are multiple possible variations and extensions to this concept but, essentially, it enables us to compute complex semantic operations by performing relatively simple vector operations.

3.2 VSM Parameters and Distance Metrics

The key point in using the VSM for computing semantic operations has to do with how lexical items are transformed in *feature vectors*. There are three essential parameters that control how such transformation is performed: (i) the choice of *feature context*, (ii) the *feature weighting function* and (iii) the choice of the *representation domain*.

Feature Context The *feature context* is the environment from which features are extracted in order to build *meaningful* vector representations. Relevant features can be found either at *lexical level* or at *syntactical level*. The choice of a specific feature context has a huge impact on the information that is *transferred* to the Vector Space, thus directly affecting the *notion of similarity* that may be inferred from feature vectors (see Sahlgren [Sah06]). For this reason, the *feature context* parameter is deeply explored throughout this thesis (more details in section 3.3).

Feature Weighting Function Another fundamental parameter for obtaining an appropriate vector representations is the *feature weighting function*. Usually, the value of each component of the feature vector is derived from a raw frequency value (e.g. the number of times a given reference context co-occurs with the lexical item). By using weighting functions (e.g. tf-idf [SM86], Pointwise Mutual Information [CH90], Log-Likelihood Ratio [Dun93], etc.) one can *promote* (or *demote*) different sections of the feature spectrum, thus compensating biases that arise from *raw frequency counts* (e.g. very frequent contexts tend to be over emphasized). Thus, the choice of a specific weighting function can have a deep impact in the final results of the operations that we wish to perform on the vector space (e.g. should "rare" features be considered important or should they be ignored?). We provide more details about feature weighting functions in section 3.4).

Representation Domain The components of the vectors are usually scalar values, either derived from direct frequency counts (which generate values in the domain $[0, \infty[$), or resulting from the application of a specific feature weighting function to raw-frequency data, (which can generate values in $]-\infty, \infty[$). When vectors have components ranging in any of these domains, we call the corresponding vector spaces *Real-Valued Vector Spaces*. It is possible to binarize a Real-valued Vector Space applying a threshold function to the real-valued components of all vectors, thus generating a *Binary Vector Space*. In Binary Vector Spaces, the values of the components of feature vectors are either “1” or “0”. In other words, the only information stored in the vector is the existence of a component regarding a specific feature (i.e. dimension). Despite this apparently excessive information loss, Binary Vector Spaces are successfully used in many data mining scenarios. There are situations in which the corresponding vector spaces are *high-dimensional* and *sparse*, and, therefore there is a very low probability of overlap between non-nil components of any two vectors. In such situations, the mere fact that two vectors share specific non-nil components might be more relevant than the scalar values that those components have, since overlap itself is a very rare event. Additionally, Binary Vector Spaces are very convenient from a computational point of view: they allow very efficient memory representations (e.g. bitmaps). Also, bit-level operations are extremely efficient at CPU level. In practice, Binary Vector Spaces can be interesting options.

Other Thresholds For practical reasons, there are also several *cut-off thresholds* that can be used for controlling which feature vectors are actually included the Vector Space. These thresholds are important when there is not enough feature information to ensure a *faithful* vector representation. Adding such underspecified vectors to the Vector Space might lead to faulty conclusions when trying to compute certain vector operations. Therefore, low-frequency lexical items are usually excluded from the Vector Space. Feature vectors with a relatively low number of non-nil components (i.e. do not carry enough diversity), also tend to be kept out of Vector Spaces.

Distance Metrics

Since we are focusing on computing similarities between lexical items – l_{σ_i,ρ_i}, l_{σ_j,ρ_j}, ... l_{σ_n,ρ_n} – the fundamental isomorphic operation to be performed on the VSM is computing the distance between the corresponding vector representations – $\overline{l_{\sigma_i,\rho_i}}$, $\overline{l_{\sigma_j,\rho_j}}$, ... $\overline{l_{\sigma_n,\rho_n}}$. Therefore, the choice of a specific *distance metric* for comparing the (weighted) vectors is crucial. As detailed in section 3.5, there are two basic groups of metrics: *geometry-based distance metrics* and *probabilistic-inspired metrics*. Geometry-based distance metrics, such as *Euclidean Distance* (L2 distance) or the *cosine metric* [SM86], assume that vector space is Euclidean. On the other hand, probabilistic-inspired metrics, such as *Jensen-Shannon Distance* [Lin91] or the *Skew-Divergence* [Lee01], do not make such assumption.

As shown in various works (see for example Curran [Cur04]), global performance of VSM approaches depends on the combination of a specific weighting function and a specific distance metric, and there is usually an optimal combination for each task. For instance, the best combination of feature weighting function and distance metric for computing the degree of synonyms between lexical items might not be (and probably is not) the best combination for finding co-hyponyms.

3.3 Feature Context

The fundamental concept of the Vector Space Model is to use *vectors of features* as *summarized representations* of objects in order to simplify a set of operations we wish to perform over them. The vector description used can be seen as a *sample* of the complete description of the object (which might even not be possible to formalize completely) and should contain the appropriate feature information to *faithfully* represent the object in the Vector Space.

Feature information can be collected from the environment in which the object exists. In our case, *objects* are lexical items and the *environment* is a given text base, such as the Web or a specific-domain document collection. Potentially, feature information includes all sort of information that can be extracted from the text environment to describe relevant properties of the lexical item at stake. The concrete set of information sources on the environment actually used for extracting feature information is called *feature context*.

3.3.1. Definition. Let l_{σ_i,ρ_i} be an arbitrary lexical item belonging to a given Lexicon, $\mathcal{L}$. Let $\mathcal{E}$ be a *text environment* in which one can find information related to l_{σ_i,ρ_i}. The *feature context* is a *restriction* over $\mathcal{E}$ from which we can collect *feature information* regarding l_{σ_i,ρ_i} in order to build the corresponding feature vector representation, $\overline{l_{\sigma_i,\rho_i}}$, on Vector Space $\mathcal{V}$.

A certain feature context is said to be *directly observable* when feature information can be readily extracted from the *feature context*. That is, for example, the case of frequency counts regarding the co-occurrence of lexical items. If, on the other hand, some form of linguistic processing is required for extracting such features (e.g. POS annotation, syntactical parsing or dependency parsing), the feature context is said to be *indirectly observable*.

3.3.1 Directly Observable Feature Contexts

Directly Observable Feature Contexts allow feature information to be directly extracted from the text environment with no (or only very simplistic) linguistic pre-processing. Directly Observable Contexts are very convenient when large amounts of text data need to be processed, since no sophisticated linguistic tools, which may be expensive from a computational point of view, need to be used. However, Vector Spaces generated using Directly Observable Feature Contexts tend to have *very high-dimensionality* and are usually *sparse*. Consequently, operations on such spaces require efficient vector representations and may need significant computational resources.

An example of a Directly Observable Feature Context is *Window Context*. In this case, the set of features generated for a given lexical item at position k in a specific text block, $l^k_{\sigma_i,\rho_i}$, are the frequencies of the lexical items that immediately surround it within a pre-defined lexical window. Usually, the context window is defined by two numbers that identify which lexical items in the neighbourhood of item $l^k_{\sigma_i,\rho_i}$ should be considered for generating the feature: (i) the number of lexical items that *precede* $l^k_{\sigma_i,\rho_i}$, and (ii) the number of lexical items that follow $l^k_{\sigma_i,\rho_i}$. For example, if we consider a [-1, +2] window, item $l^k_{\sigma_i,\rho_i}$ (at position k) would generate one count for the feature $[l^{k-1}_{\sigma_h,\rho_h}, l^{k+1}_{\sigma_j,\rho_j}, l^{k+2}_{\sigma_l,\rho_l}]$. As it is easy to understand, for a lexicon $\mathcal{L}$ of size $|\mathcal{L}|$ a vector space created using [-1, +2] context window can potentially reach a dimensionality of $|\mathcal{L}|^3$. In practice, such value is never reached but the dimensionality of space will still be extremely high.

Another example of a Directly Observable Feature Context is the *Co-occurrence Context*. In this case, the features that can be observed are all lexical items that co-occur with the lexical item at stake, within a given *text block* (e.g. a sentence). In its simplest from, the Co-occurrence Context Scope comprises only information about the form and frequency of the co-occurring Lexical Units. There are several variations that also consider information regarding the *relative position* of the co-occurrences (after or before), or the *distance* in number of items at (positive or negative values) which the co-occurrence is detected.

3.3.2 Indirectly Observable Feature Contexts

Whenever linguistic analysis tools are available, it becomes possible to use information from *Indirectly Observable Feature Contexts*. This allows us to build feature vectors using higher-level information, which is possible to extract from the environment after some linguistic processing has been performed.

For example, part-of-speech tagging allows to obtain the grammatical category of a lexical item. These can be seen as higher-level linguistic features that abstract the lexical instantiation found in text (i.e. the *directly* observable features). Thus, the number of different features at stake (i.e. components in the Vector Space) is significantly lower, allowing more compact vector representations. Notably, a Window Context can still be used to collect valid feature information (e.g. the POS tags of the previous and the two following a lexical items).

If a *syntactical analyser* or a *dependency parser* is available, then it is possible to obtain feature information with an even higher-level of abstraction, since the resulting features do not depend on the position of the lexical items as much as features extracted with fixed-size windows do. We name this type of contexts as *Relational Feature Context* because they are based on the relations that a given lexical unit establishes with other lexical units within a certain *text block* (usually a sentence or a short phrase).

The work developed by Lin [Lin98] is one example of how a Relational Feature Context is used. The author used the Minipar parser to identify several types of syntactical relations (subject_of, object_of, modified_by, etc.) in a corpus. Data obtained for each word was then used to compile feature vectors containing information about all the words with which the word at stake is grammatically related to. Based on these feature vectors, the authors were able to calculate semantic similarity (i.e. content similarity) between words.

3.3.3 Partially Observable Feature Contexts

There are some cases where the distinction between *directly observable* and *indirectly observable* feature context is not so clear. In practice, the linguistic processing involved in order to obtain "higher-level features" might be quite simple if some approximations or heuristics are applied. Such features context can thus be considered *Partially Observable Feature Contexts*.

For example, Widdows and Dorow draw attention to the importance of a very specific grammatical relation - the coordination - as source for compiling feature Vectors [WD02]. The authors used the syntactic annotation provided in the British National Corpus (BNC) to extract all pairs (Noun , Noun) that are connected in coordination ("and" or "or"). Using this information, the authors are then able to cluster nouns. The authors claim that this simple method achieves good results in building certain semantic sets/clusters. However, since coordination can be easily identified in many languages due to the presence of explicit (and mostly unambiguous) connectors ("and" or "or"), it is possible to build simple lexical filters to "process" a corpus and extract such features, even if some noisy information is generated. Such a simple approach achieved interesting results for Portuguese [Sar06b], with coordination pairs being extracted using lexical patterns and lists of stop-words over a 6GB collection of web documents, WPT03 [MS04].

3.4 Feature Weighting Functions

As seen in the in section 3.3, *raw feature* information can be obtained by extracting information from several types of contexts. But, because raw feature information is based on *individual* frequency counts, it does not express the statistical significance of the co-occurrence between the item and the feature. For example, in Portuguese most nouns are preceded by a pronoun, so the existence of a feature "preceded_by_pronoun_x" would be common to most nouns and, therefore, it provides almost no useful information when comparing vectors of two nouns.

The purpose of a feature weighting function is to promote or demote the importance of specific features in vector descriptions of (lexical) items, in order to reflect which features are actually worth to be taken into account (or not) while comparing them. Feature weighting is common need in many fields, so there is a broad range of weighting functions proposed in literature from a wide variety of fields. Tan et al. compare twenty one feature weighting functions used in data mining [TKS04], while Evert presents twenty weighting function that are more frequently used in computation linguistics [Eve05]. In the next section, we will focus on several weighting function that are used in this thesis.

3.4.1 Observations, Contingency Table and Estimations

Weighting functions are based on different assumptions regarding how to make use of the raw counts. While some functions are borrowed from Statistics, others are based on concepts from *Information Theory*. There are also many weighting functions that are *heuristic* or *ad-hoc* variations of the previous ones.

Most of the feature weighting functions can be expressed in terms of four values related to the frequency of four different cases of co-occurrence between a given feature, f_j, and a lexical item, l_{σ_i,ρ_i}:

1. $o(l_{\sigma_i,\rho_i}, f_j)$: frequency of observation of feature f_j within the context of l_{σ_i,ρ_i}. This is usually the value that is observed from the text.

2. $o(l_{\sigma_i,\rho_i}, !f_j)$: frequency of observation of features *other than* f_i in the context of l_{σ_i,ρ_i}. Usually computed by:

$$o(l_{\sigma_i,\rho_i}, !f_j) = o(l_{\sigma_i,\rho_i}, *) - o(l_{\sigma_i,\rho_i}, f_j) \tag{3.1}$$

3. $o(!l_{\sigma_i,\rho_i}, f_j)$: frequency of observation of feature f_j in contexts *other than* that of l_{σ_i,ρ_i}. Usually computed as:

$$o(!l_{\sigma_i,\rho_i}, f_j) = o(*, f_j) - o(l_{\sigma_i,\rho_i}, f_j) \tag{3.2}$$

4. $o(!l_{\sigma_i,\rho_i}, !f_j)$: frequency of *all* observations that do not involve neither l_{σ_i,ρ_i} neither f_j. This is usually obtained by computing:

$$o(!l_{\sigma_i,\rho_i}, !f_j) = O - (o(!l_{\sigma_i,\rho_i}, f_j) + o(l_{\sigma_i,\rho_i}, !f_j) + o(l_{\sigma_i,\rho_i}, f_j)) \tag{3.3}$$

   with O being the *total number* of observations made (i.e. for all lexical items and all features).

The values of these observations (obtained for each possible pair of lexical items and features) are usually compiled in a *Contingency Table*:

$$T_{cont}(l_{\sigma_i,\rho_i}, f_j) = \begin{bmatrix} o(l_{\sigma_i,\rho_i}, f_j) & o(l_{\sigma_i,\rho_i}, !f_j) \\ o(!l_{\sigma_i,\rho_i}, f_j) & o(!l_{\sigma_i,\rho_i}, !f_j) \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{3.4}$$

with values a, b, c and d being abbreviations for the four observation values presented before ($O = a + b + c + d$). In some cases, weighting functions are expressed using values of probabilities. These can be readily obtained using a *Maximum Likelihood Estimator*:

$$p(l_{\sigma_i,\rho_i}, f_j) = \frac{o(l_{\sigma_i,\rho_i}, f_j)}{O} = \frac{a}{O} \tag{3.5}$$

$$p(l_{\sigma_i,\rho_i}) = \frac{o(l_{\sigma_i,\rho_i}, f_j) + o(l_{\sigma_i,\rho_i}, !f_j)}{O} = \frac{a+b}{O} \tag{3.6}$$

$$p(f_j) = \frac{o(l_{\sigma_i,\rho_i}, f_j) + o(!l_{\sigma_i,\rho_i}, f_j)}{O} = \frac{a+c}{O} \tag{3.7}$$

3.4.2 Significance Tests as Feature Weighting Functions

Significance Tests are statistical tools that allow making informed decisions about the nature of the process that generated the observations (stored in a Contingency Table). More specifically, they allow us to reject or confirm a given *hypothesis* about the process that generated such observations.

For using Significance Tests as Weighting Functions, one has to model the event of *observing feature f_j in the context of lexical item l_{σ_i,ρ_i}* as a *random process*. The event of *observing feature f_j* (in the context of any lexical item) has probability $p(f_j)$, given by Equation 3.7. Likewise, observing lexical item l_{σ_i,ρ_i} (associated with any feature) has probability $p(l_{\sigma_i,\rho_i})$, and is given by Equation 3.6. We can then formulate the *Null Independence Hypothesis*, H_0 that assumes that the occurrence of lexical item l_{σ_i,ρ_i} and the occurrence of feature f_j are two *independent events*. Then, under H_0, the probability of co-occurrence is given by:

$$p_0(l_{\sigma_i,\rho_i}, f_j) = p(l_{\sigma_i,\rho_i}) \times p(f_j) \tag{3.8}$$

which according to Equations 3.6 and 3.7 can be expressed using the values of the Contingency Table:

$$p_0(l_{\sigma_i,\rho_i}, f_j) = \frac{a+b}{O} \times \frac{a+c}{O} \tag{3.9}$$

Then, we can compute the probabilities of the three remainder events, assuming that H_0 holds. These are given by:

$$p_0(l_{\sigma_i,\rho_i}, !f_j) = \frac{a+b}{O} \times \frac{b+d}{O} \tag{3.10}$$

$$p_0(!l_{\sigma_i,\rho_i}, f_j) = \frac{c+d}{O} \times \frac{a+c}{O} \tag{3.11}$$

$$p_0(!l_{\sigma_i,\rho_i}, !f_j) = \frac{c+d}{O} \times \frac{b+d}{O} \tag{3.12}$$

From these, and from the total number of observations, we can derive the *expected number of observations* under the Null Hypothesis, H_0:

$$o_0(l_{\sigma_i,\rho_i}, f_j) = \frac{(a+b)\cdot(a+c)}{O} \tag{3.13}$$

$$o_0(l_{\sigma_i,!\rho_i}) = \frac{(a+b)\cdot(b+d)}{O} \tag{3.14}$$

$$o_0(!l_{\sigma_i,\rho_i}, f_j) = \frac{(c+d)\cdot(a+c)}{O} \tag{3.15}$$

$$o_0(!l_{\sigma_i,\rho_i}, !f_j) = \frac{(c+d)\cdot(b+d)}{O} \tag{3.16}$$

Alternatively, we can express such information in the form of a Contingency Table of *expected observations*:

$$T^0_{cont}(l_{\sigma_i,\rho_i}, f_j) = \begin{bmatrix} o_0(l_{\sigma_i,\rho_i}, f_j) & o_0(l_{\sigma_i,\rho_i}, !f_j) \\ o_0(!l_{\sigma_i,\rho_i}, f_j) & o_0(!l_{\sigma_i,\rho_i}, !f_j) \end{bmatrix} \tag{3.17}$$

Significance Tests measure how significant is the difference between the observation values that are expected under the Null Hypothesis, $T^0_{cont}(l_{\sigma_i,\rho_i}, f_j)$, and the values actually observed, $T_{cont}(l_{\sigma_i,\rho_i}, f_j)$. These tests provide a value that quantifies how much certainty there is in *rejecting* the Null Hypothesis. In other words, the value of the significance test provides a measure of how much the co-occurrence of the item l_{σ_i,ρ_i} and feature f_j is *not* a matter of chance, but is in fact a *relevant* event. Such value can then be used as the weight of vector component for the corresponding feature: the stronger the Null Hypothesis is to be rejected, the more relevant is that feature for describing the lexical item at stake.

One can find in literature three types of *Significance Tests*:

1. **Likelihood Tests**: Likelihood Tests aim at computing the probability of observing the values expressed in the Contingency Table $T_{cont}(l_{\sigma_i,\rho_i}, f_j)$ if the Null Hypothesis is actually true. Such probability is known as *Likelihood*. Low values for the Likelihood suggest that H_0 is probably not true.

2. **Exact Hypothesis Tests**: Exact Tests compute the probability of a *Type I Error* when taking a decision about the validity of the Null Hypothesis, H_0. A Type I Error occurs when the H_0 hypothesis is incorrectly rejected. For that, Exact Tests add evidence against the Null Hypothesis for *all* possible Contingency Tables that could be generated for the event at stake. This usually involves a significant computational effort for exploring the space of the possible Contingency Tables.

3. **Asymptotic Hypothesis Tests**: Asymptotic Tests are simplifications, which can be applied under certain conditions, of the Exact Tests. When the

sample size is large enough, the *Normality Assumption* becomes valid and the computations performed by Exact Tests can be greatly simplified. These tests are, thus, used in many fields. Asymptotic Hypothesis Tests compute a statistic that allows to check how well the frequency values observed are close to what is theoretically expected under H_0. For this reason, these tests are frequently mentioned as *goodness-of-fit tests*.

In practice, because of their simplicity, *Asymptotic Tests* are the most frequently used ones. Next, we present several of these Asymptotic Tests, which are commonly used as feature weighing functions.

Pearson χ^2 Test

Pearsons χ^2 Test is a statistic based on the differences between the values observed, $o(i)$, and the values expected under H_0, $o_0(i)$:

$$\chi^2 = \sum_{i=1}^{k} \frac{(o(i) - o_0(i))^2}{o_0(i)} \tag{3.18}$$

In our case, the summation is to be made over the elements of both Contingency Tables: $T_{cont}(l_{\sigma_i,\rho_i}, f_j)$ and $T^0_{cont}(l_{\sigma_i,\rho_i}, f_j)$. For a 2×2 Contingency Table, such as the ones used in feature weighting, the χ^2 statistic can be re-written using the abbreviated notation shown in Equation 3.4:

$$\begin{aligned}\chi^2 &= \frac{(a \cdot d - b \cdot c)^2 \cdot (a + b + c + d)}{(a + b) \cdot (a + c) \cdot (b + d) \cdot (c + d)} = \\ &= \frac{(a \cdot d - b \cdot c)^2 \cdot O}{(a + b) \cdot (a + c) \cdot (b + d) \cdot (c + d)}\end{aligned} \tag{3.19}$$

The value of χ^2 statistic is a random variable that can be approximated by the χ^2 distribution with $v = (r - 1) \cdot (c - 1)$ degrees of freedom, with r and c corresponding to the number rows and columns of the Contingency Table. Thus, in our case $v = 1$. When the value of the χ^2 statistic is low, we can say that the observations follow the theoretical model and H_0 can be considered valid. On the other hand, if the χ^2 statistic is higher than a given threshold, H_0 cannot be considered valid, meaning that there is a strong association between the feature f_j and the lexical item l_{σ_i,ρ_i}. The χ^2 test can thus be directly used as a weighting function.

However, approximating the χ^2 statistic by the χ^2 distribution is only safe when the number of degrees of freedom at stake is sufficiently large. For the cases in which $v = 1$, Equation 3.18 suffers a small change known as *Yates Correction*:

$$\chi^2_{Yates} = \sum_{i=1}^{k} \frac{(|o(i) - o_0(i)| - 0.5)^2}{o_0(i)} \tag{3.20}$$

The Yates Correction factor is insignificant when the value of the expected frequency, $o_0(i)$ is very high. For low values of $o_0(i)$, the Yates Correction factor is important. For cases where $o_0(i) \leq 5$, i.e. "rare events", there may be significant deviations even when using the correction factor.

ϕ^2 Coefficient

The ϕ^2 is a statistic often used as an association measure between two binary variables. The ϕ^2 statistic is related to the χ^2 statistic by:

$$\phi^2 = \frac{\chi^2}{O} \tag{3.21}$$

Thus, by using Equation 3.19 for the χ^2 statistic, we can express ϕ^2 in an abbreviated notation:

$$\begin{aligned}\phi^2 = \frac{(a \cdot d - b \cdot c)^2 \cdot O}{((a+b) \cdot (a+c) \cdot (b+d) \cdot (c+d)) \cdot O} &= \\ &= \frac{(a \cdot d - b \cdot c)^2}{(a+b) \cdot (a+c) \cdot (b+d) \cdot (c+d)}\end{aligned} \tag{3.22}$$

Contrary to the χ^2 statistic, the ϕ^2 statistic is bounded between 0 and 1. Church and Gale have used this statistic as a measure of association for the purpose of finding bilingual matches. The authors claimed that the ϕ^2 statistic is appropriate for this type of tasks since it makes a good use of the diagonal cells of the Contingency Table, i.e. b and c, which usually have relatively high counts and, thus, provide better support for estimations [CG91].

The Z-Score

The use of the *Z-Score* statistic in computational linguistics dates back to the 70's [BR73], and it has been, since then, a popular choice for identifying collocations (e.g Smadja's Xtract system [Sma93]).

The Z-Score is related with the *Central Limit Theorem*. When n_{smp} samples are drawn from a population whose distribution is unknown but whose *average*, μ_{pop}, and *standard deviation*, σ_{pop}, are known, the distribution of the *average value of the n_{smp} samples*, μ_{smp}, tends to a Normal Distribution with average μ_{pop} and standard deviation $\sigma_{pop} \,/ \sqrt{n_{amo}}$. Therefore, the statistic:

$$Z = \frac{\mu_{smp} - \mu_{pop}}{\sigma_{pop} \,/ \sqrt{n_{smp}}} \tag{3.23}$$

tends to a *Standard Normal Distribution* (i.e with $\mu = 0$ and $\sigma = 1$). When considering the co-occurrence of lexical items and features, the "sample" is the text environment (e.g. corpus) from which we are building feature vectors. Thus, there is only one sample, i.e. $n_{smp} = 1$. In this case, the Z-Score statistic is only applicable if the population itself follows a approximately normal distribution. For very large amounts of text it is possible to approximate the discrete distribution that is usually used for modelling the co-occurrence of lexical items and features, the *Binomial Distribution*, by a Normal Distribution, keeping the same average and standard deviation parameters.

The Z-Score statistic can be used as a feature weighting function, by taking its value when used for testing the validity of the H_0 hypothesis. If the occurrence of the lexical item l_{σ_i,ρ_i} and feature f_j are independent, then:

$$p_0(l_{\sigma_i,\rho_i}, f_j) = p_0(l_{\sigma_i,\rho_i}) \cdot p_0(f_j) \tag{3.24}$$

The expected value for the number of co-occurrences in the text environment for a population of size O:

$$\mu_{smp} = O \cdot p_0(l_{\sigma_i,\rho_i}, f_j) = e_0(l_{\sigma_i,\rho_i}, f_j) \tag{3.25}$$

Then, if the population follows a *Binomial Distribution*, whose parameter p_{bin} is assumed to be $p_0(l_{\sigma_i,\rho_i}, f_j)$, the variance of the population should be given by:

$$\begin{aligned}\sigma^2_{pop} = O \cdot p_0(l_{\sigma_i,\rho_i}, f_j) \cdot (1 - p_0(l_{\sigma_i,\rho_i}, f_j)) = \\ o_0(l_{\sigma_i,\rho_i}, f_j) \cdot (1 - o_0(l_{\sigma_i,\rho_i}, f_j)/O)\end{aligned} \tag{3.26}$$

Since $\mu_{smp} = o_0(l_{\sigma_i,\rho_i}, f_j)$ and $n = 1$:

$$Z = \frac{o(l_{\sigma_i,\rho_i}, f_j) - o_0(l_{\sigma_i,\rho_i}, f_j)}{\sqrt{o_0(l_{\sigma_i,\rho_i}, f_j) \cdot (1 - o_0(l_{\sigma_i,\rho_i}, f_j)/O)}} \tag{3.27}$$

The higher the value of the Z-Score, the less likely is the Null Hypothesis to be valid.

The Equation 3.27 is the most frequent formulation found in literature for the Z-Score statistic (e.g. Schone and Jurafsky [SJ01] or Pearce [Pea02]). Evert presents a simplification for this formulation based on the fact that the total number of observations *for all lexical items and features*, O, is much larger than the number of observations for any individual combination of lexical item and feature, $o_0(l_{\sigma_i,\rho_i}, f_j)$ [Eve05]. Thus, if $O \gg o_0(l_{\sigma_i,\rho_i}, f_j)$:

$$\sigma^2_{pop} = o_0(l_{\sigma_i,\rho_i}, f_j) \cdot (1 - o_0(l_{\sigma_i,\rho_i}, f_j)/O) \simeq o_0(l_{\sigma_i,\rho_i}, f_j) \tag{3.28}$$

and Equation 3.27 can be simplified to:

$$Z = \frac{o(l_{\sigma_i,\rho_i}, f_j) - o_0(l_{\sigma_i,\rho_i}, f_j)}{\sqrt{o_0(l_{\sigma_i,\rho_i}, f_j)}} \tag{3.29}$$

Finally, we can formulate Equation 3.29 using the simplified notation to express the values of the Contingency Table:

$$Z = \frac{a - \frac{(a+b)(a+c)}{O}}{\sqrt{\frac{(a+b)(a+c)}{O}}} = \frac{a \cdot O - (a+b)(a+c)}{\sqrt{O \cdot (a+b)(a+c)}} \tag{3.30}$$

Student's t-test

Student's t-test is widely used in statistics for comparing two samples drawn from the same population. It can be used to test if the *average* of two such samples is significantly different or not. Contrary to the Z-score, the t-test does not assume that the standard deviation of the population is known, and thus it assumes that its value should also be *estimated* from the samples. The t-test is, therefore, a significance test that makes less assumptions about the population at stake than the Z-Score. The t *statistic* is given by:

$$t = \frac{\mu_{smp} - \mu_{pop}}{S_{pop}/\sqrt{n_{smp}}} \tag{3.31}$$

with:

- μ_{smp} being the average of the sample
- μ_{pop} being the average of the population, which is assumed to be the value expected under the H_0
- n_{smp} being number of samples taken
- S_{pop} being an estimate of the standard deviation of the population (σ_{pop}) computed from the n_{smp} samples taken.

The t-test follow a *Students-t Distribution* with $v = n_{smp} - 1$ degrees of freedom. As mentioned by Evert ([Eve05], pag. 82), in theory, the t-test is not applicable to frequency data taken from a corpus (i.e. one sample) since this test was developed to compare $n_{smp} > 1$ independent samples from the same population. However, the t-test can be seen as an heuristic variant of the Z-Score. Assuming that the parameter of the Binomial Distribution that describes $o(l_{\sigma_i,\rho_i}, f_j)$ (i.e. the $p(l_{\sigma_i,\rho_i}, f_j)$ parameter) can be estimated from the values observed using a *maximum likelihood estimator*, we have $p(l_{\sigma_i,\rho_i}, f_j) = o(l_{\sigma_i,\rho_i}, f_j)/O$. Then, we obtain (for a Binomial Distribution):

$$S_{pop} = O \cdot p(l_{\sigma_i,\rho_i}, f_j) \cdot (1 - p(l_{\sigma_i,\rho_i}, f_j)) = o(l_{\sigma_i,\rho_i}, f_j) \cdot (1 - o(l_{\sigma_i,\rho_i}, f_j)/O) \tag{3.32}$$

and, by making $n_{smp} = 1$ in Equation 3.31, we can derive:

$$t = \frac{o(l_{\sigma_i,\rho_i}, f_j) - o_0(l_{\sigma_i,\rho_i}, f_j)}{\sqrt{o(l_{\sigma_i,\rho_i}, f_j) \cdot (1 - o(l_{\sigma_i,\rho_i}, f_j)/O)}} \simeq \frac{o(l_{\sigma_i,\rho_i}, f_j) - o_0(l_{\sigma_i,\rho_i}, f_j)}{\sqrt{o(l_{\sigma_i,\rho_i}, f_j)}} \tag{3.33}$$

It is possible to find in literature both the *exact* and *approximate* versions of the t-test shown before (Equation 3.33). By using the simplified notation to express the values of the Contingency Table, we can obtain the following equation for the approximated value of the t-test:

$$t = \frac{a - \frac{(a+b)(a+c)}{O}}{\sqrt{a}} = \sqrt{a} - \frac{(a+b)(a+c)}{O\sqrt{a}} \tag{3.34}$$

The Log-Likelihood Ratio Test (G^2)

As mentioned by Dunning [Dun93], the χ^2 and $Z - Score$ tests are based on the assumption that the random variables at stake follow a Binomial Distribution, which can be safely approximated by a Normal Distribution. In such case, it is possible to assume that the random variable follows the Normality Assumption, a condition that is required for ensuring the validity of the χ^2 and the $Z - Score$ tests.

However, using a Normal Distribution to approximate the values of a Binomial Distribution is only valid when the value of the variance of the underlying Binomial Distributions are higher than a given threshold (usually 5). Thus, for "rare" events (i.e. those whose frequency and variance are relatively low) the Normality Assumption does not usually hold, meaning that in practice the results of the χ^2 test and $Z - Score$ tests are likely to be invalid.

For that reason, Dunning proposes the *Log-Likelihood Ratio* test, which does not depend on the Normality Assumption. The goal of Log-Likelihood Ratio test, or G^2 test, is to measure how "surprising" is a given event, even if it occurs only once. Whenever the Normality Assumption is in fact valid, the G^2 test tends to the χ^2 test.

Likelihood Function and Likelihood Ratio The Log-Likelihood Ratio test has a complex formulation. Let us assume that the occurrences of a certain event follow a given parametric statistical model $\mathcal{M}$, i.e. they follow a probability function with one or several model parameters, $m_1, ..., m_n$. The probability of a given set of occurrences described as $k_1, ..., k_m$, taking into account the parametric model $\mathcal{M}$ is given by the *Likelihood Function* of the model:

$$H_{\mathcal{M}}(m_1, ...m_n; k_1, ...k_m) \tag{3.35}$$

Defining ω as one possible instantiation of parameters within the *parameter space* Ω, and κ as a set of specific occurrences in the *occurrence space* K, we can represent the Likelihood Function in a more compact format as:

$$H_{\mathcal{M}}(\omega; \kappa) \tag{3.36}$$

The Likelihood Ratio λ is the relation between the maximum value of the Likelihood Function within a given parameter subspace Ω_0 defined by a hypothesis under test, H_0, and the maximum possible value of the Likelihood Function within the *totality* of the parameter space Ω:

$$\lambda = \frac{max_{\Omega_0} H_{\mathcal{M}}(\omega; \kappa)}{max_{\Omega} H_{\mathcal{M}}(\omega; \kappa)} \tag{3.37}$$

One important property of the Likelihood Ratio, λ, is that the statistic:

$$-2 \cdot \log(\lambda) \tag{3.38}$$

converges asymptotically to the χ^2 distribution. The convergence is very fast, even for binomial or multinomial distribution. The practical consequence of such fast convergence is that it becomes possible to perform an efficient hypothesis test even when the sample is relatively small or the expected and observed values under test are low. In other words, the Normality Assumption is not a pre-requisite for the validity of this test.

Using the Likelihood Ratio to Measure Association The Likelihood Ratio can be used for measuring the association between the occurrence of a lexical item, l_{σ_i,ρ_i}, and a feature, f_j, (and thus it can be used for feature weighting), using a formulation different from the previous three tests. Under the Null Hypothesis, H_0, we have

$$p_0(l_{\sigma_i,\rho_i}) = p_0(l_{\sigma_i,\rho_i}|f_j) = p_0(l_{\sigma_i,\rho_i}|!f_j) \tag{3.39}$$

Also, let us assume that, generically, the number of occurrences of l_{σ_i,ρ_i}, o_i, on a given text environment (e.g. a corpus) follows a Binomial Distribution with parameter p_i

$$\mathcal{P}_i(o_i = k) = \binom{O}{k} p_i^k (1-p_i)^{O-k} \tag{3.40}$$

with O being the total number of observations made on such environment. We can now define two probability functions, one for the event of l_{σ_i,ρ_i} co-occurring with f_j and the other for the event of l_{σ_i,ρ_i} of *not* co-occurring with f_j:

$$\mathcal{P}_{(i,j)}(o_i = k_1|f_j) = \binom{O_1}{k_1} p_{(i,j)}^{k_1} (1-p_{(i,j)})^{O_1-k_1} \tag{3.41}$$

$$\mathcal{P}_{(i,!j)}(o_i = k_2|!f_j) = \binom{O_2}{k_2} p_{(i,!j)}^{k_2} (1-p_{(i,!j)})^{O_2-k_2} \tag{3.42}$$

These functions correspond to the observation of k_1 occurrences of l_{σ_i,ρ_i} along with f_j in O_1 observations, and k_2 occurrences of l_{σ_i,ρ_i} not co-occurring with f_j, in O_2 observations. The Likelihood function for two binomial probability functions is given by:

$$H(p_{(i,j)}, p_{(i,!j)}; k_1, O_1, k_2, O_2) = \binom{O_1}{k_1} p_{(i,j)}^{k_1} (1-p_{(i,j)})^{O_1-k_1} \cdot \binom{O_2}{k_2} p_{(i,!j)}^{k_2} (1-p_{(i,!j)})^{O_2-k_2} \tag{3.43}$$

The Null Hypothesis, H_0, assumes that the parameters $p_1 = p_2 = p_0$ are equal, meaning that the occurrence of l_{σ_i,ρ_i} is as probable with or without f_j. Thus the Likelihood Ratio can be formulated by:

$$\lambda = \frac{max_{p_0} H(p_0, p_0; k_1, O_1, k_2, O_2)}{max_{p_1,p_2} H(p_1, p_2; k_1, O_1, k_2, O_2)} \tag{3.44}$$

The maximum values for each of Likelihood Function are obtained with:

$$p_0 = \frac{k_1 + k_2}{O_1 + O_2} \tag{3.45}$$

and

$$p_1 = \frac{k_1}{O_1} \tag{3.46}$$

$$p_2 = \frac{k_2}{O_2} \tag{3.47}$$

which allows further simplification of Equation 3.44 to:

$$\lambda = \frac{L(p_0, k_1, O_1) \cdot L(p_0, k_2, O_2)}{L(p_1, k_1, O_1) \cdot L(p_2, k_2, O_2)} \tag{3.48}$$

with $L(p, k, n) = p^k(1-p)^{n-k}$. Finally, the *Log-Likelihood Ratio*, or G^2, can be computed by making $-2 \cdot log\lambda$:

$$G^2 = 2 \cdot [\log L(p_0, k_1, O_1) + \log L(p_0, k_2, O_2) - \log L(p_1, k_1, O_1) - \log L(p_2, k_2, O_2)] \tag{3.49}$$

Since this formulation is extremely complex and does not relate to the values held by the Contingency Table, several authors (e.g. [Eve05]) have proposed other less compact, yet more explicit, formulations. If we take into account the values observed (i.e $T_{cont}(l_{\sigma_i,\rho_i}, f_j)$) and the values expected under the Null Hypothesis (i.e. $T^0_{cont}(l_{\sigma_i,\rho_i}, f_j)$) we can rewrite Equation 3.49 as:

$$\begin{aligned} G^2 = 2 \cdot [o(l_{\sigma_i,\rho_i}, f_j) \cdot \log \frac{o(l_{\sigma_i,\rho_i}, f_j)}{o_0(l_{\sigma_i,\rho_i}, f_j)} + o(l_{\sigma_i,\rho_i}, !f_j) \cdot \log \frac{o(l_{\sigma_i,\rho_i}, !f_j)}{o_0(l_{\sigma_i,\rho_i}, !f_j)} \\ + o(!l_{\sigma_i,\rho_i}, f_j) \cdot \log \frac{o(!l_{\sigma_i,\rho_i}, f_j)}{o_0(!l_{\sigma_i,\rho_i}, f_j)} + o(!l_{\sigma_i,\rho_i}, !f_j) \cdot \log \frac{o(!l_{\sigma_i,\rho_i}, !f_j)}{o_0(!l_{\sigma_i,\rho_i}, !f_j)}] \end{aligned} \tag{3.50}$$

By applying the simplified notation, we obtain the following formulation:

$$\begin{aligned} G^2 = 2 \cdot [&a \cdot \log(a) + b \cdot \log(b) + c \cdot \log(c) + d \cdot \log(d) \\ &- (a+b) \cdot \log(a+b) - (a+c) \cdot \log(a+c) \\ &- (b+d) \cdot \log(b+d) - (c+d) \cdot \log(c+d) \\ &+ (a+b+c+d) \cdot \log(a+b+c+d)] \end{aligned} \tag{3.51}$$

3.4.3 Weighting Functions based on Information Theory

In this section, we mainly focus on association measures based on concepts borrowed from the Information Theory, such as *Entropy* or *Information Gain*. Basically, when applied to lexical items and corresponding feature, these measures try to quantify how much a given feature f_j is "typical" (or not) of the lexical item l_{σ_i,ρ_i}, by measuring how much information does the occurrence of one give about the occurrence of the other. The value of such measure is then used as the weight of the feature.

Pointwise Mutual Information

Pointwise Mutual Information[2] is one of the classical association measures that is used as a feature weighting function [CH90]. It relates the probability of co-occurrence of two elements with the probabilities of occurrence of each of them individually. Therefore, assuming such elements are the lexical item l_{σ_i,ρ_i} and feature f_j extracted from the context, the Pointwise Mutual Information is given by:

$$MI(l_{\sigma_i,\rho_i}, f_j) = \log_2 \frac{p(l_{\sigma_i,\rho_i}, f_j)}{p(l_{\sigma_i,\rho_i}) \cdot p(f_j)} \tag{3.52}$$

If the occurrence of l_{σ_i,ρ_i} is independent of the occurrence of f_j then

$$p(l_{\sigma_i,\rho_i}, f_j) = p(l_{\sigma_i,\rho_i}) \cdot p(f_j) \Rightarrow MI(l_{\sigma_i,\rho_i}, f_j) = \log_2(1) = 0 \tag{3.53}$$

On the other hand, if there is a significant dependence between l_{σ_i,ρ_i} and f_j:

$$p(l_{\sigma_i,\rho_i}, f_j) > p(l_{\sigma_i,\rho_i}) \cdot p(f_j) \Rightarrow MI(l_{\sigma_i,\rho_i}, f_j) > 0 \tag{3.54}$$

We can express Pointwise Mutual Information between l_{σ_i,ρ_i} and f_j in an alternative way that allows straight-forward computation using the values stored in the Contingency Table (see Equation 3.4):

$$MI(l_{\sigma_i,\rho_i}, f_j) = \log_2 \frac{a \cdot O}{(a+b) \cdot (a+c)} \tag{3.55}$$

Issues concerning Pointwise Mutual Information One of the main criticism to Pointwise Mutual Information is the fact that it tends to over-estimate the degree of association between elements, whenever the events are "rare", i.e. whenever the values of $p(l_{\sigma_i,\rho_i})$ or $p(f_j)$ are very low [Dun93]. In fact, if $p(l_{\sigma_i,\rho_i}) \simeq 0$ or $p(f_j) \simeq 0$ Pointwise Mutual Information will be asymptotically high, which tends to promote the association of rare occurrences, such as those with infrequent words, or even spelling mistakes and other idiosyncrasies, instead of truly significant co-occurrences[3].

Another criticism to Pointwise Mutual Information is its inability to produce satisfactory rankings by degree of association [CG91]. Furthermore, this measure does not make the best use of the b and c values of the Contingency Table, whose values are usually relatively high, and thus support better estimates than those supported by a (which usually is much lower than b or c).

[2]In computational linguistics and text-mining fields, Pointwise Mutual Information is often mentioned as "Mutual Information". We also follow that convention, and every reference to Mutual Information in this thesis should be understood as *Pointwise* Mutual Information

[3]To compensate for such effect Lin and Pantel propose a correction factor to the original equation [LP02].

Mutual Dependency

Mutual Dependency was proposed by Thanapoulos et al [TFK02] and is based on the observation that Mutual Information is mainly a measure of *independency* rather than a measure of *dependency*. A measure of *dependency* can be obtained by subtracting the value of *Auto-Information* to the value of Mutual Information. Auto-Information is one of the classic measures from the Information Theory, and is defined as:

$$AI(Z) = -\log(Z) \tag{3.56}$$

Mutual Dependence (MD) is defined as:

$$MD(l_{\sigma_i,\rho_i}, f_j) = MI(l_{\sigma_i,\rho_i}, f_j) - AI(l_{\sigma_i,\rho_i}, f_j) \tag{3.57}$$

By expanding each of the terms we can transform Equation 3.57 in:

$$\begin{aligned} MD(l_{\sigma_i,\rho_i}, f_j) &= \log_2\left(\frac{p(l_{\sigma_i,\rho_i}, f_j)}{p(l_{\sigma_i,\rho_i}) \cdot p(f_j)}\right) + \log_2(p(l_{\sigma_i,\rho_i}, f_j)) \\ &= \log_2\left(\frac{p(l_{\sigma_i,\rho_i}, f_j) \cdot p(l_{\sigma_i,\rho_i}, f_j)}{p(l_{\sigma_i,\rho_i}) \cdot p(f_j)}\right) \end{aligned} \tag{3.58}$$

Using the simplified notation to express Mutual Dependency as a function of the values of the Contingency Table:

$$MD(l_{\sigma_i,\rho_i}, f_j) = \log_2\left(\frac{a/O \cdot a/O}{(a+b)/O \cdot (a+c)/O}\right) = \log_2\left(\frac{a \cdot a}{(a+b)\cdot(a+c)}\right) \tag{3.59}$$

If used for ranking associations, Mutual Dependency leads to approximately the same results as those obtained when using Pointwise Mutual Information. In fact, Mutual Dependency is essentially a rescaling of Pointwise Mutual Information function.

3.4.4 tf-idf as a Feature Weighting Function

In this section we focus on the *term frequency-inverse document frequency* (*tf-idf*) function, which can used for feature weighting although it was not initially formulated with that purpose. The original purpose of tf-idf function was to weight the relevance of the *terms* inside a *document* in a document collection, but can be easily adapted for weighting features in generic vector descriptions. The value of tf-idf function for a term t_i in document d_j is given by:

$$\text{tf-idf}(i, j) = tf_{i,j} \cdot idf_i \tag{3.60}$$

with $tf_{i,j}$ being the *normalized* frequency of term t_i in document d_j ($n_{i,j}$ is the number of times that term t_i occurs in document d_j):

$$tf_{i,j} = \frac{n_{i,j}}{\sum_i n_{i,j}} \tag{3.61}$$

and idf_i being the *normalized* inverse document frequency, i.e. the inverse of the fraction of documents in the collection containing term t_i:

$$idf_i = \log\left(\frac{|D|}{|d : t_i \in d|}\right) \tag{3.62}$$

with $|D|$ being the total number of document in the collection, and $|d : t_i \in d|$ being the number of documents in the collection containing t_i.

We can formulate feature weighting as a process similar to *term weighting*. The goal is to measure the "relevance" of feature f_j (which is treated as the analogue of the *term*) within the vector description of lexical item l_{σ_i,ρ_i} (which is treated as the *document*). Thus, we define *relative feature frequency* as[4]:

$$ff_{j,i} = \frac{o(l_{\sigma_i,\rho_i}, f_j)}{o(l_{\sigma_i,\rho_i})} \tag{3.63}$$

Likewise, *inverse item frequency* is the fraction of lexical items with which feature f_j co-occurs:

$$iif_j = \log\left(\frac{|L|}{|l : o(l_{\sigma_i,\rho_i}, f_j) > 0|}\right) \tag{3.64}$$

with $|L|$ being the total of lexical items being weighted (i.e. for which features were extracted), and $|l : o(l_{\sigma_i,\rho_i}, f_j) > 0|$ being the number of lexical items for which feature f_j was extracted.

Following the tf-idf formula, the weight, w, to be assigned to feature f_j in the vector description of l_{σ_i,ρ_i} is given by:

$$w(l_{\sigma_i,\rho_i}, f_j) = ff_{j,i} \cdot iif_j \tag{3.65}$$

3.5 Vector Distance Measures

One of the essential assumptions for using the Vector Space Model relies on the ability to measure "distances" between vectors. One can find in literature a great number of vector functions for measuring the "distance" between two vectors, or for measuring other related concepts, such as *vector divergence* or *vector dissimilarity*. Other functions measure the opposite concept, i.e. the *vector proximity* or *vector similarity* between vectors. Depending on the authors and subject field of the publication (i.e. computational linguistics, data-mining, information theory, etc.) we encounter different names for these functions: *distance measures*, *vector divergence measures*, *vector dissimilarity measure*, *vector similarity measures* or *vector proximity measures*. In practice, it is generally possible to trivially convert

[4]In order to follow previous notation, lexical items are generically denoted by l_{σ_i,ρ_i} while features are generically mentioned by f_j. Because of this, the indexes i and j used in Equation 3.63 are changed in relation to Equation 3.61

measures of *distance* (or *dissimilarity*) into measures of *proximity* (or *similarity*). Therefore, in the next descriptions we do not treat such functions separately according to that criterion, but we explicitly state the type of each function at stake (i.e. *distance* vs. *proximity*). Additional information and comparisons between some of these measures can be found in works by Lee [Lee99], Dagan et al. [DLP99], Weeds et al. [WWM04], Strehl et al. [SGM00] and also St-Jacques and Barrière [SJB06].

There are several vector distance functions specially designed for *Real-valued* Vector Spaces and others for *Binary* Vector Spaces. Most measures for Real-Valued Vector Spaces can also be used in Binary Vector Spaces. In the next sections, we present an overview of several vector distance (or proximity) measures.

3.5.1 Distance Measures over Real-Valued Vector Spaces

We consider two different approaches for measuring distances in Real-Valued Vector Spaces: *Geometric Measures* and *Probabilistic Measures.* The following notation conventions are used:

- $\overline{x}$ is the vector representation of item x
- $\overline{x}(k)$ is the value of the element at component k of vector $\overline{x}$
- $||\overline{x}||$ is the *Euclidean Norm* of vector $\overline{x}$
- $\overline{x} \cdot \overline{y}$ is the *dot product* of vectors $\overline{x}$ and $\overline{y}$
- $abs(x)$ is the absolute value of the scalar x
- $|\mathcal{V}|$ is the dimensionality of the Vector Space $\mathcal{V}$

Geometric Measures

These measures are based on traditional concepts about space in Euclidean Spaces:

L1 Distance The *L1 Distance*, also known as *Block Distance*, *Manhattan Distance* or *Variational Distance* is given by:

$$D_{L1}(\overline{x}, \overline{y}) = \sum_{k=0}^{|\mathcal{V}|} abs(\overline{x}(k) - \overline{y}(k)) \tag{3.66}$$

$$0 \leq D_{L1}(\overline{x}, \overline{y}) < +\infty \tag{3.67}$$

L2 Distance The *L2 Distance*, or *Euclidean Distance*, is one of the most elementary distance measures, and it is simply the norm of the difference between the two vectors:

$$D_{L2}(\overline{x}, \overline{y}) = ||\overline{x} - \overline{y}|| = \sqrt{\sum_{k=0}^{|\mathcal{V}|} (\overline{x}(k) - \overline{y}(k))^2} \tag{3.68}$$

$$0 \leq D_{L2}(\overline{x}, \overline{y}) < +\infty \tag{3.69}$$

Cosine Measure The *Cosine Measure* [SM86] is a very widespread measure of proximity. For two vectors, this measure gives the value of the cosine of the angle between them. The Cosine Measure is given by:

$$cos(\overline{x}, \overline{y}) = \frac{\overline{x} \cdot \overline{y}}{||\overline{x}|| \cdot ||\overline{y}||} = \frac{\sum_{k=0}^{|\mathcal{V}|} (\overline{x}(k) \cdot \overline{y}(k))}{\sqrt{\sum_{k=0}^{|\mathcal{V}|} \overline{x}(k)^2} \cdot \sqrt{\sum_{k=0}^{|\mathcal{V}|} \overline{y}(k)^2}} \tag{3.70}$$

The Cosine Measure is a very convenient measure because contrary to the Euclidean Distance that is unbounded, it produces values in the interval [-1,1]. If two vectors are parallel on the hyperspace, then the value of the cosine is either 1, if they have the same direction, or -1 if they are in opposite directions. A value of 1 means that the two vectors have the same features in the same proportions, i.e. they differ only by a *positive* scaling operation. A value of -1, means that vectors have the same features but with symmetric values. A value of 0 means that the vectors are totally perpendicular, i.e. they have no common features at all.

Extended Jaccard Similarity Although the "original" Jaccard Similarity measure (which is presented in section 3.5.2) was initially conceived as a measure for binary attributes, an extended version for real-valued attributes has been proposed by Strehl and Gosh [SG00]. The *Extended Jaccard Similarity* is described by the following formula:

$$J_{ext}(\overline{x}, \overline{y}) = \frac{\overline{x} \cdot \overline{y}}{||\overline{x}|| \cdot ||\overline{y}|| - \overline{x} \cdot \overline{y}} \tag{3.71}$$

Probabilistic Measures

Another way of comparing vectors in Real-Value Vector Spaces consists in formulating such task as a problem of comparing two *probability distributions*. One of the first descriptions of such an approach for word clustering purposes is given by Pereira et al. [PTL93]. In this work, the authors tried to classify a set of nouns $n_i \in \mathcal{N}$ according to their distribution as direct objects of a set verbs $v_j \in \mathcal{V}$. A parser was used to compile the frequency of co-occurrence of pairs (v_j, n_i), $f_{v_j n_i}$. Then, for each $n_i \in \mathcal{N}$, its empirical verb distribution is given by:

$$p_n(v) = \frac{f_{vn}}{\sum_v f_{vn}} \tag{3.72}$$

The comparison between distribution p_n and distribution q_n (i.e. verb distributions for two nouns) is done using the Kullback-Leibler (KL) distance:

$$D_{KL}(p||q) = \sum_v p_n(v) \log \frac{p_n(v)}{q_n(v)} \tag{3.73}$$

There are some advantages in using such types of measures. The main advantage is that they all naturally operate over Real-valued Vector Spaces, eliminating the need for a *threshold operation* over the Vector Space, and thus avoiding introducing undesirable non-linearity in the Space.

Another advantage is that these measures, contrary to the Cosine Measure or the Euclidean Distance, do not assume an *Euclidean Space*. The assumption that the Vector Spaces we are considering are in fact Euclidean is not well justified, although many works so far have accepted it as granted (refer to [PTL93] for a list of other important advantages).

However, one problem with Kullback-Leibler Distance is that it is not defined when $q_n(v) = 0$ but $p_n(v) > 0$. This problem can be alleviated by using smoothing techniques for zero frequencies, but it involves additional complexity. Description of such techniques is out of the scope of this document.

Next, we describe some other measures that are related to the Kullback-Leibler Distance.

Information Radius or Jensen-Shannon Distance The *Information Radius or Jensen-Shannon Distance* [Lin91] is a dissimilarity measure that is directly related to KL by the following formula:

$$D_{JS}(p||q) = \frac{1}{2} D_{KL}(p||m) + \frac{1}{2} D_{KL}(q||m) \tag{3.74}$$

where M is the average of the two distributions:

$$m = \frac{1}{2}(p+q) \tag{3.75}$$

Contrary to the KL Distance, the Jensen-Shannon Distance is a *symmetric* measure.

Skew Divergence The *Skew Divergence* is yet another formula derived from the Kullback-Leibler Distance [Lee01]. It is defined as:

$$s_\alpha(p,q) = D_{KL}(p||\alpha q + (1-\alpha)r) \tag{3.76}$$

For $\alpha = 1$ it becomes the KL Distance.

Confusion Probability The *Confusion Probability* is a *similarity* measure that estimates the probability of lexical item l_1 being "confused" for lexical item l_2, based on feature information extracted from their contexts [ES92]. A large value of confusion probability indicates that the two lexical items l_1 and l_2 appear mostly in similar feature contexts so that they can be "easily confused" (or substituted). The confusion probability can be expressed by (f_i is one of the possible features):

$$P_{Conf}(l_1, l_2) = \sum_i P(l_1|f_i)P(f_i|l_2) \tag{3.77}$$

Tau Metric Kendall's τ Metric [Ken38] is a measure of *association* between two random variables that can be used to calculate distributional similarity. It is used to compare different rankings (orderings) and to measure their agreement / correlation. However, it has been applied to measure distributional similarity of features, under the following assumption: if the probability functions p and q that describe the occurrence of the features of two lexical are similar, then the *rankings* of the features of each lexical item, based on the frequencies observed in a corpus, should be the same, independently of the actual frequencies found. The τ metric is often given by:

$$\tau(p, q) = \frac{\sum_i^{|\mathcal{V}|} \sum_j^{|\mathcal{V}|} sign\left[(p(f_i) - p(f_j)) \cdot (q(f_i) - q(f_j))\right]}{2 \cdot \binom{|\mathcal{V}|}{2}} \tag{3.78}$$

This formula means that for each pair of features f_i and f_j, we check if their relative positions in terms of ranking (i.e. lower or greater probability) are the same for both distributions, p and q. If so we add one, otherwise we subtract 1. The denominator of the equation is the number of all possible pairings times 2 because the summation repeats the comparison in both ways. An easier formulation is given by:

$$\tau = \frac{n_c - n_d}{|\mathcal{V}| \cdot (|\mathcal{V}| - 1)} \tag{3.79}$$

where n_c number of concordant pairs (ranked consistently) and n_d is the number of discordant pairs (i.e. ranked differently).

Therefore, for exactly similar rankings, we will have $\tau(p, q) = 1$, indicating total similarity between distributions. For totally opposite rankings, we will have $\tau(p, q) = -1$, indicating that the distributions are totally dissimilar.

3.5.2 Distance Measures over Binary Vector Spaces

Next, we present some of the most frequent distance vector metrics used in binary vector spaces. The following notation conventions are used:

- $||\overline{x}||$ is the *binary norm* of vector $\overline{x}$, that is the number of "1" elements in vector $\overline{x}$

- $\overline{x} \bigcap \overline{y}$ is the *intersection* between vector $\overline{x}$ and vector $\overline{y}$, that is the number of common "1" elements in vector $\overline{x}$ and vector $\overline{y}$.
- $\overline{x} \bigcup \overline{y}$ is the *union* between vector $\overline{x}$ and vector $\overline{y}$

Matching Coefficient The *Matching Coefficient* is the most elementary measure of similarity, but it has several important drawbacks: (i) it does not take into account the length of the vectors, (ii) it does not consider the total number of non-zero elements, and (iii) it is not normalized. It is given by:

$$M(\overline{x}, \overline{y}) = ||\overline{x} \bigcap \overline{y}|| \tag{3.80}$$

The Matching Coefficient is mentioned here mainly for reference purposes and for a better understanding of the following measures.

Dice Coefficient The *Dice Coefficient* is a very popular measure of similarity. It is an improvement of the *Matching Coefficient* because it provides a normalized value in [0,1] range. It is given by:

$$D(\overline{x}, \overline{y}) = \frac{2 \cdot ||\overline{x} \bigcap \overline{y}||}{||\overline{x}|| + ||\overline{y}||} \tag{3.81}$$

Overlap Coefficient The *Overlap Coefficient* is variation of the *Matching Coefficient* that takes into account the maximum possible value of the intersection, which is limited to $min(||\overline{x}||, ||\overline{y}||)$. The *Overlap Coefficient* is defined as:

$$O(\overline{x}, \overline{y}) = \frac{||\overline{x} \bigcap \overline{y}||}{min(||\overline{x}||, ||\overline{y}||)} \tag{3.82}$$

The *Overlap Coefficient* has a value of 1 whenever one of the binary vectors is completely included by another, that is $\overline{x} \subseteq \overline{y}$ or $\overline{y} \subseteq \overline{x}$.

Jaccard Coefficient The *Jaccard Coefficient* or *Jaccard Index*, J, is the ratio between the module of the intersection of two vectors and the module of their union:

$$J(\overline{x}, \overline{y}) = \frac{||\overline{x} \bigcap \overline{y}||}{||\overline{x} \bigcup \overline{y}||} \tag{3.83}$$

The Jaccard Coefficient ranges from 0, meaning total dissimilarity between the binary attributes, and 1, in case all attributes match (except when they are all 0). It penalizes a small number of shared entries more than the Dice Coefficient because the union of the vectors is used instead of the average of their norms.

The *Jaccard Distance* is given by:

$$J_\delta(\overline{x}, \overline{y}) = 1 - J(\overline{x}, \overline{y}) = \frac{||\overline{x} \bigcup \overline{y}|| - ||\overline{x} \bigcap \overline{y}||}{||\overline{x} \bigcup \overline{y}||} \tag{3.84}$$

Estimating Proximity of Binary Vectors by *Min-Hashing* Instead of directly comparing two vectors, there are ways of estimating the distance / proximity using statistical approaches that allow significant computational savings. These approaches are particularly useful when the number of vectors to be compared is very high and there is only the need to obtain an approximate list of the nearest-neighbours (i.e closest vectors). Min-Hashing (Min-wise Independent Permutation Hashing) is a Locality Sensitive Hashing scheme that can be used for such purpose [IM98].

The key component of Min-Hasing is a *hash function*, $\mathcal{H}(\overline{x})$, that returns the index of the first *non-null component* of a given feature vector $\overline{x}$ from the space. It is possible to show – see Broder [Bro97] and Indyk and Motwani [IM98] – that, when random permutations on the positions of the components of the vector space are performed, the probability of two vectors hashing to the same value, i.e. $\mathcal{H}(\overline{x}) = \mathcal{H}(\overline{y})$, is equal to their Jaccard coefficient $J(\overline{x}, \overline{x})$. Conversely, the Jaccard Coefficient of two vectors can be *estimated* by repeating the random permutation process and checking the *fraction* of times in which the values of the corresponding hashes *collide*. The number of repetitions can be set according to the desired precision of such estimate and to a pre-defined limit on the computational effort.

Since only positions of components are taken into account, and not their *values*, Min-Hashing is especially suited for Binary Vector Spaces.

3.6 Conclusion

In this chapter we summarized the basic concepts regarding the use of the Vector Space Model as the fundamental mathematical tool for grounding the computation of similarity. We provided a detailed description and explanation of the main parameters of the VSM namely *feature contexts*, *feature weighting functions* and *vector distance measures*.

We described several possibilities for defining features contexts. The choice of the appropriate feature context is one of the main issues addressed from Part II to Part IV. Despite the fact that the information regarding *feature weighting functions* and *vector distance measures* presented here does not constitute an original contribution, our main achievement in this chapter is to have presented a comprehensive overview of relevant information spread throughout a vast body of literature.

Chapter 4

Evaluation

In this chapter, we address several issues regarding the evaluation of procedures for computing similarity between lexical items. We begin by differentiating *direct* and *indirect* evaluation strategies. Then, for direct evaluation strategies, we discuss the most common evaluation scenarios and, for each, present performance measures (essentially those that are used in the remainder of the thesis). We proceed by describing resources that can be used as gold-standard references in evaluation procedures. We divide these resources in three different groups, depending on the type of lexical items for which they provide gold-standard information: common lexicon, named-entities and Web 2.0 tags. Finally, we present a formal description of indirect evaluation strategies, and we discuss possible performance criteria that can be applied in such evaluation scenarios.

4.1 Direct vs. Indirect Strategies

There are two main evaluation strategies: *direct* and *indirect* evaluation. Direct evaluation strategies consist in *directly* comparing the output of a given procedure (in our case one for computing a certain notion of similarity between lexical items) against a *ground-truth reference* or *gold-standard resource* containing explicit information about the correct output(s) expected. On the other hand, indirect evaluation presupposes that the procedure being evaluated is part of a larger processing chain whose result is the one that will be (directly) evaluated. Thus, when indirect evaluation strategies are used, the performance of the procedure under evaluation is only assessed through the quality of the results to which it contributes, which themselves have to be evaluated appropriately.

Direct evaluation strategies are particularly suited for assessing the performance of procedures related with *content* or *type* similarity. For example, for evaluating the performance of a synonymy finding procedure, one can directly compare the results obtained with the information contained in a WordNet-like resource. In this case, the challenge consists in obtaining an appropriate gold-standard resource, which may not exist or may not be easily (or freely) available. Whenever available, this type of resources might still have significant recall gaps that impact negatively in the results of the evaluation. Alternatively, different sources of ground-truth information may be used, such as, for example, *standard language tests* (e.g. TOEFL tests [Tur01]), or *psycholinguistic evidence* indicating how people "perceive" the degree of similarity between words.

Procedures for computing *functionally similarity* fit almost perfectly in *indirect evaluation* strategies. For example, a procedure for finding functionally similar keywords for query expansion can be evaluated in a traditional IR setting. A better procedure should provide better functional equivalents that should lead to better retrieval results, which can be measured. However, indirect evaluation strategies have their own set of challenges. First, they usually require a more complex set-up, comprising both the similarity computation procedure (under indirect evaluation) and the system which depends on the results of such procedure. Second, the evaluation of the external system may itself be complex and require gold-standard resources that might also be hard to obtain. Finally, and more important, it might not always be possible to establish a *clear and direct correlation* between a change in the performance of the external system and the performance of the procedure being evaluated. There simply may be too many factors that are difficult to isolate or control. Thus, indirect evaluations usually involve a larger number of performance measures to allow tracking multiple dimensions of the system.

One can generically say that direct evaluation is preferable since it provides a more straightforward way of comparing the performance of a function to the gold-standard without interferences from other factors. While this is indisputable, one should also not forget that gold-standards are usually created manually following a given set of (linguistic) assumptions, which imprints them a certain semantic bias (see [VPF91]). Depending on whether the same bias is followed (or not) by the procedure being evaluated, direct evaluation may overestimate (or underestimate) the true performance of the procedure. Direct evaluation over multiple gold-standard resources, when available, may help to alleviate this type of problems.

On the other hand, when there is a text processing goal at stake, the "quality" of the specific similarity computation function is not relevant *per se*, because only the final result of a longer processing chain is what really matters. In that case, the performance values obtained when using direct evaluation may not be significant in comparison with those obtained through indirect evaluation. Thus, it is important to consider both strategies, since different scenarios require different evaluation needs.

4.2 Direct Evaluation Strategies

Given a procedure to compute content or type similarity between lexical items, there are essentially three possible evaluation scenarios depending on the output of such procedure:

1. **Pairings**: The simplest situation is when the procedure produces *pairing information* taking into account a certain notion of similarity, as, for example, a pair of *synonyms* or *co-hyponyms*. In this situation, evaluation focus on determining whether such pairings are true or not, and whether all correct pairings have been found.

2. **Ranked Lists**: In other cases, the procedure outputs a ranked list of items according to their similarity to a given input item, such as for example, a list of synonyms with a grade indicating how similar the items in the list are to a reference word. Evaluation consists in measuring not only the *correctness* and *completeness* of the items in the ranked list, but also measuring how *accurate* is the ranking.

3. **Clusters**: In other situations, the procedure under evaluation generates clusters of lexical items (e.g. synsets) based on some similarity criteria. In such cases, evaluation consists in verifying whether these clusters are valid based on a number of quality criteria (e.g. "purity").

In the next sections we formalize these three scenarios and present possible evaluation measures.

4.2.1 Evaluating Pairings

Let π_i be a list of pairings (possibly only one) that was returned by a procedure that finds items to be matched with lexical item l_i according to a given notion of similarity. Let γ_i be the *gold-standard set* containing a list of correct pairings (ideally all) for lexical item l_i. We can compute Precision, p, and Recall r figures for the results of the pairing procedure for items l_i as:

$$p(\pi_i, \gamma_i) = \frac{|\pi_i \cap \gamma_i|}{|\pi_i|} \tag{4.1}$$

$$r(\pi_i, \gamma_i) = \frac{|\pi_i \cap \gamma_i|}{|\gamma_i|} \tag{4.2}$$

with $|\pi_i|$ and $|\gamma_i|$ being the number of pairing in π_i and γ_i respectively, and $|\pi_i \cap \gamma_i|$ being the number of correct pairs found in π_i.

If we now consider n lexical items and we wish to evaluate the list of pairings produced for them, $\Pi = [\pi_1, \pi_2 ... \pi_n]$, against the corresponding gold-standard pairing sets, $\Gamma = [\gamma_1, \gamma_2 ... \gamma_n]$ we can compute *global* precision and recall measures by *micro-averaging*:

$$P_{micro}(\Pi, \Gamma) = \frac{\sum_i^n |\pi_i \cap \gamma_i|}{\sum_i^n |\pi_i|} \tag{4.3}$$

$$R_{micro}(\Pi, \Gamma) = \frac{\sum_i^n |\pi_i \cap \gamma_i|}{\sum_i^n |\gamma_i|} \tag{4.4}$$

Alternatively, we can compute global precision and recall figures by *macro-averaging*:

$$P_{macro}(\Pi, \Gamma) = \frac{1}{n} \cdot \sum_i^n \frac{|\pi_i \cap \gamma_i|}{|\pi_i|} \tag{4.5}$$

$$R_{macro}(\Pi, \Gamma) = \frac{1}{n} \cdot \sum_i^n \frac{|\pi_i \cap \gamma_i|}{|\gamma_i|} \tag{4.6}$$

4.2.2 Evaluating Ranked Lists

Let λ_i be a list of lexical items returned by a procedure that identifies items similar to a given input lexical item l_i:

$$S(l_i) \rightarrow \lambda_i \tag{4.7}$$

Let element in λ_i, s_{ij}, be ranked by degree of similarity to l_i, s_{ij}:

$$\lambda_i = [(l_{i1}, s_{i1}), (l_{i2}, s_{i2}), \ ... \ (l_{im}, s_{im})] \tag{4.8}$$

with:

$$s_{ij} \geq s_{ik} \ \ \forall \ j < k \tag{4.9}$$

Let γ_i be the *gold-standard set* for l_i, i.e. the set of lexical items actually known to be similar to l_i (γ_i does not contain any information about the degree of similarity between its elements and l_i, so there is no *ranking* in γ_i):

$$\gamma_i = \left[l_{i1}^{\gamma}, l_{i2}^{\gamma}, ... l_{ig}^{\gamma} \right] \tag{4.10}$$

Usually, the number of elements in λ_i is much larger than the number of elements in the gold-standard:

$$|\lambda_i| \gg |\gamma_i| \tag{4.11}$$

The scenario we have just describe is typical of a traditional *information retrieval* setting, in which we wish to evaluate the performance of the retrieval system that generated a ranked (and long) list of documents in response to a query. Therefore, we can adapt IR performance measures to evaluate λ_i given γ_i.

More specifically, we can compute three precision figures. The first is *Precision at Rank 1*, $P_{@1}(\lambda_i, \gamma_i)$, which consists in checking whether the top ranked element of λ_i is part of the gold-standard γ_i or not:

$$P_{@1}(\lambda_i, \gamma_i) = \begin{cases} 1 & \text{if } l_{i1} \in \gamma_i \\ 0 & \text{if } l_{i1} \notin \gamma_i \end{cases} \tag{4.12}$$

We can extend the previous measure to compute a more comprehensive precision figure that takes into account other elements than just the top ranked one. Let λ_i^r be a list resulting from the truncation of λ_i to the top r elements (i.e. containing only elements l_{i1} to l_{ir}). *Precision at Rank* r, $P_{@r}$, is given by:

$$P_{@r}(\lambda_i, \gamma_i) = \frac{|\lambda_i^r \cap \gamma_i|}{r} \tag{4.13}$$

Since there are $|\gamma_i|$ items in the gold-standard, and usually $|\gamma_i| \ll |\lambda_i|$, one particularly useful point at which precision can be computed is at rank $|\gamma_i|$:

$$P_{@|\gamma_i|}(\lambda_i, \gamma_i) = \frac{|\lambda_i^{|\gamma_i|} \cap \gamma_i|}{|\gamma_i|} \tag{4.14}$$

$P_{@|\gamma_i|}$ maximizes the use of the gold-standard information available. Finally, *Average Precision*, $P_{avg}(\lambda_i, \gamma_i)$, provides a global view of the precision by combining the values of the precision at various ranks:

$$P_{avg}(\lambda_i, \gamma_i) = \frac{\sum_{r=1}^{R} P_{@r}(\lambda_i, \gamma_i) \cdot rl_{@}(\lambda_i, \gamma_i, r)}{|\gamma_i|} \tag{4.15}$$

with R being the number of elements in λ_i to be considered ($R \leq |\lambda_i|$) and $rl_{@}(\lambda_i, \gamma_i, r)$ being a binary function indicating if the element of λ_i at rank r is an element of γ_i (1) or not (0):

$$rl_{@}(\lambda_i, \gamma_i, r) = \begin{cases} 1 & \text{if } l_{ir} \in \gamma_i \\ 0 & \text{if } l_{ir} \notin \gamma_i \end{cases} \tag{4.16}$$

If we now consider the scenario where there are n ranked lists to be evaluated, $\Lambda = [\lambda_1, \lambda_2 ... \lambda_n]$, against the same number of gold-standard sets, $\Gamma = [\gamma_1, \gamma_2 ... \gamma_n]$, we can compute *global performance* measures by averaging the values obtained with Equations 4.12, 4.14 and 4.15 for the n corresponding lexical items, i.e. l_1, l_2 ... l_n. We can thus define three global precision figures:

1. *Average Precision at Rank 1*, $P_{@1}^{avg}(\Lambda, \Gamma)$:

$$P_{@1}^{avg}(\Lambda, \Gamma) = \frac{\sum_i^n P_{@1}(\lambda_i, \gamma_i)}{n} \tag{4.17}$$

2. *Average Precision at Rank* $|\gamma|$, $P^{avg}_{@|\gamma|}(\Lambda, \Gamma)$:

$$P^{avg}_{@|\gamma|}(\Lambda, \Gamma) = \frac{\sum_i^n P_{@|\gamma_i|}(\lambda_i, \gamma_i)}{n} \tag{4.18}$$

3. *Mean Average Precision*, MAP:

$$MAP(\Lambda, \Gamma) = \frac{\sum_i^n P_{avg}(\lambda_i, \gamma_i)}{n} \tag{4.19}$$

The previous measures can only be computed for lexical items for which there is gold-standard information. However, usually the overlap between the set of lexical items for which ranked lists where generated and those for which there is gold-standard information is only partial. If L is the list of lexical items for which ranked lists were generated, and G is the list of lexical items for which there is gold-standard information, the measures previously presented can only be applied to a sub-set of L, L^{eval}, such that:

$$L^{eval} = L \cap G \tag{4.20}$$

Taking into account the number of lexical items that can be evaluated, we can thus propose an additional measure of performance, *coverage* ($\mathcal{C}$), which quantifies the fraction of entries in gold-standard covered by the automatic procedure:

$$\mathcal{C} = \frac{|L \cap G|}{|G|} \tag{4.21}$$

4.2.3 Evaluating Clusters

Evaluating clusters is a complex problem. When no gold-standard clusters are available, the quality of clusters can only be assessed based on *internal criteria*, such as *intra-cluster similarity*, which should be high, and *inter-cluster similarity*, which should be low. However, these criteria do not necessarily imply that the clusters obtained are appropriate for a given practical application [MRS08]. In our case, internal criteria might not reflect the performance of the similarity computation procedure that leads to the generation of the clusters.

When gold-standard clusters are available, one can perform evaluation based on *external criteria* by comparing clustering results with the existing gold-standard. Several measures have been proposed for measuring how "close" test clusters are to reference (gold-standard) clusters.

Simpler measures are based on frequency counts. For example, *Pair Measures* [HBV01, Mei07] evaluate each pair of items taking into account whether they belong (i) to the same test cluster and the same gold cluster; (ii) to different test clusters and different gold clusters; (iii) to the same test cluster but different gold clusters, or (iv) to different test clusters but the same gold cluster. Several clustering quality evaluation measures can be computed based on these four contingency values (e.g. Jaccard Coefficient or Rand Coefficient). However, these measures are not invariant under scaling, i.e., they are sensitive to the number of items being evaluated.

Entropy-based measures, on the other hand, are invariant under scaling, so many authors have proposed entropy-based options. For example, *Information Variation* (IV) [Mei07] measures how much information we gain or lose if we remap items from test clusters to gold clusters. If the two sets of clusters are exactly the same, IV is zero, otherwise, IV is a positive value. IV has several interesting mathematical properties including the fact that it is a metric, and it has provable bounds.

Another alternative for evaluating clustering results is the *B-Cubed algorithm*, which was initially proposed for evaluating results of co-reference resolutions algorithms. The B-Cubed algorithm differs from other methods because it computes precision and recall figures *for each item* in the test clusters. For example, precision for item i_i of class c_j, which is part of test cluster t_k, is obtained by:

$$P(i_i) = \frac{number\ of\ elements\ in\ t_k\ of\ class\ c_j}{number\ of\ elements\ in\ t_k} \tag{4.22}$$

An equivalent recall figure can be also computed. Then, global evaluation figures can be obtained by performing a weighted average over all items (usually using uniform weights).

Amigó et al. propose four formal constraints that are based into simple intuitions about quality of clusters [AGAV08]: (i) *cluster homogeneity*, (ii) *category connectedness*, (iii) *existence of a "rag bag"* (loose elements being grouped together instead of being incorrectly assigned to "good" clusters) and (iv) *item distribution* (cluster size vs. number of clusters). They compare several measures regarding the satisfaction of the previously defined constraints and concluded that only B-Cubed metrics are able to satisfy all constraints simultaneously, while others have to be combined in order to capture all aspects of clustering quality.

However, even with several possible available measures, extrinsic evaluation of clustering is still not trivial since multiple equivalently good, yet different, solutions may exist. Additionally, there may be hidden dependencies between the data to cluster, the algorithm and the measures used for evaluation. Delling et al. present a clustering evaluation framework that tries to incorporate these dependencies and isolate them in specific unit-tests [DGG$^+$06]. Haldiki et al. make a comprehensive analysis of different evaluation measures on several types of clustering algorithms [HBV01].

Two Entropy-based Measures: Cluster Entropy and Class Dispersion

In this thesis we opted for entropy-based measures because they are *invariant to scale*, i.e. they do not depend on the number of items being clustered but only on item *distribution*. Let us assume that there is a procedure that produces clusters of lexical items taking into account a specific notion of similarity in relation to a given input lexical item l_i. We wish to evaluate clusters produced for lexical item l_i, i.e. the *test clusters* T_i with $|T_i|$ clusters, using the corresponding set of *gold clusters*, G_i with $|G_i|$ clusters. In other words, we wish to evaluate how well clusters in T_i, t_1, t_2,...$t_{|T_i|}$ represent the clusters in G_i, g_1, g_2,... $g_{|G_i|}$. For that, we first obtain the Intersection Matrix, I, with $|T_i|$ lines and $|G_i|$ columns. Elements i_{xy} of matrix I indicate the number of items in common between the test clusters t_x and gold clusters g_y.

Ideally, all the elements in a given test cluster, t_x, should belong to only one of the gold clusters. Such t_x cluster is considered "pure" if it contains only items of a unique gold cluster as defined by the gold-standard. If, on the other hand, elements from t_x are found to belong to several gold clusters, then the clustering algorithm was unable to correctly delimit the class at stake.

To quantify how elements in test cluster t_x are distributed over the gold-standard clusters, we use the entropy of the distribution of the elements in t_x over all the clusters g_y. High quality clusters should be very "pure"[1] and thus have very low entropy values. Let $I_t(x)$ be the total number of elements of cluster t_x that *were found in gold clusters*. Then, the *Impurity of cluster* t_x is the entropy of cluster t_x over all gold-standard clusters, and is given by:

$$e_t(t_x) = \sum_{y=0}^{|G_i|} -\frac{i_{xy}}{I_t(x)} \cdot \log_2\left(\frac{i_{xy}}{I_t(x)}\right) \tag{4.23}$$

This measure depends solely on item *distribution* and not on the *sizes* of clusters. For all test clusters obtained for item l_i, we can compute *Impurity* $E_t(T_i, G_i)$ as the *weighted average* of the entropy values $e_t(t_x)$ obtained for each test cluster, t_x:

$$E_t(T_i, G_i) = \frac{\sum_{x=0}^{|T_i|} |t_x| \cdot e_t(t_x)}{\sum_{x=0}^{|T_i|} |t_x|} \tag{4.24}$$

with $|t_x|$ being the number of items in cluster t_x, including those not found in gold clusters, and $|T_i|$ the number of test clusters obtained for item λ_i. Good quality clusters imply low (i.e. close to zero) Impurity values.

We also need to measure if elements of gold-standard clusters are dispersed throughout the test clusters produced. Again, we would like to have all elements

[1]The notion of *Purity* here defined is distinct from what is used in [ZK01], which measures the relative weight of majority class inside each cluster, while *Inverse Purity* measures the relative weight of the larger cluster for each class.

belonging to each gold-standard cluster concentrated in the least number of test clusters possible, ideally only one. Then, for each gold cluster, g_y, we can also compute the entropy value, $e_g(g_y)$, to measure how the elements of a gold-standard cluster g_y are spread over the clusters we are testing. The *Dispersion* of gold cluster g_i, $e_g(g_y)$ can be computed by a formula similar to that of Equation 4.23, substituting references to test cluster by reference to gold clusters, and vice-versa:

$$e_g(g_y) = \sum_{x=0}^{|T_i|} -\frac{i_{xy}}{I_g(y)} \cdot \log_2\left(\frac{i_{xy}}{I_g(y)}\right) \tag{4.25}$$

Similarly, a global Dispersion figure for l_i, $E_g(T_i, G_i)$, can be obtained by performing a weighted average over the Dispersion figures found for each gold cluster, $e_g(g_y)$, (similar to Equation 4.24):

$$E_g(T_i, G_i) = \frac{\sum_{y=0}^{|G_i|} |g_y| \cdot e_g(g_y)}{\sum_{y=0}^{|G_i|} |g_y|} \tag{4.26}$$

Good quality clusters imply very low levels of Dispersion.

Finally, we need to evaluate *recall*, i.e., the proportion elements in gold clusters that are in fact found in *any* test cluster. If $I_g(y)$ is the total of elements in cluster g_y that are found in test clusters, we may define the *item recall* metric for gold cluster g_y as:

$$r_g(g_y) = \frac{I_g(y)}{\sum_{y=0}^{|G_j|} |g_y|} \tag{4.27}$$

An overall Recall figure for the lexical item l_i, R, could be obtained again by doing a weighted average of $r_g(g_y)$ over all gold clusters:

$$R_g(T_i, G_i) = \frac{\sum_{k=0}^{|G_i|} |g_y| \cdot r_g(g_y)}{\sum_{j=0}^{|G_i|} |g_y|} \tag{4.28}$$

We can now consider a generalized scenario where clusters are produced for n lexical items, i.e. l_1, l_2 ... l_n. The corresponding set of test clusters $[T_1, T_2, T_3, ...T_n]$ is denoted by T, and the matching set of gold-standard clusters $[G_1, G_2, G_3, ...G_n]$ is denoted by Γ. Three global performance metrics can be computed by averaging performance figures obtained for each lexical item l_i:

1. *Average Impurity*, $E_t^{avg}(\mathrm{T}, \Gamma)$:

$$E_t^{avg}(\mathrm{T}, \Gamma) = \frac{\sum_i^n E_t(T_i, G_i)}{n} \tag{4.29}$$

2. *Average Dispersion*, $E_g^{avg}(\mathrm{T}, \Gamma)$:

$$E_g^{avg}(\mathrm{T}, \Gamma) = \frac{\sum_i^n E_g(T_i, G_i)}{n} \tag{4.30}$$

3. *Average Recall*, $R_g^{avg}(\mathrm{T}, \Gamma)$:

$$R_g^{avg}(\mathrm{T}, \Gamma) = \frac{\sum_i^n R_g(T_i, G_i)}{n} \tag{4.31}$$

Ideally, *Average Impurity*, $E_t^{avg}(\mathrm{T}, \Gamma)$ and *Average Dispersion*, $E_g^{avg}(\mathrm{T}, \Gamma)$, should be as low (i.e. close to 0) as possible, while R_g^{avg} should be close to 1.

4.2.4 Gold-Standard Resources for Direct Evaluation

The performance measures presented in the previous sections require gold-standard resources against which results of automatic procedures can be directly compared. While in some situations there may exist gold-standard resources that exactly match the evaluation we wish to perform (e.g. a WordNet-like structure for evaluating procedures to identify synonyms), in other cases gold-standard resources need to be created specifically for the evaluation at stake, by collecting or adapting information from other resources.

In the next sections, we briefly describe several resources that can be directly used as gold-standard, or as source for deriving custom gold-standard information. We divide these resources into three groups, according to the type of lexical items and corresponding similarity relations for which they can provide gold-standard information: (i) common lexicon, (ii) entity names and (iii) Web 2.0 tags.

Gold-Standard Resources for Common Lexicon

The main source of gold-standard information concerning similarity relations between words of the common lexicon, mainly nouns, adjectives and verbs, are thesauri and WordNet-like resources. Such type of resources contain information about synonyms (e.g. synonyms pairs or synsets), antonyms and co-hyponyms (usually by hypernymy/hyponymy links), so they are particularly suited for evaluating procedures for computing *content* and *type* similarity between items from the common lexicon. We focus mainly on resources for the Portuguese language, since it is the main language addressed in this thesis (in [TSO10] we report the results of comparing four of these resources in term of the verb synonym information they contain). In all cases, however, there are versions of similar, or alternative resources, for other languages. In fact, in many situations it is much easier to find gold-standard resources for other languages, especially for English, than it is for Portuguese.

Wiktionary Wiktionary is a community-edited resource that stores a wide variety of information about words, such as grammar, pronunciation, etymology, definitions, a list of synonyms, a list of related terms, a list of derived terms, and also translations to other languages. The Wiktionary project started in 2002, and it currently has more than 6.9 million entries for 172 languages. The Portuguese version of Wiktionary[2] was started in 2004 and contains approximately 92,000 entries.

PAPEL PAPEL (*Palavras Associadas Porto Editora Linguateca*) is a freely available[3] lexical database for Portuguese, automatically constructed by extracting relations from a large commercial Portuguese dictionary [OSGS08]. The process consisted in parsing 237,246 dictionary definitions (using a chart parser), and using a set of manually-developed lexical-syntactical patterns to mine 10 semantic relations: (i) *hypernym*, (ii) *cause_of*, (iii) *part_of*, (iv)*means_to_end*, (v) *place_of*, and its corresponding inverse relations. PAPEL contains information about 99,783 words: 55,372 are nouns, 24,089 verbs, 18,933 adjectives and 1,389 adverbs. It contains 79,035 pairs of *synonym* relations.

TeP TeP (*Thesaurus Electrônico para o Português do Brasil*) is a freely-available[4] electronic thesaurus for the Brazilian Portuguese, strongly inspired by WordNet [dS00]. TeP development was based on four dictionaries (used as reference corpora) and relies on three main concepts: (i) *synsets*, i.e. sets of synonyms, (ii) the *lexical matrix*, which defines a biunivocal correspondence between synsets and *senses*, (iii) and the *index*, which establishes the antonymy relation by using indexes. TeP contains information about approximately 44,000 words: 17,000 nouns, 15,000 adjectives, 11,000 verbs and 1,000 adverbs.

OpenThesaurusPT OpenThesaurusPT is the official Portuguese thesaurus used for OpenOffice, and is freely available online[5]. The OpenThesaurusPT project started as an effort to migrate the German OpenThesaurus project [Nab04] to the Portuguese Language. The structure of the OpenThesaurus project is based on WordNet, using synsets to hold information about synonyms and antonyms. It also stores some information about superordinate/subordinate (i.e. hypernymy/hyponymy) relations between synonym sets. The current available version of OpenThesaurusPT dates from 2006-08-17, and contains 4,010 synsets for 12,941 words.

Gold-Standard Resources for Entity Names

Usually, resources related to common lexicon, as the ones we described in the previous section, do not include information about names or about the corresponding entities. Traditionally, information about entities is compiled by au-

[2]`http://download.wikimedia.org/ptwiktionary/20090809/ptwiktionary-20090809-pages-articles.xml.bz2`

[3]`http://www.linguateca.pt/PAPEL/PAPELv1.0.zip`

[4]`http://www.nilc.icmc.usp.br/tep2/download.htm`

[5]`http://openthesaurus.caixamagica.pt/index.php`

thoritative resources such as encyclopaedias or domain-specific ontologies. This type of resources usually group entities in classes according to a given taxonomic (similarity) criteria, and store information about several relations between then (e.g. "equivalent_to", "related_to"). Such information can be readily used as "high-quality" gold-standard information.

More recently, with the advent of Web 2.0, several domain-specific social networks have been created (e.g. music, computer games) allowing the community to add information about relevant entities for the domain at stake, and to *explicitly* (e.g. by cross-linking) or *implicitly* (e.g. by tagging) establish relations between entities. Although there is no assurance about the quality of the information stored in community-edited resources, these tend to become broad repositories of information, and are, thus, potentially interesting for being used as gold-standards for very specific domains.

General and Domain-Specific Ontologies One of the largest freely available ontologies is OpenCyc[6], which is the open source version of the Cyc *general* and *common-sense* ontology, for English[7]. OpenCyc contains hundreds of thousands of terms and millions of relations between them. Alternatively, there are several *domain-specific* ontologies. One of the domains with more resources available is the geographic domain. There are several gazetteers and geographic ontologies freely available, such as *Getty Thesaurus of Geographic Names*[8] the *Alexandria Digital Library Gazetteer Server*[9] or, focusing on Portuguese geographic scope, *GeoNet.pt*[10]. These geographic ontologies can be used, for example, as gold-standard for algorithms that compute type-similarity between entities (i.e. cities within a certain geographic scope). There are also ontologies for other domains that are useful when evaluating algorithms for computing equivalent names for the same entity (i.e. content similarity). One example is the MeSH ontology that contains entries about medical subjects (names of diseases, substances, etc.) and includes information about equivalent alternative names, such as, for example, "[Vitamin C] $\leftrightarrow$ [Ascorbic Acid]".

The Wikipedia Probably one of the broadest and most versatile resources that can be used as gold-standard is Wikipedia. There are Wikipedia versions for more than *270 languages*, containing a total of more than *14 million articles*. The English Wikipedia contains more than three million articles, and the Portuguese contains more than half a million. Because they are community-edited resources, the multiple versions of Wikipedia are constantly growing in size, diversity and density. Most of the articles in Wikipedia refer to entities from a very wide spectrum

[6] `http://www.opencyc.org`

[7] There is no similar resource for the Portuguese language.

[8] `http://www.getty.edu/research/conducting\_research/vocabularies/tgn/`

[9] `http://www.alexandria.ucsb.edu/gazetteer/`

[10] `http://linguateca.di.fc.ul.pt/index.php?l=geonetpt`

of entity types that ranges from *people* to *bacteria*, from *astral bodies* to *computer games*, and many others. Besides the actual content of the article, Wikipedia provides also a broad *classification scheme* that allows users to connect an article to *categories*. For example, the Wikipedia article about the famous music group "The Beatles" is linked to several categories, as "English musical groups", "Music from Liverpool", "1960s music groups", "Parlophone artists", "Rock and Roll Hall of Fame inductees", "Grammy Award winners", among others. The categories assigned to each article can be used as gold-standard information regarding the type-similarity or relatedness between entities: the more categories in common, the more similar the entities should be. From a different perspective, all entities that have been assigned to a given category form an *explicit* set of type-similar entities (under the category at stake). The categories can correspond to rather generic concepts (e.g., "*List of English novelists*"), as well as to very specific topics (e.g., "*List of fractals by Hausdorff dimension*"). This grouping information can be directly used as gold-standard information for procedures that compute *type similarity*. There are also several *equivalence links* between names of entities that can be used as gold-standard information for content similarity. Finally, Wikipedia provides valuable information regarding *ambiguous names*, by listing all possible entities in a single "disambiguation" page. For example, there are several dozens of *well-known* entities named "Paris" with one dedicated article in Wikipedia, and they are all listed along with a small differentiating description in a dedicate disambiguation page[11].

One of the main obstacles in using Wikipedia as a gold-standard is the fact that Wikipedia articles are formatted in a special mark-up language that does not allow an easy programmatic access to all potentially relevant information. The DBPedia project[12] was started with goal of extracting information from Wikipedia articles and store it tabular format to allow simple access, hence facilitating the use of information from Wikipedia for a variety of purposes, including for serving as gold-standard. However, the DBPedia project has essentially focused in producing tabular information from the English and German versions of Wikipedia. Thus, the amount and variety of tabular information available in other languages (including Portuguese) is significantly inferior.

Social Networks There are several social networks focusing on a particular domain that foster the development of knowledge resources, which can later be used for gold-standard purposes. For example, Last.fm[13] is a Web 2.0 radio platform that allows users to listen to music and, at the same time, tag both artists and songs. Users are able to associate different types of tags to artists such as those referring to music genre (e.g. "acid jazz"), locale (e.g. "japan"), artist/band

[11]`http://en.wikipedia.org/wiki/Paris_(disambiguation)`
[12]`dbpedia.org`
[13]`http://www.last.fm`

structure (e.g. "duo") or instrumentation (e.g. "guitar"). This allows, for example, creating groups of artists based on the tagging provided by the community of users (e.g. all artists tagged "acid jazz"). Since this information is usually made available for download, it can be used as gold-standard for procedures that compute type similarity between this type of entities. There are many other domain-specific social networks that allow users to tag or relate items and which can be source of gold-standard information, such as *Shelfari*[14] or *Library Thing*[15] for literature, books and writers, *The Internet Movie Data Base*[16] for movies, actors, etc. or *Snooth*[17] for wines.

Gold-Standard Resources for Web 2.0 Tags

The best source of gold-standard information for evaluating procedures that compute similarity between Web 2.0 tags is the tag information itself extracted from social networks. There are two options for extracting such information. The first is to directly use information about "similar tags" that Web 2.0 sites usually provide. This is interesting if the procedure that is used by the Web 2.0 site to compute tag similarity combines evidence from multiple sources, namely from community usage patterns, and is thus *different* from the procedure whose results are being evaluated. For example, Last.fm web radio computes artist similarity taking into account several sources of evidence including the preferences that users explicitly or implicitly express (which Last.fm keeps private) when listening to the radio stream. Also, Last.fm makes available through a web-service API[18] a procedure that returns the list of (type) similar artists for each artist, along with a normalized similarity value. This data can be used for evaluating procedures for computing type similarity in the domain of music artists.

The second option consists in using *diachronic information* about tag assignment. Let $\tau(i, t_0)$ be the set of tags associated with a given element e_i of the network, at time t_0, and $\tau(i, t_1)$ be the set of tags associated to the same element, e_i, in a *later* point in time, $t_1 > t_0$ (e.g. six month after). The difference between the two sets, $\tau_{diff}(i) = \tau(i, t_1) - \tau(i, t_0)$ contains important information about tags that the users believed to be important to *add to* or *replace from* the tag bag of item e_i during that time interval. Thus, from $\tau_{diff}(i)$ we can obtain gold-standard information *asymptotically validated* by the community users, for evaluating procedures aiming at automatically expanding $\tau(i, t_0)$, i.e. procedures that find tags *functionally similar* to $\tau(i, t_0)$ (*functional* in the sense that tags are mainly used for aiding item retrieval).

[14]`www.shelfari.com`
[15]`http://www.librarything.com`
[16]`http://www.imdb.com/`
[17]`http://www.snooth.com`
[18]`http://www.lastfm.com.br/api`

4.3 Indirect Evaluation Strategies

Indirect evaluation of a given procedure presupposes the existence of an application that consumes the result of the procedure under evaluation. It is, thus, the most appropriate type of evaluation either for procedures related to *functional* similarity, or whenever there is not a very clear notion of what the "true" value of similarity is but, instead, there is a notion of what should be the correct behaviour of a broader system that uses such similarity information.

Let $\mathcal{S}(x)$ be a procedure for performing a similarity based computation on a given input x (e.g. a list of lexical items). Let $\mathcal{A}(y)$ be an application for achieving some higher-level language processing goal. Indirect evaluation consists in evaluating $\mathcal{S}(x)$ by evaluating $r_{\mathcal{S}}$, the result of $\mathcal{A}(\mathcal{S}(x))$:

$$r_{\mathcal{S}} = \mathcal{A}(\mathcal{S}(x)) \tag{4.32}$$

There are mainly two possible evaluation sub-scenarios for indirect evaluation. In the first scenario, $r_{\mathcal{S}}$ is evaluated *individually*, either against a gold-standard resource (customized for $\mathcal{A}(y)$) or compared against some desired target performance values (e.g. number of correct factoids extracted, if $\mathcal{A}(y)$ is an information extraction application).

The second sub-scenario consists in performing a *comparative* evaluation against the results obtained by a reference input, y_{ref} or, more generically, by a reference function $\mathcal{S}_{ref}(x)$ (e.g. a baseline or a legacy function). Let r_{ref} be a reference value produced when $\mathcal{S}_{ref}(x)$ is the input of $\mathcal{A}(y)$:

$$r_{ref} = \mathcal{A}\left(\mathcal{S}_{ref}(x)\right) \tag{4.33}$$

Then, $r_{\mathcal{S}}$ can be compared to r_{ref} using the appropriate gold-standard $\mathcal{A}(y)$. If no such gold-standard exists, then $r_{\mathcal{S}}$ and r_{ref} can be compared according to a pre-defined set of performance criteria.

However, by moving the focus of the evaluation from a low-level procedure to a higher-level procedure or application, we transform a relatively constrained performance measurement into a much broader evaluation exercise, which may include, for example, assessing facets that are relevant to *end-users* of such application. The set of criteria used for evaluating the performance of higher-level application tends to be multi-faceted and, thus, more difficult to establish. Some potentially interesting criteria to evaluate results achieved with $\mathcal{A}(y)$ may be related to:

- *user satisfaction*: is $r_{\mathcal{S}}$ a better result than r_{ref} from the point of view of the user of application $\mathcal{A}(y)$? For example, if $\mathcal{A}(y)$ is an information retrieval application, do documents in $r_{\mathcal{S}}$ represent better answer to the user's information need (does he/she clicks more on those documents)?

- *computational performance*: does $\mathcal{S}(x)$ allow $\mathcal{A}(y)$ to be computed more efficiently (CPU / RAM / Storage) that $\mathcal{S}_{ref}(x)$ for the same or equivalent results?

- *engineering requirements*: is $\mathcal{S}(x)$ easier to set up, to integrate or to maintain within $\mathcal{A}(y)$ architecture in comparison with $\mathcal{S}_{ref}(x)$ for the same or equivalent results?

- *economic factors*: if $\mathcal{A}(y)$ generates revenue, does the integration of $\mathcal{S}(x)$ in $\mathcal{A}(y)$ lead to higher revenues, even if $\mathcal{S}(x)$ is *less accurate* from a conceptual point of view than $\mathcal{S}_{ref}$?

4.4 Conclusion

In this chapter, we summarized the main concepts regarding *direct* and *indirect* evaluation of procedures for computing similarity between lexical items. One of the main contributions of this chapter is the systematization and formalization of several possible evaluation strategies and scenarios, including related performance criteria and measures. Another important contribution presented is the comprehensive overview of resources that can provide gold-standard information for direct evaluation settings, specially, but not only, when considering the Portuguese language. We divided those resources in three groups based on the type of lexical items for which they provide gold-standard information, namely common lexicon, names of entities and Web 2.0 tags. Additionally, we explained how to *adapt* or *derive* gold-standard information from those resources when such information is not *directly* accessible.

Part II

Common Lexicon

In this part of the thesis, we focus on the computation of similarity-based functions for problems involving items of the *Common Lexicon*, i.e. the lexicon of *common words*. In chapter 5, we address a quite traditional task, namely automatically finding synonyms, more specifically for verbs. We use a Vector Space Model (VSM) approach, and we perform an exploration over the parameter space of the VSM in order to find the best configuration for the specific problem at stake. In chapter 6, we focus on the problem of inferring rules for generating paraphrases of job titles. Basically, given a set of job title *noun-phrases*, we wish to learn rules for generating other equivalent job titles.

Both of these problems have many things in common. The first one, more obvious, is that both involve issues related with *content similarity*. In one case, we wish to find verb *synonyms*. In the other, we wish to be able to generate *paraphrases*, i.e. two equivalent expressions. The second thing in common is that both of these problems have been inspired by very practical needs related to two information extraction tasks over on-line news sources.

In fact, the need for automatically finding verb synonyms comes from an applications that constantly mines on-line news for quotations (e.g. "Obama says that..."). We had no verb dictionary available containing a large enough number of verbs related to *speech-acts* for supporting the quotation extraction procedure. We, thus, decided to infer a network of verb synonyms, and then use such network to find relevant verbs starting from a small set of known verbs related to speech-acts and following the synonymy links that were obtained automatically.

The work regarding the inference of rules for generating paraphrases of job titles was also motivated by practical questions related to an entity tracking task. Because there are so many distinct ways of expressing equivalent job titles – which are required for finding indirect mentions to a set of target entities – the problem of entity tracking on media sources becomes very difficult to solve. Since manual development of rules for dealing with such variation is not practical, we opted for automatically inferring such relations. Thus, we developed a data-driven way of acquiring valid rules for paraphrasing job titles that allows generating a large number of valid job title variations for a given entity, starting from a set of a few known ones (e.g. the most frequently used ones).

Chapter 5

Identification of Verb Synonyms

In this chapter[1] we assess the virtues (and limitations) of the Vector Space Model (VSM) in helping to solve a long standing problem in semantics, namely that of automatically finding *synonyms*. As explained in chapter 2, synonymy is an instantiation of *content-similarity*. We focus on a specific case: finding synonyms for verbs in Portuguese. The motivation for finding verb synonyms came from the need to expand a lexicon of verbs related to communication (i.e. *speech-acts*) that is used in a quotation extraction application[2].

The traditional approach for finding synonyms using the VSM consists in projecting feature information in the Vector Space and then computing distances between Feature Vectors. Therefore, our main research questions at this stage are related to quantifying the impact of several core parameters of the VSM in the final result of the synonym pairs obtained. For practical reasons, we use only *directly observable features* and, thus, avoid the need for complex linguistic pre-processing. We investigate which *feature weighting function* (see section 3.4) leads to better results, and which type of *context window* (see section 3.3) contains more information. We also test the impact of considering only feature vectors with a given minimum number of features, so as to estimate which is the minimum amount of vector information that is required for the task of synonym finding.

To allow a comprehensive parameter exploration, we perform *automatic* evaluation of the VSM-based synonym computation procedure, using *gold-standard* information from two publicly available thesaurus. As it will become clear, automatic evaluation of synonymy procedures is extremely challenging since most gold standard resources lack many important synonym relations. Nevertheless, automatic evaluation is enough for the purpose of optimizing parameters. Next, for obtaining a clearer notion of the real performance of the VSM-based approach

[1] Most of the material found in this chapter was presented in *Luís Sarmento, Paula Carvalho and Eugénio Oliveira "Exploring the Vector Space Model for Finding Verb Synonyms in Portuguese"* [SCO09].

[2] See `http://verbatim.labs.sapo.pt`

we propose, we *manually* evaluate the synonyms obtained for two sets of verbs: *declarative verbs* and *psychological verbs*. These sets of verbs pose different challenges to the automatic synonym finding procedure: while it should be relatively hard to automatically find synonyms for *psychological verbs*, since they are highly polysemous and almost always have several antonyms, *declarative verbs* are expected to be easier to tackle since they do not exhibit such high levels of polysemy nor they tend to have so many antonyms. Results obtained for these verbs allow us to estimate the *best* and *worst* case scenario for the task of automatically finding verb synonyms in Portuguese, using direct data-driven approaches.

5.1 Introduction

Large-coverage and fine-grained linguistic resources are crucial for the majority of the applications in natural language processing, but they are still scarce and, in most cases, they do not satisfy every particular information need. Manual creation of linguistic resources is time-consuming and requires linguistic expertise. Therefore, there is a rising interest in developing automatic or semi-automatic methods and techniques for building language resources with minimal human intervention. However, automatic methods usually involve a large set of parameters, whose impact on final results is difficult to assess, and thus to optimize. In this chapter, we address the task of automatically creating a lexicon of verb synonyms for Portuguese using the Vector Space Model (VSM). We explore the impact of three of the core parameters of the VSM on the quality of the final synonymy relation obtained. The parameters explored are: (i) the *context* used for extracting vector features, (ii) the function used for weighting features, and (iii) the *cut-off threshold* for removing vectors with insufficient feature information. We rely on n-gram information collected from a large dump of the Portuguese web, in order to obtain distributional statistics for verb lemmas. For performing parameter exploration, we evaluate results automatically using gold-standard information extracted from the OpenOffice thesaurus and from Wiktionary. Fine-grained evaluation is achieved by manually assessing the synonym candidates obtained for a sample of two syntactic-semantic classes of verbs: *psychological verbs* and *declarative verbs*.

We chose these two specific verb classes for two reasons. First, they exhibit different syntactic and semantic behaviour, and thus present different challenges for the task of synonymy finding. In fact, psychological verbs do not have a prototypical syntactic structure and they usually convey a plurality of meanings, which can only be disambiguated in context. In contrast, declarative verbs are less ambiguous and the syntactic structure where they occur is better defined. Second, these two verb classes are crucial in several information extraction tasks such as, for example, *quotation extraction from news* or *opinion mining*, so they provide a realistic and practical test-bed.

To the best of our knowledge, this study is pioneer for the Portuguese language. Since our approach relies only on minimal linguistic processing, the results presented can be considered a *baseline* for other methods that try to perform the same task, using additional linguistic information.

5.2 Related Work

Curran follows an experimental methodology for testing several parameters of the VSM in the process of automatically computing a language thesaurus – the context for extracting features, functions for weighting those features, functions for computing vector similarity, cut-off thresholds for input data and algorithms for computing pairwise vector similarity [Cur04]. The author performs large scale experimentation on the parameter space and evaluates results automatically by computing precision at several ranks, inverse ranks (InvR) and direct comparison with a gold standard built by aggregating 5 thesauri: the *Roget's Thesaurus*, the *New Roget's Thesaurus*, the *Moby Thesaurus*, the *New Oxford Thesaurus of English* and the *Macquire Encyclopedic Thesaurus*. WordNet was also used to automatically check if results on synonymy are contaminated with antonyms, hyponyms or meronyms. Detailed error analysis was performed for a sample of 300 words. Results show that when the number of features associated to vector drops below 1000, or for words with frequencies below 5000, performance decays significantly. Additionally, direct comparison and InvR measures tend to increase for words with multiple senses with larger number of senses while the precision measures are fairly stable. Results also demonstrate that it is more difficult to find synonyms for words related with certain Wordnet classes such as *entities* and *abstractions*.

Sahlgren builds vector spaces for capturing either *paradigmatic* or *syntagmatic* relations, and tests how such spaces can then be used for different tasks – thesaurus generation, synonym finding, antonym detection and POS guessing [Sah06]. The author evaluates the impact of several VSM parameters such as (i) the *type of context* (paradigmatic vs. syntagmatic), (ii) the *size of the context window* (narrow vs. wide and small vs. large) and (iii) the *weighting of the windows* (constant vs. aggressive decay) feature weighting functions (raw frequency vs. binary vs. tf-idf vs. logarithmic). For the specific task of finding synonyms, the author concludes that spaces built using paradigmatic contexts clearly outperform those built using syntagmatic contexts. Additionally, vectors built by extracting word features from *narrow windows* (with two or three context words around the headword) lead to better performance. Interestingly, wide windows lead to better results for the task of finding *antonyms*.

im Walde presents a set of experiments on clustering German verbs (by synonymy) [iW06]. Verbs are described by vectors whose features are extracted from 3 types of contexts with increasing levels of semantic information: (i) syntactical relations (from a set of 38 possible frames); (ii) syntactical relations plus information about prepositional preferences, and (iii) 15 possible semantic categories of the verb arguments (mostly nouns and noun phrases) taken from GermaNet. In a first experiment, a set of 168 German verbs was manually classified into 43 semantic classes, to be used as gold-standard for the clustering experiments. On a second experiment, clustering of 883 verbs (including the previous 168) was attempted. Clustering was performed using the *k-means* algorithm. Several parameters related to clustering (k-means initialization, linkage, metric) and to the set of features used (i.e. context) were experimented. The author concludes that the addition of more informative features – from (i) to (iii) – has a positive effect on clustering results. Also, she observes that (a) similarity metrics such as the Kullback-Liebler and its variants tended to produce better results in larger data-sets, and (b) low-frequency verbs have a negative impact in the quality of the clusters. More importantly, she concludes that the choice of features and the overall success of the clustering approach greatly depends on definition of *verb group* one wishes to replicate automatically.

Takeuchi describes a co-clustering approach to building sets of verb synonyms in Japanese [Tak08]. A dependency parser is used to extract verb-name structures, and a information theoretic co-clustering procedure is then applied to cluster both the verbs and the nouns simultaneously. The author experiments applying the co-clustering algorithm over frequency and binary vector representations, and perform additional comparison with a vector-based approach (Probabilistic Latent Semantic Indexing - PLSI). Evaluation is performed by comparing clusters obtained automatically using several different corpora with a manually constructed thesaurus containing 4400 verbs, and computing *Purity*. Results show that both co-clustering configurations outperform the vector-based approach, and the frequency-based vector representations outperform binary-based representation in the co-clustering approach. However, performance figures are always modest with the best configuration achieving a value of Purity of 0.234.

There are also other works that deal with verbs but in different settings. For example, the work by Chklovski and Pantel addresses the problem of automatically finding several semantic relations between verbs, namely *similarity*, *strength*, *antonymy*, *enablement* and *happens-before* [CP04]. The procedure involves querying a search engine for co-occurrences of pairs of verbs in specific lexical-syntactic patterns that indicate that the verbs might establish one of such relations. Results were evaluated by human assessors. The work by Resnik and Diab focus essentially of method comparison and evaluation [RD00]. They compare the results of applying three different automatic methods for establishing verb similarity against ratings provided by 10 human assessors. Methods compared include measuring similarity using (i) taxonomic information from WordNet, (ii) distributional information extracted from a corpus and (iii) lexicon with semantic frames. They concluded highest correlation with human assessors was obtained with methods relying on WordNet.

Even more generic, Riloff and Shepherd present a bootstrapping method for automatically expanding sets of semantically related words [RS97]. The system receives as input a seed set that defines extensively a category and expands that list using distributional information taken from corpus. Evaluation was made by human assessors. A similar bootstrapping method is presented by Roark and Charniak, also involving manual evaluation [RC98]. Lin uses a broad-coverage parser to obtain grammatical relationships between pairs of words [Lin98]. Each word is then represented by a vector whose features are derived from the set grammatical relations it establishes with other words. Raw frequency values are weighted using a variation of the Mutual Information function. Pairs-wise similarity between nouns, verbs and adjectives/adverbs that occurred at least 100 times was computed, using several similarity metrics. Then, for each word, a thesaurus entry was created using the top most similar words. Evaluation was performed using WordNet and the Roget Thesaurus.

Related work on VSM generally takes advantage of significant linguistic information, usually extracted from *annotated* corpora. In this study, we rely mostly on the information derived directly from data, in particular, on raw n-grams statistics taken from a large non-annotated collection of web documents. Apart from dictionary-based filtering and lemmatization, no additional linguistic processing (e.g. POS annotation, word-sense disambiguation) is used. Given the increasing availability of large databases of n-grams computed from non-annotated terabyte web collections (e.g. Goolge's N-gram database) and the lack of publicly available resources for Portuguese with refined semantic information, we believe that this is a promising approach.

Also, in our work we do not follow a clustering approach for computing word synonyms, as it is done in several other related works. In fact, for tasks of finding pairs of synonyms there is no need to perform any clustering procedures since we are only interested in finding *weighted links* between pairs of words, and not *groups* of semantically related words. Given the weighted links between words, i.e. the link graph, there are several graph-based clustering algorithms that can be used for finding the clusters – and dealing with other situations such as ambiguity (e.g. [WD02]) – which could eventually be applied. However, the scope of our study is solely that of exploring parameters related to the task of finding the set of nearest synonyms for verbs ranked by degree of similarity, such as in [Cur04] and [Sah06]. Although some clustering-based approaches to the same task are possible, they introduce additional parameters to the problem (e.g. stopping criteria, strategy for dealing with outliers, option between "hard" and "soft" clusters) that are difficult to set and to correlate with the quality of final results.

5.3 VSM for Verb Synonyms

As mentioned before, we wish to investigate the impact of considering different feature contexts for the task of finding verb synonyms. Concretely, we confined the context window to the *four* words around the verb, i.e. a [-2 : +2] window. Since Portuguese is a Subject Verb Object (SVO) language, we believe that such context contains, in the majority of the cases, relevant information about *verb-object* and *subject-verb* relations[3]. The right and the left contexts are specially important for the case of *transitive* and *intransitive verbs*, respectively. We also assume that features extracted from such contexts might be compiled independently, so that feature vectors can be created by aggregating the two sources of statistical evidence.

For obtaining verb context information we used BACO [Sar06a], a database of n-gram statistics compiled from a dump of the Portuguese web, totalling about 1000 million words. We scanned 3-gram information of the form $\langle w_1, w_2, w_3, f \rangle$ for cases where either w_1 or w_3 were verbs. N-gram information in this collection is *not* POS-tagged. Nevertheless, since the majority of verb forms are inflected, they can be unambiguously recognized using a simple dictionary (at least for the vast majority of possible forms). Hence, we used a dictionary to filter out ambiguous verb forms – i.e. those that could not be uniquely assigned to an unique (verb) lemma – so that only the 3-grams matching either of the two following selection patterns were chosen (v_{uf} = unambiguous verb form):

- *Pattern 1* = $[w_1 = v_{uf}$ & $w_2 = *$ & $w_3 = *]$
- *Pattern 2* = $[w_1 = *$ & $w_2 = *$ & $w_3 = v_{uf}]$

Verb forms (at w_1 or at w_3) are lemmatized in order to obtain feature tuples of the form $\langle$ verb lemma, "X w_2 w_3", frequency $\rangle$ and $\langle$ verb lemma, "w_1 w_2 X", frequency $\rangle$, with X signalling the original position of the verb in relation to the extracted features. Feature information extracted for the various forms of the same lemma is merged so to that a single feature vector is obtained for each verb lemma. At this point, feature vectors contain raw frequency information regarding features extracted from the two words before the verb and from the two words after the verb. Features can then be weighted according to given weighting function to produce *weighted feature vectors*, which should be able to reflect more faithfully the association between verbs and features.

Next, weighted feature vectors are compared so that we obtain all pairwise similarities. Synonyms for verb v_i are obtained among the other verbs, v_j, whose feature vectors $\overline{v_j}$ are more similar to $\overline{v_i}$. By this procedure, we are not producing *closed sets* of verb synonyms. Instead, we are building a network of similarities

[3]It should be stressed, however, that this context window is not sufficient for all cases, namely when there is a modifier between the verb and one of its arguments.

which enables a verb to be synonym of many other verbs, depending on the different senses it conveys.

The overall procedure we use to automatically compute the verb thesaurus for Portuguese can be described by the four following steps:

1. compile *feature vectors* by filtering the 3-gram database with selection patterns;

2. compile statistics regarding the feature and generate the set of *weighted feature vectors* using a given *weighting function*;

3. compute pairwise vector similarity using the metric of choice (e.g. cosine metric);

4. for each verb v_i obtain the top n vectors closest to its word vector, keeping the corresponding words as "verb synonyms".

However, we know in advance that the context scope chosen will not allow to differentiate between synonyms and antonyms. Opposite sense verbs tend to occur in the *same contexts*, since they usually select identical arguments and allow the same modifiers (e.g. "Please, open the door!" and "Please, close the door!"). Nevertheless, we decided to analyze how VSM performs in the detection of synonyms in Portuguese and assess the true impact of this limitation. Furthermore, we assume that antonyms could be identified in a subsequent *post-processing* step by using techniques based on typical lexical-syntactic patterns such as the ones described in [CP04].

5.4 Evaluating Verb Synonyms

We used a publicly available resource as a gold-standard for automatic evaluation: the OpenOffice thesaurus for Portuguese[4]. From the OpenOffice thesaurus we collected $\langle$ verb $\rightarrow$ list of synonyms $\rangle$ mappings for 2,783 verbs, each having 3.83 synonyms in average. However, this information refers only to about 50% of the verb lemmas one can find in standard on-line dictionaries for the Portuguese language (e.g. [AP94]). More important, there are serious *recall* problems for the mappings collected. For example, many high-frequency verbs have *only one* synonym in OpenOffice thesaurus: "ganhar" (to "win") $\rightarrow$ "poupar" ("to save (money)");"afirmar" ("to state") $\rightarrow$ "declarar" ("to declare"); "chamar" ("to call") $\rightarrow$ "invocar" ("to invoke"), among many others. In order to minimize this problem, we extracted additional verb synonym information from the Portuguese version of the Wiktionary project[5]. We thus obtained additional (verb $\rightarrow$ list

[4]Available from http://openthesaurus.caixamagica.pt/. The most recent version is dated from 2006-08-17.

[5]Available at http://download.wikimedia.org/ptwiktionary/

of synonyms) mappings for 2,171 verbs, each having in average 1.95 synonyms. By merging mappings extracted from both resources we obtained a larger gold-standard covering 3,423 verbs, with 4.53 synonyms per verb. This larger gold-standard still has coverage and recall problems, but we believe that it provides a good solution for the purpose of performing *parameter exploration*.

Nevertheless, we chose to perform a more thorough evaluation by manually analyzing results obtained for two subclasses of verbs. We selected two groups of verbs with different syntactic and semantic properties (see Table 5.1). The first group includes 25 *declarative verbs*, such as "dizer" ("to say") or "mencionar" ("to mention"), and will be referred as V_{com}. The second group includes 25 *psychological verbs*, such as "gostar" ("to like") and "envergonhar" ("to shame"), and will be mentioned as V_{emo}. V_{emo} are related to the expression of a sentiment or an emotion, which can be experienced by the human noun occupying the subject or the complement position, according to the verb at stake. The level of polysemy of verbs in V_{com} is relatively low. On the other hand, verbs in V_{emo} are highly polysemous. This fact is somehow reflected by the vast list of possible antonyms, with various degrees of strength, that can be associated to verbs in V_{emo}. Sets V_{com} and V_{emo} can be placed in opposite ends of the spectrum regarding the performance that one expects to achieve in the task of synonym finding: performance for V_{com} should be higher than for V_{emo}.

Table 5.1: Verb groups chosen for manual evaluation.

	Verbs
V_{com}	acrescentar, adiantar, afirmar, alertar, anunciar, avisar, comunicar, confessar, contar, comentar, declarar, defender, destacar, dizer, esclarecer, explicar, frisar, indicar, mencionar, nomear, responder, referir, revelar, salientar, sublinhar
V_{emo}	aborrecer, adorar, agradar, amar, angustiar, assustar, atemorizar, chatear, decepcionar, detestar, emocionar, enternecer, entristecer, entusiasmar, envergonhar, fascinar, gostar, humilhar, impressionar, intimidar, irritar, lisonjear, orgulhar, preocupar, ridicularizar

Performance Metrics

Let V_{gold} be the set of verb entries in the *gold standard verb thesaurus*, and let V_{auto} be the set of verb entries for which synonyms mappings were obtained by the automatic method. Also, let $S_{gold}(v_i)$ be the set of verb synonyms defined for entry v_i in the gold standard thesaurus (i.e. the "true" synonyms), and $S_{auto}(v_i)$ be the set of synonyms inferred automatically for v_i. As a result of the automatic process, elements in $S_{auto}(v_i)$ are ranked according to the degree of synonymy they have with v_i. Thus, traditional metrics used in information retrieval can be used for evaluating the ranked sets of verb synonyms, $S_{auto}(v_i)$, against those in $S_{gold}(v_i)$. Because verb mappings contained in the gold standard are far from being complete, we will not compute recall figures and we will mainly focus on evaluating *precision*.

More specifically, for each verb entry $v_i \in (V_{auto} \cap V_{gold})$, we will compute three precision figures. The first is *Precision at Rank 1*, $P_@(v_i, 1)$ (see Equation 4.17). The second is *Precision at Rank* $N_{gold}(v_i)$, $P_@(v_i, N_{gold}(i))$, with $N_{gold}(v_i)$ being the number of true synonyms contained in $S_{gold}(v_i)$ (see Equation 4.18). The third is *Average Precision*, $AP(v_i)$, which gives a global view of the precision by combining the values of the precision at various ranks (see Equation 4.19).

Global performance figures can be obtained by averaging each of the previous metrics, i.e. $P_@(v_i, 1)$, $P_@(v_i, N_{gold}(v_i))$ and $AP(v_i)$, over all entries for which evaluation was possible, i.e. for $v_i \in (V_{auto} \cap V_{gold})$. This allows us to compute three global precision figures: $P_@^{avg}(1)$, $P_@^{avg}(N)$ and MAP. A global *coverage* figure, $\mathcal{C}$, can be computed by dividing the number of entries evaluated by the total number of entries in the gold standard thesaurus: $\mathcal{C} = |V_{auto} \cap V_{gold}|/|V_{gold}|$.

For manual evaluation, we are no longer limited by the number of "true" synonyms contained in the gold standard for a given entry, so we can compute the value of precision at several ranks up to a reasonable value (although we still can not list all possible synonyms of a verb). We chose to compute precision at ranks 1, 5, 10 and 20, which will be represented by $P_@^{man}(v_i, n)$, with $n \in \{1, 5, 10, 20\}$.

5.5 Experimental Setup

The main goal of this work is that of evaluating the impact of three of the core parameters of the VSM on the overall quality of the synonymy mappings than it allows to obtain. Thus, we performed three sets of experiments, one for each parameter at stake.

Experiments Set 1 First, for assessing the impact of different *weighting functions*, we will run the complete procedure for automatically generating synonym mappings iteratively, keeping the same context scope – a window of $[-2, +2]$ words – while using different feature weighting functions. We will try several weighting functions documented in chapter 3, namely: tf-idf [SM86], Log-Likelihood Ratio (LL) [Dun93], Z-Score [Sma93], Pearson's χ^2 test, ϕ^2 test [CG91], [Eve05], Student's T test [Eve05], Mutual Information [CH90] (MI) and Mutual Dependency (MD) [TFK02]. We also run the complete experiment without using any weighting function, i.e. using raw frequencies. For this set of experiments, we arbitrarily set the cut-off threshold on the minimum number of features to 1. Additionally, pairs with cosine similarity lower than 0.1 will be excluded (which can lead to different coverage values).

Experiment Set 2 The second parameter to be explored is the context window used for extracting features. Experiment Set 2 consists in executing the complete synonymy finding procedure using only features extracted from a $[-2, 0]$ window (i.e. the two words preceding the verb) and from a $[0, +2]$ window (i.e. the two words following the verb). These experiments will be run using the *best performing* weighting function found in the previous experiment. The third parameter we wish to investigate is the *cutoff threshold* to be applied to raw frequency feature vectors based on the number of non-null features.

Experiment Set 3 In *Experiment Set 3* we will select the *best performing* weighting function found in Experiment Set 1, and repeat the complete synonym finding process with increasing cutoff thresholds. We expect to obtain increasing precision values, while coverage should slowly decrease.

Finally, for refining the figures obtained by automatic evaluation, we will manually evaluate two subsets of verbs that lie on the opposite ends of the spectrum in what performance is concerned. The main purpose is to define a possible baseline for the task of automatic synonym finding, knowing in advance that the VSM approach we used is almost purely lexical (it relies on a minimal set of linguistic features) and does not try to address issues related with antonymy and ambiguity. We will chose the best performing configuration found in Experiment 3 in terms of $P_{@}^{avg}1$, and manually evaluate candidate synonyms found for 25 V_{com} verbs (related to *communication*) and 25 V_{emo} verbs (related to the *expression of emotion*). Results for verbs in V_{emo} are expected to be substantially worse than those for V_{com}.

5.5.1 Feature Information

Feature information was obtained from our n-gram database ([Sar06a]). There are 173,607,555 distinct 3-grams available in the database. The *Selection Pattern 1* (see section 5.3) allowed collecting feature information for 4,972 verbs, described in a space with 2,002,571 dimensions. Likewise, *Selection Pattern 2* allowed to collect feature information for 4,962 verbs over 2,066,282. Globally, by aggregating information from both patterns, we were able to collect information for 5,025 verbs in a space with 4,068,853 dimensions.

Table 5.2 presents an histogram regarding the number of word vectors per number of features.

5.6 Results and Analysis

Global precision figures $P_{@}^{avg}1$, $P_{@}^{avg}N$ and MAP (mean average precision) for Experiment Sets 1, 2 and 3 (automatic evaluation) are presented in Tables 5.3, 5.4 and 5.5. Results of manually evaluating synonym identification for the 25

Table 5.2: Number of vectors per number of features

# feat.	# vec.
< 10	541
10 - 19	220
20 - 29	145
30 - 39	136
40 - 49	112
50 - 99	353
100 - 199	471
200 - 499	777
500 - 1k	580
1k - 2k	456
2k - 5k	497
5k - 10k	306
10k - 50k	382
$\geq$ 50k	49

verbs related to *communication*, V_{com}, and the 25 verbs related the *expression of emotion*, V_{emo}, are presented in Table 5.6 (synonym candidates were obtained by setting the cutoff threshold to 200, i.e. corresponding to best $P_{@}1$ found in Experiment Set 3).

The most relevant, yet expected, fact regarding results from automatic evaluation is that precision values are all quite low, even for the best configurations (*simeq*0.22). This is not surprising since the gold standard used has serious recall gaps, so it is possible that many correct top found synonyms can be evaluated, thus decreasing precision figures. In [Sah06], even lower precision figures are reported. Also, we knew in advance that the context chosen for generating feature vectors does not allow to effectively differentiate between a verb and its possible opposite senses. Still, performance values obtained can be interpreted from a *relative* point of view.

Results presented in Table 5.3 confirm that the impact of the weighting function is very relevant. The best performing weighting function (Mutual Information) leads to a Mean Average Precision figure that outperforms the one obtained using the worst performing weighting function with comparable coverage (Log-Likelihood) by over 300%. Notably, the two best performing weighting functions are Mutual Information and Mutual Dependency, both grounded in information theoretic concepts (the two metrics are actually very similar). A well-known effect of these types of metrics is that they tend to asymptotically over-promote rare features. This suggests that rare features might be of crucial value in the task of finding semantically similar verbs.

It is also quite surprising to see that most weighting functions score worse

Table 5.3: Experiment Set 1: context = [-2, +2] and cutoff threshold = 1

Weighting	$P_{@}^{avg}1$	$P_{@}^{avg}N$	MAP	$\mathcal{C}$
MI	0.221	0.121	0.125	0.800
MD	0.164	0.083	0.083	0.800
Z	0.134	0.096	0.067	0.712
χ^2	0.087	0.075	0.030	0.392
ϕ^2	0.084	0.075	0.027	0.375
raw	0.083	0.041	0.043	0.798
tf-idf	0.076	0.038	0.039	0.800
T	0.073	0.040	0.040	0.800
LL	0.059	0.034	0.037	0.796

than performing no weighting at all (*raw*). This happens even in the case of popular weighting functions such as tf-idf. One possible reason for this is having set the cut-off threshold on the minimum number of non-nil features to 1, which resulted in considering many verb vectors with insufficient statistical information (see Table 5.2). Some of the weighting functions used seem to be particularly sensitive to this effect, and actually lead to worse results than performing no weighting at all. Another observation is that the coverage obtained using the χ^2 and ϕ^2 weighting functions was approximately half of that obtained for the others. Since coverage is a functions of the minimum cosine similarity (which was kept 0.1 for all weighting functions), we confirms that there is in fact a considerable interaction between the choice of the weighting function and the similarity metrics used.

Results from the Experiment Set 2 (Table 5.4) show that using feature information from *both* the left and the right sides of the verb lead to better results that using any of the two sides individually. From a relative point of view, the two words following the verb (i.e. context [0, +2]) appear to carry more information regarding verb synonymy than the two previous words (i.e. context [-2, 0]), which seems quite natural since most verbs are transitive.

Table 5.4: Experiment Set 2: weighting function = Mutual Information and cutoff threshold = 1

Window	$P_{@}^{avg}1$	$P_{@}^{avg}N$	MAP	$\mathcal{C}$
[-2, 0]	0.136	0.078	0.079	0.779
[0, +2]	0.196	0.107	0.111	0.798
[-2, +2]	0.221	0.121	0.125	0.800

As for Experiment Set 3, results shown in Table 5.5 confirm expectation: increasing the cutoff threshold lead to better precision values, at the cost of reducing coverage. However, if threshold is set too high (≥ 200), values of precision do not increase anymore, while the global coverage figure falls continually. For even higher thresholds (≥ 500), precision figures actually drop, since by excluding word vectors below the threshold we are also removing correct word synonyms of verbs that were not filtered out, leading to a decrease in precision values for these more frequent verbs.

Table 5.5: Experiment Set 3: weighting function = mutual information and context window [-2, +2]

cut.	$P^{avg}_{@}1$	$P^{avg}_{@}N$	MAP	$\mathcal{C}$
1	0.221	0.121	0.125	0.800
10	0.251	0.136	0.136	0.783
20	0.263	0.142	0.141	0.767
50	0.277	0.149	0.149	0.736
100	0.288	0.154	0.154	0.695
200	0.297	0.155	0.155	0.632
500	0.297	0.146	0.146	0.507
1000	0.290	0.141	0.141	0.398
2000	0.294	0.140	0.141	0.300

.

Results shown in Table 5.6 suggest that automatic evaluation underestimates performance. This is due mostly to the low *recall* of the gold-standard used. Also performance achieved for V_{com} is very high. Top ranked synonyms found for V_{com} are correct most of the times. More specifically, the values of $P^{man}_{@}1$ (0.88) and $P^{man}_{@}5$ (0.71) confirm that antonyms seem not to represent such a severe problem for the case of V_{com}. On the other hand, for verbs in V_{emo}, antonyms populate the top ranked positions, and in some cases are best ranked candidate. An interesting case in V_{emo} is the verb "gostar" ("to like"), which scored 0, or close to 0 precision, at all ranks tested, despite being a very frequent verb. As expected, performance figures obtained for V_{emo} are much lower than those obtained for V_{com}. Due to the simplicity of the VSM approach we followed, the figures obtained for V_{emo} can be considered baseline values for other automatic approaches aiming at finding verb synonyms for Portuguese.

Table 5.6: Manual evaluation of sets V_{com} and V_{emo}

Group	$P^{man}_{@}1$	$P^{man}_{@}5$	$P^{man}_{@}10$	$P^{man}_{@}20$
V_{com}	0.88	0.71	0.56	0.44
V_{emo}	0.60	0.44	0.37	0.27

5.7 Conclusions

We confirmed that the weighting function chosen has a crucial impact on the performance obtained when using the VSM for finding verb synonyms in Portuguese. Results achieved by combining the cosine distance with the Mutual Information weighting function suggest that the low frequency features carry most of the information regarding verb similarity. We showed that information obtained from both sides of the verb is important for identifying possible synonyms, but the two following words seem to carry more information than the two preceding words. Also, we showed that it is beneficial to exclude word vectors with less than 50 non-nil features, but when the cutoff threshold is set too high, both precision and coverage figures will be affected. Manual evaluation showed that the performance obtained by the VSM approach varies greatly depending on the linguistic and semantic properties of the verbs at stake. Results for verbs related with communication show that the VSM approach can potentially lead to very high performance figures. Results with the much more complex class of psychological verbs related with the expression of emotion exposed the limitations of this method in coping with antonymy. Because of the almost absence of linguistic preprocessing of our approach, such results – specially $P_{@}1 \simeq 0.60$ and $P_{@}5 \simeq 0.45$ – can be seen as *baseline* values for the task of automatically finding verb synonyms for Portuguese.

Chapter 6

Paraphrasing Job Titles

In this chapter we focus on additional issues related to *content-similarity*, which, again, have been motivated by practical problems encountered in an *entity-tracking* task. Entity-tracking consists in identifying all references made to *Named Entities* (NE's) within a certain text, including not only those references made by *name* but also those made by different sequences, such as *job titles*. However, since job titles might be relatively long *noun-phrases*, they usually have *multiple paraphrases*, which greatly complicates the task of identifying them robustly. Generically, job titles paraphrases can be generated by performing *chunk substitutions* using (i) *synonyms*, (ii) *equivalent formulations* (e.g. *acronyms*) or (iii) alternative formulations with different degrees of *specialization* (e.g. *hypernym noun-phrases*). In this sense, the *content-similarity* situations that we are addressing in this chapter are more generic than the ones presented in the previous chapter, which focused on one relation only (synonym), and on one particular class of words (verbs).

Robust entity tracking requires being able to match all these possible job title paraphrases for each entity being tracked. In practice, this involves generating all possible valid paraphrases for the entities at stake in order to be able to match them in text. The problem is that it is obviously not feasible to *manually* create rules for generating all possible job title paraphrases. Such rules would not only be *too many*, but they would also require the knowledge about certain equivalence relations (including synonymy and hyponymy/hypernymy relations) that are usually *domain-specific*, and can sometimes even be *highly metaphorical*.

We use a data-driven method for *inferring* such *domain-specific substitution rules*. The method takes as input a set of known job titles paraphrases for several entities. For each entity, we *pair* the corresponding job titles that have *partial overlap* and extract the *differences* between them. We expect some of these different substrings to be interesting *short paraphrases*. For finding those that are actually *short paraphrases* and, from them, inferring valid *substitution rules*, we use a well-known data mining algorithm: the *Apriori* rule inference algorithm.

Again, evaluation is a challenging issue, since no gold standards exist for assessing the validity of this type of rules. Therefore, we perform two types of evaluation procedures. First, multiple judges *manually* check the rules inferred to test their *correctness*. Then, we use the rules obtained to generate paraphrases of job titles of entities frequently mentioned in news, and we test the validity of such paraphrases over a corpus of news.

6.1 Introduction

Many Information Extraction (IE) applications focus on monitoring on-line news sources to find relevant information (events, quotations, opinions) related to specific human entities. Named entity recognition and resolution are crucial pre-processing tasks, identifying relevant (mentions to) entities in text and then associating them with a real-world referent. This involves processing both *direct mentions*, such as proper names (e.g. "Obama"), and *indirect mentions*, such as *job titles* (e.g. "the U.S. President"). Processing indirect mentions is particularly challenging because they are usually compositional constructions, allowing a wide range of lexical and syntactic variations. In some cases such variations follow a more or less predictable structure. For example, adjectives of nationality can be paraphrased by a noun phrase headed by the correspondent locative name (e.g. Barack Obama is frequently mentioned by "the (American|US|U.S.) (president|leader)" or "the president of the (US|United States)). Nevertheless, in some cases variations might involve some *domain-specific* knowledge (e.g. "the 2009 Nobel Peace Prize winner", for mentioning the president Barack Obama) and the use of commonly accepted metaphors (e.g. mentioning a political party by a *color*, or a soccer team by a *nickname*).

A basic approach to deal specifically with job title variation would consist in compiling extensive lists of job titles for each relevant entity, in order to build a comprehensive database of possibilities. However this does not ensure that *all* possible job title paraphrases are covered for *all* entities and it does not allow generalizing the process through which paraphrasing job titles is achieved. More specifically, it does not allow to generalize the process through which paraphrasing job titles is achieved. Thus the knowledge about a large number of valid job paraphrases for one specific entity can not be used for paraphrasing the job titles of other entities. For example, if a given entity has the job title "president of X", and if we know of several other valid job titles for the same entity that are based on paraphrasing "X", then it should be possible to apply such knowledge to obtain valid job titles for another entity who is known to be "vice-president of X".

Hence, we propose an unsupervised method for inducing sets of *substitution rules* that govern the generation of paraphrases for job titles. Our method takes as single input a large list of tuples of the form $\langle$*entity_name*, *job_title*$\rangle$. We assume that if there are several job titles associated to a *specific* entity, then they are supposed to be top-level paraphrases of each other (i.e. they are different formulations of the same job title). Thus, using the *Apriori* algorithm [AS94], we try to infer *word* or *phrase* substitution rules of the form X ⇒ Y, that allow transforming these job descriptions into their corresponding paraphrases. The substitution rules we seek to infer are *not* necessarily symmetric, that is, they do not need to relate two *synonyms*: they might relate hyponyms/hypernyms, allowing valid job title generalizations. As we will show, this method makes it possible to infer hundreds of correct domain-specific substitution rules. This leads to the generation of previously unknown job title paraphrases for a significant number of entities contained in the input set. We will also demonstrate that the new job titles generated are both correct and productive, by checking the number of occurrence of pairs ⟨*entity_name*, *new_job_title*⟩ in news corpora.

6.2 Related Work

Our work has connections with research on (i) for automatic detection of paraphrases in news sources, (ii) finding alternative names with different levels of formality, and, indirectly, (iii) matching name variations and performing name normalization.

Shinyama et al. present an approach for automatically acquiring *domain-specific* paraphrastic expressions from news articles [SSS02]. Their method consists in trying to match sentences found in articles from different newspapers but covering the same topic. The key assumption is that references to named entities (people, locations, values, etc.) are expected to be preserved among paraphrastic sentences. Therefore, two sentences mentioning the same entities, but belonging to different news articles covering the same topic, have a certain likelihood of being paraphrases. For each paired sentence a dependency tree is computed and, if the number of common named entities in certain parts of each tree exceeds a certain threshold, then such parts are considered paraphrases. The authors experimented their method on a corpus containing news published during one year in two Japanese newspapers: 294 pairs of articles describing *arrest events* and 289 articles describing *personal affairs* (hiring and firing of executives) were retrieved. After running an information extraction system on these texts for finding relevant patterns, they were able to automatically match 136 pairs of potential paraphrases. Manual evaluation showed that this method achieves 49% precision in finding paraphrases for the arrest events and 94% for personal affairs events. One of the reasons for the worse performance in finding paraphrases for arrest events was the large number of sentences describing these events involving one NE (e.g. "*NE was arrested*"). Such sentences end up being matched to other sentences that mention the same entity but do not convey comparable information. Ironically, another limitation of this system is that it is not able to deal with paraphrases related with the names of the entities such as “New York City”, “NYC” or “the city”.

Barzilay and Lee propose a more general procedure for finding *sentence-level* (i.e. long) paraphrases [BM01]. The procedure has three steps and involves comparable corpora. For each corpus they obtain lattices (graph representations) from groups of structurally similar sentences. The lattices are obtained by first clustering sentences reporting similar events (based on n-gram overlap and substituting named-entities by generic tokens) and then finding multiple sequence alignments, which expose commonalities. These lattices can then be transformed into patterns taking into account the variability found for each possible slot: a large variability in a given section of the lattice will correspond to an argument in the pattern. The next step consists in pairing the lattices found in the two comparable corpora. Two lattices are matched if by assigning the same parameters to the corresponding slots they instantiate sentences from news of the *same day* on the *same topic* (yet belonging to different corpus). Generating paraphrases becomes then straight-forward: each path of one of the matched lattices can be paraphrased to any of the paths of the other. The system was evaluated on a pair of comparable news corpora: Reuters and Agence France-Presse (AFP). Among the 43 slotted lattices found for Reuters and 25 for AFP it was possible to match 25 lattices. These 25 matched lattices allowed the generation of 6,534 paraphrase pairs. Then, the authors adapted the DIRT system ([LP01]) for producing paraphrase pairs on the same data set. The top scoring 6,534 pairs where selected and a sample of 500 paraphrases from both systems was manually evaluated by 4 judges. Results show that the system proposed by the authors outperforms DIRT by 38% in generating correct paraphrase pairs. The proposed system was also compared against a baseline system that generates paraphrase by performing substitution using synonym information from WordNet. Results achieved revealed that the proposed system consistently outperforms the baseline showing that it is able to generate paraphrase that go beyond simple synonym substitutions.

Elhadad and Sutaria [ES07] propose a method for matching technical terms with lay equivalents. The authors start by aligning 367 news stories summarizing research in the field of coronary diseases with the abstracts of the technical papers that report such research (this process required some manual intervention since, although *mentioned* in the news piece, the papers are not *linked*). Term identification on both news and abstracts was made with the support of the UMLS ontology (only terms related to diseases and therapies were considered). Then for all pairs of technical terms (i.e. terms occurring in at least one paper abstract) and lay terms (i.e. terms present in at least one news story) contingency statistics based on the co-occurrence of such terms in aligned documents were computed. For matching the equivalents, the authors used a combination of three different *association measures* (χ^2, λ and odds-ratio) and a filter based on the semantic type of each concept as provided by the UMLS (two terms can only be equivalent if the semantic types are the same). Evaluation against a gold standard set for all possible pairs of equivalents revealed that the method could match the correct equivalent with a precision of 57.9% and a recall of 77%, thus clearly outperforming the baseline method (equivalent pairs directly provided by the UMLS for the terms under evaluation).

Ananthanarayanan et al. *manually* define a set of rules for generating synonyms / paraphrases of known named entities, in order to be able to match such entities with potential variation found in technical documentation [ACDK08]. The proposed rules try to cope with several spelling variations, sub-entity/super-entity mentions and abbreviations. The results obtained show a significant improve in the *recall* of entity recognition. Jijkoun et al. [JKMdR08] start by presenting a baseline name detection and normalization method for *noisy user-generated content* that combines a series of in-document heuristics (e.g. trying to find in the document a candidate for the complete name given a possible last name), with attempts to match the name with Wikipedia titles. Then additional heuristics were added to the method such as, for example, (i) attempting soft-matches against a list of known entities to identify name variations (using Levenshtein distance), and (ii) using the text of anchors in Wikipedia that sometimes link incomplete versions of a name to the page that refers the normalized entity name. These additional heuristic led to improvements in precision, recall and accuracy of name recognition / normalization over the baseline method.

As Barzilay and Lee, we wish to automatically generalize paraphrase matching by learning domain-specific *substitution rules*, in order to make possible to generate paraphrases combinatorially. In spite of seeking relatively short paraphrases, as in Elhadad and Sutaria, our approach does not require any manual intervention, and is not dependent on any external knowledge resources. We wish to follow, as much as possible, a data-driven approach and avoid manually defining rules for identifying potential paraphrases, as done in Ananthanarayanan et al.

6.3 Method Description

The method we propose takes as input a list of pairs of the form $\langle n_i, t_{ij} \rangle$, where n_i is the *canonical name* of a certain entity and t_{ij} is a *distinct* job title for that entity. By canonical name, we mean a *two name* expression such as “Hilary Clinton” or “Lady Gaga”. The core of the method lies in the observation that each entity may have several valid job titles, and most of them are likely to be mutual paraphrases. For now, we assume that names of entities in the input list are *not* ambiguous, i.e. that there are no two entities sharing the same canonical name. We thus assume that there are not two *radically different* job titles under the same name, each one referring to two different entities possessing the same name. Although this might look like a very strong simplification at first, in practice, the vast majority of the entities that are *frequently* mentioned in the news are *not* ambiguous when referred by *the canonical name* (i.e. an expression with at least two names), despite some notable exceptions (e.g. "George Bush"). Our method, however, does not require the existence of a single name per entity (the opposite situation to ambiguity). If one entity is frequently mentioned by two different canonical names, which can happen due to different transliterations of the same name (e.g. "Viktor Yushchenko" vs. "Victor Iuchenko") or because some middle name is optionally used (e.g. "George Bush" vs. "George W. Bush"), this will not represent much of a problem to the mining process, if *enough* valid job titles are known for each of those possible names.

6.3.1 Finding Potential Short Paraphrases

The algorithm for finding potential short paraphrases is relatively straightforward. For each name n_i, we first group all (distinct) job titles, t_{ij}, to obtain the list T_i:

$$\langle n_i, T_i \rangle = \langle n_i, [t_{i1}, t_{i2} ... t_{in_{max}}] \rangle \tag{6.1}$$

where T_i is the list of *top-level* paraphrases from which we obtain shorter paraphrases that will allow us to infer a set of substitution rules. From T_i we proceed by generating P_i, the list of all possible combinations of *pairs* of distinct job titles for the name n_i:

$$T_i \longrightarrow P_i = [\langle t_{i1}, t_{i2} \rangle, ... \langle t_{i1}, t_{in_{max}} \rangle, \langle t_{i2}, t_{i3} \rangle ... \langle t_{i2}, t_{in_{max}} \rangle, ...] \tag{6.2}$$

For all pairs of job titles we look for *partial alignments* between *distinct* substrings of each job title. This is done by first identifying the *longest common substring* between each pair of job titles. Let t_1 and t_2 be two paired job titles. Without any loss of generality, we can represent these two job titles as $t_1 = L_1 C R_1$ and $t_2 = L_2 C R_2$ with C being the longest common substring between t_1 and t_2, and L_1 / L_2 and R_1 / R_2 being the corresponding substrings positioned at left and right side of C (C can be an empty string, and if not, any of L_1, L_2, R_1 and R_2 can also be empty). For example, let t_1 be "recently elected president of the U.S." and t_2 be "president of the United States of America." The longest common substring, C, will be "president of the". For t_1 we will have L_1 = "recently elected" and R_1 = "U.S.", while for t_2 we will have L_2 = "" and R_2 = "United States of America".

For producing potentially interesting substring alignments (i.e. those that can yield shorter paraphrases), we have to consider three possible cases (see Figure 6.1):

(a) parallel alignments - the substrings on the left side (L_1) and right side (R_1) of the *longest common substring* from one job title (t_1) are aligned with the corresponding left (L_2) and right (R_2) substrings of the other job title (t_2). This case covers situations such as "the leader [of the] communists" and "the general-secretary [of the] communist party."

(b) cross alignments - the substrings on the left side (L_1) and right side (R_1) of the *longest common substring* from one job title (t_1) are aligned with the substrings on the opposite sides (i.e. R_2 and L_2, respectively) of the other job title (t_2). The cross alignment is useful for matching cases such as "the French [First Lady]" and "[First Lady] of France."

(c) merge alignments - a merge alignment consists in removing the longest common substring of both job titles and producing a single alignment between the concatenation of the remainder substrings. This case is intended to deal with situations such as "the Brazilian Minister [of Defense]" and "the Minister [of Defense] of Brazil."

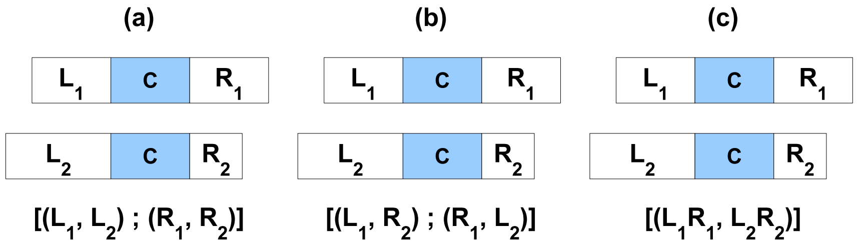

Figure 6.1: All potential segment alignments that can be derived from a pair of top level paraphrases: parallel alignments (a), cross alignments (b) and merge alignments (c).

Using these three alignment strategies, we can generate pairs of substring alignments, $\langle s_{1i}, s_{2i} \rangle$ from each pair of job titles, $\langle t_1, t_2 \rangle$:

$$\langle t_1, t_2 \rangle \longrightarrow [\langle s_{11}, s_{21} \rangle, \langle s_{12}, s_{22} \rangle ... \langle s_{1n}, s_{2n} \rangle] \tag{6.3}$$

These alignments are expected to be (potentially interesting) short paraphrases, i.e. s_{1i} can potentially be a valid substitution for s_{2i}, and vice-versa. In some cases, either s_{1i} or s_{2i} can be an empty string. This can occur if after finding a common string between two job descriptions, one of the remainder substrings – R_1, R_2, L_1 or L_2 – is empty. This happens, for instance, when attempting a parallel match between "the recently elected [President of] U.S." and "[President of] the United States", where "the recently elected" needs to be matched with an empty string (i.e. "NULL"). These cases are likely to occur when of the job descriptions being matched have a higher degree of specialization, or provide an additional, yet dispensable, piece of information.

6.3.2 Inferring Substitution Rules

Given pairs of string alignments $\langle s_{1i}, s_{2i} \rangle$, we wish to mine substitution rules. There are two possible scenarios regarding how substitutions between s_{1i} and s_{2i}

can occur:

Implication, $s_{1i} \Rightarrow s_{2i}$**:** s_{1i} is a specialization of s_{2i}. Therefore, despite some loss of information, s_{1i} can always be replaced by s_{2i}, but not the reverse. For exemple, "president" $\Rightarrow$ "leader", or "midfielder" $\Rightarrow$ "(soccer) player".

Equivalence, $s_{1i} \Leftrightarrow s_{2i}$**:** s_{1i} and s_{2i} are strict paraphrases in the sense that they convey the same information. They are always valid substitutions of each other. This includes (i) pairs of names and corresponding acronyms (e.g. "U.S." $\Leftrightarrow$ "United States") (ii) shorter versions / simplifications of the same name (e.g. "Popular Republic of China" $\Leftrightarrow$ "China") or (iii) domain-specific synonyms (e.g. "red" $\Leftrightarrow$ "communist").

Inferring implication rules from data sets is a relatively well-known problem (equivalence rules can obviously be seen as bidirectional implication rules). One of the most well studied settings is mining *association rules* of the form *whoever buys X also (very likely) buys Y* (i.e. $X \Rightarrow Y$) over basket data. Basket data is the aggregation of information regarding transactions made by buyers. Typically, the information kept for each transaction is the list of identifiers of the products bought, b_1, b_2 ... b_n and a unique identifier for the transaction, id_i, such as:

$$\begin{array}{ccc} id_1 & \rightarrow & [b_1, b_2 \ldots b_i] \\ id_2 & \rightarrow & [b_j, b_k \ldots b_l] \\ & \ldots & \\ id_x & \rightarrow & [b_m, b_n \ldots b_q] \end{array}$$

There are several algorithms for mining association rules from these data sets. We chose one of the well-known algorithms, the *Aprori* algorithm [AS94]. The application of *Aprori* algorithm to our problem is straightforward, since each pair of short paraphrases $\langle s_{1i}, s_{2i} \rangle$, obtained as explained in section 6.3.1, can be seen as one *transaction* (each substring is seen as a "product"). The *Apriori* algorithm will then infer rules of the form $s_{1i} \Rightarrow s_{2i}$ (or $s_{2i} \Rightarrow s_{1i}$). Simultaneously, the algorithm will compute the values of *support* – the number of transactions in which such rule is observable – and the value of *confidence* – the probability $P(s_{2i}|s_{1i})$ (or $P(s_{1i}|s_{2i})$). If the values of *support* and *confidence* for a given association rule are considered to be high enough, then the rule can be adopted. If both $s_{1i} \Rightarrow s_{2i}$ and $s_{2i} \Rightarrow s_{1i}$ are adopted, then we conclude that $s_{2i} \Leftrightarrow s_{1i}$.

6.4 Experimental Setup

For obtaining ⟨name, job title⟩ tuples to serve as input to our rule inference process, we mined news corpora for two specific patterns often found in news:

1. "... the [potential job title] [name] [speech act]"
   (e.g. "... the British Prime-Minister Gordon Brown announced ...")

2. "[name], [potential job title], [speech act]"
   (e.g. "... Carla Bruni, the French First Lady, said...")

where [potential job title] is a sequence of words containing at least one known ergonym (i.e. a title, a position or designation such as "minister", "soccer coach"), and [speech act] is one of 118 possible verbs related to communication (which were manually selected). These patterns have very high precision, at the cost of a relatively low recall, as confirmed in previous work. However, since we are mainly looking for high-quality ⟨name, job title⟩ tuples, this pattern matching process fulfils our needs. The news corpus consists of a database of news snippets (title plus one to three sentences) collected from the official RSS feeds of eight Portuguese news sources (news agencies, newspapers, TV and Radio Broadcast) between 11/20/2008 and 12/23/2009. The database contained about 245.200 short news from which we extracted 2615 distinct ⟨name, job title⟩ tuples for a total of 1730 names, by matching the two previously mentioned patterns. Since the names with only one job title do not contribute to the rule inference process, we only consider the names that are associated to two or more job titles: 1279 ⟨name, job title⟩ tuples for 394 names (which shows the lack of job title alternatives for the vast majority of the names extracted).

Initial experiments showed that both *cross alignments* and *merge alignments* were not productive enough and led to the inference of mostly incorrect rules. We thus decided only to use *parallel alignments* in the remainder of our experiments. By running the alignment algorithm for the 1279 job titles found for the 394 names, we generated 1781 alignment pairs (i.e. *transactions*). Running the *Apriori* algorithm and imposing *no* threshold on the minimum amount of support and confidence we obtained 2234 rules. We then tried to select a smaller set of rules that combined a reasonable value of support (s_i) and confidence (c_i). This was done by selecting those rules, r_i, for which the value of the heuristically defined *utility function*:

$$u_i = c_i \times {s_i}^2 \tag{6.4}$$

is higher than a predefined threshold t_{min} ($u_i \geq t_{min}$). In our experiments we set $t_{min} = 0.25$ and we obtained 843 rules with supports varying from 1 to 30. Table 6.1 shows a sample of the rules inferred (and one possible translation to English[1]), ranked by the value obtained by the utility function (Equation 6.4). Rules of the type "X ⇒ NULL" mean that the "X" element in a job description can simply be removed, hopefully generating a valid yet more generic job description.

[1]In some cases there is no direct translation to English, so we try to provide an expression that conveys similar meaning. We will adopt this strategy in all other situations where a translation to English is required.

Table 6.1: Sample of the 843 substitution rules selected, ranked by the value given by the utility function $u_i = c_i \times {s_i}^2$.

#	u_i	s_i	c_i	substitution rule
1	14.2	16	0.94	"eleições europeias" ⇒ "europeias" ("European elections" ⇒ "European (elections)")
2	10	10	1	"cabeça de" ⇒ "líder da" ("head of (list of candidates)" ⇒ "leader of")
3	10	10	1	"ainda" ⇒ NULL ("in exercise" ⇒ NULL)
4	8	8	1	"da América" ⇒ NULL ("of America" ⇒ NULL)
5	7.74	30	0.51	"às eleições europeias" ⇒ NULL ("to the European elections" ⇒ NULL)
6	7.6	14	0.74	"social-democrata" ⇒ "do PSD" ("Social Democratic" ⇒ "from/of PSD")
7	7.11	16	0.67	"europeias" ⇒ "eleições europeias" ("European (elections)" ⇒ "European elections")
8	7	7	1	"Municipal de Lisboa" ⇒ "de Lisboa" ("Municipal of Lisbon" ⇒ "of Lisbon")
9	7	7	1	"Estados Unidos" ⇒ "EUA" ("United States" ⇒ "USA")
...	...	...	...	...
36	4	4	1	"PSD/Porto" ⇒ "Porto do PSD" ("PSD/Porto" ⇒ "Porto (section) of PSD")
37	3.72	9	0.64	"secretário-geral" ⇒ "líder" ("general secretary" ⇒ "leader")
38	3.51	13	0.52	"do PS" ⇒ "socialista" ("of/from PS" ⇒ "socialist")
...	...	...	...	...
129	1.68	3	0.75	"brasileiro" ⇒ "do Brasil" ("Brazilian" ⇒ "of/from Brazil")
130	1.68	3	0.75	"do Brasil" ⇒ "brasileiro" ("of/from Brazil" ⇒ "Brazilian")
...	...	...	...	...
219	1	1	1	"treinador do F. C." ⇒ "técnico do F.C." ("coach of F. C." ⇒ "manager of F.C.")
220	1	1	1	"extremo" ⇒ "futebolista" ("forward" ⇒ "soccer player")
...	...	...	...	...
842	0.27	3	0.3	"de Portugal" ⇒ "português" ("of/from Portugal" ⇒ "Portuguese")
843	0.27	3	0.3	"CDS-PP" ⇒ "CDS/PP"

6.4.1 Evaluation

As pointed out by Callison-Burch et al., there is no consensus regarding the best methodology for evaluating the quality of paraphrases, and, thus, of rules or of systems that generate them [CBCL08]. Due to the lack of gold-standard data, such evaluation is performed manually most of the times, with all the corresponding limitations in terms of objectivity, consistency and repeatability. On the other hand, it is extremely difficult to build gold-standards for paraphrase evaluation, because it is infeasible to ensure that all the correct paraphrases for a given expression are contained in the gold-standard. Additionally, since paraphrases are usually domain-specific, an hypothetical gold-standard for one domain would be of limited applicability in other domains.

Therefore, in order to reduce the impact of the previously described methodological limitations, we performed two types of evaluation. First, we evaluated the substitution rules that were automatically inferred by *manually* checking their correctness. Then, we assessed the validity and usefulness of the *new* job titles that can be generated using such rules, by automatically checking their productivity in the news corpus.

Evaluation of Substitution Rules

For evaluating the correctness of the rules, three judges manually checked each of the 843 rules selected (see Equation 6.4). Since the *Apriori* algorithm generates association / implication rules (i.e. $s_{1i} \Rightarrow s_{2i}$), we evaluated each rule according to three possible cases:

1. *correct* (C): s_{1i} can generally be safely substituted by s_{2i} in the context of job title paraphrases;

2. *possible* (P): s_{1i} can be substituted by s_{2i} in some specific cases, but such rule is not safely generalizable. We also included in this category substitutions that involved spelling mistakes (e.g. "president" ⇒ "persident"), since these idiosyncrasies were part of the original dataset;

3. *incorrect* (I): s_{1i} can not be (safely) substituted by s_{2i}.

Two precision figures can be computed based on the evaluation provided by the judges. The first is *Strict Precision* which only considers valid *Correct* rules:

$$P_{str} = \frac{\#C}{\#C + \#P + \#I} \tag{6.5}$$

The second is *Relaxed Precision* which considers valid all *Correct* and *Possible* rules:

$$P_{rlx} = \frac{\#C + \#P}{\#C + \#P + \#I} \tag{6.6}$$

These two precision figures can be computed at different *support* and *confidence* thresholds.

Evaluating new Job Titles

For testing the validity of the new *job titles*, we first generated a set of new job titles for each name by paraphrasing known job titles with the previously found rules. The new job titles were generated using a subset of 70 rules with support equal or larger than 5. For each name, we excluded all job titles that included or were included in any of the previously known job titles for the corresponding name. We thus tried to generate job titles that were more than mere *trivial variations* of the known job titles. Under such conditions we were able to generate 1686 new job titles for 890 names (notably, the rules were inferred from data obtained from *only* 394 names). For each name, n_i, we tested the productivity of the corresponding new job titles, t_{ij}^{new}, by trying to match both n_i and t_{ij}^{new} *simultaneously* on the corpus. We performed such test on the corpus of 245.200 short news. In principle, if a match of both n_i and d_{ij}^{new} occurs within such a relatively short text passage (usually less than 300 chars), d_{ij}^{new} is likely to be a valid job title for n_i. Productive job titles were manually checked in the end. However, if a new job title is not productive in the corpus, it does not necessarily imply that it is incorrect. In any case, since we are essentially trying to determine the usefulness and correctness of the new job titles, this simple automatic test is adequate.

6.5 Results and Analysis

6.5.1 Substitution Rules

Each of three judges - A, B and C - separately evaluated 843 cases, following the guidelines described in the previous section. Inter-annotator agreement achieved was quite high. The Cohen's Kappa Coefficient (for three classes) between judges A and B was 0.96, between B and C was 0.92, and between A and C was 0.95. Table 6.2 contains the values of (averaged) *Strict Precision*, P_{str} (see Equation 6.5), and (averaged) *Relaxed Precision*, P_{rlx} (see Equation 6.6), for different values of minimum support.

Not surprisingly, the higher the support the higher the values of both P_{str} and P_{rlx}. Also, for relatively high values of support, the difference between P_{str} and P_{rlx} tends to decrease, since the number of "dubious cases" (i.e. those that end up being evaluated as "Possible") also tends to decrease. Generically, the precision figures obtained for support larger than one are relatively high, ranging from 0.69 to 0.86. In fact, the vast majority of the rules are correct, including not only relatively straight-forward rules, but also some non-trivial cases.

Table 6.2: *Strict* (P_{str}) and *Relaxed* (P_{rlx}) Precision for the rules obtained, at different thresholds on the corresponding minimum support value.

s_{min}	# rules	P_{str}	P_{rlx}
1	843	0.63	0.75
2	220	0.70	0.81
3	126	0.71	0.83
4	89	0.72	0.82
5	70	0.73	0.82
6	58	0.69	0.80
7	40	0.75	0.83
8	36	0.75	0.81
9	31	0.77	0.85
10	27	0.81	0.86

Some rules are very productive in Portuguese, and involve syntactic-semantic strategies, such as the ones listed below:

- Reduction of specific modifiers, such as adjectives (e.g. *"**actual** seleccionador nacional"* ⇒ *"treinador da selecção"* / *"**current** national team coach"* ⇒ *"national team coach"*; "treinador da selecção argentina" ⇒ *"seleccionador"* / "coach of the Argentina national team" ⇒ *"coach"*). More complex expressions, such as phrases, can also be removed resulting in a job title generalization (e.g. *"presidente **ainda em exercício**"* ⇒ *"presidente"* / *"**the still in office** president"* ⇒ *"president"*).

- Replacement of relation adjectives by equivalent noun phrases introduced by preposition, namely the ones involving political affiliations (e.g. *"democrata-cristão"* ⇒ *"do CDS-PP"* / *"Christian Democrat"* ⇒ *"from CDS-PP"*; *"cabeça de lista do **PSD**"* ⇒ *"cabeça de lista **social-democrata**"* / *"main candidate from **PSD**"* ⇒ *"main **social-democratic** candidate"*) and nationality (e.g. *"dos Estados Unidos da América"* ⇒ *"norte-americano"* / *"of the United States of America"* ⇒ *"North-American"*).

- Movement of constituents (e.g. *"[candidato] do Bloco de Esquerda **às europeias**"* ⇒ *"[candidato]**às eleições europeias** pelo BE*" / *"[candidate] of Bloco de Esquerda (party) **to the Europeans**"* ⇒ *"[candidate] **to the European elections** for the BE (party)*").

- Association of acronyms with respective expansion (e.g. *"[Fe]deração [N]acional dos Sindicatos dos [Prof]essores ⇒ FENPROF"*).

- Substitution of words or expressions that are synonyms in Portuguese (e.g. *"primeiro ministro"* ⇒ *"chefe do governo"* / *"prime-minister"* ⇒ *"head* *of government"*; *"**antigo** ministro"* ⇒ *"**ex-**ministro"* / *"**former** minister"* ⇒ *"**ex-**minister"*; *"candidato "número um""* ⇒ *"cabeça de lista"* / *"first candidate"* ⇒ *"top candidate in the list"*). Some constituents, which may be not equivalent when found in general compositional structures, can be synonyms within the scope of particular job titles (e.g. *"[candidato] às europeias"* ⇒ *"[candidato] ao Parlamento Europeu"* / *"[candidate] to the European (elections)"* ⇒ *"[candidate] to the European Parliament"*). Partial synonymy is also productive, and often involves the replacement of a word or expression by a possible hyperonym (e.g. *"presidente do Governo"* ⇒ *"líder"* / *"president of Government"* ⇒ *"leader"*; *"parliamentarian of PS who was responsible for the final report"* ⇒ *"report author"*).

The linguistic strategies previously described can be used in combination, resulting in paraphrases that would be difficult to identify by using simple rule-based approaches. The most interesting cases involve the relation between free constructions, whose interpretation requires up-to-date knowledge of the world and society, as illustrated by the following examples:

1. *"[Secretária] de Estado da nova administração de Barack Obama"* ⇒ *"[Secretária] norte-americana de Estado"* / *"[Secretary] of State of the new Barack Obama's adminstration"* ⇒ *"North-American [Secretary] of State"*

2. *"cabeça de lista do Movimento Esperança Portugal às [europeias]"* ⇒ *"candidata independente do MEP às eleições [europeias]"* / *"head of the list (of candidates) of Movimento Esperança Portugal to the European (elections)"* ⇒ *"independent candidate of MEP to the [European] elections"*.

In the first example, the constituent of the compound noun "secretária de Estado" ("secretary of State") is discontinuously mentioned and the noun phrase "nova administração de Barack Obama" ("new Barack Obama's adminstration") is replaced by the adjective "norte-americana" ("North-American"), roughly equivalent within this particular context. In the second example, the noun "candidata" ("candidate") and "cabeça de lista" ("head of the list (of candidates)" $\simeq$ "main candidate") are synonyms, despite not being possible to establish this symmetric relation in general circumstances.

There are also some typical cases of errors. One frequent error arises from the inference of rules of the type *organization A* ⇒ *organization B* (and vice-versa). These rules are inferred as a result of job changes, which are quite frequent in some domains (e.g. sports or finance). For example, if a soccer manager and a few players move from one team to another during the time period covered by our input data set, then we will probably have tuples referring to these two teams in the list of job titles for the corresponding names, such as $\langle n_i$, "manager/midfielder of Team A"$\rangle$ and $\langle n_i$, "manager/midfielder of Team B"$\rangle$. Our method will thus possibly derive "Team A" ⇒ "Team B" and "Team B" ⇒ "Team A". This

Table 6.3: Statistics regarding simultaneous matches of names and new job titles, $\langle n_i, t_{ij}^{new} \rangle$, and also names and previously known job titles.

	$\#_n$	$\#_{(n,t)}$	$m_{(n,t)}$
$\langle n_i, t_{ij}^{new} \rangle$	130	175	1609
$\langle n_i, t_{ij}^{old} \rangle$	890	1566	33147

suggests that temporal information should be taken into account when pairing input job titles (see Equation 6.2). One relatively simple alternative is to avoid pairing input job titles that were extracted from news that are separated by more than a given amount of time (e.g. a week). Another common error, which occurred specially in rules with low support, was the inference of bidirectional rules of the form "specific_case $\Leftrightarrow$ general_case" when only one of the direction was valid (namely the one that states "specific_case $\Rightarrow$ general_case"). We believe that the solution for this type of errors will be achieved by increasing the quantity (and variety) of input data, which will allow collecting enough evidence to exclude the incorrect cases.

6.5.2 Evaluating new Job Titles

Table 6.3 shows statistics regarding the simultaneous match of both the name, n_i, and the new job title t_{ij}^{new}, against the corpus of short news. We had previously obtained 1686 new job titles. The table contains information about (i) the number of names for which a match and a new job title was found, $\#_n$, (ii) the number of distinct $\langle n_i, t_{ij}^{new} \rangle$ matched, $\#_{(n,t)}$, and (iii) the total number of matches in the corpus for all the $\langle n_i, t_{ij}^{new} \rangle$ combinations, $m_{(n,t)}$. For comparison purposes, we performed the same matching procedure using the initially known job titles (i.e those that where used as input for the inference process), t_{ij}^{old}, for each of 890 names under test (for which new job titles could be generated using rules with support ≥ 5).

About 10.4% (175 in 1686) of the new $\langle n_i, t_{ij}^{new} \rangle$ tuples were productive on this corpus. This represents 11.2% of the total number of *previously known patterns*, $\langle n_i, t_{ij}^{old} \rangle$, that were also matched (1566). The number of matches produced using the newly found job titles was 1609, while the number of matches obtained using the previously known patterns, which are supposed to be the most frequently used, was 33147. Basically, we were able to increase our ability to detect references to entities by about 10.4% in number of *distinct* job titles and about 4.8% in number of matches.

Another relevant fact is that at least one new job title was productive for 130 names, i.e. about 14% of the names. For some entities the number of new job titles found was quite high. For example, there were 10 new valid job titles for *Vital Moreira*, who was the "socialist main candidate for the 2009 European Elections." Our input data contained 18 possible job titles, including others not related to the elections. The 10 new productive job titles were generated as a result of having inferred rules for substituting many sub-sections of the complete job title (i.e. for "socialist", for "main candidate" and for "2009 European Elections"). As a rule of thumb the number of productive job titles generated is proportional to the relevancy/coverage of the entity at stake and to the length (in words) of its previously known job title.

Manual evaluation of the 175 productive $\langle n_i, t_{ij}^{new} \rangle$ tuples showed they were *all correct* (see Table 6.4 for some illustrative examples) except for 3 cases. For *Barack Obama*, *Hugo Chávez* and *Lula da Silva,* one of the productive job titles found was "leader", which was generated from one of the most frequently matched rules we inferred: "president" $\Rightarrow$ "leader." Although not incorrect, we believe that this job title is too generic to be useful in finding additional references of each of these entities, since they can be used to refer any other leader that might be present in the context. This is partially shown by the relatively high number of matches found. This type of generic job titles can be filtered out in a subsequent step by checking the number of different names for which they were generated. If many entities have such a job title, then it is probably too generic and should be excluded or marked as so.

6.6 Conclusions

We presented an unsupervised method that takes as input a list of valid ⟨name, job title⟩ tuples and infers substitution rules for paraphrasing known job titles into new and valid ones. We showed that the rule inference process is capable of obtaining hundreds of correct rules with precisions over 70%. We also showed that the subset of *productive* job titles generated from these rules is almost 100% correct, and leads to a 10% increase in the overall number of distinct job titles that were previously known. Although our experiments have been developed in the Portuguese language, we believe that our method can be easily ported to other languages, with similar results.

Table 6.4: The list of new job titles (t_{ij}^{new}) found to be productive for a sample of names. The number of matches for each job title is shown in round brackets.

name (n_i)	new job title paraphrase matched (t_{ij}^{new})
Alex Ferguson	técnico (11) (manager) técnico do Manchester United (1) (manager of the Manchester United)
Barack Obama	**líder** (76) (**leader**) líder dos Estados Unidos (1) (leader of the United States) líder dos Estados Unidos da América (1) (leader of the United states of America) líder dos EUA (1) (USA leader) líder norte-americano (3) (North-American leader)
Eric Holder	procurador-geral dos EUA (4) (Attorney General of the USA) procurador-geral norte-americano (6) (North-American Attorney General)
Gerry Adams	presidente do Sinn Féin (1) (president of the Sinn Féin)
Malam Bacai Sanhá	presidente da Guiné-Bissau (20) (president of Guinea-Bissau)
Max Mosley	líder da FIA (3) (leader of FIA)
Robert Gates	secretário da defesa dos estados unidos (2) (Secretary of Defense of the United States)

Part III

Names and Entities

Most of the contents on the Web are not about *abstract concepts* nor about *deep theoretic formulations*: they are about everyday *entities*, specially about *people*, *locations* and *organizations*. Undoubtedly, practically all news content focus essentially on these types of entities, so an important function of any *news monitoring* application is actually processing the *names* of (such) entities. But, if we widen the scope of what an entity can be, to include types of entities other than just the more traditional ones (i.e. people, locations and organizations), then one must recognize that virtually every facet of the Web is *essentially* about entities. For example, web-commerce sites are about *products*, which have names. Media content sites are about *movies*, *music tracks*, *bands*, which also have names. If we look at more recent web phenomena, we can see that blogging and micro-blogging environments provide platforms for people expressing opinions about *anything* from *software components* to *biking parts*, while social network sites connect *people* that share information about common interests, whatever these may be. All these have names.

Automatically processing mentions of entities is very challenging, particularly in the Web environment. There are many reasons for this, namely:

1. The universe of possible *entities* to be tracked is *extremely large* and is always *expanding*, contents regarding *new* entities are always being produced. Additionally, the set of types in which such entities can be classified is also very large and expanding.

2. The set of *names* used for referring those entities is also extremely large, and only partially known, which makes *dictionary-based approaches* infeasible.

3. The relation between names and entities is far from being a simple 1-1 mapping: a single entity can be mentioned by several different names, and the same name can be used to mention an indefinite number of entities, possibly of different types (name *ambiguity*).

4. The mapping between entities and the corresponding set of names used to mention them is also very *dynamic*, as people tend to use *metaphors* and other linguistic phenomena (e.g. *metonymy*) to produce (instant) alternatives for mentioning entities.

5. Names, especially names of foreign entities and commercial products, are sometimes hard to spell. The amount of variations that such names can suffer is significant, even in *industrial media* (e.g. on-line news). In *user-generated contents* the problem is even worse since standard spelling conventions, such as capitalization, are not followed. Thus, the apparently trivial task of robustly *identifying* names is not a simple one.

In this part, we try to contribute to the solution of some of these problems. In chapter 7, we focus on the development of a technique for finding *type similar* entities, i.e. those that can be said to belong to the same type (or class). This problem is important for gazetteer creation and named-entity recognition: knowing that an unknown entity is *type-similar* to another known entity (e.g. a soccer team) allows us to tag its type based on such similarity. We propose a *data-driven* approach to infer the degree of type similarity between entities. The method uses information about how entities co-occur in *coordination structures*, which can be obtained using very simple methods.

In chapter 8, we propose a solution to the Named-Entity Disambiguation (NED) problem on the Web: do two occurrences of the *same name* in *different* Web documents refer to the *same entity* or not? This is an extremely important question in the scope of Web document retrieval in general, and in *entity tracking* applications in particular, especially when *merging* information from different documents is required. We propose a clustering-based approach to the NED problem, in which we try to group mentions of the same name that exhibit enough (distributional) similarity so as to be considered to refer to the same *entity*. Since we are trying to tackle named-entity disambiguation in the Web, several algorithmic issues need to be addressed in order to allow efficient scaling of our method to such a large data set.

In both works, we follow data-driven approaches and we use the Vector Space Model as the main framework for computing similarity. Also, in both works, evaluation is performed using gold-standard information extracted from Wikipedia.

Chapter 7
Finding Type Similar Entities

In this chapter[1] we focus on *type similarity* (see section 2.3.2). Our goal at this stage is to develop a corpus-based mechanism for inferring the degree to which a set of entities can be considered type similar, or, alternatively, the degree to which they belong to the same class. The core of this mechanism is a *membership function* (see Equation 2.11), which allows us to grow a set of type similar entities, given an initial set of seeds that represent (by extension) a target class. The membership function we propose relies on information about the co-occurrence of *names* in corpora in the scope of *coordination structures*. Our hypothesis is that coordination structures tend to relate entities of the same type (e.g. "I visited [Rome] *and* [Paris]") and, thus, provide the information needed for inferring type similarity. Under certain conditions, coordination structures are relatively simple to identify from corpora (i.e. they are *partially observable features* as described in section 3.3.3), which allows an easy deployment of out method in a practical application scenario, such as, for example, the semi-automatic construction of gazetteers.

The expansion method itself is based on the Vector Space Model. Having extracted features for representing name/entity items as vectors, the membership function is computed by calculating distance between two vectors: the one referring to the items being tested, and the one representing the reference class (by aggregating the seed elements). For evaluating our method we use information taken from Wikipedia, namely lists that aggregate entities that are supposed to belong to some class (e.g. "*List of English novelists*"). We evaluate our method by choosing a couple of elements from the gold-standard list to serve as seeds, and then checking how much of the initial list can be reconstructed using the expansion mechanism proposed. The evaluation scheme we used is one of the major contributions of this work.

[1]The material contained in this chapter was published in *Luís Sarmento, Valentin Jijkoun, Maarten de Rijke, Eugénio Oliveira. ""More like these": growing entity classes from seeds"* [SJdRO07]

7.1 Introduction

Lexical resources such as thesauri and ontologies form essential ingredients of many intelligent information access applications, including question answering, digital libraries, and cross-lingual search engines. Creating, maintaining, and expanding such lexical resources by manual means is a tedious, time-consuming and expensive task. As a consequence there has been a lot of research on finding methods for automatically extending lexical resources. Traditionally, a significant portion of this line of work has focused on learning taxonomic relations and constructing semantic hierarchies; see e.g. [Car01, CW03, GBM03, RC98, SJN04].

We focus on a different, yet related problem in the setting of lexical acquisition: extending classes with *type-similar* items. We are aiming specifically at expanding sets of entities. Given a set of *seed* lexical items referring to entities, $E^s = \{e^s_{\sigma_1,\rho_1}, e^s_{\sigma_2,\rho_2}, ...\}$ that are supposed to belong to some unknown class c_i, and a set E^c of candidate entities, $E^c = \{e^c_{\sigma_1,\rho_1}, e^c_{\sigma_2,\rho_2}...\}$, we wish to determine which of the entities in E^c belong to class c_i[2]. In other words, we want to "grow" a class c_i from a few seed examples, E^s, by choosing elements from E^c. An example is provided by the seed set {*Raphael, Michelangelo, Leonardo da Vinci*}, which might be expanded so as to include for example *Sandro Botticelli*.

We propose and evaluate one corpus-based method that uses a small seed set obtained from the user and "grows" those seeds into sets of similar entities, together forming the extension of some implicit concept. The actual extension depends on the specific corpus used, since it may contain different entity information. Therefore, the previous seed set could be expanded with other non-Italian yet Renaissance, artists such as *El Greco* or *Jan van Eyck*. One advantage of our method is that it can be deployed on arbitrary text collections, almost regardless of the language. Another important and original contribution in this chapter is the innovative evaluation framework for this type of task, based on list data from Wikipedia.

7.2 Related Work

Automatic and semi-automatic extraction of (typed) lexicons from free text has received a good deal of attention in the natural language processing community. For example, Roark and Charniak [RC98] describe a corpus-based method for expanding a nominal category from a small set of exemplar "seed" words. The method selects new members of the category by looking how often they co-occur with the seed words. The co-occurence statistics is based on noun conjunctions, lists and appositives. Unlike in our work, though, [RC98] focuses on common nouns ("*car*", "*pickup trucks*") and does not report results for named entities.

[2]The notation used for identifying entities (e_{σ_i,ρ_i}) is intended to be compatible with the notation presented in chapter 2.

A similar bootstrapping approach, due to Thelen and Riloff [TR02], relies on a large body of extraction patterns that capture information about behaviour of a word. For example, such patterns as "*X was arrested*" or "*murdered X*" are likely to extract candidates of the category *People*. A word is considered a good candidate to include in a category if it is extracted by patterns that also have a tendency to extract known category members. This bootstrapping method requires a large set of linguistic patterns that allows it to identify entities of a specific category. It is sensitive both to the language of the corpus and the specificity of the categories.

Ghahramani and Heller propose a probabilistic Bayesian framework for the task of expanding a class from seed entities [GH05]. The method estimates probability that a candidate belongs to a (hidden) class, based on the available information. The authors compare their class expansion algorithm to Google Sets[3] and show a significant improvement.

Finally, the task we are addressing in this chapter is similar to the List Completion task [4] that is to be evaluated at INEX (Initiative for the Evaluation of XML Retrieval) [FL07] in 2006.

7.3 Expansion using Membership Functions

In principle, the task of expanding a set of seed entities, E^s, which implicitly defines a class c_i, by adding elements from a set of candidate entities, E^c, can be tackled as a binary classification problem: elements of E^c need to be classified as being of class c_i or not. However, this view on the class expansion task is problematic. First, c_i is only implicitly defined, and its size is unknown. Second, there might be many different meaningful, but distinct, classes that contain E^s. Thus, we approach the class expansion problem by making used of the *class membership functions* we defined in section 2.3.2: $\mu(e_{\sigma_i,\rho_i}, c_k, \mathcal{C})$ (see Equations 2.10. For simplification purposes, in this chapter, we ignore the role of the context, $\mathcal{C}$, in the computation of the membership function. Henceforth, we use the following more compact notation: $\mu(e_{\sigma_i,\rho_i}, c_k)$.

However, since in the class expansion task the actual class, c_k, is not known, we approximate $\mu(e_{\sigma_i,\rho_i}, c_k)$ by $\mu(e_{\sigma_i,\rho_i}, E^s)$, the latter function using E^s, the set of seeds, as a model for c_i. If, given E^s, we can compute the value of $\mu(e_{\sigma_i,\rho_i}, E^s)$ for any candidate entity $e^c_{\sigma_1,\rho_1}$ in E^c, then we can obtain the n^{th}-*expansion* of E^s by collecting the set of n elements of E^c with the highest $\mu(e_{\sigma_i,\rho_i}, E^s)$ values. The problem of class expansion of E^s given E^c, becomes thus reduced to the task of computing $\mu(e_{\sigma_i,\rho_i}, E^s)$ for all $e \in E^c$.

[3]`http://labs.google.com/sets`

[4]`http://inex.is.informatik.uni-duisburg.de/2007/xmlSearch.html`

7.3.1 Implementing a Class Membership Function

We consider the membership function as a kind of similarity function between a set of seed entities, E^S, and a single candidate entity, e^c. We propose to compute the values of the membership function using the vector space model (VSM) approach (see chapter 3). We will represent both the seed set E^s and the candidate entity e^c by a vector of numerical features, and then use standard vector distance measures, such as the cosine similarity, to compute similarity.

Based on the fact that humans easily group and list similar objects, we assume that many explicit enumerations of entities that we find in text are in fact lists of objects of similar classes. Therefore, if we can identify such enumerations in text, we may be able to gather information about class similarity between the elements contained in such lists. Our assumption (also used in, e.g., [RC98, TR02]) is that if two elements consistently co-occur in lists, they are likely to be of a similar semantic class.

This assumption may be seen as a particular case of Harris' Distributional Hypothesis [Har54], according to which words that occur in the same "contexts" tend to have "similar" meanings. In our case, the context being considered is "occurrence in the same lists," which, we argue, is closely related to type (or class) similarity.

7.3.2 Identification of Lists

Identification of lists in text may not be trivial. If we consider semi-structured text, such as HTML documents, we might find some explicit clues about lists that might enable list extraction. But even in HTML documents, lists may be expressed in tables or without any specific layout clues. For unstructured text, a complete identification of lists might require robust parsing tools since lists can be expressed in a variety of ways.

We do not aim at solving the full list identification problem for free text. Instead, we propose a simple approximation that tries to identify *pairs* of elements that belong to textual lists. We assume that lists are composed of sequences of pairs of coordinated elements which are either connected by explicit coordination elements ("*and*", "*or*" ...) or by commas. Note that this intuition has been corroborated by other studies that focus specifically on using information about coordinated words to cluster them in semantic categories [WD02].

In order to identify entity pairs that belong to lists, we look for structures like "*... e_a, e_b and e_c...*" where e_a, e_b, etc are named entities. E.g., "*I've lived in NY, Paris and Amsterdam.*" Another possibility would be "*...e_a, e_b or e_c ...*" as in "*Experience with Oracle, PostgreSQL or MySQL is required*"). When instances of such patterns are found in a corpus, we can easily conclude that the pairs (e_a, e_b) and (e_b, e_c) co-occur in coordination. Note that in our method we will not make any conclusions about e_a and e_c, since, for simplicity reasons, we are only looking for entities that co-occur contiguously.

7.3.3 Building Feature Vectors

Having extracted such co-occurrence information for all entities in the corpus, we can represent our entities and entity sets as vectors encoding the frequency of co-occurrence. Specifically, if $e_1, \ldots, e_N$ are all (named) entities in the corpus that occur in coordination constructions, then the j-th component of the vector $\overline{e_i}$ is defined as the number of times e_i and e_j co-occur in coordination. Similarly, for an entity set E_k, $\overline{E_k}(j)$ is defined as the number of times n_j co-occurs with *any* element of E_k in coordination. The dimension of the vectors used to represent our elements is, thus, equal to the number of *distinct* entities in the corpus that occur in coordination constructions at least once (i.e., N).

For instance, in the example above, if ne_a, ne_b and ne_c are all named entities $S_{ab} = \{e_a, e_b\}$, and we have encountered a single occurrence of "*... e_a , e_b and e_c...*" in our corpus, the vector space $\mathcal{VS}'$ will have three dimensions and the elements will be represented as follows:

$$\begin{array}{rcl} \overline{ne_a} & = & (0,\ 1,\ 0) \\ \overline{ne_b} & = & (1,\ 0,\ 1) \\ \overline{ne_c} & = & (0,\ 1,\ 0) \\ \overline{S_{ab}} & = & (1,\ 1,\ 1). \end{array} \tag{7.1}$$

One might argue that pairs of elements connected by comma are subject to a lot of noise. In fact, surface text structures of the form "*... X, Y, ...*" may occur often without implying any class similarity between "*X*" and "*Y*", but introduce, e.g., apposition, or clarification, instead. Therefore, a possible option may be to take only information about pairs connected by explicit coordination such as "*... e_a **and** e_b ...*" or "*... e_a **or** e_b...*". The features collected with this additional restriction are likely to be less noisy but, on the other hand, less feature information will be collected, which can lead to recall problems later. Using the same examples given previously, we would obtain the following representations in the restricted vector space $\mathcal{VS}^X$:

$$\begin{array}{rcl} \overline{ne_a} & = & (0,\ 0,\ 0) \\ \overline{ne_b} & = & (0,\ 0,\ 1) \\ \overline{ne_c} & = & (0,\ 1,\ 0) \\ \overline{S_{ab}} & = & (0,\ 0,\ 1). \end{array} \tag{7.2}$$

Note that this representation is sparser: the feature vectors contain more zeros.

From now on, we will refer to vector spaces built using only explicitly coordinated pairs by $\mathcal{VS}^X$, and to those build from explicit coordinations or commas as $\mathcal{VS}'$. It is important to note here that, when applying our method, we will always be able to guarantee that "X" and "Y" are, in fact, named entities (as we explain later in section 7.5.1).

7.3.4 Calculating membership function

With a fixed feature representation and a vector space, the membership function (that relates a candidate entity to an entity set) can be computed using one of the standard distance measures for vector spaces. In our experiments we used the *cosine similarity measure* [SM86]. For calculating the degree of membership of entity e^c, to the (model) entity set E, we can simply compute the value of the cosine between the two corresponding vectors $\overline{e^c}$ and $\overline{E}$:

$$\mu(e^c, E) = \cos(\overline{e}, \overline{E}) \tag{7.3}$$

Depending on the vector space used to obtain the vector representation of the entities and sets, we can have different definitions of the membership function. We will differentiate between μ', which is calculated using vector spaces generated with the explicit coordination connectors and the comma (i.e.,$\mathcal{VS}'_{EN}$ and $\mathcal{VS}'_{PT}$), and μ^X, which is calculated using the vector spaces generated considering only pairs of entities connected by explicit coordination connectors (i.e., $\mathcal{VS}^X_{EN}$ and $\mathcal{VS}^X_{PT}$). We will use μ when this difference is immaterial.

7.4 Evaluation Using Wikipedia

In this section, we present an evaluation framework for the class expansion task that is based on information extracted from Wikipedia. We choose Wikipedia for several reasons. First, as we described in section 4.2.4, Wikipedia contains many human-generated lists that can serve as gold standard for class expansion algorithms. Therefore, we can obtain test data from different domains and of different levels of specificity.

Second, Wikipedia has *explicit* information regarding the delimitation of named entities in text. We can explore the fact that many articles in Wikipedia actually correspond to named entities, and moreover, articles explicitly link to each other. Whenever in the text of an article, A_1, we find a (hyper)link to another Wikipedia article, A_2, such that the anchor text of the link starts with a capital letter, we can safely assume that the anchor text is a mention of the named entity: the title of article A_2. Using this simple heuristic, we automatically identify and delimit mentions of named entities in the content of Wikipedia articles. This allows us to circumvent the problem of *identifying* named entities in text and focus on evaluating membership functions and class expansion algorithms, rather than entity extraction.

Moreover, Wikipedia editors often assign each article to one or more *Categories*. We will use this explicit category information in section 7.4.3 to develop a simple baseline for the membership function.

7.4.1 Test sets and performance measures

From the definition of the class expansion task (section 7.3), we see that there are three inputs at stake: (i) the set of seed entities E^s, (ii) the set of candidates entities E^c, and (iii) the number of elements n that one intends to add to E^s. However, the Wikipedia lists that we will use to create our test sets are not guaranteed to be complete. Therefore, we chose not to rely on the exact sizes of these lists and we will not evaluate the resulting expanded set directly. Instead, we will evaluate the quality of rankings over E^c produced by our membership functions.

Specifically, for a gold standard class c_{gold}, which corresponds to a Wikipedia list, we will pick a subset, $\mathcal{P}$, containing only positive examples of entities of such category ($\mathcal{P} \subset c_{gold}$). We will also construct the $\mathcal{N}$ set that contains entities *outside* class c_{gold}, but which are somehow *related* to elements of c_{gold}. In general, $|\mathcal{N}| \gg |\mathcal{P}|$.

For a given c_{gold} class, and the corresponding sets of *positives*, $\mathcal{P}$, and *negatives*, $\mathcal{N}$, we define *test case* as a seed set taken from $\mathcal{P}$, $E^s_{gold} \subset \mathcal{P}$. To evaluate the quality of a membership function μ on this test case, we construct the set of candidates $E^c_{gold} = \mathcal{P} \cap \mathcal{N} \setminus E^s_{gold}$, rank the elements E^c_{gold} by $\mu(\cdot, E^s_{gold})$ and assess the quality of the resulting ranking R using *Average Precision* (see Equation 4.15). A *test set* is a collection of all test cases for a given c_{gold}, $\mathcal{P}$ and $\mathcal{N}$. The overall quality of a membership function μ on a test set can be assessed using the *Mean Average Precision* of the rankings produced for all test cases (see Equation 4.19).

7.4.2 Test Set Construction using Wikipedia

The construction of the actual test sets was done in two stages, and was based on XML encoded dumps of Wikipedia[5]. We produced test sets for English and for Portuguese. Table 7.1 gives some descriptive information about the English and Portuguese XML dumps used.

Table 7.1: Sizes of the Wikipedia dumps.

Lang.	files	paragraphs	text in MB
EN	1602048	8763404	2,950
PT	209198	757712	227

[5]Available from the University of Amsterdam in March 2007: `http://ilps.science.uva.nl/WikiXML`.

List Extraction from Wikipedia

First, we detect lists in Wikipedia pages. Such pages can be easily identified by an expressive title: “List of...” for English and “Lista de...” for Portuguese. We only consider the subset of those pages which present information using explicit *HTML list structures*. All Wikipedia pages which present list information using *tables* are ignored because the extraction of their elements is not trivial in most of the cases. The title of the list is saved for future identification of the test set.

Then, for each element of the extracted lists we obtain (i) the corresponding frequency in Wikipedia (i.e. number of times that it occurs as a link, as explained previously), and (ii) the URL of the Wikipedia article that addresses that entity. For the elements that do not posses a corresponding article in Wikipedia (i.e. point to a page were the user is invited to start an article about that topic) we still extract the element but we keep information regarding the absence of the article. Elements that are entirely numeric are ignored, to avoid long lists of dates or other numerical values (e.g. telephone codes) which are useless for our evaluation purposes.

Table 7.2 presents some statistics regarding the lists extracted from the English and the Portuguese Wikipedia. We will name these set of lists as $\mathcal{L}_{en}$ and $\mathcal{L}_{pt}$. The statistics address the number of lists extracted, the average number of elements (named entities) in those lists, and the average number of elements which have a dedicated article in Wikipedia. The last column is the overall average of frequency of the elements in lists, considering only the elements for which there is a page in Wikipedia.

Table 7.2: Statistics of lists extracted from Wikipedia

Lang.	Lists	#NE	NE w/ art.	avg_f(NE) w/ art.
EN	17594	92.4	58.4	286.2
PT	1390	90.3	43.4	32.5

The number of lists extracted from the English Wikipedia is obviously much larger, despite the fact that the *average number* of named entities contained in each list is quite close. The great difference however is in the average frequency of the elements contained in list, which is much higher for English.

Construction of $\mathcal{P}$ and $\mathcal{N}$ sets

To generate the sets of positives, $\mathcal{P}$, and the corresponding sets of negatives, $\mathcal{N}$, from the previously extracted lists, $\mathcal{L}_{en}$ and $\mathcal{L}_{pt}$, we followed several steps. We first selected only those lists which contained a minimum number of elements – min_a – with a dedicated Wikipedia article. This filtering was done to guarantee that the topic addressed by the list is reasonably well covered in Wikipedia. We set $min_a = 10$ for English and Portuguese and we thus obtained two more restrictive sets of list, $\mathcal{L}_{en}^{min_a}$ and $\mathcal{L}_{pt}^{min_a}$, which were then processed in order to obtain the final $\mathcal{P}$ and $\mathcal{N}$ sets.

For each list $\lambda(i)$ in $\mathcal{L}_{en}^{min_a}$ and in $\mathcal{L}_{pt}^{min_a}$ the following procedure was executed:

1. Choose all items in λ_i that have a dedicated article and whose frequency in Wikipedia is higher than a given threshold f_{min}. These will constitute $\mathcal{P}_{cand}(i)$, the candidates to set $\mathcal{P}$.

2. For each element in $\mathcal{P}_{cand}(i)$ extract all entities from the corresponding Wikipedia article. Add such entities to set $\mathcal{N}_{cand}(i)$, except those that belong to list $\lambda(i)$. The set $\mathcal{N}_{cand}(i)$ will thus be composed by all sort of entities related to elements from $\mathcal{P}_{cand}(i)$ but which are known not to belong to the initial list $\lambda(i)$. We also keep information about the number of times each element in $\mathcal{N}_{cand}(i)$ is found in the pages corresponding to elements in $\mathcal{P}_{cand}(i)$. This information reflects the degree of "relatedness" that each element of $\mathcal{N}_{cand}(i)$ has to the overall concept addressed by $\lambda(i)$.

3. Since sets $\mathcal{P}_{cand}(i)$, $\mathcal{N}_{cand}(i)$ can be extremely large the final $\mathcal{P}(i)$ and $\mathcal{N}(i)$ test sets are obtained by truncating the candidate sets. Thus, only the top n_p most frequent elements from $\mathcal{P}_{cand}(i)$ are chosen (if there are less than n_p, all are chosen). These will become set $\mathcal{P}(i)$. Again, we are trying to ensure that the membership function μ is tested on sufficiently frequent items. From $\mathcal{N}_{cand}(i)$, we choose the top n_n most "related" elements (as previously explained) with n_n being set to twice the number of elements in $\mathcal{P}(i)$. These elements will become set $\mathcal{N}(i)$.

4. Exclude all $\mathcal{P}(i)$ and $\mathcal{N}(i)$ sets for which the number of elements in $\mathcal{P}(i)$ is less that 5.

We set $f_{min} = 100$ both for English and Portuguese. Also, in both cases we set $n_p = 20$. For practical reasons, this number can not be higher because the test procedure will involve combinations of elements of $\mathcal{P}(i)$, which grow very quickly with n_p. Table 7.3 presents some statistics regarding the test sets generated and also some details about the corresponding $\mathcal{P}(i)$ sets.

Table 7.3: Statistics regarding test sets generated

	#	$\#\mathcal{P}$=20	10≤ $\#\mathcal{P} \leq 19$	5≤ $\#\mathcal{P}$ ≤9	f_{avg}
EN	3219	35%	29%	36%	1758.3
PT	75	36%	20%	44%	623.6

The number of tests generated for English is obviously much higher. We would be able to generate more tests for Portuguese if we lowered the f_{min} threshold. For example, $f_{min} = 50$ would allows us to generate 167 tests. We decided, however, to keep the same settings for English and for Portuguese because, for now, we do not have an informed way of defining an appropriate f_{min} value (based for example on parameters such as the size of the Wikipedia collection at stake).

It is possible to see that only about one third of the tests do actually reach the n_p limit. There seems to be also a difference between the two test sets regarding the relative weight of smaller tests ($5 \leq \#\mathcal{P} \leq 9$). For Portuguese, almost half the tests involve $\mathcal{P}(i)$ with 9 or less elements. This is related with the fact that we ended up setting a proportionally higher value for f_{min} (given the relative size of the collections) and this excluded many possible elements from the initial $\mathcal{P}_{cand}(i)$ sets. Another important difference is on value $\bar{f}_{avg}$. This value is the average over all $\mathcal{P}(i)$ of the average frequency of its elements and can be seen as an indicator of how frequent are the elements in the $\mathcal{P}(i)$ sets. For English this value is much higher, which reflects the fact that the English Wikipedia is not only larger in size, but is also much denser in entities than the Portuguese Wikipedia.

7.4.3 A Baseline using Wikipedia Categories

The set of Wikipedia Categories to which a given article is assigned provides valuable information that can be used for devising a simple baseline membership function. We observed that in the English Wikipedia there are a total of 172,762 categories and each article is assigned in average to 3.0 categories (in March 2007). For the Portuguese Wikipedia the total number of categories is 20,736, and the average number of categories to which an article is assigned is 2.2.

Let us represent the complete set of categories of Wikipedia by

$$C^{wk} = \{c_1^{wk}, c_2^{wk}, ...c_w^{wk}\} \tag{7.4}$$

For a given article A_z, describing entity e_z, let $\overline{e_z^{wk}}$ be a binary vector of the same dimension of C^{wk} that contains information regarding the Categories to which article A_z has been assigned. If A_z is assigned a given category c_j^{wk}, then the value of $\overline{e_z^{wk}}$ at index j is 1, otherwise is zero.

Let us now consider the set E_k^{wk} composed of an arbitrary set of named entities, each addressed by an article in Wikipedia. The corresponding set of articles is denoted by $A_{E_k^{wk}}$. Let us now define $\overline{E_k^{wk}}$ as binary vector of the same dimension of C^{wk} that contains information about the categories to which the articles in $A_{E_k^{wk}}$ were assigned. The value of $\overline{E_k^{wk}}$ at position j will be one if and only if *at least one* element of $A_{E_k^{wk}}$ is assigned to category c_j^{wk}. If not, the value of $\overline{E_k^{wk}}$ at position j will be 0.

If there is a high degree of overlap between the category vectors $\overline{e_z^{wk}}$ and $\overline{E_k^{wk}}$, as calculated by a standard vector distance measure, then we may conclude that the corresponding articles, A_z and $A_{E_k^{wk}}$, are assigned to many common categories. In that case, we obtain a strong indication that the entity e_z has been assigned many of the same categories that have been assigned to entities in E_k^{wk}. The higher the overlap between the category vectors, the higher should be the degree of membership of e_z to the set E_k^{wk}.

We thus propose $\mu_{base}(e_{\sigma_i,\rho_i}, E^s)$, a baseline membership function based on the categories vectors obtained for the article addressing entity e_i, and for the articles addressing the entities contained in the set E^s (which we will now denote by $\overline{e_i^{wk}}$ and $\overline{E_s^{wk}}$). We chose to calculate overlap between $\overline{e_i^{wk}}$ and $\overline{E_s^{wk}}$ using the Jaccard coefficient over such vectors:

$$\mu_{base}(e_i, E^s) = J(\overline{e_i^{wk}}, \overline{E_s^{wk}}) \tag{7.5}$$

The Jaccard coefficient produces values in the range [0,1] so direct comparison between $\mu(e_{\sigma_i,\rho_i}, E^s)$ and $\mu_{base}(e_{\sigma_i,\rho_i}, E^s)$ is possible.

7.5 Experimental Setup

7.5.1 Extracting entities and features

We collected pairs of coordinated named entities both for the English and for the Portuguese, based on the previously described Wikipedia XML dumps. The fundamental reason for having chosen Wikipedia as source corpus for grounding our membership function μ is that we can thus avoid the complex problem of identifying / delimiting named entities in text. As explained in section 7.4, by using a very simple heuristic based on links found in Wikipedia, we can easily identify named entities in the articles.

We should emphasise that the only source of information used for extracting the pairs of coordinated named entities was the text contained in paragraphs inside articles (e.g., excluding category and language links, information boxes, etc.). Explicit list information (lists and tables) was also filtered out.

For both languages, the paragraphs of wikipedia articles were scanned for structures of the form "(e_a) (*coordination connector*) (e_b)". Pairs (e_a, e_b) were extracted, counted and stored in a database. We separately compiled data using *only* explicit coordination connectors, and using *both* explicit coordination connectors *and* the comma connector, so as to be able to create two different vector spaces for each language, as described in section 3. We used the following sets of explicit coordination connectors:

- for English: *"and the", "and a", "and", "or the", "or a", "or",*
- for Portuguese: *"e o", "e um","e a", "e uma", "e do", "e da", "e", "ou o", "ou um", "ou a", "ou uma", "ou".*

We extracted features defining four vector spaces: $\mathcal{VS}_{EN}^X$ and $\mathcal{VS}_{PT}^X$ (using only explicit coordination), and $\mathcal{VS}'_{EN}$ and $\mathcal{VS}'_{PT}$ (using explicit coordination and comma). Table 7.4 provides some basic statistics about these spaces. The number

Table 7.4: Statistics about the named-entities extracted from Wikipedia for the corresponding Vector Space

	NE Pairs	Distinct NE Pairs	Dim($\mathcal{VS}$)
$\mathcal{VS}'_{EN}$	2,172,790	1,255,204	819,379
$\mathcal{VS}^{X}_{EN}$	1,755,603	516,415	500,980
$\mathcal{VS}'_{PT}$	154,836	119,174	85,494
$\mathcal{VS}^{X}_{PT}$	44,919	36,751	46,601

of distinct NEs defines the dimension of a vector space, and the number of distinct pairs is the number of non-zeros in the vector representations of all NEs. Not surprisingly:

1. The dimension of English Vector Spaces is substancially larger than for Portuguese ones, since English Wikipedia is also about 13 times larger;

2. The dimensions of Vector Spaces built using only explicit coordinations is inferior to the dimension of the Vector Spaces built using both explicit coordination and commas (61% for English and 54% for Portuguese)

To illustrate the extracted data, below we show the top 10 most frequent coordination pairs compiled for each of the four cases, together with their counts:

$\mathcal{VS}'_{EN}$: (Black, African American, 6157), (Ontario, Canada, 3631), (Pennsylvania, United States, 2610), (Minnesota, United States, 2514), (New South Wales, Australia, 2442), (London, England, 2325), (British Columbia, Canada, 2164), (Biographies, Books, 1995), (Books, Companies, 1995), (Companies, Fiction, 1995)

$\mathcal{VS}^{X}_{EN}$: (Black, African American, 6156), (British, Commonwealth, 1327), (United States, Canada, 1202), (Australia, New Zealand, 897), (England , Wales, 486), (World War I, World War II, 424), (Europe, Asia, 413), (Canada, United State, 405), (United States, United Kingdom, 308), (India, Pakistan, 299)

$\mathcal{VS}'_{PT}$: (Rio de Janeiro, Brasil, 135), (São Paulo, Brasil, 127), (Rio Grande do Sul, Brasil, 125), (Curitiba, Paraná, 113), (Portugal, Espanha, 90), (Los Angeles, Califórnia, 89), (São Paulo, Rio de Janeiro, 87), (Paris, França, 85), (Hyuuga Neji, Rock Lee, 84), (Londres, Inglaterra, 83)

$\mathcal{VS}^{X}_{PT}$: (Momochi Zabuza, Haku, 82), (Jiraya, Tsunade, 81), (Portugal, Espanha, 66), (BR-101, RS-389, 64), (Arizona, Novo México, 44), (Estados Unidos, Canadá, 42), (Espanha, Portugal, 42), (Santa Catarina, Rio Grande do Sul, 40), (Rio de Janeiro, São Paulo, 39), (Gagaúzia, Transnístria, 39)

Despite the fact that we are just observing a very small sample of the coordination pairs obtained, the pairs obtained for $\mathcal{VS}'_{EN}$ and $\mathcal{VS}'_{PT}$ seem to be much noisier than those obtained $\mathcal{VS}^X_{EN}$ and $\mathcal{VS}^X_{PT}$. Some of the most frequent tuples for $\mathcal{VS}'_{EN}$ and $\mathcal{VS}'_{PT}$ do not express the type similarity relation we are looking for. One of the most frequent noisy cases is the (location, wider location) pair which is collected from text passages such as "... Ontario, Canada...". On the other hand, as explained in section 7.3.3, $\mathcal{VS}^X_{EN}$ and $\mathcal{VS}^X_{PT}$ tend to be *much sparser* than $\mathcal{VS}'_{EN}$ and $\mathcal{VS}'_{PT}$, so one should also expect the latter spaces to have, globally, more information for inferring type similarity.

7.5.2 Evaluation setup

We conducted evaluation of our membership function μ for both types of spaces, for English and Portuguese. We will use μ^X when referring to the cases where μ is computed over $\mathcal{VS}^X_{EN}$ or $\mathcal{VS}^X_{PT}$, and μ' when μ is computed over $\mathcal{VS}'_{EN}$ and $\mathcal{VS}'_{PT}$. Evaluation was performed using all pairs of sets $\mathcal{P}(i)$ and $\mathcal{N}(i)$ obtained for each language. Each $\mathcal{P}(i)$ set had up to a maximum of 20 elements and each $\mathcal{N}(i)$ set had exactly twice as many elements as the corresponding $\mathcal{P}(i)$.

One important parameter in this experimental set-up is the *size* of the seed sets. Due to the combinatorial nature of the evaluation procedure we restricted the size of the seed sets to only two elements. Increasing this value would greatly increase the time required for evaluating the membership functions over the entire test-set collection. One might argue that using only two values is an ad-hoc decision, which does not guarantee the definition of the class(es) underlying a given seed set: "growing" such small set might lead to many possible (yet admissible) sets, so evaluation becomes rather fuzzy. However, one might also argue that it makes sense to evaluate the membership functions under such extreme conditions. The reason is that, in practice, for very specific classes it might be very difficult to obtain a larger set of seed examples, so evaluating a membership function with small seed set can be closer to an actual application scenario.

For each pair of $\mathcal{P}(i)$ and $\mathcal{N}(i)$ set the following procedure was followed:

1. Generate all possible seed combinations of 2 elements from $\mathcal{P}(i)$ (which would never be more than 380 combinations). For each combination of seed entities, E^s_k do:
   (a) Obtain $\overline{E^s_k}$ the vector representation of E^s_k in the corresponding Vector Space ($\mathcal{VS}'$ or $\mathcal{VS}^X$).
   (b) Use μ' and μ^X to compute the degree membership of each of the elements in $(\mathcal{P}(i) \setminus E^s_k)$ (i.e. all positive elements except seed entities) to the set E^s_k. For that we use the vector representations of each element of the set $(\mathcal{P}(i) \setminus E^s_k)$ taken from the corresponding Vector Space ($\mathcal{VS}'$ or $\mathcal{VS}^X$).

(c) Likewise use μ' and μ^X to compute the degree of membership of each of the elements from set $\mathcal{N}(i)$ to the set E^s_k.

(d) Rank all elements of sets $(\mathcal{P}(i) \setminus E^s_k)$ and $\mathcal{N}(i)$ according to the previously computed value of membership. Compute Average Precision for this seed combination.

2. Compute Mean Average Precision using all values of Average Precision obtained for each seed combination.

A similar procedure was followed for obtaining the performance of the baseline function μ_{base}. Thus for each pair of sets $\mathcal{P}(i)$ and $\mathcal{N}(i)$, i.e. 3219 for English and 75 for Portuguese, we obtained the values of MAP of the three membership function μ', μ^X and μ_{base}.

7.6 Results

Table 7.5 contains the average values of MAP computed over all test sets. In order to provide a more encompassing outlook, Figures 7.1 and 7.2 show the values of Mean Average Precision for μ', μ^X and μ_{base} for all English and Portuguese test sets (test set are ordered by performance of μ').

Table 7.5: Average values of MAP on the entire test sets for EN and PT

	#test	μ'	μ^X	μ_{base}
EN	3219	0.424	0.289	0.399
PT	75	0.542	0.426	0.333

For both languages the average performance obtained by μ' is higher than the performance obtained for μ^X (+0.135 for English and +0.116 for Portuguese). According to the one-tail sign test, the difference is consistent in both cases, μ' ($p < 0.0001$ in both cases). This probably results from the fact that, as mentioned before, μ^X is operating over much sparser spaces. When comparing to the baseline, μ' outperforms μ_{base} but the difference for English (+0.025) is not as expressive as it is for Portuguese (+0.209). However μ^X is not able to outperform the baseline function μ_{base} in the case of English. The baseline function obtains a higher score for English than for Portuguese.

In the case of Portuguese, the results regarding μ' and μ^X are significantly better than for English. One possible reason for this is related to the frequency threshold used for selecting the elements that compose the set of Positives, $\mathcal{P}(i)$, which was the same both for the English and the Portuguese test sets (f_{min} = 100). This means that threshold was relatively higher for Portuguese than for English, taken the relative sizes of both Wikipedias. Therefore, the resulting test sets for Portuguese tend to contain elements that are very frequent, from a

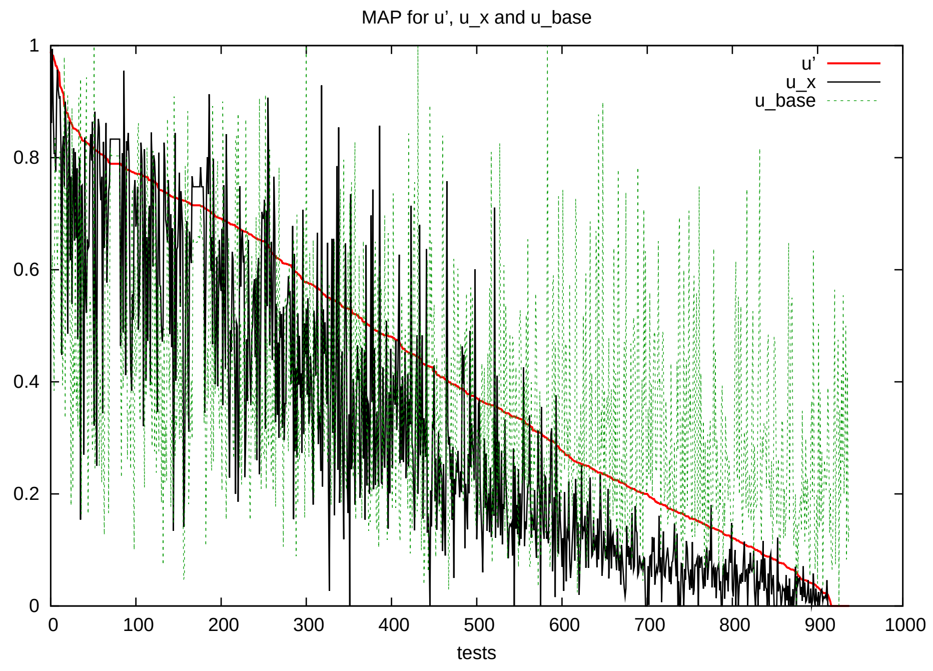

Figure 7.1: Plot of Mean Average Precision for μ', μ^X and μ_{base} obtained for the 3219 test sets prepared for English (order by performance of μ')

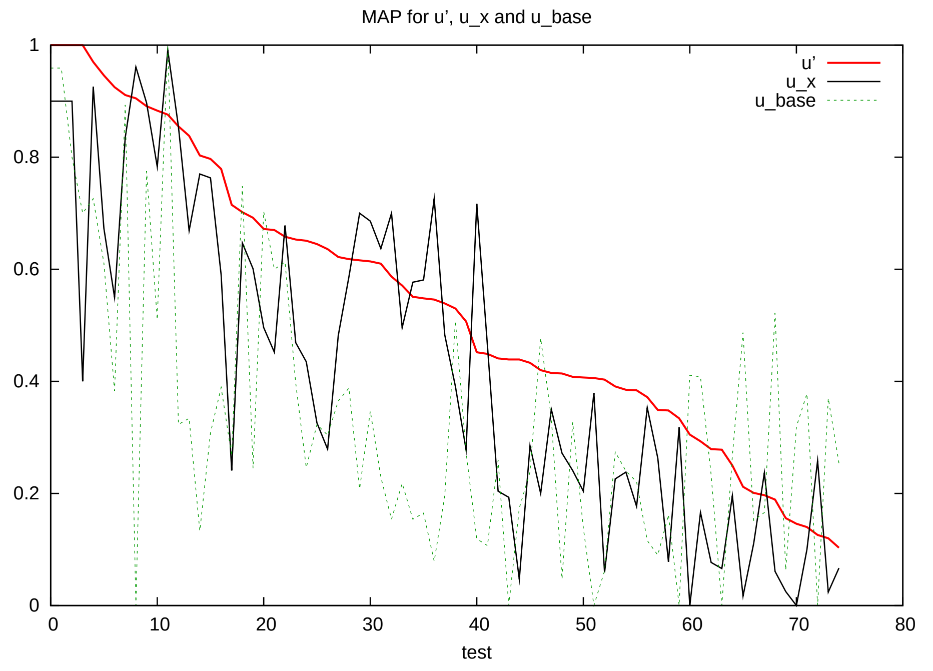

Figure 7.2: Plot of Mean Average Precision for μ', μ^X and μ_{base} obtained for the 75 test sets prepared for Portuguese (order by performance of μ')

relative point of view, in the Portuguese Wikipedia which benefit corpora-based methods such as the one we propose.

7.6.1 Results on least frequent elements

In order to further assess the importance of the frequency values in the performance of μ, we developed additional test sets for Portuguese. In the previous set of tests, the elements chosen for the sets $\mathcal{P}(i)$ (i.e. the positives) were the *most frequent* elements (up to a maximum of 20) from the sets of candidates $\mathcal{P}_{cand}(i)$ (elements in sets $\mathcal{P}_{cand}(i)$ were already restricted to those whose frequency in Wikipedia is higher than $f_{min} = 100$). For this second set of experiments we generated test sets composed by less frequent elements, by choosing this time *the least frequent elements* found in $P_{cand}(i)$ (also with $f_{min} = 100$). This allowed us to obtain new $\mathcal{P}(i)$ sets that contained exactly the same number of elements as in the previous test, but with a lower average frequency (for those $\mathcal{P}_{cand}(i)$ set which had less than 20 elements, there is no change). From now on we will refer to these newly created set of Positives as $\mathcal{P}^{-}(i)$. The corresponding sets of negatives, $\mathcal{N}(i)$, was kept the same: the only difference between the new test sets and the previous ones is that we have now another set of Positives, $\mathcal{P}^{-}(i)$. The value of $\bar{f}^{-}_{avg}$ (average over all $\mathcal{P}^{-}(i)$ of the average frequency of its elements) for the new test sets dropped to 327.6, which is approximately 53% of the previous $\bar{f}^{-}_{avg}$ (see Table 7.3). This clearly indicates that we are now dealing with sets of Positives containing much less frequent elements.

We repeated the evaluation procedure for all the membership functions μ', μ^X and μ_{base} on the newly created test sets. The results, along with those obtained with the previous tests sets, are show in Table 7.6:

Table 7.6: Average values of MAP on the entire test sets($\mathcal{P}^{-}$, $\mathcal{N}$) and ($\mathcal{P}$, $\mathcal{N}$) for Portuguese.

	#tests	$\bar{f}_{avg}$	μ'	μ^X	μ_{base}
($\mathcal{P}$, $\mathcal{N}$)	75	623.6	0.542	0.426	0.333
($\mathcal{P}^{-}$, $\mathcal{N}$)	75	327.6	0.499	0.328	0.344
Δ		296.0	0.043	0.099	-0.011

The performance of both μ', μ^X decreases for the test sets ($\mathcal{P}^{-}$, $\mathcal{N}$). The decreases occur consistently over the 75 test sets, as illustrated by the values of the test sign: $p < 0.01$ for μ' and $p < 0.0001$ for μ^X). The drop for μ^X is more significant, which seems quite understandable since the associated vector space $\mathcal{VS}^X_{PT}$ is much smaller than $\mathcal{VS}'_{PT}$, and the probability of not finding a vector representation for some of the entities test has increased.

However, only 28 of the new tests ($\mathcal{P}^{-}$, $\mathcal{N}$) are actually different from the previous ($\mathcal{P}$, $\mathcal{N}$) because, as indicated in Table 7.3 only 36% the original $\mathcal{P}_{Cand}$

had more than 20 elements. Considering only the test sets that refer to those 28 cases (we will denote them by ($\mathcal{P}_{28}$, $\mathcal{N}_{28}$) and ($\mathcal{P}^{-}_{28}$, $\mathcal{N}_{28}$)) the results become more extreme (Table 7.7). On one hand, the performance of both μ' and μ^X over the sets ($\mathcal{P}_{28}$, $\mathcal{N}_{28}$) is the highest of all the tests. On the other hand, the results over ($\mathcal{P}^{-}_{28}$, $\mathcal{N}_{28}$) drop significantly. Again, μ^X is more sensible to changes with performance dropping sharply to 46.5% of the value obtained in the original test, while μ' also drops for 78.8%. In both cases the changes are consistent and significant as assessed by the *sign test* ($p < 0.0001$).

These changes confirm that the performance of proposed membership functions frequency increase with frequency of the elements to be tested.

Table 7.7: Average values of MAP on the test sets($\mathcal{P}^{-}_{28}$, $\mathcal{N}$) and ($\mathcal{P}_{28}$, $\mathcal{N}$) for Portuguese.

	#tests	f_{avg}	μ'	μ^X	μ_{base}
($\mathcal{P}_{28}$, $\mathcal{N}_{28}$)	28	982.2	0.547	0.493	0.290
($\mathcal{P}^{-}_{28}$, $\mathcal{N}_{28}$)	28	189.4	0.431	0.229	0.320
Δ		792.8	0.115	0.264	-0.030

7.7 Conclusions

In this chapter, we presented a corpus-based method for the class expansion task: given a number of seed entities for a class, our method selects additional candidates for inclusion in the class. The method uses a class membership function, estimated from the co-occurrence statistics in a text corpus, more specifically, co-occurrence of named entities in coordination constructions. Results show that coordination information does in fact convey information about type similarity for a wide variety of entity classes. We also presented an original evaluation framework based on entity lists automatically extracted from Wikipedia. The method presented outperforms a simple baseline that uses category tags of Wikipedia articles assigned by human editors. The performance of the method improves as the frequencies of the candidate entities in the text corpus (which are related to the corpus size) increase. Thus, having large enough corpora available, the method presented here can effectively be used for helping the expansion of lexicons, or their enrichment with information about type similarity.

Chapter 8

Web-Scale Entity Disambiguation

In this chapter[1] we address *ambiguity*, one of the fundamental issues in natural language. We propose a solution to some ambiguity situations related to *names of entities*: many distinct entities share the *same name*, and hence, a mention by name in *two different documents* can not be univocally matched to an entity. This task is generally known as *Named-Entity Disambiguation* (NED).

Ambiguous situations regarding names can be quite complex. They occur not only *intra-type*, i.e. when the entities that could be mentioned by certain name all belong to the same type or class (e.g. people with the same name), but also *inter-type*, when the entities with the same name belong to different types (e.g. "Pluto" can be the name of an astral body, a Roman God, any of several existing rock bands, a fiction character, among many other possibilities[2]).

As explained in section 2.4, disambiguation procedures can be formulated as a *similarity search* among ambiguous items. Thus, disambiguating mentions (i.e. occurrence of a name in a certain document) can be achieved by finding those mentions in different documents whose *profile* is similar enough to be considered references to the *same* entity. On the other hand, mentions whose profiles differ substantially are probably references to different entities.

Following this rationale, we propose a clustering approach to NED. The goal is to cluster together ambiguous mentions (i.e. those made using the same name) so that each cluster contains mentions of only one entity, while different clusters group mentions of different entities. The key point in this approach is choosing the appropriate set of features to describe the mentions. We use information regarding name *co-occurrence*. Our hypothesis is that entities exhibit a more or

[1]The material in this chapter results from work developed during a six-month internship at Google NY in 2007. Most of this material was presented in two papers published during 2009: (i) *Luís Sarmento, Alexander Kehlenbeck, Eugénio Oliveira and Lyle Ungar "An Approach to Web-scale Named-Entity Disambiguation"* ([SKOU09a]), and (ii) *Luís Sarmento, Alexander Kehlenbeck, Eugénio Oliveira and Lyle Ungar "Efficient clustering of web-derived data sets"* ([SKOU09b]).

[2]Check `http://en.wikipedia.org/wiki/Pluto_(disambiguation)`

less stable and distinctive entity co-occurrence profile. Thus, different entities (with the same name) will tend to co-occur in text with different sets of entities. Therefore, name co-occurrence information should help us to differentiate between "Paris", *"France"* from "Paris", *"Texas"*, since they are expected to co-occur with different sets of entities (["Eiffel Tower", "Moulin Rouge", etc.] vs. ["Lamar County", "East Texas", etc.]).

Although conceptually elegant, clustering approaches face several practical barriers, specially when attempting named-entity disambiguation on the whole Web, as we do: the number of documents to be processed, the number of mentions to be processed and the size of information contained in the feature vectors that need to be compared represent serious challenges to the scalability of any algorithm. We propose a solution that involves the *partition* of the problem in several small chunks, and then merging the partial results in a subsequent operation. Also, we take advantage of *intrinsic distributional properties* of Web-derived data to propose an optimized strategy for performing vector comparisons that scales relatively well in our scenario, specially when compared with naive vector comparison strategies (e.g. *all-against-all*).

But, as we show, scaling NED to the Web involves more than just algorithmic challenges. In fact, during our experiments, we noticed that the performance of our method did not increase with the size of the web sample to be disambiguated as much as we expected. This was a somewhat surprising due to the *data-driven nature* of our our approach. Manual inspection of the clustering results led to some interesting conclusions regarding (i) the *sets of features* required for actually performing *realistic* NED at Web-scale in an efficient fashion, and (ii) about the *definition* of the named-entity disambiguation task itself.

8.1 Introduction

Realistic named-entity disambiguation (NED) of Web data involves several challenges that have not yet been considered simultaneously. First, when moving NED to the web we need to deal with very high levels of ambiguity. Since there are so many documents in the Web, the same name will often refer to hundreds of different entities. This makes the problem much harder as compared with NED approaches for small collections where one needs to disambiguate only among a few possibilities. Second, distributions of mentions on the web are highly skewed. For each ambiguous name, there are usually one or two dominant entities to which the vast majority of mentions refer, even when many entities share the same name. For example, most mentions of the name "Paris" found on the web refer to the capital of France (and a smaller number to Paris Hilton), while there are dozens of well-known entities with that name[3]. Table 8.1 shows hit counts

[3]See the Wikipedia disambiguation page for "Paris": `http://en.wikipedia.org/wiki/Paris_(disambiguation)`

for five queries sent to Google containing the word “Paris” and additional (potentially) disambiguating keywords. These values are merely indicative of the orders of magnitude at stake, since hit counts are known to change significantly over time. The real challenge is to be able to disambiguate between mentions of the less frequently mentioned entities, for which there is proportionally much less information and more noise. Third, most solutions to NED presented so far involve processing relatively small data-sets. Realistic NED involves processing web-scale collections (terabyte size), requiring computationally efficient ways of representing and processing data and, sometimes, involving practical decisions that might affect negatively final results for some cases.

Table 8.1: Number of Google hits obtained for several entities named “Paris”

# query	# hit count (x10^6)	%
paris	583	100
paris france	457	78.4
paris hilton	58.2	9.99
paris greek troy	4.130	0.71
paris mo	1.430	0.25
paris tx	0.995	0.17
paris sempron	0.299	0.04

There are also other fundamental questions that have not yet been investigated. Many of the solutions to NED involve data-driven techniques, such as clustering. Such techniques usually benefit from processing larger amounts of data. Therefore, one would expect to obtain better NED results as the size of the collection to be disambiguated increases. However, as the size of the collection to be disambiguated becomes larger, the variety of different entities and contexts that have to be dealt with also increases. As the contexts in which mentions occur become more diverse, data-driven approaches potentially become harder. The exact balance between these two effects has yet to be quantified.

In this chapter, we present a clustering-based approach to disambiguating entities on the Web. The algorithm we propose is capable of dealing with an arbitrarily high number of entities types, is scalable to the number of mentions on the web, and can be distributed over a cluster of machines to process large web collections. For evaluating the results of the disambiguation procedure we developed a gold standard based on entity information extracted from Wikipedia. We experimented disambiguating samples of the web with increasingly large sizes to test how well the algorithm scales and whether or not more data leads to better results. Results suggest that as the size of the collection increases, more complex cases of ambiguity emerge, making the definition of the NED task itself less clear. This seems to be an intrinsic characteristic of highly heterogeneous document collections, and suggests the existence of fundamental upper limits on the performance of clustering-based approaches to NED based only on name co-occurrence information.

8.2 Related Work

There are currently two main lines of research that explore different strategies. The more traditional (e.g. [GA04, Mal05, MY03, YE07]), involves a clustering approach: the hypothesis is that mentions of the same entity share a significant amount of contextual features that allows to cluster them together. Mentions clustered together should refer to the same entity and each cluster is related with a single disambiguated entity. The other line of work (e.g. [BP06, Cuc07, DEG^{+}03]) focuses on using external sources of knowledge containing information about entities of the world (e.g. Wikipedia) to perform disambiguation. The core of these methods consists in trying to project each mention found in text on the external knowledge source by comparing the corresponding feature profiles. Usually, these methods achieve full entity resolution because mentions found in text are matched with a specific entity contained in the knowledge resource.

8.2.1 Approaches based on clustering

Mann and Yarowsky present a disambiguation procedure focused on personal names [MY03]. An hierarchical clustering procedure is used to group Mentions referring to the same entity. The clustering is achieved in three steps. In the first step all Mentions are compared using an all-against-all strategy (cosine measure over vector representation), and clustering proceeds until a given proportion of the Mentions are clustered and the corresponding clusters achieve a minimum relative size. These first *seed* clusters are expected to be “pure” and represent already the main entities. In a second step the remainder Mentions are assigned to these seed clusters using a nearest neighbor policy. This step tries to solve the problems associated with outliers and Mentions with an insufficient number of features. In a third step clustering proceeds until no more clustering is possible. The final clusters are expected to represent each of the (main) entities involved. The authors tested the performance of the method using several different features to describe the Mentions, ranging from a simple bag-of-words approach to a combination of statistically filtered words, co-occurring proper names and automatically extracted biographic features. These biographic features are extracted using patterns learned thought a bootstrapping approach. Results show that using biographic features, in combination with other features, improve performance. Evaluation is made both against a test set of ambiguous pseudo-names (documents referring two different people are shuffled and each Mention of the two different names is replaced by a unique random pseudo-name to artificially create ambiguity), and against 4 sets of 60 documents containing ambiguous names which had been manually disambiguated.

The method, however, has two important limitations. First, it makes a very strong simplification by considering that the disambiguation involves deciding only between two classes, corresponding to only two distinct entities, or in the best case three classes which also include an “other entity” class. This is not the case for most of the personal names. The other limitation is the fact that the method performs an all-against-all comparison between Mentions prior to the first stage of clustering. Such a procedure is certainly not scalable, since for very large data sets names can easily be mentioned millions of times.

Another set of clustering methods aiming at person name disambiguation is presented by Gooi and Allan [GA04]. Each Mention is described by a vector composed of the terms taken from a 55 word window centered around it. Each term is tf-idf weighted taking into account its in-windows occurrence, the number of documents and the global number of documents that contain the term. The authors compare three different clustering methods based on this set of features. The first method, named *incremental vector space*, uses a variation of stream-clustering technique. The first feature vector is taken from the set to be disambiguated and is considered a cluster (chain). Each of the subsequent vectors is compared with all the mentions of the existing clusters (initially only one), using an average-link comparison strategy and the cosine metric. If a vector is far from any of the existing clusters, a new cluster is created containing only this vector. Otherwise the vector is assigned to the closest cluster. The second method varies from the previous one by modeling each feature vector as a probability distribution and by comparing each of these distributions using the KL divergence measure (with auxiliary smoothing to deal with 0 probabilities). A similar incremental approach is used. This time, however, instead of keeping information about all probability distributions contained in a given cluster, each cluster is described by a probability distribution that results from the aggregation of each of its individual distributions (i.e. a “centroid” probability distribution). Subsequent probability distributions are compared only with this aggregate distribution and are included in the closest cluster if distance is below a pre-defined threshold. Since the first two methods are very sensitive to outliers and are dependent on the order in which Mentions are processed, Gooi and Allan propose a third method based on an agglomerative clustering. In each step all clusters are compared with each other using the cosine metric, and the two closest are merged, if their distance is below a given threshold. The procedure is repeated until no more merging is achieved.

The authors evaluated the three methods against two test-sets. The first was the “John Smith” corpus (first presented by Bagga and Baldwin [BB98b]) which is composed by a set of 197 articles containing mentions of the naturally ambiguous name ”John Smith” and its variations. There are 35 different entities name “John Smith” in this corpus. The second corpus, the “person-x corpus” is an artificially ambiguous corpus generated by conflating Mentions of different name into a ambiguous "person-x" name. The corpus contained 33.404 entity-annotated documents, and from each one a single person entity was randomly selected in order to be made ambiguous. All Mentions of the selected name in the document (along with several obvious variations) where replaced by the ambiguous "person-x" name. Reference to the original name was kept, and when the same name was found in multiple documents, the authors manually checked to see if it really referred the same person. This procedure generated 34.404 mentions corresponding to 14.767 entities. 46.66% of the names selected only occur once, but 16.67% occur 9 or more times. Performance of the disambiguation task was measured the using B-CUBED algorithm [BB98a]. Results achieved show that the agglomerative clustering method leads to better precision and recall figures on both test corpora, and it showed higher stability to changes in parameters (similarity threshold and test corpora partitioning). In general both vector space approaches seemed to lead to better results that the KL-Divergence approach. Additionally, the authors compared the three methods with the method presented in [BB98a]. This method had only been evaluated against the "John Smith" corpus and, interestingly, when re-evaluated against the larger "Person-X" corpus it was not able to achieve a comparably good performance, especially regarding recall and precision trade-off. This led the authors to suggest that methods that are successful in small corpora might not be able to maintain their performance in larger corpora.

Despite the fact that the approaches presented by Gooi and Allan do involve the disambiguation of names into many possible entities (up to 9 in the "Person-X" test set), they only consider person entities (although an extension to other types looks possible). Additionally, the best-performing methods show serious scalability problems that would not allow them to deal with web-scale disambiguation.

In [YE07], Yates and Etzioni present RESOLVER, a system focusing on a related, yet different problem: *Synonym Resolution*. In this case the goal is to find alternative forms of names that refer the same entity. The definition of the task, as defined by the authors, does not involve solving the name ambiguity problem (several entities may be referred by the same name). However, because the Synonym Resolution can be seen as a subtask of a complete disambiguation procedure and because this is one of the few works that actually tries to deal with a web-scale data set we will make a brief revision of this work. Each name is initially considered to refer to a different entity and synonym resolution process is achieved through name clustering. Names are compared using a combination of two pieces of evidence: (i) similarities between strings - *String Similarity Model* (SSM) - and (ii) the distributional similarity of the set of relations found for each name using IE techniques - *Extracted Shared Property Model* (ESPM). The String Similarity Model aims at solving situations that involve lexical variations of the same name: substrings, acronyms, abbreviations, etc. The probability of two names being synonyms is computed by a formula that includes contributions of the Monge-Elkan and the Levenshtein string similarity functions, and other user-defined parameters. On the other hand, the Extracted Shared Property Model is intended to deal with the situation where a completely different name is used to mention the same entity, such as for example, "Mars" and "Red Planet". In this case, assertions like ⟨*lacks, Mars, ozone layer*⟩ and ⟨*lacks, Red Planet, ozone layer*⟩, which can be extracted from large corpora using IE techniques, would provide evidence that "Mars" and "Red Planet" are names that might mention the same entity. The probability of two different strings being synonyms based on the existence of common relations follows a comprehensive model based on the work initially presented by Downey et al. ([DES05]).

The RESOLVER system combines evidence from the two probabilistic models into a joint probability. This probability is then generalized to allow estimating the probability of the two clusters including synonyms. RESOLVER uses an agglomerative clustering algorithm, which relies on this last probability function to compare clusters, to iteratively merge clusters (initially each name is a single cluster). Iterations run in O(N log N) on the number of extracted assertions. The authors report that RESOLVER usually finishes in a few iterations (maximum found was 9). The performance of RESOLVER was tested using a data set of 2.1 million assertions extracted from a Web crawl. From this data set only assertions that included name strings and relation strings whose frequency was higher than 25 were considered. The test set contained assertion relating 9.797 distinct object strings and 10.151 distinct relation strings. For control purposes, the authors also tested the performance of the method using only one of the two available models (the SSM and the ESPM). The Cosine Similarity Metric (CSM) was also used as a simple baseline method for cluster comparison. A gold standard was manually created by starting from the top 200 most frequent object strings. For each of these strings, the authors manually matched their synonyms taken from the set of remainder names (approximately 9.600 names) included in the test set. Fifty-one clusters of size greater than one were found, with the average cluster size being 2.9. Performance of the 4 models (1 combination + 2 partial + 1 alternate) was measured by comparing all generated hypothesis clusters (of size greater than two) which contained at least one high frequency name among the gold standard clusters. Results show that the full RESOLVER system significantly outperforms all other options. However, rather surprisingly the simpler SSM model outperforms the ESP in the synonym resolution task, which author argue that is mostly due to noisy input. ESP had superior performance than the CSM baseline. Error analysis showed two main types of errors: (i) incorrect resolution of names that share a large number of relations because they are entities of the same (very specific) type ("US News" vs. "World Report"), and (ii) problems that arise precisely from ambiguous names ("President", "Army", "President Bush").

Two other methods based on clustering and graph partitioning are presented and compared by Malin ([Mal05]). As in most of the other works, the disambiguation is done exclusively over personal names. The basic principle is, again, using information regarding social networks to disambiguate person entities. Such information is inferred from the names with which the possible ambiguous name co-occurs. The first approach is based again on document clustering. Each source (i.e. document) s_i, from a set of sources S, is represented by a binary vector containing information about the names it contains (each name is considered unambiguous inside each document): $s_i = \{e_{i1}, e_{i2}, ...e_{in}\}$, where e_{ij} is 1 if the name e_j is found is source s_i. In this research the author only considered sources having no more than a single occurrence of an ambiguous name (according to a ground truth test-set which we will explain later). Sources are then clustered using an hierarchical clustering procedure. Each cluster is initially composed by one source and clustering is achieved iteratively by merging the most similar clusters into one new cluster. Clusters are compared using an average linkage criterion and are merged if their distance is below a given threshold. Cluster distance is the average distance of all pair-wise distances between elements of each cluster, measured using the cosine similarity between (binary) vectors. Clustering proceeds until a given stopping criteria is achieved or a unique big cluster is obtained. The second method uses a random walk strategy over a graph and tries to incorporate in the disambiguation process not only information about name co-occurrence within the same page but also information regarding indirect connections among social networks. Disambiguation is achieved by partitioning the graph into separate connected components, each hopefully corresponding to one entity.

A social network is built by taking each name in S and making a node in a graph. An edge exists between any two nodes if they co-occur in at least one source. The weight between the nodes is adjusted inversely to the number of names that exist in the corresponding source, so that if two names co-occur in a document that contains a very large number of other names (e.g.: a list of names), such co-occurrence is considered as less relevant, and a smaller weight is assigned to the edge. A separate social network is built for each ambiguous name. Each occurrence of a ambiguous name in the set of sources (i.e. each Mention) leads to a *separate node* in the graph. Therefore, ambiguous names generate multiple nodes in the graph. The overall goal of the method is to partition such graph so that mentions corresponding to the same entity end up in the same connected component. To achieve that goal, a random walk is started from each ambiguous name observation. The probabilities of transition between to nodes are normalized taking into account the weight assigned to each node. The random walk continues either until another ambiguous node is found or until a maximum number of step is reached (the limit was 50). When the random walk finishes the probability of reaching an ambiguous node from any other node have been calculated. These probabilities are taken as similarity levels between two (ambiguous) nodes for a single link clustering procedure. Edges corresponding to similarity levels below a given threshold are removed. Each of the resulting connected components will contain disambiguated mentions to a given entity.

Malin used a subset of the data contained the Internet Movie Database (IMDB), which holds references to manually disambiguated entities (each entity in IMDB has a unique id). The subset chosen includes about 37000 movies with more than one author made between 1994-2004 (10 years) and includes about 180000 distinct entities. To create an ambiguous set, entities were replaced by their last names, resulting in about 85000 distinct names. The resulting set included about 12000 names ambiguous between two entities. The F-score was used to evaluate the performance of the clustering methods. For each cluster produced, c_j, Recall and Precision figures are computed. Let t_j be the set of sources in cluster c_j, e_i be a given ambiguous entity and s_i be the set of sources which mention the e_i entity (using the ambiguous name). Then Recall is calculated as $R(e_i, c_j) = |s_i \cap t_j|/|s_i|$ and Precision is calculated as $P(e_i, c_j) = |s_i \cap t_j|/|t_j|$. These values are used to calculate a F-score for each pair (e_i, c_j). For each entity e_i the cluster with highest F-score, $\max(F(e_i, c_j))$, is chosen as the best match for e_i. An overall F-score for all entities is calculated by doing a weighted average over all the corresponding $\max(F(e_i, c_j))$. Additionally, two simple baselines were created. For one of the baselines all ambiguous names are considered different entities (*AllSingletons* baseline) while for the other all names where considered a single entity (*OneClusterOnly* baseline).

Results show that agglomerative clustering only outperforms the baseline *OneClusterOnly* baseline when the similarity threshold higher than 0.8. On the other hand, if the similarity threshold becomes too high (> 0.99) the results tend to all *AllSingletons* baseline. However, the best results with agglomerative clustering do not go better than F = 0.7. The random walk method seems to perform significantly better, with F-scores higher than 0.8 for a considerable interval of threshold values. It reaches very high performance levels even when the threshold (below which edges are removed) is still relatively low. A performance peak was observed when such threshold is set to 0.12. As the threshold increases, the results of the random walk method tend to the *AllSingletons* baseline. The author conclude that the use of indirect information, through social networks, provides an advantage over using direct features for comparing similarity and achieve disambiguation. They also point out the limitation of their work that results from considering only situations where only one name is ambiguous, which in practice should not be the case since most (all?) the names are ambiguous to a certain extent. Again, because the work addresses a very small data-set with only 12000 ambiguous names and about 37000 documents, authors did not focus on computational efficiency and scalability issues, which is fundamental in order to process web-scale data sets.

8.2.2 Approaches using external knowledge

Bunescu and Pasca present two approaches to disambiguation based on entity information taken from Wikipedia ([BP06]). The authors first create a dictionary of names by extracting information from the title of Wikipedia articles, using a set of heuristics (e.g.: capitalization in title and in article) to select which strings are in fact names. The mapping between the name, q_i, and the article e_k (which refers to an unambiguous entity) is kept, so as to maintain a relation between names and to the entities mentioned by that name. Then for each 1-to-n pairing $\langle q_i, e_k \rangle$, a feature vector is created that includes information about all words found in the Wikipedia article e_k inside a 55-word window around the occurrence of the name q_i (tf-idf weighting is used). This will be the *disambiguation data set*. Mentions found in text will be compared against the disambiguation data set using their corresponding feature vector (built in a similar way from the context). Disambiguation, or rather full entity resolution, is achieved by ranking entities from the disambiguation data set according to the similarity of their feature vectors to that of the Mention to be disambiguated. Vectors are compared using the standard cosine metric.

Because this context-only method seems to be unable to solve many cases, especially in situations where the Wikipedia article used to build the disambiguation vector is too short or incomplete, or when the context taken from the mention contains equivalent but different wordings (synonyms), the authors left this method as a simple baseline method and propose an alternative one. The new method tries to reduce the possible mismatch between the features in context vectors (from mention and from disambiguation data set) by using additional information from Wikipedia categories. The intuition is that certain context words (e.g: "conducted", "concert") are more strongly correlated with certain Wikipedia categories (e.g: "Musicians", "Composers") than with others (e.g. "Professional Wrestlers"), and can thus provide information that allows disambiguating between two entities whose articles are indexed under two different Wikipedia categories (e.g.: a musician vs. a professional wrestler with the same name). A *taxonomy kernel* composed by the combination of the previously defined cosine-based similarity and a weight vector model for the word-category correlation was trained using the SVM-light package. Disambiguation is achieved by projecting the context vector of the ambiguous mention over the disambiguation data set using the learned taxonomy kernel: mentions are considered to refer the top ranked entity in the data set. Entities not found in Wikipedia are dealt with by considering an "out-of-wikipedia" generic entity e_{out}, to which mentions are assigned whenever the maximum similarity obtained by the taxonomy kernel is below a given minimum threshold.

Evaluation of both the cosine metric and the taxonomy kernel was made under four evaluation scenarios. In each scenario the disambiguation data $\langle q_i, e_k \rangle$ is split into two disjoint sets: the training and the testing set. Only ambiguous names q_i were chosen, and names are either in the training set or in the testing set. In scenario S_1, the set of categories considered included only the top 110 level categories (in number of articles) under *People by Occupation* (12288 mentions for training and 48661 for testing). Scenario S_2 generalizes S_1 by considering all categories under *People by Occupation* that have at least 200 articles, which lead to 540 categories (17970 mention for training and 70468 mentions for testing). Scenario S_3 generalizes even further S_2 by including all categories under *People by Occupation* that have at least 20 articles, resulting in 2847 categories (21185 mentions for training and 75190 for testing). To make the evaluation more realistic the authors created scenario S_4 that contained the same categories as in S_2 with an additional 10% of mentions from articles that do not belong to the *People by Occupation* category. Disambiguation performance was measured using an *accuracy value* that is basically the Precision achieved in entity resolution (computed by dividing the number of mentions correctly resolved by the total number of mention to resolve). Results show that the taxonomy kernel clearly outperform the cosine similarity baseline in the first three scenarios (S_1: 77.2% vs. 61.5%; S_2: 68.4% vs. 61.5%; S_3: 68.0% vs. 55.4%), while the improvement was not so substantial in S_4 (84.8% vs. 82.3%). Bunescu and Pasca claim that results do confirm the intuition that word-category correlations help the disambiguation process, but simple methods based on cosine similarity may be good enough for cases where disambiguation decisions are to be made over only two possible target entities, as was the case of most mentions in the disambiguation set.

Another disambiguation approach based on Wikipedia is presented by Cucerzan. In [Cuc07], Wikipedia is used to extract information that will be allow building vector representations of entities against which vector representations of ambiguous mentions will be compared. However, the method does not rely on direct comparison between vector representations of entities and each individual mentions but, instead, it tries to maximize the agreement between *all* the disambiguation hypothesis of *all* mention in the document simultaneously (i.e. disambigaution is formulated as a sort of *global* optimization problem to be solved at document level). Also, this method attempts to solve both name ambiguity (the same name is used to mention different entities) and name variation (the same entity is mention using several different names) problems. The method relies on several relevant pieces of information extracted from Wikipedia. The first step consist of obtaining a mapping between surface forms and entities using the link information in Wikipedia articles. The various names found in internal links pointing to the same Wikipedia article (which represent one unambiguous entity) were collected. This allowed not only finding ambiguous names, but also finding the several alternative name spellings for the same entity. The mapping obtained contained information about 1.4 millions entities with an average of 2.4 names per entity. Then, for each entity found two sets of features were extracted: (i) the related category tags and (ii) the information about entities co-occurring in certain contexts. The titles of the Wikipedia pages containing lists (e.g.: "List of animated television series") were used to tag all entities included in the list. 540k $\langle$entity, tag$\rangle$ pairs were created this way. A larger number of category tags were extracted from the user-added tags attached to each article. Because some of these tags are irrelevant for disambiguation purposes (e.g.: "Articles with unsourced statemets", "1929 births"), some basic filtering was made to exclude noisy tags. 2.65 million (entity, tag) pairs with 139.029 distinct category tags were found.

Context information for each entity was extracted both from the corresponding Wikipedia article and from other articles that point back to it. First, the appositives contained in the title of the article were extracted and saved as contextual features. For example, "TV Series" is extracted as contextual feature for the entity "Texas (TV Series)". The other features were based on information about co-occurring entities. All entities mentioned in the *first paragraph of the article* were included in the context feature set. Additionally, all entities found in the entity's article whose pages point back to it are also considered relevant contextual features. Thirty-eight million (entity, context) pairs were extracted according to these rules. Let T = $\{t_1, t_2, t_3, ..., t_N\}$ be the set of Wikipedia categories tags found and C = $\{c_1, c_2, c_3, ..., c_M\}$ the set of extracted contexts. Then, each entity will thus be represented by a feature vector binary vector $\delta_e \in \{0, 1\}^{N+M}$, indicating the presence (1) or absence (0) of the N tag features feature and M context features.

The actual disambiguation procedure over a given document D is performed after named-entity recognition and in-document co-reference resolution. The systems start by retrieving all possible disambiguation options for each surface form in D, s_j. Each surface form will possibly lead to multiple disambiguation options. Then, all the associated feature vectors δ_{e_i} (one for each possible entity for all disambiguation forms) are combined in a *extended document vector*. The key point here is that within one document, all possible disambiguation forms will converge on one feature centroid for a certain number of compatible features, because incompatible disambiguation options will tend not to contribute with the same features for the *extended document vector*, and will thus become "background noise". Disambiguation is achieved by comparing the vectors of all possible entities for each surface form and maximizing the similarity to the extended document vector.

The method was evaluated using two different test collections. The first collection was a set of 350 of Wikipedia articles from which links to other Wikipedia articles had been taken. The disambiguation procedure implied finding the original links, which pointed to unambiguous entities. The automatic evaluation was made over 5.131 surface forms. The second test collection was a set of 20 news stories containing 629 surface forms for test. In this case, evaluation was made by manually checking the output of the system. In both scenarios, the system was compared with a simple baseline method whose output was the most frequently mentioned entity in Wikipedia for the ambiguous surface form at stake. The disambiguation accuracy for the set of Wikipedia articles was 88.3% for the system and 86.2% for the baseline method. The author points out that the small difference in performance is not significant and that such result is due to the fact that the most of the surface forms in the test set were not ambiguous. The difference between both systems becomes significant when the test is restricted to a smaller subset containing only ambiguous forms. The accuracy obtained in the news stories test set was 91.4% against 51.4% for the baseline method. In this case the system clearly outperformed the baseline.

One interesting point about this work is that it tries to perform disambiguation (resolution) of an open set of entity types, not just People. However, it does fails to illustrate “large-scale” disambiguation since the method was not tried on a large collection.

8.3 A Clustering Approach to NED

In this work we focus on the *disambiguation problem*, that is, the problem of determining whether occurrences of the *same name* in *different documents* refer to the same entity, or to different ones that share the same lexical representation (following standard practice – [GCY92] – we assume that a name inside a document can only refer to one entity). For example, the name “Amsterdam” can be used refer to many different geographic locations, to a novel, to several songs, to a ship, to a pop music band, and to many other entities[4]. We do not address the related problem of *conflating mentions* that use different names to refer the same entity (e.g., “George W. Bush”, “George Bush”, “Mr. Bush”, “President Bush”, “the President”, “Dubya”). Solution to the *name conflation* problem can be built on top of the solution provided for the name ambiguity problem (for an interesting approach to large-scale name conflation check [YE07]).

NED can be formulated as a clustering task. Let m_{ij} represent a *mention*, i.e., the occurrence of name n_i in document d_j, and let $M_{all} = \{m_{11}, m_{21}, ...m_{ik}\}$ be the set of all mentions found in a given document collection $C = \{d_1, d_2, ...d_k\}$. Disambiguation can be achieved by clustering together all mentions in M_{all} that refer to the same entity e_j. The goal is to partition M_{all} in several disjoint clusters of mentions, M_1, M_2, M_3 ... M_n, so that each of them contains mentions that refer to one and only one entity e_j. Also, all mentions of a given entity e_j should end up in a single cluster.

8.3.1 Feature Vector Generation

We start by assuming that a mention of a given name can be disambiguated using information about the names with which it co-occurs *within the same document*. For example, mentions of “Amsterdam” that refer to the capital of the Netherlands will probably co-occur with mentions of “Netherlands”, “Utrecht” or “Rijksmuseum”, while those mentions of Amsterdam that refer to the novel, will probably co-occur with “Ian McEwan” or “Amazon”. Under this assumption, describing mentions using the set of co-occurring names as features ({“Netherlands”,

[4]Check `http://en.wikipedia.org/wiki/Amsterdam_(disambiguation)`

"Utrecht", "Rijksmuseum"...} vs. {"Ian McEwan", "Amazon"...}) should lead clusters that group mentions that refer unambiguously to one specific entity (the capital of the Netherlands vs. the novel).

Let N(d_k) be the set of names found in document d_k. The mention of name n_j in document d_k, m_{jk} will be described by a feature vector of name - value pairs, (n_i, v_i):

$$\overline{m}_{jk} = [\langle n_1, v_1 \rangle, \langle n_2, v_2 \rangle, \langle n_3, v_3 \rangle, ... \langle n_i, v_i \rangle] \tag{8.1}$$

with $n_i \in N(d_k) \setminus n_j$, and v_i being a value obtained through a generic feature weighing function (for example TF-IDF or Mutual Information).

The input for our clustering procedure is an annotated collection of documents, C_{annot}. Therefore, it requires names to be previously *identified* in each document, although *type classification* is not needed.

8.3.2 Clustering Procedure Overview

The procedure we propose for performing NED over a collection of annotated documents C_{annot} starts by extracting all names from each document d_k to generate mention feature vectors $\overline{m}_{jk}$ (a mention is the occurrence of a name in a document). Feature vectors are then grouped by name, so as to have a set of mention feature vectors *per name*: $M(n_j) = \{\overline{m}_{j1}, \overline{m}_{j2} ... \overline{m}_{jx}\}$. Vectors inside each set $M(n_j)$ are then compared according a given comparison strategy and similarity metric $sim(\overline{m}_{nj}, \overline{m}_{nk})$ (e.g: Cosine or Jaccard Distance). Finally a clustering algorithm is applied to each $M(n_j)$, using information about vector similarity computed in previous step.

The algorithm itself is generic in the sense that it does not establish any specific strategy for comparing feature vectors prior to clustering, nor a specific choice for the clustering technique. At this point, we assume only that an efficient algorithm exists for performing vector comparison and clustering. For example, Min-Hash techniques [IM98] provides a efficient way for computing an approximation to the nearest-neighbor problem, which can be used for computing distances between vectors. Clustering by Committee [PL02] and variations of streaming clustering techniques [GMM$^+$03] might be an option for the clustering stage. In any case, one important advantage of this algorithm is that it provides a natural way for distributing computational load. Since feature vectors are grouped by name, all information that is required to resolve ambiguity for each name is aggregated and can be processed separately: both the calculation of vector similarities and the clustering process can be distributed over a cluster of machines, on a per-name basis, thus helping scalability.

8.4 Clustering-based Approaches to Web-Scale NED: Challenges

As explained, the size of the clustering problem at stake - millions of names and thousands of millions of mentions - requires distributed algorithms that can be deployed on large computer clusters. Right from the beginning our method was designed to be run on a Map-Reduce [DG04] platform, a data intensive supercomputing paradigm that simplifies the distribution of data (hundreds of gigabytes) and tasks over thousands of computer nodes (typical commodity computers). Map-reduce provides a generic framework for scaling algorithms to very large data sets but in order to choose an appropriate clustering method for NED, some specific characteristics of the dataset and of the problem should be taken into account.

First, the mention distribution is highly skewed, and is dominated by one or two most popular entities. Thus, the clustering algorithm chosen should be able to handle unbalanced data distributions and still produce correct clusters both from dominant and non-dominant entities. Second, the number of entities in which the set of mentions $M(n_j)$ should be mapped, and thus the final number of clusters, is not known in advance. Therefore, the stopping criteria for the clustering procedure should not depend on a predefined number of final clusters desired, which is difficult to estimate. Instead, it should depend on parameters related with input data and cluster properties.

Finally, web-derived datasets usually involve *sparse*, *high-dimensional* feature spaces. To illustrate this, we took a sample of 2.6 million name-annotated web documents, with 5.1 million mentions annotated corresponding to 52,000 names. We then generated feature vectors according to the procedure explained in section 8.3.1 (the corresponding feature space has dimension 52,000). Table 8.2 shows the number of feature vectors per interval on the number of features. About 63% of the features vectors have fewer than 100 features, and another 20% have between 100 and 199. The vast majority of feature vectors have fewer than 1% of the possible features. On the open Web, the set of possible names - and hence the feature space dimensionality - is on the order of millions, so sparsity becomes especially severe.

In such spaces, comparing items (which is required for clustering) is particularly challenging, not only because of problems arising from high-dimensionality [AHK01], but also because is it very likely that the corresponding feature vectors share only very few or none common attributes, even when the items being compared belong to the *same class*. In other words, two mentions of the same name referring to the same entity, may co-occur with two totally different set of names, and thus the corresponding feature vectors will not share any common attribute. When comparing such feature vectors, the similarities will be zero, despite the fact that they are related to the *same entity*.

Table 8.2: Distribution of the number of features in mention vectors extracted from a 5.2 million documents web sample

# features	# mentions	% mentions
1-99	3235958	63.47
100-199	1022151	20.05
200-299	386880	7.59
300-399	165384	3.24
400-499	84528	1.66
500-599	55905	1.1
600-699	32941	0.65
700-799	28938	0.57
800-899	16997	0.33
900-999	10088	0.2

The impact of these *false negatives* during item comparison can be extremely damaging for the computational performance of clustering algorithms. We showed elsewhere – [SKOU09b] – that *streaming clustering* methods [GMM$^+$03], which because they have linear computational complexity and modest RAM requirements (under ideal conditions) are usually considered viable options for clustering very large date sets, tend to perform sub-optimally when exposed to the *false negative* problem. Therefore, clustering methods for performing NED on the Web need to be both computational efficient and robust to the idiosyncrasies of Web-derived data.

8.5 Efficient Web Clustering by Finding Connected Components

We propose using a graph-based clustering approach for performing NED on the Web. For each name n_j, we start by computing pairwise distances between feature vectors to build the link graph $\mathcal{G}(n_j)$. Two mentions are linked in the graph if their similarity is higher than a given threshold s_{min}. Then, we find the connected components of the Link Graph $\mathcal{G}(n_j)$. The retrieved connected components represent the clusters we seek. The only parameter of this approach is s_{min}; there is no need to set the target number of clusters to be produced. So far we have not yet found an automatic method for estimating the s_{min} parameter. Values used in our experiments range from 0.2 to 0.4.

Naive approaches to building $\mathcal{G}(n_j)$ would attempt an all-against-all comparison strategy. For large data sets that would certainly be infeasible due to time and RAM limitation. However, an all-against-all strategy is not required. If our goal is simply to build the Link Graph for finding the true connected components, then we only need to ensure that we make enough comparisons between items to obtain a *sufficiently connected* graph, $\mathcal{G}_{min}(n_j)$, which has the same set of connected components as the complete Link Graph $\mathcal{G}(n_j)$. This means that $\mathcal{G}_{min}(n_j)$ only needs to contain the sufficient number of edges to allow retrieving the same connected components as if a complete all-against-all comparison strategy had been followed. In the most favorable case, $\mathcal{G}_{min}(n_j)$ can contain only a single edge per node and still allow retrieving the same connected components as in $\mathcal{G}(n_j)$ (built using an all-against-all comparisons strategy). Since efficient and scalable algorithms exist for finding the connected components of a graph ([CLR90], [HT73]), the only additional requirement needed for obtaining a scalable clustering algorithm that is robust to the problem of false negatives is a scalable and efficient algorithm for building the link graph.

The fact that the distribution of mentions among the entities is highly skewed turns out to be advantageous for building the link graph $\mathcal{G}(n_j)$. If we pick mentions randomly from the set $M(n_j)$, for any of the mentions belonging to the dominant entities (one or two) it should be possible to quickly find another one that turns out have a higher than threshold similarity (because there are so many of them). Then, for mentions of the dominant entities, we can obtain a significant decrease in the number of comparisons while almost surely keeping enough connectivity to retrieve the connected components. For mentions belonging to non-dominant entities many more comparisons will be needed to find another "similar enough" mention, since such mentions are, by definition, rare. But since *rare* items are *rare*, the total number of comparisons is still much lower than what is required under a complete all-against-all-strategy.

We use a simple procedure: for each mention keep comparing it with the other items until k_{pos} similar mentions are found, so as to ensure enough connectivity in the Link Graph.

More formally, we will start by shuffling items in set $M(n_n)$ to obtain $M_{rand}(n_n)$. Each mention $\overline{m}_{ni}$ in $M_{rand}(n_n)$ is given a sequential number i. Then, for all the mentions starting with i = 0:

1. take item at position i, $\overline{m}_{ni}$
2. Set j = 1
3. Repeat until we find k_{pos} positive comparisons (edges)
    (a) Compare item $\overline{m}_{ni}$ with item $\overline{m}_{n(i+j)}$
    (b) Increment j

One can show (Appendix A) that the average computation cost under this "amortized comparison strategy" is:

$$\tilde{O}\left(\frac{|M(n_n)| \cdot |E(n_n)| \cdot k_{pos}}{1 - p_{fneg}}\right) \tag{8.2}$$

with $|M(n_n)|$ being the number of mentions for name n_n, $|E(n_n)|$ the number of different entities for that name, p_{fneg} is the probability of false negatives and k_{pos} the number of positive comparisons, corresponding to the number of edges we wish to obtain for each item. This cost is vastly lower than what would be required for a blind all-against-all comparison strategy, without significantly reducing the chances of retrieving the same connected components. Notice that computation cost is rather stable to variation of p_{fneg} when $p_{fneg} < 0.5$. For $p_{fneg} = 0.5$ the cost is just the double of the ideal case ($p_{fneg} = 0$). One can also show (Appendix A) that the expected value for the maximum number of mentions that have to be kept in memory during the comparison strategy, m_{RAM} is equal to $E(m_{RAM}) = k_{pos}/(p_{min} \cdot (1 - p_{fneg}))$, where p_{min} is the fraction of mentions that correspond to the least frequent entity. If only 0.1% of the mentions to be disambiguated belong to such entity, and if $k_{pos} = 1$ and $p_{fneg} = 0.5$, then $E(m_{RAM}) = 2000$. It is perfectly possible to hold information in RAM for that many vectors with standard computers. Imposing a hard-limit on this value (for e.g. 500 instead of 2000) will mostly affect the connectivity for less represented classes.

Another important property of this strategy is that link graphs produced this way do not depend too much on the order by which mentions are picked up to be compared. One can easily see that, ideally (i.e., given no false negatives), no matter which mention is picked up first, if we were able to correctly identify any pair of mentions of the same class as similar, then the link graph produced would contain approximately the same connected components although with different links. In practice, this will not always be the case because false negatives may break certain critical edges of the graph, and thus make the comparison procedure order-dependent. A possible solution for this issue is to increase the number of target positive comparison, k_{pos}, to create more alternatives to false negatives and thus reduce the order dependency.

8.6 Additional Scalability Issues

There are still some important scalability problems that we need to solve. First, there are so many mentions on the web for the most frequent names that the corresponding feature vectors cannot be simultaneously fit into the RAM of a single machine to perform comparisons between them. For illustration purposes, we present in Table 8.3 the number of documents (hence mentions under our definition) found by Google for a few very frequent, and ambiguous, names (we use the number of possible entities found in the corresponding Wikipedia disambiguation page for each name as a rough indicator of its ambiguity). Second, even if they did fit simultaneously in RAM, processing these very frequent names would require much more time than processing less frequent names (which may have only a few hundred mentions), leading to extremely long tails in the overall processing time. Therefore, we need to break the set of mentions for each name into smaller partitions, each with n_{max} mentions, so that they can be distributed more evenly across machines.

Table 8.3: An illustration on the number of Google hits found on the web for some frequent names (hits may change), and the corresponding number of entities found in Wikipedia.

name	# Wiki Entities	Google Hits ($\times 10^6$)
Paris	90	583
Amsterdam	35	185
Jaguar	34	73.4
Pluto	25	13.8

However, by splitting the data into multiple partitions and placing them in different machines, we loose the ability to compare all mentions that would be required to find appropriate (i.e. complete) clusters. In fact, for each (frequent) name we are breaking the corresponding clustering problem into several *independent* clustering problems. Many of these partitions will produce clusters that correspond to the same entity, and so they need to be merged afterwards. Since after the first clustering pass we should have many fewer clusters than mentions, re-clustering these clusters is certainly a more tractable problem. Clusters can be described by the feature vectors generated from the aggregation of feature vectors of the mentions they contain (e.g., their centroid). Comparisons can then be made using any vector distance metric over such vector descriptions, also on a per-name basis.

After the first stage of clustering, the size of the resulting clusters should also follow a highly skewed distribution. There will be several larger clusters corresponding to the few dominant entities, and many smaller clusters corresponding both to non-dominant entities and to (small fragments of) dominant entities. Taking into account this typical distribution (that we systematically found in our experiments), we developed a dedicated re-clustering procedure to merge results from partitions. This procedure is applied independently for each name, and thus it can be trivially run in parallel. For each name, we group all clusters obtained in each partition and divide them in two groups: Big Clusters, C_{big} and Small Clusters, C_{small}. C_{big} is composed of the 10% biggest clusters produced in the first clustering pass, while all others are included in C_{small}. We then use the following re-clustering strategy:

1. **Pre-assign Small Clusters to Big Clusters:** Start by trying to assign each small cluster to one big cluster. This assignment is made using a nearest neighbor strategy (with a minimum similarity threshold), and thus tends not to make many incorrect assignments, while greatly reducing the total number of clusters. Cluster descriptions are updated accordingly.

2. **Merge Small Clusters:** Try to merge all the unassigned small clusters with each other. The main goal here is to make sure that some of the less represented entities grow into medium size clusters, so they get enough "critical mass" to be kept, even if we simply filter out the smaller clusters. Cluster descriptions are updated accordingly.

3. **Merge Big and Medium Clusters:** Try to re-cluster the medium and big clusters based on only a *few top features*. The intuition is that big clusters can usually be "described" by a small number of features (e.g., their top 3), which will be highly discriminative for the entity at stake. We thus achieve cluster consolidation, while reducing the risk of performing incorrect merge operations due to noisy features.

4. Repeat 2 and 3 to reduce fragmentation.

Note that Big clusters and Small Clusters are never compared simultaneously, (i.e. all-against-all), which avoids the problems that might come from comparing elements of with significant size differences.

8.7 Evaluation Framework

As mentioned in section 4.2.3 there are several methods and metrics can be used for evaluating clustering. We will use the two entropy-based metrics to be computed against gold-standard (which will be described next). For each name, n_j, we will compute:

- $E_t(n_j)$: Cluster Entropy of the obtained (test) clusters over the gold-standard clusters (see Equations 4.23 and 4.24); and

- $E_g(n_j)$: Class Dispersion of the gold-standard clusters over all clusters obtained (see Equations 4.25 and 4.26).

As mentioned before, the advantage of these metrics is that they are dependent only on the distribution of items and not on the sizes of clusters. Ideally, the values of Class Entropy and Class Dispersion should be as close to zero as possible.

For evaluating *mention recall* obtained for each name, $R_m(n_j)$, i.e. the proportion *mentions* from gold cluster that are in fact found in any of the resulting clusters, we will use the *item recall measure* presented in Equations 4.27 and 4.28. Similarly, we can compute $R_e(n_j)$ which measures how many of the *entities* included in the gold standard clusters for n_j are found in the corresponding test clusters. This figure is important because mention distribution among entities is expected to be very unbalanced.

The previous figures are calculated for each name, $n_j \in \mathcal{N}$. For assessing the global performance of the clustering-based NED procedure for all names in $\mathcal{N}$, we need to combine the performances obtained for the individual names, n_i. To do so, we use the arithmetic average of the previous metrics over all names: E_t, E_g, R_m and R_e.

8.7.1 Preparing the Gold Standard

The English version of the Wikipedia served as the basis for developing a gold standard for evaluating NED (although the procedure can be replicated for other languages). The basic assumption is that each article in Wikipedia can be related to one *unambiguous* entity / concept. Let $W_{seed}(n_j)$ be the set of Wikipedia articles found for name n_j (n_j can usually be easily identified by the article title). If the number of articles for n_j is greater than one, then n_j is know to be ambiguous, and each possible entity is unambiguously related to one of the articles.

The set $W_{seed}(n_j)$ can be used as seed for obtaining more documents that unambiguously refer entities mentioned using name n_j. For each page in $W_{seed}(n_j)$, which refers to an unambiguous entity e_k, we find all its immediate neighbours in the web link graph, both inside and outside Wikipedia. These linked pages will probably have mentions of the name n_j, which can be assumed to refer to the same entity e_k described by the Wikipedia article to which they are linked. The output of the expansion procedure is a set of gold clusters for each name, n_j. These gold clusters are a set of pages that mention name n_j and that can be uniquely assigned to one Wikipedia article (which stands for a specific entity). A problem arises when such pages are linked to more than one Wikipedia article that describes entities mentioned by the same name, i.e. to more than one article from the same seed set $W_{seed}(n_j)$. In those cases, we cannot automatically decide which entity is in fact being mentioned, and thus all occurrences of the corresponding name in that document have to be considered ambiguous. Thus, those documents are excluded from the gold clusters for the name at stake (n_j). Using such expansion and filtering procedures, we obtained a gold standard with around 9.3 million mentions for about 52,000 ambiguous names[5].

In Table 8.4 we present the distribution of the gold names in four classes based on the entropy of the corresponding gold clusters. Low entropy values correspond to names where there is clearly one dominant entity to which the vast majority of the mentions belong, while high entropy values are related with names for which mention distribution among entities is less skewed.

8.8 Experimental Setup

In order to investigate how scalable our algorithm is and whether or not NED performance improves as the amount of data to be disambiguated grows, we

[5]This gold-standard information was made available by Nemanja Petrovic (Google @ NY).

Table 8.4: Internal entropy of the names in the gold standard

Entropy	# names	% names
0 to 0.1	768	1.5
0.1 to 0.5	7610	14.5
0.5 to 1	29304	56.0
1 or more	14657	28.0

experimented clustering different portions of a large web collection with over a billion documents (in English). The web collection had been previously analyzed by a wide scope named-entity recognition system [WKPU08], so we were able to use name annotations in each document to produce feature vectors for the clustering procedure. We first took a 1% sample of the complete web collection (randomly choosing 1% of the documents) and we performed the complete NED procedure several times while slowly increasing the value of the s_{min} parameter, i.e. the minimum similarity for two mention vectors to be considered linked in the Link Graph. This allowed us to obtain several reference points for the values of E_t, E_g, R_m and R_e for a 1% sample. We then proceeded by performing NED over samples of different sizes - 0.5%, 2% and 5% - so that we could compare the results with the ones previously obtained for 1%. To allow a fair comparison, we matched the results obtained for the 0.5%, 2% and 5% samples with those obtained for one of the 1% samples with the *closest value for* E_t, i.e., similar "purity" values. Results were evaluated against the gold standard (see section 8.7.1).

All code was implemented in the Map-Reduce [DG04] paradigm and experiments were run in parallel over 2048 machines. Because of limited RAM and load balancing issues, names were divided in partitions of maximum size 3000. For very frequent names, this may lead to a considerable fragmentation, because there can be hundreds of thousand of mentions for such names. Each mention vector was limited to having, at most, 5000 features (i.e., corresponding to co-occurring names in the same document). We use the Jaccard Metric to compare vectors (we previously perform filtering of less significant features based on minimum tf-idf and frequency values). At the end of first stage of clustering, all clusters with less than 5 elements are filtered out to reduce the total number of clusters to be processed in the second stage. This can have obvious impacts on final recall values, if there are too many such small clusters at the end of the first stage.

8.9 Results and Analysis

Table 8.5 contains the values for E_t, E_g, R_m and R_e for the 0.5%, the 2% and the 5% samples, and corresponding values for the 1% samples with the closest E_t obtained. It also presents the value of s_{min} with which each result was obtained, and the *clustering ratio* parameter, C_{rat}, which gives the relation between the number of clusters obtained (after filtering) and the number of clusters in the gold standard.

Table 8.5: Performance metrics for three different comparison scenarios.

%@s_{min}	E_t	E_g	R_m(%)	R_e(%)	C_{rat}
0.5@0.3	0.0003	0.0056	0.024	1.16	1.23
1.0@0.4	0.0001	0.0085	0.055	1.74	1.82
1.0@0.25	0.0042	0.0226	0.135	3.70	2.06
2.0@0.3	0.0042	0.0312	0.294	5.43	3.27
1.0@0.2	0.0103	0.0212	0.186	5.00	2.18
5.0@0.3	0.0140	0.0797	0.912	12.4	6.91

One first observation is that for keeping the values of E_t comparable, the s_{min} parameter of the larger sample has to be higher than that of the smaller sample. This was expected, because as the number of mentions to be disambiguated increases, the corresponding vector space tends to become more dense. Thus, in order to avoid noisy clusters we need to increase s_{min} to make sure that only mention vectors that are really close in the vector space actually become linked in the Link Graph, and thus generate pure clusters. Increasing s_{min} should, however, lead to higher fragmentation and to producing many small clusters. The C_{rat} parameters increases both when the size of the sample increases, and when s_{min} increases for the same sample size (the 1% sample), which confirms that fragmentation does in fact increase.

Recall values, R_m and R_e, seem very low. However, one has to take into account that the number of gold standard documents in the sample is proportional to the sample size. Thus, for the 1% sample, recall values cannot be higher than 1% (if sampling is unbiased as we expect it to be). We are primarily interested in observing the relative changes of recall with sample size. For that, we computed the ratios between the recall figures (R_m and R_e) obtained for the larger and the smaller samples that are being compared in each pair of rows. Table 8.6 shows the value of these two parameters $r_m^{+/-}$, $r_e^{+/-}$ for the three comparison situations. For the *0.5% vs 1%* and the *1% vs 2%* scenarios, we can see that even with better

Table 8.6: Ratio between Recall values R_m and R_e of larger and smaller samples.

% vs %	$r_m^{+/-}$	$r_e^{+/-}$
0.5% vs. 1.0%	2.28	1.5
1.0% vs. 2.0%	2.17	1.48
1.0% vs. 5.0%	4.9	2.48

(i.e., lower) values for E_t, the mention recall R_m increased faster than the data size; in both cases the recall ratio $r_m^{+/-}$ is higher than the data increase ratio (twice as many documents). For the *1% vs 5%*, the 5-fold increase in the number of documents did not lead to a 5-fold increase in R_m, although it almost did. However, if we look at the $r_e^{+/-}$ ratio for the entity recall, we see that it is not increasing as fast as the data size is, meaning that we are losing entities (found in the gold standard) as we process more data. The combination of these two factors indicates that for the entities being kept we are able to cluster more and more mentions, but we are losing all the mentions for some more obscure entities. Additionally, recall ratios are systematically decreasing as we increase the size of the data sets to be disambiguated. We believe that there are two main reasons for this.

The first reason is a consequence of the compromises we had make in our algorithm to allow it to process web-scale collections. As we increase the size of the sample, and thus the number of mentions to be disambiguated, the number of partitions made for each name also increases (each partition has 3,000 mentions). The overall clustering problem is thus divided into a large number of smaller independent clustering problems whose solutions should ideally be merged in the re-clustering stage. However, for less frequent entities, the partitioning procedure will disperse the mentions over too many partitions, which, in combination with high values for s_{min}, will lead to generation of more but much smaller clusters. Chances are that most of these clusters end up being filtered out after the first stage of clustering and do not even get the chance of being merged in the second clustering stage. Since our gold standard contains some quite exotic entities mentioned in Wikipedia that are probably under-represented in the web collection, the corresponding clusters will be relatively small and will eventually be completely filtered out. This progressively affects R_t, and also R_m, as we the sample gets larger, compensating possible positive effects that would result from having more data and a more dense vector space. These positive effects were only visible when partitioning was not too problematic (i.e., for the 0.5%, 1.0% and 2.0% samples).

The second reason has to do with a more fundamental issue for NED, and it only became obvious after manually inspecting the results for very frequent names, such as "Amsterdam". As we increased the size of the data to be disambiguated, and s_{min} accordingly, we noticed that results for such type of names were composed of many clusters concerning the several possible entities, as expected, but for the dominant entities at stake (for example Amsterdam, the Dutch capital) there was a surprisingly high number of medium and large clusters. These clusters should have been merged together into a single very large cluster since they all rather obviously (based on inspection of their features) seemed to refer to the same (dominant) entity.

However, each of these clusters appeared to contain mentions that referred to specific *scopes* to which the entity occurs, or to different *facets* that the entity could assume. For example, some clusters referred to "Amsterdam" as *world capital*, for which the typical features of the clusters (co-occurring names) were other large cities of the world, such as "Paris", "New York" or "London", while others clusters would refer to "Amsterdam", a *city in the Netherlands*, and would have as typical features names of cities in the Netherlands. In other cases, the clusters produced had features that apparently were not related to the entity, but that were in fact associated with specific contexts of the entity at stake. For example, since there are many scientific editors based in Amsterdam, we found relatively large clusters whose typical features are names of editors (such as "Elsevier" or "Elsevier Science"), and other names related to scientific conferences and societies. There are many other similar examples, where the clusters refer to distinct possible facets of the entities, such as different geographic scopes or different times in history ("Paris" nowadays v.s during the French Revolution). Interestingly, most clusters corresponding to different and highly specialized facets of a dominant entity contained many more mentions than the "main" clusters of non-dominant entities (e.g. "Amsterdam" the novel, or "Paris" of Troy from Greek mythology).

From a clustering point of view, the different, yet consistent, name co-occurrence patterns that dominant entities exhibit can be seen as distinct "sub-entities", leading to smaller clusters in both clustering stages. The resulting fragmentation effect only becomes obvious when one tries to disambiguate very large and heterogeneous data-sets such as the web: as the size of the corpus increases, more facets of the same entity tend to emerge and make this fragmentation effect more visible. The key point is that, even if we had enough RAM and CPU resources to avoid the partitioning of mentions, fragmentation for these dominant entities would probably still occur. The problem arises from the features used to describe each mention, i.e., the set of co-occurring names, which does not carry sufficient information for merging the existing facets.

Conceptually, this situation is close to the *homonymy* vs. *polysemy* problem ([Kro97]), which is often encountered in word-sense disambiguation tasks. While homonyms have no related senses ("river bank" vs. "bank account"), polysemous words do share some relation ("the Stoic school" vs. "the school room"). In our case, different entities with the same name ("Amsterdam" the city vs. "Amsterdam" the novel) should be seen as homonynmy, while the multiple "facets" found for the same entity can be seen as the multiple "senses" of a polysemous name ("Amsterdam" a world capital vs. "Amsterdam" a city in the Netherlands). Recently, some Named-Entity Recognition (NER) evaluation programs, such as ACE [DMP$^+$04] and HAREM [SSCV06], have recognized the existence of inherently ambiguous situations, especially those that exhibit a more or less systematic pattern. For example, ACE introduced the notion of *geo-political entities* for entities such as countries, that contain a population, a government, a physical location, and a political existence, and that can thus be mentioned by several different facets. However, the large number of possible facets that we observed in our experiments, some quite specialized (e.g. "Amsterdam" as an important city in the field of scientific publishing), does not allow a simple and systematic identification of all relevant cases.

Ideally, we would want to merge all facets belonging to the same entity but still keep information about the distinct facets (whose meaning might be understandable at a later stage). What our results show is that name co-occurrence information is not sufficient for merging facets and that more specialized information is required. For instance, e-mail addresses or biographic features might help merging different facets of people entities, as geographic related information (geo-codes) might help in the case of locations. More generally, web link information might provide good clues for merging facets of arbitrary types of entities. Mentions of the same name in highly connected parts of the web graph indicate that we are probably dealing with the same entity, even if the corresponding mentions have been placed in different clusters. All this additional information might be used in a third clustering stage to merge potential facets (i.e. clusters) of the same entity.

8.10 Conclusion

We have presented a wide-scope NED algorithm that is scalable and explicitly handles the power law distribution of entities in the web, allowing us to cluster a billion mentions. We also presented a novel evaluation strategy that uses information extracted from Wikipedia to automatically generate a gold-standard. Our experiments do not provide a complete solution to web-scale NED. Instead, they raise several fundamental questions (both theoretical and practical) that have so far been neglected by most approaches to NED. We showed that NED on the web involves dealing not only with obvious scaling issues, but with less obvious and more fundamental problems related to the intrinsic variety of web data. As the data volume grows, new facets of entities become apparent, making NED a more complex and less clearly defined task. We showed that name co-occurrence information is not sufficient for merging distinct facets of the same entity.

Part IV

Labels, Web 2.0 Tags and other User-Generated Keywords

Contrary to what we presented in parts II and III, in part IV we focus on lexical items that are less typical of traditional text but which are very popular in the Web: *Labels*, *Web 2.0 Tags* and other types of *user-generated keywords*. These types of lexical items are essentially used with a *functional* purpose. *Labels* are assigned by *content developers* (e.g. newspaper editors) to their *own* content (e.g. news items) to help indexation and subsequent retrieval. *Tags* are used for characterizing possibly complex media elements (e.g. music tracks, videos), and are assigned *and* updated by the *Web community*. Other types of *user-generated keywords* are used with other functional goals, ranging from "hidden" terms placed in web pages for influencing search engine rankings, to keywords assigned to ads so that these can be properly matched to web pages or queries.

Apart from their essentially *functional nature*, these types of lexical items share another two characteristics, namely the *loosely-defined semantics* and the *minimalistic syntactic system*. Each individual keyword – either a label, a tag or other user-generated keyword – can virtually represent any type of *referent*, or can describe any *property* the corresponding referent. Obviously, these two characteristics have the advantage of making keywords very simple to use for all types of users. This is probably the main reason for their global dissemination over the Web.

However, the loose semantics and the almost absence of syntax create a series of problems. One of those problems is *inconsistency in usage*: different users exhibit different keyword usage behaviours. Actually, the problem is even more complex since individual users also show inconsistencies in how they use keywords. For example, when assigning a topic label to a specific news item, a journalist can either assign one label characterizing the *broad subject* of the news (e.g. "Sports"), or assign one characterizing it in a more specific way (e.g. "African Cup"). Or, instead categorizing the topic, the journalist may assign a label that characterizes a different *facet* of the news, such as for example the *location* of the event reported. In any case, the choice is left to the journalist, and there is no guarantee that he/she will be consistent in similar situations.

Another important characteristic of user-generated keywords is the inevitable presence of many *idiosyncrasies*. Besides frequent *spelling mistakes*, and irregular use of common writing conventions (e.g. *non-standard capitalization*), users tend to use a lot of creativity when assigning keywords (e.g. *swapping characters* while keeping equivalent phonemes, use of *excessive punctuation*, use of *emoticons*, etc.). In fact, these idiosyncrasies are such an intrinsic part of this type of lexical items that users already account for them. For example, when assigning trigger keywords to web ads, advertisers often include keywords with spelling mistakes in order to match potential spelling mistakes by the users (e.g. in a search query).

In this part, we address some issues related to this type of user generated contents. First, in chapter 9, we focus on *topic labels* assigned to news items by editors. The problem here is that topic label assignment is performed in a very inconsistent way: very similar news items (possibly covering the same event) are labelled differently (though compatibly) by different news sources (or even by the same news source). Different labels can result from different topic assignment policies, or from different spelling conventions. In any case, when trying to use label information to learn *automatic topic classifiers*, we face several problems. First, there is an extremely large number of distinct labels, (i.e. classes) and the vast majority of them have only a few positive examples. From the point of view of classification, this is already quite problematic. Additionally, since label assignment tends to be inconsistent, very similar items can be found in "different", although in practice similar, classes. This inconsistent item-to-class assignment creates even more problems to classification procedures. Thus, we propose a method for solving some of these inconsistencies, and automatically increasing the number of labels assigned to each news item, so that each class holds more positive examples.

Next, in chapter 10, we look at a different problem: how can we automatically expand the set of the tags assigned by the community to a given media object, in order to improve the (tag) description of the media object? This is specially useful in cases where users have assigned only a few tags to the items. Having less tags, these items tend not to become as accessible to users as others, and, as a consequence, the chances being retrieved and potentially receiving additional tags are lower. This leads to a *starvation effect*. We focus on one specific application scenario: tags attributed to artists by the Last.fm radio community. Our strategy consists in using an *external knowledge resource*, Wikipedia, for finding tags that are *functionally similar* to ones already attributed to media items. Tag information is extracted from the abstract of the Wikipedia article that describes each artist. Evaluation of this tag expansion procedure is made using *diachronic data* from Last.fm, allowing us to check if we can replicate the *real* tagging behaviour of users.

Finally, in chapter 11, we address yet another problem related to the expansion of lists of user-generated keywords. Web advertisers need to assign good *trigger keywords* to their ads. Since this, however, is not an easy task, we propose a method for generating keyword suggestions. Our method mines a database containing a data set of ads previously submitted by advertisers, and infers relations of *functional similarity* between keywords. Then, using these relations, it becomes possible to suggest relevant and non-trivial keywords to advertiser, who only need to provide a few seed keywords. The mining process tries to find keywords that advertisers use alternatively while characterizing ads (i.e. they may assign either one of them to the same ad). We call these keywords *local synonyms*, since they can be replaced by each other in this specific (i.e. local) context. One interesting point in this chapter is the *evaluation methodology*: we compare our keyword suggestion system against a legacy suggestion system in a *real-world* on-line scenario. Also, we propose several novel metrics for evaluating these types of systems in on-line environments. As in chapter 10, we believe that the evaluation methodology presented, and more specifically the set of *performance measures* proposed, is a significant contribution of this thesis.

Chapter 9

Conflating News Topic Labels

In this chapter[1], we focus on a problem related to *text classification*, which has been motivated by the need to perform *fine-grained* topic classification of quotations in an automatic quotation extraction application (see `http://verbatim.labs.sapo.pt`). Our initial goal was to use *topic labels* that editors implicitly assign to news items, by *prepending* those label in the *titles* of some news items, to automatically generate a data set for training a news classifier. However, we soon realized that several inconsistencies in this label assignment process made by editors generated many problems to our approach. In some cases, different *synonym* or *quasi-synonym* label were being assigned to practically equal items (e.g. two news items covering the same issue). In other cases, labels assigned had different *degrees of specialization*. Yet in others, labels assigned to very similar items were related with two distinct possible *facets* (e.g. topic vs. location). Finally, all examples where only *partially labelled*: only one label is assigned by the editors when several labels are possible. As a result, *item to class* assignment is *incomplete*, *inconsistent* and *very fragmented* (the number of classes is almost on the same order of magnitude as the number of items).

In this chapter, we investigate how to solve some of these problems. Our strategy consists in introducing a *pre-processing stage*, before generating the training set, in which we try to *propagate* label information among news items. This should help us increase the number of labels assigned to each item. As a consequence we can increase the number of *positive examples* for each label (i.e. class), and thus reduce the fragmentation problem. Also, in this way it becomes possible to identify distinct labels that are not *incompatible*. Some of these non-incompatible labels are *domain-specific synonyms*, which can potentially be *merged* in a subsequent step, and thus reduce the number of classes that actually exist in this classification problem.

[1]The material presented in this chapter is based on *Luís Sarmento, Sérgio Nunes, Jorge Teixeira and Eugénio Oliveira "Propagating Fine-Grained Topic Labels in News Snippets"* [SNTO09].

From the point of view of similarity, our problem is that of trying to find *functionally-similar* labels (including *content-similar* ones) by looking at a set of labelled news items. Again, we will support our method in the Vector Space Model. The work presented in this chapter is mainly that of exploring the best features for grounding the discovery of functional similarity in this particular scenario. Our exploration will be made only over *directly observable features* (see section 3.3.1), namely *1-grams*, *2-grams*, *3-grams* and *4-grams* taken from the *title* and the *body* of the news item. Due to the lack of gold-standard resources, we will have to perform *manual evaluation* of our propagation method.

9.1 Introduction

News feed items are usually composed by a title, a short body (1-3 sentences) and some meta information, such as date, source or authorship. In some cases, these items also include a label that describes the *high-level topic* of the news – such as "Sports" or "Politics" – usually associated with a specific thematic section of the news source that published the feed. More focused topic information is *sometimes* found in the title of the news feed item, when the title begins by a label that provides additional fine-grained description of the topic (e.g. "*G20*: spotlight falls on police again"[2] or "*Centres de rétention*: la Cimade dénonce une "mascarade" de l'appel d'offre"[3]).

However, attempts to use such fine-grained topic labels for feed classification face several challenges. First, one can easily find hundreds or thousands of such very specific topic label in a given set of RSS (Really Simple Syndication) feeds, as opposed to only few dozen high-level topic labels that are usually associated with news sections. Second, many fine-grained topic labels can potentially be assigned to a given news source with different *scopes* of specialization. For example, for a news item describing a soccer match between two teams playing the European Champions League, it is possible to find in its title labels such as [Sports], [Soccer], [Champions League], or the name of any of the two teams. In other cases, title labels may express different, yet compatible, *perspectives* about the event covered in the news item. For example, a news source might label news regarding the pirate attacks on ships in the coast of Somalia either by the theme – [Piracy] or [Pirate attack] – by geography – [Somalia] or [Indian Ocean] – or by specific event – [Liberty Sun][4]. For classification algorithms this may represent a problem, since *very similar* items (i.e. RSS feeds from different news source reporting the same event) are considered to belong to different classes (i.e. were assigned different topic/class labels). This problem is worsened by the fact that when labeling with such fine-grained topic labels, it becomes very difficult to maintain

[2]Taken from the `http://www.guardian.co.uk`.

[3]Taken from the `http://www.lemonde.fr`.

[4]Liberty Sun is a U.S.-flagged cargo ship attacked by Somali pirates in April 2009.

a consistent label assignment policy among annotators over time, even inside the *same* news source. If multiple spelling conventions or (domain dependent) synonyms are used, then the classification problem becomes even harder since multiple "equivalent" classes are labelled differently.

In this chapter, we present an approach that tries to reduce the impact of topic label fragmentation for classification purposes. The motivation for this work comes from previous work done on quotation extraction from on-line news feeds [SN09]. In this context, extracted quotations are classified into specific topics (i.e. *someone* said *something* about a *specific topic*), and the fine-grained topic labels found in news titles are used to train the topic classifier. Because of the previously described problems, only the larger classes are kept (i.e. those associated with more frequent topic labels) during training and classification. However, due to the Zipfian-like nature of topic distribution, this strategy leaves out many interesting topics. In a nutshell, we investigate if we can automatically assign additional topic labels to a given news feed in order to improve the description of the feed with labels related with alternative but equally valid *scopes* or *perspectives*, and thus reduce the problem of topic fragmentation.

9.2 Related Work

Most of the work that has some connection to ours has been developed in the field of Topic Detection and Tracking. Pons-Porrata et al. [PPBLRS03] propose an incremental hierarchical clustering algorithm for news with the purpose of organizing news items both in a hierarchy that includes news *topics* (e.g. "Kosovo - Peace Agreement") and *events* that occur under such topics (e.g. "Agreement Sign"). News items are represented by a vector containing three types of features: (i) document terms, (ii) temporal references (dates) and (iii) geo-entities. Document similarity is computed by a function that explicitly takes into account document content and temporal-spatial proximity. Evaluation of the clustering algorithm is performed on a collection of 452 newspaper documents covering 48 topic and 68 non-unitary events. Results show that temporal information is beneficial for the news clustering process while spatial information tends to generate noisy results.

An algorithm for performing *unsupervised* discovery of topics labels from news is presented by Sista et al. [SSLM02]. The system is trained in three steps. First, it finds *descriptive phrases* in the collection of training documents, including names of entities (people, places and organizations) and n-grams with high level of lexical cohesion. Second, an initial set of topic labels is assigned to each training document based on the descriptive phrases they contain. Finally, the Estimate-Maximize procedure is used to find the *support words* associated with each topic. Final topic label assignment is performed using these support words. Thus, a topic label can be assigned to a document that does not include it. Evaluation was performed by manually computing label assignment precision at ranks 1-5 for a sample of 100 documents from two collections of newspaper documents in English and Arabic. Precision values ranged from 96% to 82% for English and 88% to 75% for Arabic. Entity name features were found to contribute only marginally to overall precision.

Bigi et al. [BBH$^+$01] compare five statistic models for topic identification on newspaper text using a text-classification framework approach. Models were trained using a 80M words corpus divided in seven topics, and each news item had either one or *two* manually assigned label. The best results in topic propagation were achieved with an unigram "cache" model that is dynamically updated over time. However, results for all five models in assigning the two correct labels to the corresponding texts were quite modest (precision $< 45\%$).

Overall, our work differs from existing research since we are explicitly developing an unsupervised pre-processing step for optimizing subsequent text classification procedures. As far as we know, this is a novel approach since most news classification systems avoid dealing with such a large number of classes by truncating them to the top frequent ones. Also, contrary to most related works, we are processing very short texts (40-60 words), while dealing with thousands of classes.

9.3 Propagating Topic Labels

We explore the fact that news about the same topics should have similar contents. This should be specially so for news provided by different sources but covering the same event. If two news feeds items are found to have *very similar* content but have lexically different topic labels, we might assume that the topic label of each feed item can be propagated to the other, so that both of them will have one additional (hopefully valid) topic label.

More formally, let $\mathcal{F} = \{f_1, f_2, ...f_{|\mathcal{F}|}\}$ be a set of $|\mathcal{F}|$ news feed items. Each feed item, $f_i = [l_i, t_i, b_i]$, is composed of a *topic label*, l_i, a *news title*, t_i, and a *short text body* b_i (we will ignore related metadata in this work). Let $\overline{f_i}$ be a vector representation of the *content* of the feed item f_i, which includes features extracted from both the title t_i and the body b_i. For example, $\overline{f_i}$ can be a *bag-of words* representation of t_i and b_i, with features weighted by tf-idf [SM86] over the feed set $\mathcal{F}$. The content of two news feeds, f_i and f_j, can be compared by using a vector similarity metric over the corresponding vector representations, $\overline{f_i}$ and $\overline{f_j}$. Using such vector representation and similarity metric, it is possible to obtain all the pairwise similarities between news feeds in $\mathcal{F}$.

Let $\mathcal{N}(f_i)$ be the ordered set of neighbours of f_i, i.e. the set of feeds item in $\mathcal{F}$ ordered by degree of similarity with f_i. Let $\mathcal{N}_1(f_i)$ be the nearest neighbour of f_i in such metric space. $\mathcal{N}_2(f_i)$ will be the second nearest neighbour, $\mathcal{N}_3(f_i)$ the third and so on. For each feed item f_i, let l_{i1}, l_{i2} ... l_{ik} be the topic labels associated with each of its neighbours, $\mathcal{N}_1(f_i)$, $\mathcal{N}_2(f_i)$ ... $\mathcal{N}_k(f_i)$. We will define l_i^{top} as the label of the closest neighbour of f_i whose label is *different* from l_i, given that the *inter-item distance* (or *inter-item similarity*) is lower (or higher) than a pre-defined threshold. Thus, l_i^{top} can be considered the best option for assigning a *new* topic label to f_i. Our method consists in propagating the l_i^{top} to f_i. We will not immediately propagate l_i back to the closest neighbour of f_i, f_i^{ngbr}, because f_i might not be its closest neighbour (there might be other feed items closer to f_i^{ngbr}).

9.4 Experimental Set-Up

We collected RSS feeds from eight distinct mainstream Portuguese news sources for a period of about 6 months (mid November 2008 to mid April 2009). This allowed us to collect 90,780 feeds items. Among these, a significant fraction (30.4%) had titles with the structure we wish to explore, i.e. "[topic label]: remainder of the title...". The use of this pattern in titles varied greatly, ranging from less than 1% in two cases to more than 70% in one news source. We performed additional filtering using a dictionary containing names of entities that are frequently mentioned in news to exclude cases where the title actually refers to a quotation (e.g. "Obama: Economy improving, crisis not over"). In the end we obtained 18,309 news feeds containing a valid topic label corresponding to 3,082 different topic labels ranging from very frequent and generic labels, such as "Futebol/Soccer" (886 items), "Música/Music" (393) or "EUA/USA" (386)", to long tail labels that were found in only one document such as "Cogumelos silvestres/Wild Mushrooms", "Grammy Awards" or "Botox".

News feed items were vectorized by generating n-gram features from the title and the body (we differentiate features coming from the title from n-grams coming from the body). In our experiments we explored 4 different types of features: (i) unigram features, (ii) 2-gram features, (iii) 3-gram features and (iv) 4-gram features. With these four options we wish to measure the balance between a more compact feature set which ignores all word ordering information (i.e. unigram features) and order sensitive yet much sparser feature set (i.e. 4-gram features). In any case, these four options represent a rather straight-forward approach, which does not require any more sophisticated language pre-processing. Features were weighted by tf-idf to demote those that occur in many news items. The resulting vectors were compared using the cosine metric [SM86].

We evaluate the propagation of topic label by manually comparing l_i^{top} with the originally assigned label l_i and consider 5 different possibly correct (C) cases:

C1: Different Perspective – l_i^{top} addresses an alternative perspective of the news (e.g. location vs. time);

C2: Generalization/Specialization – l_i^{top} is a generalization or a specialization of the concept described by l_i (e.g. "Sports" vs. "Soccer");

C3: Non-Obvious Synonym – l_i^{top} is equivalent to l_i but there is no lexical intersection (i.e. words in common) between both labels that would suggest such equivalence beforehand.

C4: Obvious Synonym – l_i^{top} is equivalent to l_i, but there is sufficient lexical overlap between both to make such equivalence rather obvious (e.g. l_i^{top} is lexically included in l_i).

C5: Spelling Variation – l_i^{top} and l_i differ only in minor spelling variations (sometimes one is a misspelled version of the other).

Deciding cases C1, C2 and C3 may require consulting external sources (for example the corresponding complete news item). All other possibilities not listed before are considered incorrect (I).

We compared the results of the label propagation algorithm using the four different options for generating vector features (unigrams, 2-grams, 3-grams and 4-grams). The four runs were configured to obtain almost equal recall figures so that the corresponding values of the precision regarding l_i^{top} could be compared. The relation between desired recall and precision can be controlled by setting a threshold on the minimum value of similarity between feature vectors. The larger this threshold, the less probable is to obtain false positives – which can result from noisy or ambiguous features – and the higher should be the value of precision at the cost of recall. We are mostly interested in looking at the high-precision section of the precision-recall curve since l_i^{top} is intended to be used as class label in subsequent text classification procedures.

9.5 Results and Analysis

Table 9.1 presents results (in %) regarding the propagation of l_i^{top} (cases C1 to C5 described before), for each of the four feature generation options experimented. Manual evaluation was performed over a random sample of 30% of the resulting topic label attributions. The position on the recall vs. precision curve was controlled by manually setting the minimum inter-item similarity threshold parameter in order to achieve high precision ($\simeq$90%) at comparable recall values. The minimum value for the inter-item similarity (i.e. cosine) varied from to 0.2 (for unigram features) to 0.33 (when using 4-gram features). The resulting recall values obtained oscillated between 7.1% and 7.2% for all the runs (approximately to 1,310 l_i^{top} label assignments in a set of 18,309 news feeds), which allowed a fair comparison of the corresponding precision values achieved.

Results show that for equivalent recall figures, precision achieved by using 2-gram and 3-gram features is significantly higher than that obtained using either unigram or 4-gram features. This result is important since it is very common to use unigram features in several news classification procedures, when, as seen, 2-gram and 3-gram features seem to carry more information about the topics. There may be several reasons for this, but the fact that 2-gram and 3-gram features keep the names of most entities mentioned in news feeds intact (specially names of people and organizations) may be particularly significant. On the other hand, on such small text snippets, 4-gram features tend to generate a too sparse feature space which affects the efficiency of feature vector comparison, since it tends to relatively promote the importance of topic-unrelated features, such as common stylistic formulations or language specific fixed-expressions. Also, most label propagations added information either about different *perspectives* (case C1) or different *scopes*. Still for about 5% of the cases the methods propagated *non-obvious synonyms* (C3).

Further analysis of incorrect label propagation revealed two major type of errors, which were common to the four runs. The first, and by far the most common, was related to news about local events. These news usually refer to accidents (e.g. car accident or a burning house) or crimes events (e.g. a shop robbery) that occurred in a specific city or neighbourhood and are usually labeled using the name of that location. In many cases, the topic label that was propagated to these feeds items is actually the name of *another* and unrelated location (e.g. another city). The reason for this lies in the fact that, apart from mentioning different locations, most news about accidents and crime events at local level are actually quite similar in structure and content, i.e. they report very similar occurrences using similar wordings. Also, they include mentions to certain typical entities (e.g. the Police or the Fire Department) and describe common standard procedures (e.g. an arrest). This specific type of errors can probably be avoided by using a geographical ontology to identify such news and either excluding the assignment of incompatible labels (e.g. by taking into account distance of the locations) or by only propagating labels that refer to different perspectives (e.g. "Crime").

The second most frequent type of error was the propagation of an incorrect yet related label. For example, news about the *European* Euromillion Loto received

Table 9.1: Precision (in %) at Recall 7.1%-7.2%.

	unigram	2-gram	3-gram	4-gram
C1	24.9	31.1	33.9	30.7
C2	42.0	39.6	35.5	30.7
C3	6.3	4.8	4.6	6.1
C4	8.7	8.0	11.6	11.6
C5	7.9	10.4	8.3	9.7
$\sum$C	89.8	93.9	93.8	88.7
I	10.2	6.1	6.2	11.3

labels that are associated with the *Portuguese* Loto (which is independent of the european one), or news about a specific soccer club or player were assigned a label about another soccer club or player. This type of errors seems to be more difficult to solve and might require looking to labels other than the top suggested one, l_i^{top}, and using more complex criteria to select the correct label (e.g. minimizing the distance to *several* news feed items *simultaneously*).

9.6 Conclusions

We showed that it is in fact possible to automatically propagate topic label between news items using an unsupervised process based on the content of the news. The majority of labels propagated add relevant *perspective* or *scope* information to items. We also showed that 2-gram and 3-gram features seem to carry more topic-related information, and can thus be used in subsequent classification tasks.

For improving one of the major limitations of this work, the low recall, we can change the label ranking procedure so that information coming from multiple neighbours is combined: if a new label is found in *several* of the nearest neighbours of an item, then it might be considered a good topic suggestion even if it is not the top ranked one. We can also perform multiple iterations of the algorithm to propagate novel label information to nodes which could already be close enough to their neighbours but had the *same* label.

Chapter 10

Expanding User-Defined Tag Sets

In this chapter[1], we address a problem that results from the *distributed* nature of the *tag assignment process* made by the community of users of Web 2.0 sites. Because no uniform tag assignment procedure is enforced, *tag distribution* within such sites is *not evenly balanced*: while the most popular items tend to receive many tags (which then go through several iterations of validation), the least popular items usually have few or no tags, since only a small number of users actually interacts with them, and only a small fraction of these is willing to assign tags. The problem is that less tagged items have less chances of being retrieved, and thus tend to maintain only a small number tags. Such a short-loop *starvation cycle* is difficult to be broken spontaneously.

For solving this cycle we propose a strategy for *automatically assigning additional tags* to items. These tags are supposed to be *functionally-similar* to those already assigned to the items at stake (or other items within the same Web 2.0 site). Tag assignment is made taking into account information *mined* from knowledge resources *external* to the site affected by the problem, but which potentially contain *alternative* information regarding the same *universe of items*.

We will focus on one particular Web 2.0 site that suffers from the starvation effect described before: Last.fm, a web radio community. In this specific scenario, we will try to assign additional descriptive tags (e.g. "indie", "pop rock") to the *artist profiles* indexed by Last.fm, some of which do not have any tags assigned. Information about new tags will be mined from the *abstracts of the Wikipedia articles* that address the artists.

Using this Wikipedia-based method, we will attempt to obtain new tags for a list of artists *crawled* from Last.fm in February 2008. For evaluating the tag suggestions generated by our method, we will compare them with the ones that were actually assigned by *real users* during a period of ten months. The information about the tags assigned by users was by obtained by performing another crawl

[1]This chapter is based on the following paper: *Luís Sarmento, Fabien Gouyon and Eugénio Oliveira "Music Artist Tag Propagation with Wikipedia abstracts"* [SGO09].

in October 2008 and computing the difference in relation to the tag information obtained originally. This evaluation methodology, which involves comparing an automatic tagging procedure with *diachronic data* reflecting the behaviour of *real users*, is, to our better knowledge, an original contribution of this thesis.

Results show that the tag assignment strategy we propose, i.e. mining information from other knowledge resources to complement tag descriptions in Web 2.0 sites, is in fact capable of suggesting *valid and non-trivial new tags* to a significant percentage of artist profiles in Last.fm site, including to those that had only very few tags assigned.

10.1 Introduction

One of the most interesting advances of the Web 2.0 is the possibility for users themselves to add and edit meta-information about content by assigning *descriptive tags* to media items. Such social tagging process leads to the emergence of meaningful textual descriptions of web content that can be extremely helpful in information retrieval tasks, especially when automated content analysis is still not accurate enough (e.g. video or music). However, social tagging mechanisms tend to lead to unbalanced tag distributions: while popular items are abundantly described with tags, less popular items might not have enough tags — both in quantity and diversity — to have meaningful, and stable descriptions. Some authors, such as Schein et al. [SPUP02], Lamere [Lam08] and Turnbull et al [TBL08], refer to this as the "cold start problem". If tag information is used for retrieval, then less popular items will probably be retrieved less times, degenerating in a retrieval *starvation effect*. If tags are used for (e.g. music) recommendation, the most tagged items (e.g. artists) end up biasing recommendations (see Park and Tuzhilin [PT08] and Celma and Cano [CC08]).

In this work we focus on a specific Web 2.0 music site, `http://www.last.fm`, which allows users to tag both artists and songs. As mentioned by Lamere [Lam08], the type of tags that users associate to artists is varied, and includes tags related to: music genre (e.g. "acid jazz"), locale (e.g. "japan"), artist/band structure or instrumentation (e.g. "duo"), personal experiences ("seen live"), opinion ("weird") and all sorts of miscellaneous tags (e.g. "Eurovision"). Figure 10.1 shows the distribution of the number of artists with respect to the number of user-assigned tags for a universe of 583,497 artists, using data taken from Last.fm webservice in February 2008 (see section 10.4). The vast majority (almost 80%) of the artists in this data set has 5 or less tags, and 47% has not been tagged at all. This denotes a "Long Tail" distribution, typical of many social phenomena.

We propose an information extraction approach for tackling the problem of unbalanced tag distribution. We use Wikipedia abstracts (in English) to extract, rank and finally assign *one* additional tag to Last.fm artists, including artists which had not been previously tagged by users. We do not address the subse-

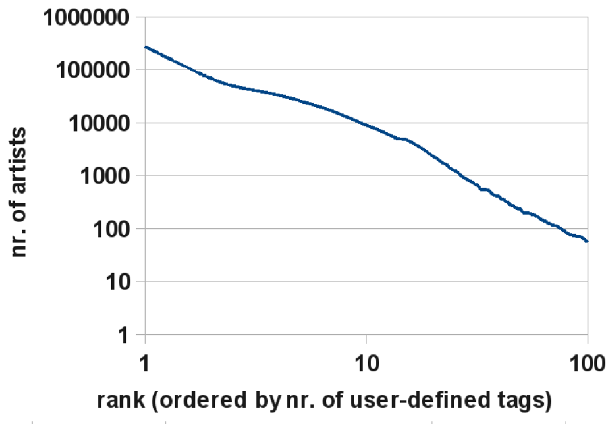

Figure 10.1: Distribution of the number of artist by number of tags (logarithmic scale) in Last.fm (tags per artists are limited to a maximum of 100.

quent problems of tag-based item retrieval or recommendation. Furthermore our method is *community-contained* in the sense that it only suggests tags that are already part of the Last.fm artist tag folksonomy.

It is not obvious that the idea of pulling information from one repository of socially-edited data, such as Wikipedia, and using it to leverage data sparsity in another repository can actually lead to useful results: items with incomplete descriptions (i.e. "not popular") in one social data repository (e.g. Last.fm) might very well be "not popular" in other ones (e.g. Wikipedia). Last.fm and Wikipedia are developed independently and they turn out to represent quite different (musical) universes: about 95% of the artists listed in Last.fm are not matched by any page in the English Wikipedia. On the other hand, this means that Wikipedia may still be helpful for around 5% of Last.fm entries, many of which have still no user-assigned tags. In fact, around 33% of the Last.fm artists for which there is a page in the English Wikipedia have 5 or less tags in Last.fm, and can thus be considered to lie in Last.fm's Long Tail.

Additionally, despite the fact that current work only uses the *English* version Wikipedia, it is likely that a significant number of Last.fm artists, in particular less popular ones (i.e. with few tags) are better described in Wikipedia pages written in other languages. Preliminary tests show that over 1/3 of Last.fm artists present in the Portuguese Wikipedia pages do not have any information associated in English Wikipedia pages. This is also true for Last.fm artists present in the French Wikipedia pages. Analysis and extraction of potential niches of data in repositories of different languages is left for future work.

10.2 Related Work

There are a number of proposals in the literature for suggesting tags to web items (e.g. Xu et al. [XFMS06], Calefato et al. [CGL07], Symeonidis et al. [SNM08]). Tag suggestion is especially important for recommendation systems working with sparsely categorized data (e.g. Park and Tuzhilin [PT08] and Gunawardana and Meek [GM08]). Assigning automatically relevant tags to a specific web item for specific users usually implies discovering inter-user, inter-item or inter-tag similarities, hence calling for some modeling of users and user behaviors (e.g. collaborative filtering) and/or modeling of items, and/or tags (e.g. content-based analysis) ([SPUP02] [SNM08]).

Recent efforts have also been made in the specific domain of music data, including the issues linked to unbalanced tag distributions of music items. For instance, following image tagging research (von Ahn et al. [vAGKB07]), some authors propose to add information to music items via entertaining games (Law et al. [LvAD07]), Mandel and Ellis [ME08] and Turnbull et al. [TLBL07].

Another line of work focuses on propagating tags from popular artists (i.e. tagged frequently) onto other, less frequently tagged artists. A technique to do this is to project artists into a similarity space, and to propagate tags to close neighbours in that space. Such a similarity space can be constructed by content-based similarity computation such as in Sordo et al. [SLC07] or Bertin-Mahieux et al. [BMEML08], i.e. by assuming that music items that "sound alike" should be tagged alike (this has been referred to as *autotagging*). Alternatively, the similarity space can be obtained via *collaborative filtering* or *co-occurrence analysis* (Celma and Lamere [CL07]), by assuming that music items that are commonly found together in different users' playlists or in webpages [SKPW08], should be tagged alike. Hybrid methods combining content and context description have also been advocated (Symeonidis et al. [SNM08]), such as for example.

For a complete review, and list of applications of music social tagging, we refer to [Lam08]. Data and bibliographic links can also be found on `http://SocialMusicResearch.org`.

10.3 Tag Propagation Method

Let $\mathcal{B}_{last}(a_i)$ be the "bag" of user-defined tags found in February 2008 on Last.fm for artist a_i. Our approach consists in using *semi-structured* third-party information sources to perform tag propagation on Last.fm artists. Namely, we mine abstracts from the English Wikipedia for a_i to find relevant tags to be added to $\mathcal{B}_{last}(a_i)$.

For example, the Wikipedia abstract for the band "!!!" is: "*!!! (pronounced as chk chk chk, to simulate mouth-clicking sounds) is an American dance-punk band that formed in autumn 1996 from the former band members of The Yah Mos,* *Black Liquorice and Popesmashers.*" This small text passage contains information that could be used for tagging "!!!", namely [American] and [dance-punk]. From Last.fm page dedicated to "!!!" we can in fact confirm that users assigned both [american] and [dance-punk] to the band (i.e. these tags are indeed included in $\mathcal{B}_{last}$(!!!)). More important cases are those for which there are *no user-defined* tags in Last.fm. For example, at the date of writing, there are no Last.fm tags for the band "Brixx" (i.e. $\mathcal{B}_{last}$(Brixx) is empty), while the corresponding Wikipedia abstract describes the band as follows: *Brixx was a Danish pop group which represented Denmark in the Eurovision Song Contest 1982, in which it sang "Video, Video".* This contains valuable tagging information as e.g. [Danish pop], [Denmark], [Eurovision Song Contest 1982].

Because tags can be extremely diverse in nature, we opted for considering only those tags that have already been assigned to some artist in Last.fm. This can be achieved by building a Tag Dictionary, $\mathcal{T}_{last}$, from tags used in Last.fm, and matching only elements that are part of that dictionary against Wikipedia abstracts. The tag propagation procedure for a given Last.fm artist a_i can thus be performed using the following procedure:

1. Check if there is a Wikipedia article written in English for artist a_i. This is done by matching the artist name with the article title and ensuring that certain music-related words (e.g., "singer", "band", "music", "artist", "composer", "group") are found in the abstract to reduce probability of processing irrelevant / ambiguous names;

2. If a Wikipedia article is found, then try to match tags from $\mathcal{T}_{last}$ on the article abstract. This will create $\mathcal{T}(a_i)$ containing all tags matched.

3. Remove from $\mathcal{T}(a_i)$ all tags already in $\mathcal{B}_{last}(a_i)$ and rank each remaining tag according to a relevance function (see section 10.5).

10.4 Data

Some of the data related to Last.fm radio (artist, users, etc.) is freely available through a dedicated web-service API[2]. For a period of approximately a week (from January 30 to February 4 2008) we consulted Last.fm web-service to obtain a local copy of data concerning artists and their user-defined tags. We obtained basic information for 583,497 artists (name, "popularity" index within Last.fm community) and information regarding 2,774,068 tag attributions. On average we found 4.76 tags per artist, but many artists do not have any tag assigned (see Figure 10.1). There are 208,565 distinct tags, a surprisingly high number. The 10 most commonly used tags are: "seen live" (54,660 artists), "rock" (41,854), "electronic" (33,108), "indie" (27,913), "alternative" (25,401), "pop" (24,010), "punk"

[2]`http://www.audioscrobbler.net`

(20,555), “electronica” (18,781), “metal” (17,419) and “experimental” (16,680). The most frequently used tags describe common music genres. Interestingly, the most used tag is “seen live”, which reflects previous experience of listeners with these artists.

Part of Wikipedia’s content has been converted into tabular format by the DBpedia[3] project [AL07], allowing a simple access to certain parts of the content (e.g. infoboxes) without the need for performing complex parsing operations. In our work, instead of directly consulting Wikipedia articles, we used the *short abstract* data (only those written in English language) provided by DBpedia, which contains abstracts (1-3 sentences) for 2,491,442 entities/concepts identified by Wikipedia page title (the data we used was downloaded on October 20 2008). The short abstract information was chosen because it provides very focused information about the artist, therefore reducing the chances of matching tags that are irrelevant for that artist (e.g. from other artists that are also mentioned in the complete Wikipedia article).

10.5 Experimental Setup

Our Tag Dictionary $\mathcal{T}_{last}$ is composed by 182,556 tags: 71,875 with 1 word, 77,643 with two words and 33,038 with 3 words (all tags were converted to their low-case representation to avoid duplication derived from case variation). We ignored longer tags (4+ words) to optimize the matching procedure. For each tag t_j in $\mathcal{T}_{last}$ we computed $\#_{last}(t_j)$, the number of artists in Last.fm tagged with it by users. This statistic reflects the importance of tag t_j in Last.fm user-defined tag folksonomy. From the initial set of 583,497 artists we matched 28,607 with a Wikipedia abstract. A successful match between an artist and its corresponding article required two conditions to be fulfilled. First, the name of the artist had to match the title of the article. Second, in order to avoid incorrect matches due to ambiguous artist names, the abstract of the article had to contain words that could be directly related to music artists, such as “singer”, “band”, etc. We also excluded abstracts that contained expressions that are usually found in Wikipedia disambiguation pages, such as “may refer” or “may stand”. Only 3 Wikipedia abstracts that fulfilled these conditions did not match any single tag from $\mathcal{T}_{last}$. This was a surprisingly low number but the explanation lies in the fact that there are many very frequent common words among user-defined tags, such as “a”, “for”, “with”, “is”, “he”. Since we have not performed any tag filtering on $\mathcal{T}_{last}$, practically every abstract matched at least one element in $\mathcal{T}_{last}$. However, only 30,114 tags of the 182,556 tags in $\mathcal{T}_{last}$ were matched. For each of the 30,114 tags matched, we computed $\#_{wiki}(t_j)$, the number of Wikipedia abstracts which matched the tag t_j. Table 10.1 shows some illustrative examples of $\#_{wiki}(t_j)$ for several tags.

[3]`http://dbpedia.org/`

Table 10.1: Examples of tags, corresponding number of Wikipedia abstracts matched (in English language) and weights computed by our ranking function.

matched tag t_j	$\#_{wiki}(t_j)$	$w(t_j)$
a	24,480	0.002
band	7,831	0.066
american	3,177	2.088
rock band	2,593	0.247
songwriter	1,133	0.839
new york	517	11.520
country music	471	0.110
heavy metal band	228	0.039
classically trained	36	0.333
italian baroque	10	4.400
traditional instruments	8	1.500
swedish black metal	4	222.75
electro-indie	2	30.500
warehouse raves	1	4.000

Tags were ranked according to the following weighting function, inspired by TF-IDF weighting:

$$w(t_j) = \frac{(n_{wrd}(t_j))^2 \cdot \#_{last}(t_j)}{\#_{wiki}(t_j)} \tag{10.1}$$

with $n_{wrd}(t_j)$ being the number of words of tag t_j (one, two or three words). With this weighting function we seek to:

1. *demote* tags that have been matched with many Wikipedia abstracts (e.g., “a”, “and”, “is”, “the”, “in”, “of”), since they have a very high probability of being noisy;

2. *promote* tags which we know have already been assigned by users to many Last.fm artists, since this means they are relevant within Last.fm tag folksonomy;

3. *boost the relevance* of relatively *long tags* (2 and 3 words) both because they are naturally more informative and have less chances of being noisy (hence the square power).

We could have used a list of stop words to filter out noisy tags. However, we did not proceed to this step as we found defining “noisy tags” a difficult task in the context at hand.

Using this ranking function, the top 5 weighted tags are “seen live” ($w = 218,640$), “female vocalists” ($w = 14,906$), “drum n bass” ($w = 11,124$), “brutal death metal" ($w = 6,951$) and "folk metal" ($w = 6,192$). In the list of ranked tags for each artist a_i, all tags with $w(t_j) < 0.25$ were removed to avoid noisy assignments (albeit excluding some tags such as "band", "rockband", see e.g. Table 10.1). Following these steps, our method could propagate one new specific tag from $\mathcal{T}_{last}$ onto 27,157 artists.

10.5.1 Evaluation

We performed both *automatic* and *manual* evaluation on the *best ranked tag suggestion* only, $t^{sug}_{@1}(a_i)$. Automatic evaluation consisted in comparing $t^{sug}_{@1}(a_i)$ with the new tags *actually* assigned by Last.fm users to artist a_i *between* February 2008 and November 2008 (recall that $\mathcal{B}_{last}(a_i)$ data was obtained in February 2008). For the 27,157 artists for which our method assigned a new tag, we queried Last.fm webservice to obtain *current* tag information. We found new user-defined tags for 20,872 artists (76.8% of 27,157), each having 15.4 new tags on average. Let us call this set of artists $\mathcal{AS}_1$ (for "Artist Set 1"). The remaining 6,275 artists (not further tagged from February 2008 to November 2008) will be called $\mathcal{AS}_2$. For artists a_i in $\mathcal{AS}_1$, the set of new user-defined tags (i.e. those tags that were not in $\mathcal{B}_{last}(a_i)$ but that users assigned to a_i since February 2008) will be named $\mathcal{B}^{nudt}_{last}(a_i)$ ("nudt" standing for "new user-defined tags").

For each artist in $\mathcal{AS}_1$, we computed the precision measure $P^{exact}_{@1}(a_i)$. $P^{exact}_{@1}(a_i)$ is 1 iif $t^{sug}_{@1}(a_i) \in \mathcal{B}^{nudt}_{last}(a_i)$. This measure gives us an indication on whether our system can *replicate* the tagging behavior of Last.fm users during a 10-month period. We also defined the following, more permissive, yet informative precision measures: $P^{all}_{@1}(a_i)$ and $P^{some}_{@1}(a_i)$. $P^{all}_{@1}(a_i) = 1$ iff *all* words of $t^{sug}_{@1}(a_i)$ are comprised in $\mathcal{B}^{nudt}_{last}(a_i)$, such as, for instance, when $t^{sug}_{@1}(a_i)$ is "Punk Rock" and *both* "Punk" and "Rock" are in $\mathcal{B}^{nudt}_{last}(a_i)$. On the other hand, $P^{some}_{@1}(a_i) = 1$ iff *some* words of $t^{sug}_{@1}(a_i)$ are comprised in $\mathcal{B}^{nudt}_{last}(a_i)$, such as when $t^{sug}_{@1}(a_i)$ is "Punk Rock" and *either* "Punk" or "Rock" are in $\mathcal{B}^{nudt}_{last}(a_i)$.

We extended these two last measures to the set of tags $\mathcal{B}_{last}(a_i)$ (i.e. tags assigned before February 2008) instead of $\mathcal{B}^{nudt}_{last}(a_i)$, defining hence $P^{all_{old}}_{@1}(a_i)$ and $P^{some_{old}}_{@1}(a_i)$, so we can measure the relevance of $t^{sug}_{@1}(a_i)$ taking into account *already existing* tags. These five measures were computed on a *mutually exclusive basis*, in the order presented above. For example, $P^{all}_{@1}(a_i)$ is only computed if $P^{exact}_{@1}(a_i)$ was found to be 0. Thus, for artist a_i we can automatically compute a global "tag propagation relevance" measure, $P^{sum}_{@1}(a_i)$, by summing all the above.

For the 6,275 artists in $\mathcal{AS}_2$ (those not further tagged from February 2008 to November 2008) we performed *manual evaluation* for a random sample of 125 artists, i.e. about 2%. Using the information available in Wikipedia and in Last.fm artist pages, we manually computed the precision figure $P^{manual}_{@1}$. $P^{manual}_{@1}(a_i) = 1$ iff $t^{sug}_{@1}(a_i)$ relates to:

1. a possible music genre for the artist, or

2. a specific style/attitude of the artist, or

3. a geographic location relevant to the artist's biography, or

4. relevant relations of the artist with other musical items (other artists, as e.g. former bands, record labels, etc.).

$P^{manual}_{@1}(a_i) = 0$ otherwise, i.e. incorrectly extracted, incomplete, ambiguous, irrelevant or uninformative tags were considered incorrect.

10.6 Results and analysis

Results of automatic evaluation on $\mathcal{AS}_1$ and manual evaluation on $\mathcal{AS}_2$ are presented in Table 10.2.

Table 10.2: Results of automatic evaluation for artist set $\mathcal{AS}_1$ (20,872 artists) and of manual evaluation for a random 2% sample of $\mathcal{AS}_2$ (125 artists).

$\mathcal{AS}_1$	%	# artists (out of 20,872)
$P^{exact}_{@1}$	6.30%	1315
$P^{all}_{@1}$	11.40%	2379
$P^{some}_{@1}$	8.05%	1681
$P^{all_{old}}_{@1}$	18.07%	3771
$P^{some_{old}}_{@1}$	3.21%	669
$P^{sum}_{@1}$	47.0%	9815

2% $\mathcal{AS}_2$	%	# artists (out of 125)
$P^{manual}_{@1}$	56.8 %	71

Global precision is 47.0%. This can be broken down as follows: In 25.75% of the cases, the suggested tag corresponds fully or partially to a tag actually attributed by users in the 10-months period in question. Otherwise, in 21.28% of the cases, some parts of the suggested tag correspond fully or partially to a previously attributed tag. Manual evaluation of artists from $\mathcal{AS}_2$ yields a better result, 56.8%. In addition to global values shown in Table 10.2, in Figure 10.2 we present the detailed performance of our tag suggestion method with respect to the number of tags previously attributed to each artist (i.e. tags in $\mathcal{B}_{last}(a_i)$). The number of tags attributed to a specific artist ranges from 0 to 100 (note however that although the maximum number of tags allowed in Last.fm is set to 100, tags in an artist's "tag bag" can vary over time).

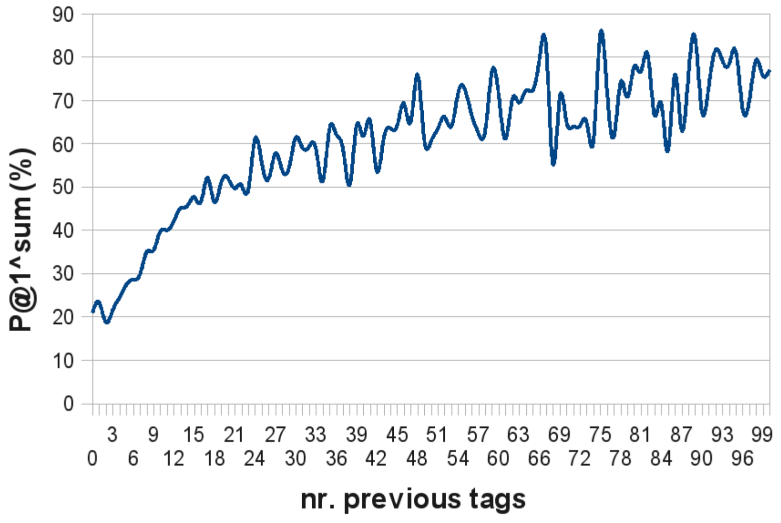

Figure 10.2: Variation of precision with respect to number of previous tags.

10.6.1 Automatic vs manual evaluations

Since there is a difference of about 10 percentage points between the results for $P^{manual}_{@1}$ and for $P^{sum}_{@1}$, we performed additional manual evaluation on $\mathcal{AS}_1$ in order to test whether such difference is due to differences between the two artist sets $\mathcal{AS}_1$ and $\mathcal{AS}_2$, or due to a possibly restrictive nature of the automatic evaluation procedure we proposed. By manually evaluating a 1% random sample of $\mathcal{AS}_1$ (i.e. 208 artists) we found 132 relevant $t^{sug}_{@1}(a_i)$ suggestions, corresponding to a precision value of $P^{1\%}_{@1} = 63.5\%$. This is considerably higher than $P^{sum}_{@1}$, suggesting that our automatic evaluation procedure is probably too strict (see section 10.6.3).

10.6.2 Particular case of Long Tail artists

In Figure 10.2, we can see that our method has higher precision for artists with larger number of tags. For instance, it appears that for artists with 5 or less tags, our method has a precision of only around 20%. However, one should take into account that the automatic evaluation procedure depends on the overlap between the tags that our method suggests and the gold-standard tags (both the newly- and previously-assigned tags). Hence, the probability of matching gold-standard tags is naturally higher for artists with many tags, and lower for artists with very few tags. Therefore, the apparently low precision for artists with few tags shown in Figure 10.2 may be misleading. This led us to manually evaluate the results of our method on a random sample of 100 artists with 5 or less tags in $\mathcal{B}_{last}(a_i)$. This manual evaluation resulted in a precision of 61%, confirming that the automatic evaluation procedure is indeed too strict, especially for Long Tail artists, and that performance of our method on these artists is amenable to that obtained for the "most popular" artists in Last.fm.

Furthermore, the automatic performance measure hides the fact that our data is extremely unevenly distributed: artists with a lot of tags represent a very small portion of the data while artists with very few tags or not at all are the great majority. So, a smaller precision for artists with few tags may still mean many good tag attributions. Thus, it is also interesting to look at our data under a difference angle: considering *only* the 9815 artists to which our method did suggest a correct new tag. Figure 10.3 shows the distribution of those artists with respect to the number of previous tags.

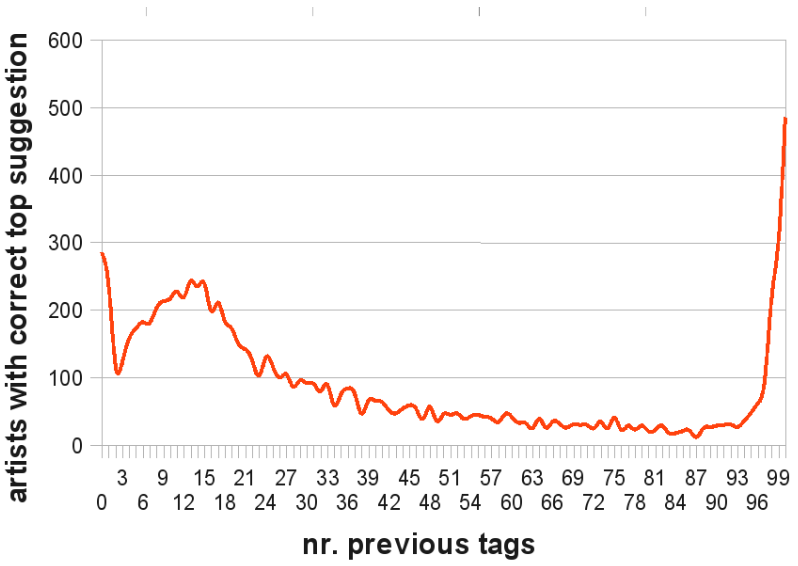

Figure 10.3: Distribution of artists with correct tag suggestions with respect to number of previous tags.

In Figure 10.3 we can observe that when our method does assign new correct tags, it does so mostly at the two edges: (i) artists that had none or very few tags and (ii) artist with many tags (> 98 tags). That is, artists in the Long Tail, as well as "popular" artists. While there might be evaluation-dependent reasons that can explain the performance peak for "popular" artists (e.g. larger overlap with gold-standard), the performance for artists with few tags shows that our method does a relatively good job in the Long Tail, which is precisely where it is most needed.

10.6.3 Further error analysis

With the manual calculation of $P^{1\%}_{@1}$ for a sample of 208 artists and its comparison with $P^{sum}_{@1}$, we argued above that our automatic evaluation procedure is too restrictive, and that the manual evaluation may give a better picture of the actual performance of our method. Indeed, a deeper analysis of the differences between automatic and manual evaluation unveiled many situations (42% of the 208 artists sample) where tag propagation does add novel and relevant information about the artist, but the 5 automatic performance measures fail to score it accordingly, because user tagging behaviour between February and November 2008 has been different (but not necessarily incompatible).

For example, there are several cases where $t^{sug}_{@1}(a_i)$ refers to record labels (e.g. [universal records] for the band "Denver Harbor", or [infectious records] for "The D4"), or to additional activities of the artist (e.g. [project runway] for artist "Heidi Klum", reflecting presence of the artist in a TV program). In other cases the additional tags assigned by users are actually irrelevant while $t^{sug}_{@1}(a_i)$ is correct (e.g. [hip hop] is a valid tag suggested by our method for "Gloria Velez", while Last.fm users have not provided any relevant tag for the artist).

Error analysis revealed that there are two main causes of error: (i) incomplete tag extraction in 26% of the cases, and (ii) incorrect matching of Wikipedia page due to ambiguity in names in 8% of the cases. For instance artist "Ella Koon" was assigned the tag [french], when the relevant tag would be [french polynesia]. However, the tag [french polynesia] does not exist in Last.fm tag folksonomy, so only the known part of it (i.e [french]) was extracted. Another example is the suggestion of [outstanding] to artist "Heinrich Wilhelm Ernst", while the relevant tag would have been [outstanding violinist]. "Oil on Canvas" is a rather obscure band listed in Last.fm but is listed in Wikipedia as a live album by the British band "Japan". Our simple disambiguation mechanism based on frequent music-related keywords, while relatively efficient in avoiding ambiguous names from other domains, is not able to avoid these ambiguous cases inside the music domain.

It is interesting to note that some tags considered correct by our automatic evaluation procedure seem to be relatively *uninformative*. These include both very frequently used (and thus highly ranked) tags such as "new", "music" or "best", as well as relatively obscure, vague or even noisy tags such as "pablo", "sven", "oc", "e", which end up being promoted by our ranking function because they are included in only a small number of Wikipedia abstracts. All these tags *are*, nevertheless, part of Last.fm folksonomy. Other borderline cases are those of "redundant" tags. For example, when the tag [charlotte perrelli] is assigned to artist "Charlotte Perrelli", and [cowie] to "Chris Cowie". These are valid tags (i.e. used by Last.fm users), but we may wonder whether they are really informative in the context of these artists.

10.7 Conclusions

We presented a method for propagating tags mined from Wikipedia abstracts to Last.fm artists which achieved encouraging results. We showed that our method has good performance even for artists that had very few tags previously assigned by users. Therefore, we claim that the proposed method represents a useful contribution for addressing the "cold-start" problem typical of Web 2.0 social tagging environments.

We also illustrated the potential usefulness of transferring information from one socially-edited environment to another. Thus, despite the fact that we have only addressed one very specific scenario, i.e. we used Wikipedia to mine relevant tags for artists in Last.fm, we believe that our method can be generalized in two ways. First, we can use Wikipedia to obtain relevant tag information about other entities (i.e. not just music artists) addressed by other Web 2.0 community sites. Second, we might use alternative socially-edited media, not just Wikipedia, for mining additional relevant tag information (e.g. from blog RSS feeds). With minor changes, we believe that this method can be applied to many other situations.

Chapter 11

The Suggestion of Keywords for Ads

In this chapter[1], we will continue to analyse questions related with less traditional lexical items and, again, we will look at a practical problem: the automatic expansion of a seed set of keywords to help advertisers assign the best keywords to their on-line ads. Due to its very high economical impact on web related business, the problem of keyword suggestion for on-line advertisement purposes has received significant attention lately. Similarly to Web 2.0 tags and topic label, keywords related to ads exist in a very specific semantic space and have their own set of idiosyncratic characteristics. We will formulate the keyword suggestion task on top of the notion of *functional similarity*: suggested keywords are supposed to be *functionally similar*, in the scope of web advertisement, to those provided as seed by the advertiser. We will further refine this idea and introduce the concept of *local synonym* which will provide the basis for our keyword suggestion mechanism. More specifically, local synonyms are lexical items that can be interchanged seamlessly in a well defined scope (i.e. they play the same function in a certain region of the semantic space). In this setting it becomes straightforward to use a Vector Space Model approach for computing these local synonyms based on information about keyword co-occurrence mined from previous ads.

What is not straightforward is how this procedure should be evaluated, since there are no gold standards for this type of tasks, nor it is possible to manually verify the validity of the local synonyms obtained. Thus, we propose a *novel* evaluation framework for this type of systems, which consists in comparing the proposed method against a *legacy method*, in a *real-world* on-line scenario. We will also propose several metrics for quantifying the *usefulness* of method presented from the point of view of the advertiser, and the potential impact on the *revenues* of the ad broker as a result of better trigger keywords.

[1]Most of the material in this Chapter was originally published in *Luís Sarmento, Paulo Trezentos, João Gonçalves and Eugénio Oliveira "Inferring Local Synonyms for Improving Keyword Suggestion in an On-line Advertisement System"* [STGO09].

11.1 Introduction

Internet advertisement is one of the main funding sources for many web services, and is the key component for a cost-free Web for the user. There are two types of ads: (i) *display ads*, which are composed by a banner (possibly animated) and are mainly used by larger advertisers, and (ii) *text ads*, which include a *title*, a short *abstract* describing the product or the service being announced, and *URL* pointing to the web site of the advertiser. In this chapter, we will focus on *text ads* because, typically, display ads are not based in keyword-targeting.

When setting a campaign with text ads, advertisers are asked to associate a list of *keywords*[2] that describe the context in which the ad should be placed[3]. *Keyword Targeted Advertisement Systems*, such as Google's AdWords, places ads in the *result page* of web searches. They try to match search terms with the keywords associated with the ads to select the most relevant ones. On the other hand *Content-Targeted Advertisement Systems*, such as Google's AdSense, focus on placing ads on content-rich sites, like newspapers or web logs. These systems perform content analysis in order to extract descriptive terms that will be matched against the keywords ads, so that the most appropriate ads for the content at stake are selected. Thus, advertisers are specially interested in describing their ads using *more* and *better* keywords so they can increase the number of times their ads are printed and clicked by web users, while keeping their overall *cost per click* (CPC) low, and thus getting a better value for their campaigns. However, because of lack of experience or support, in many cases advertisers end up associating only few keywords (1 to 5) to their ads, which are not enough for describing all possible contexts where their ads would be relevant. Hence, many otherwise relevant and highly focused ads are not even considered for placement because the keywords provided by the advertiser are not enough (in number and diversity) to be matched with the appropriate target contexts (either search terms or descriptive terms extracted from content). When no better option exists, brokers place generic ads, but since these are not really targeted for the specific context at stake they have less chances of being clicked by web users. In a *pay-per-click* scenario, this means less revenue for the broker. In order to reduce this problem, brokers provide *keyword suggestion tools*, such as Google's AdWords Tools[4] or Overture's Keyword Selection Tool[5]. Given an initial list of keywords provided by the advertiser, keyword suggestion tools generate a ranked list of *relevant* and (ideally) *non-obvious* keywords for the advertiser to choose from and associate to

[2]The business terminology for these "keywords" is "*bids*", since they are later involved in a process where ads are chosen to be printed through an *election*.

[3]In fact, keywords (or bids) are usually associated with a *group* of ads that are shown alternatively in order to provide some diversity. For simplicity reasons, throughout this chapter, we will use the term "ad" for referring to such "group of ads".

[4]`https://adwords.google.com/select/KeywordToolExternal`

[5]`http://sem.smallbusiness.yahoo.com/searchenginemarketing/`

its ad, thus enriching the corresponding keyword description.

In this chapter, we present a keyword suggestion mechanism intended for supporting *content-targeted advertisement systems*. The system proposed mines a database containing previously submitted ads to infer similarity relations among the corresponding keywords, and uses such information for suggesting relevant and non-obvious keywords to assist new advertisers. By mining the ads database, we try to identify keywords that can be *interchanged*, i.e. which can be considered *local synonyms* in the universe of known keywords already associated to ads. Suggested keywords are ranked using a function that takes into account both the overlap and the average similarity with keywords provided by the advertiser. The ranking procedure we propose provides an implicit sense-disambiguation, and ensures that suggestions are sense-compatible with the keywords previously given by the advertiser. We perform *on-line experiments* and compare the results of our method with an alternative legacy method. Additionally, We propose several novel evaluation measures, and we show that our keyword suggestion method outperforms the legacy suggestion method on all these measures, in practically all situations.

11.2 Related Work

Briefly, keyword suggestion tools operate according to few different high-level strategies, sometimes in combination. One approach consists in using information extracted from *web search logs*. Keyword suggestions are those *queries* that lexically include some of the keywords given by the advertiser (e.g. "holidays in Corfu" given "Corfu" or "holidays"). The major limitations of this strategy are (i) the inability of the system to suggest keywords that are relevant but lexically dissimilar, and (ii) the inability to filter out suggestions generated from ambiguous seed keywords. Also, obtaining access to web search logs is not trivial. A second type of approaches consists in using existing lexical-semantic resources, such as thesauri, to perform suggestions. The main limitation of this type of methods is the low recall of the suggestion method since existing resources usually have very low recall and coverage (specially for languages other than English) on many important word classes, such as toponyms, names of entities, and words associated with brands or product models (e.g. "Minolta x-700"). A third strategy, which can be applied when the ads database has a large number of ads and enough variety of keywords, consists in mining keyword co-occurrence information from the ads database itself in order to infer relations between keywords chosen by previous advertisers. New keyword suggestions are chosen taking into account such learned keyword relations. This strategy explores the fact that keywords stored in the ads database have been subjected to manual selection, and are thus probably relevant for "similar" ads. In this chapter, we present a keyword suggestion method that follows such strategy. However, this type of methods are unable to suggest keywords out of the set of those that have already been used (although as the ads database becomes larger and more diverse this might not be such a severe problem). To overcome such limitation, some methods try to infer keyword relations by mining the web. In these cases, possible suggestions are found among the keywords that co-occur with user provided keywords on a set of documents / snippets found on the web (e.g. using a search engine). The main problems of this type of strategies are (i) the complexity of the procedure for extracting keywords from web documents (which can lead to many noisy suggestions), and (ii) the less than ideal response times for real-time processing, since pre-processing is not viable.

Joshi and Motwani introduce TermNet [JM06], a graph-based technique for identifying semantic relations between keywords using information extracted from search engines. Given a keyword k_i, the method builds a *characteristic document* from text snippets extracted from the top 50 documents found querying a search engine for that keyword. Text snippets are sentences from those documents containing k_i. Using the corresponding *characteristic documents* it becomes possible to compute the *directed relevance* between pairs of keywords. The authors consider that relevance between two keywords should not be symmetric, in the sense that if k_j is a relevant keyword suggestion for k_i, it does not necessarily mean that k_i is a relevant suggestion for k_j. The directed relevance of k_j to k_i is computed as the frequency of k_j found in the characteristic document of k_i. The resulting directed graph, which expresses the directed relevance between pairs of keywords, is used to perform suggestion. The most interesting keywords for k_i are those whose edges point to k_i. Additional filtering based on *tf-idf* weights [SM86] can be performed to remove very frequent, and thus less interesting, suggestions. Evaluation is performed by comparing the list of keywords generated for 100 keywords by TermNet and by five other systems. Manual assessors where asked to judge the list for *Relevance* and for *Non-Obviousness* (i.e. suggestion does not contain the initial keyword), and metrics of precision and recall for these indicators were computed. For the top 50 suggestion, precision of TermNet is close to 1 for both *Relevance* and for *Non-Obviousness*. When comparing to other methods TermNet performs better in almost all studied indicators.

Abhishek and Hosanagar present *Wordy* [AH07], a similar approach that aims at suggesting keywords that are relevant but at the same time have low frequency and, thus, smaller bidding cost for the advertiser[6]. For a given keyword, the idea is to suggest many relevant words with a frequency/cost as low as possible in order to allow the same effect of using high frequency/cost keyword, but with much less global cost for the advertiser. *Wordy* starts by mining the website of the advertiser to find a set of seed keywords based on their *tf-idf* values. These

[6]This is a rather strong simplification assumption because in practice the relation between frequency and cost of the keyword is not direct. In fact, some low frequency keywords can have very high bidding costs since they are very discriminative of a marketing target.

will compose the initial dictionary D_0. In order to expand D_0, a search engine is queried and the top ranked documents retrieved for each keyword in D_0 are added to the corpus. Filtering the corpus by *tf-idf* allows to expand D_0. As in [JM06], a search engine is queried for each keyword in D_0, and the top ranked documents are used to build the corresponding vector description (again weighted and truncated by *tf-idf*). Pairwise keyword similarity is computed using a kernel equivalent to inner product of the corresponding vectors. The resulting graph is used to perform suggestions: cheaper keywords are found by searching the graph for keywords that are similar and at the same time have lower frequency. The author, however, does not conduct evaluation of the proposed method. In particular, this approach can potentially lead to a decrease in the *conversion rate* (the ratio of users that actually perform the desired action, such as for example purchasing a product), since lower cost keywords may be too far away from the initial context and thus attract users who will find themselves being offered something that they were not expecting.

An alternative approach is proposed by Chen et al. [CXY08]. Instead of using co-occurrence statistics for finding related keywords, the proposed method maps seed keywords onto a concept hierarchy, which is supposed to capture the advertisers goal better than the list of keywords. Once seed keywords have been mapped to the hierarchy, new keywords can be suggested by selecting phrases associated with the closest nodes in the hierarchy. The authors derive the concept hierarchy from the Open Directory Project (ODP), which contains a large set of web pages. First, concepts are derived from the categories of the ODP and are assumed to be organized under a *is-a* hierarchy. Then, phrases associated with the web document under each category are matched with the corresponding concept node (and with all the parents up the hierarchy). A *tf-idf*-like formula is used to compute the degree of association of phrases with each node, ensuring that phrases that are more peculiar to a given category have a higher weight in the corresponding node. Given a list of seed keywords, the method is able to find the nodes in the hierarchy with higher degree of similarity with them (which may include, for example, the common ancestor of several matching nodes), and then suggest the corresponding phrases as keywords. When ambiguous seed words are given (i.e. which may be mapped to nodes with distinct senses, having no common ancestor except the top node of the hierarchy) suggestions can be made separately, thus avoiding list of suggestions that mix multiple senses. Authors have shown this property by comparing results of their method against suggestion generated by Google's, Overture's and WordTracker's suggestion tools for the word "matrix". The authors, however, do not address the issues related with the low concept coverage that usually affects concept hierarchies.

Although not explicitly addressing keyword suggestion, the work by Carraco et al. can be useful while developing keyword suggestion methods [CFLZ03]. The authors propose a method for clustering the *advertisers* $\times$ *keyword* bipartite graph, with the purpose of finding sets of advertisers and sets of keywords that are more strongly connected than the rest of the data set. Such cluster can be seen as *submarkets*, i.e. groups of advertisers that show a common bidding behaviour, and keywords related to the same marketplace. Thus, keywords corresponding to such well defined market places can be suggested to advertisers that have been found to belong to the corresponding group. However, such strong stereotyping will tend to reduce the diversity of keyword suggestion, thus, excluding many otherwise interesting suggestions, specially for those advertisers who do not fit perfectly in a *submarket*. Also, inside each submarket the cost per click for each keyword will tend to increase, since all advertisers will be bidding for approximately the same set of keywords.

11.3 Method Description

In this section, we will describe the keyword suggestion method we propose. Let $\mathcal{A}$ be the set of advertisements composed of $|\mathcal{A}|$ ads $\mathcal{A} = \left\{a_1, a_2, ...a_{|\mathcal{A}|}\right\}$. Each ad can be represented by a tuple of the form $a_i = (t_i, d_i, K_i)$, where t_i is the title, d_i is a short description and K_i is the list of $|K_i|$ keywords provided by the advertiser, $K_i = \left\{k_{i1}, k_{i2}, ...k_{i|K_i|}\right\}$. Given a list of seed keywords $K^0 = \{k_1^0, k_2^0, ...\}$ we wish to develop a *keyword suggestion function* $\mathcal{F}_s$ that generates a ranked list of keyword suggestions k_1^s, k_2^s, ... k_n^s that are both *relevant* and *non-obvious* expansions of the initial K^0 list:

$$\mathcal{F}_s(K^0) = \{k_1^s, k_2^s, ...k_n^s\} \tag{11.1}$$

Let us for now assume that K^{seed} is composed by a single *unambiguous* keyword, i.e. $K^0 = \{k^0\}$. Keyword k^s may be considered a good suggestion for K^{seed} if it is capable of describing the same context described by k^0, or an alternative but *equally relevant* context for the ad at stake. That is, both k^0 and k^s should be comparable in terms of representing equally relevant (but not necessarily the same) contexts for the ad, given the ads domain, $\mathcal{A}$. In other words, if advertisers could *only* associate one keyword from the domain defined by $\mathcal{A}$ to ads, they would chose keyword k^0 or keyword k^s instead with approximately equal probabilities. Keywords k^0 and k^s are, thus, said to be *inter-changeable* in the domain defined by $\mathcal{A}$. From now on, we will use the term *local synonyms* in $\mathcal{A}$, or simply *synonyms*, to refer to keywords that can be inter-changed among ads in $\mathcal{A}$ while maintaining the level of relevancy of the contexts described. We will use the term *synonym* since the behaviour of synonym keywords in the $\mathcal{A}$ domain is similar to the behaviour of synonyms words in common language (i.e. they can be inter-changed without adulterating meaning). Note that in this scope, synonym keywords need not, and probably are not, equivalent to synonyms in the domain of *common language*.

We propose a suggestion method whose definition of relevance is related to the notion of keyword inter-changeability: keyword s is considered a relevant suggestion for an initial seed list $K^0 = \{k_1^0, k_2^0, ...\}$, if it is inter-changeable with one or more elements from K^0. Additionally, the larger the number of elements of K^0 with which k^s can be swapped, the more relevant will k^s tend to be as a suggestion for K^0. In fact, as we will show later, when ambiguous keywords exist in the initial seed list the level of inter-changeability can be used for suggesting correct keywords.

11.3.1 Computing Keyword Synonymy

Two words can be considered synonyms in the domain of keywords associated with ads set $\mathcal{A}$, if they systematically co-occur with the same *previously known* keywords (i.e. with those that have already been selected by advertisers up to that point). The intuition here is that, if two keywords k_i and k_j are in fact inter-changeable, advertisers will associate either one of them to an ad. Thus, assuming that there are ads concerning the same range of product, in some cases advertisers will choose to associate k_i while in other similar ads will associate k_j, while keeping the rest of relevant keywords. Let us assume that we have a dataset composed of a large number of ads, collected over the time. We can compile statistics regarding the co-occurrence of keywords in the lists of keywords associated with each ad, K_j. Let $\overline{c(k_i)}$ be the vector of keyword co-occurrences for keyword k_i, which contains information about the number of ads (or keyword lists K_j) in which keyword k_i co-occurs with all other keywords:

$$\overline{c(k_i)} = [\langle k_1, f_{i1}\rangle, \langle k_2, f_{i2}\rangle, ...\langle k_i, 0\rangle...\langle k_n, f_{in}\rangle] \quad (11.2)$$

Let us also assume that there is a *feature weighting function* $\mathcal{W}$ that is used to ponder features by computing the degree of association between the co-occurring keywords. $\mathcal{W}$ will allow to reduce the relative importance of the co-occurrence of k_i with very frequent, and thus less strongly related, keywords. Let $\overline{c_{\mathcal{W}}(k_i)}$ be the vector that results from applying the feature weighting function to $\overline{c(k_i)}$:

$$\overline{c_{\mathcal{W}}(k_i)} = \mathcal{W}(\overline{c(k_i)}) \quad (11.3)$$

Some common weighting functions have been described in section 3.4. The degree of synonymy s_{ij} between two keywords, k_i and k_j can now be computed by applying a generic vector similarity metric $\mathcal{S}$ to the two corresponding feature-weighted co-occurrence vectors:

$$s_{ij} = \mathcal{S}\left(\overline{c_{\mathcal{W}}(k_i)}, \overline{c_{\mathcal{W}}(k_j)}\right) \quad (11.4)$$

Possible instantiations of $\mathcal{S}$ have been described in 3.5. A keyword synonymy graph, $\mathcal{G}_s$, can be obtained by computing pairwise similarity between all co-occurrence vectors. For some keywords pairs, s_{ij} will be null or very low. Filtering by s_{ij} will allow to keep only the strongest synonym links and remove noisy edges. To allow a more intuitive visual understanding of the graph, s_{ij} weights (corresponding to strength of the synonymy relation) can be substituted by distance weights, which vary inversely with the strength of the synonymy. Thus, nodes with high degree of synonymy will be located closer to each other than nodes with a lower degree of synonymy.

11.3.2 Keyword Suggestion and Ranking

The similarity graph $\mathcal{G}_s$ can be used for keyword suggestion in a straight-forward way. Given a seed keyword, k_1^0, relevant keyword suggestions may be found amongst the nodes of $\mathcal{G}_s$ closest to the k_1^0 node, i.e. those corresponding to keywords with highest synonymy values. Suggestions can be ranked by the length of edges to the seed node: the best keyword suggestions will be located closer to the seed node. Looking at Figure 11.1 this corresponds to saying that $\mathcal{F}_s(k_1^0) = \{k_A^s, k_B^s, k_C^s, k_D^s\}$, ranked by the order shown.

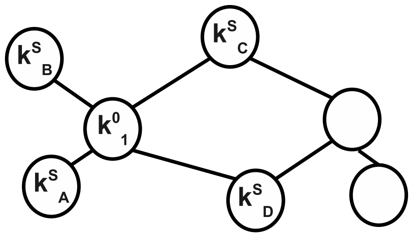

Figure 11.1: Keyword selection and ranking for one seed keyword.

If more than one seed keyword is given then more information describing the goal of the advertiser is available, which should be used for improving keyword suggestions. Obviously, extra seed keywords allow finding *more* relevant keywords since the number of nodes around seed keywords is expected to grow. But more important, with more keywords there is the chance of resolving possible ambiguity related with some ambiguous seed keywords, and suggest only keywords that are related to the sense that is relevant to the advertiser.

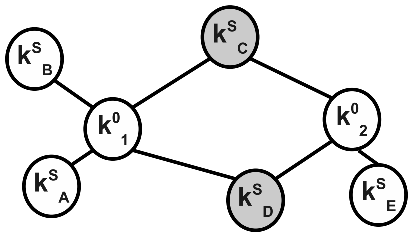

Figure 11.2: Keyword selection and ranking for two seed keywords.

Consider, for instance, that the seed keyword k_1^0 from example given in Figure 11.1 is ambiguous, such as $k_1^0 =$ "orange" (see Definition 2.4.1). It is possible that the set of four keywords suggested may correspond to relevant keywords, but only relevant to one of the different senses of k_1^0 (e.g. $k_A^s =$ "apple", $k_B^s =$ "banana", $k_C^s =$ "yellow" and $k_D^s =$ "red"). If an additional seed keyword is known, k_2^0, then it might be possible to infer which sense of k_1^0 is relevant for the advertiser and exclude incorrect suggestions that are derived from other senses. This can be done by considering that suggestion that correspond to the correct sense of k_1^0 will probably also be synonyms of k_2^0. By intersecting the sets of synonyms of k_1^0 and of k_2^0 we will find such keywords, even if they are not the ones that, individually, have the highest level of synonymy with each of the seed keywords. Figure 11.2 illustrates on such situation (assume for example that k_2^0 = "blue" and k_E^s = "green"), where keywords, k_C^s and k_D^s, despite not being the ones that have higher synonymy with any of the individual seed keywords, are the ones that are synonym with both k_1^0 and of k_2^0, and should thus be ranked higher than k_A^s, k_B^s and k_E^s.

11.3.3 Overview of the Suggestion Procedure

In summary, given a list of seed keywords, $K^0 = \{k_1^0, k_2^0, ...\}$, with one or more keyword seeds k_i^0, and the synonymy graph, $\mathcal{G}_s$, our suggestion algorithm works as follows:

1. for each $k_i^0 \in K^0$ obtain the set of direct neighbours in $\mathcal{G}_s$

   $K_i^n = \{\langle k_{i1}^n, s_{i1}\rangle, \langle k_{i2}^n, s_{i2}\rangle ...\}$

   where s_{ij} represents the degree of synonym between k_i^0 and k_{ij}^n as taken from $\mathcal{G}_s$;

2. merge all sets of neighbours K_i^n in a single set K^N, while computing two parameters for each keyword $k_j^N \in K^N$:

   (a) *keyword overlap*, o_j, as the number seed keywords k_i^0 to which k_j^N is a direct neighbour (i.e. for which k_j^N is found in the corresponding neighbour set, K_i^n);

   (b) *average degree of synonymy*, $avg(s_j)$, as the average degree of synonym between keyword k_j^N and each of the seed keywords to which k_j^N is direct neighbour.

   to obtain $K^N = \big\{\langle k_1^N, o_1, avg(s_1)\rangle, \langle k_2^N, o_2, avg(s_2)\rangle ...\big\}$;

3. Rank K^N first by descending values of o_i and then by descending values of $avg(s_i)$. Keyword suggestion, k_s are taken from the top ranked keywords in K^N.

Therefore, keywords that are synonyms of many seed keywords always rank higher than those that are synonyms of only a few seed keywords. As explained before, this provides implicit filtering against irrelevant suggestion generated from ambiguous seed keywords.

One possible criticism regarding our method is that it might introduce certain distortions in the bidding process, since it suggest keywords submitted by *previous* advertisers to help new advertisers. In a competitive environment, as advertisement bidding is, it would be unfair if new advertisers could benefit from the good keywords ideas that were submitted by a previous advertisers, who would then have additional competitors in the bidding process for such keywords.

However, our ranking method indirectly prevents that from happening. As explained before, suggested keywords that overlap with several input keywords are ranked higher than those that overlap with only a few keywords. This means that, in practice, keywords that have already been submitted by more advertisers, and thus co-occur with more keywords, are placed higher in the ranked lists of suggestions (suggestions ranked lower are simply filtered out). Thus, our suggestion system tends to suggest keywords that have already been submitted by more advertisers, while keeping individual "business secrets" hidden. In certain way, what our keyword suggestion method provides is the "wisdom of the crowd", specially to the less experienced advertisers.

In any case, advertisers that rely *exclusively* on the suggestions proposed, are always more exposed to competitive bidding. Ideally, advertisers should apply a strategy that combines automatically suggested keywords, which represent mainstream traffic, with more "exclusive" keywords, which are less exposed to competition.

11.4 Evaluation: an On-line Method

Let us start by defining *campaign publishing session* as the complete set of steps that an advertiser has to follow to submit one advertisement to the web advertisement system. A *campaign publishing session* includes (i) choosing the name of the campaign, (ii) choosing the content of the ad and (iii) choosing the keywords for the ad, either by typing them in directly, or by selecting keywords provided by the suggestion system. Obviously, the suggestion systems require at least one initial seed keyword provided by the advertiser. Once a list of suggestions has been generated, the advertiser can either pick keywords from a that list or type in new keywords. The advertiser can then request new keywords suggestion, which will be generated using the previously added keywords as new seeds. The process continues iteratively until the advertiser is happy with the list of keywords chosen and performs final submission of the ad.

Performing evaluation of keyword suggestion systems is not trivial, because it is an highly application-oriented task, and because no well established standards exist. Therefore, evaluation can focus on two possible perspectives: (i) semantic (i.e. relevancy) or formal (e.g. "non-obviousness") criteria, and (ii) user feedback (i.e. is the system useful for advertiser?). Most works so far have performed evaluation by manually testing a small sample of suggested keywords to check if they are relevant and non-obvious. When there are ambiguous keywords in the seed lists, tests focus on checking if the system is able to either separate the suggestion by possible senses, or automatically infer the sense that best expresses the purpose of the advertiser. Frequently, authors manually compare keyword suggestions generated by their methods against those generated by Google's AdWord's or Overture's suggestion tools. Instead, we will perform *on-line* evaluation of our method, and compare it against a legacy keyword suggestion system that has been providing suggestions for the `http://anuncios.sapo.pt` web advertisement platform for about 5 years (since March 2004). Evaluation will be based on statistics gathered directly from the behaviour of advertisers. As far as we know, this is the first time that evaluation of keyword suggestion systems for web advertisement platforms based on real usage is reported.

We wish to evaluate our method and compare it with the legacy suggestion system. Moreover, both systems will be assessed regarding (i) the *usefulness* of the method for the advertisers, and (ii) how much On-line impact does the method have in the revenue it is able to generate. More specifically:

- Method Usefulness - Do advertisers choose keywords suggested by the new method frequently? Do advertisers accept suggestions ranked high by the new system, or do they choose suggestion ranked lower? What is the ratio of the number keywords directly typed-in by the advertiser vs. the number of automatically suggested keywords?

- Impact on Revenue - Are ads which received keywords suggested by the new system selected for being *printed* on web pages more frequently than the ones which received keyword suggestions generated by the legacy system? Do they end up being *clicked* more often, and hence generate more revenue for the broker?

Given a legacy keyword suggestion function, $\mathcal{F}_L$, and a new keyword suggestion function $\mathcal{F}_N$, one can perform comparative on-line evaluation by having both systems running in parallel and randomly choosing one or the other to assist the advertiser in a given *campaign publishing session*. This means that the keyword suggestion function is chosen for the *entire* session: all keywords suggested in that session are generated by one and only one of the two competing systems. Let $\mathcal{F}_N$ be activated with probability p_N, and $\mathcal{F}_L$ be activated with probability $p_L = 1 - p_N$. For each new ad, a_i, submitted to the system we keep information about which function used to generate suggestions (either $\mathcal{F}_N$ or $\mathcal{F}_L$). Then, for each individual keyword that the advertiser chooses to associate with the ad, we log if it was directly typed in by the advertiser, or if it was selected from the set of keywords generated by the suggestion system that was previously activated for that specific session.

11.4.1 Evaluating Method Usefulness

For each *campaign publishing session* the following information is logged for all keywords, k_{ij}, associated with ads submitted by advertisers, a_i:

- *source*, $\int(k_{ij})$, whether the keyword was directly typed in by the user ($\int(k_{ij}) = U$), or was suggested by one of the two available suggestion functions, the *legacy* suggestion function, $\mathcal{F}_L$ ($\int(k_{ij}) = L$), or the *new* suggestion function $\mathcal{F}_N$ ($\int(k_{ij}) = N$).

- *rank*, $r(k_{ij})$: the position at which the keyword was ranked in the list of suggestions, when suggested by $\mathcal{F}_L$ or $\mathcal{F}_N$.

- *iterations until selection*, $i(k_{ij})$: the number of the suggestion iterations used by the advertiser until the keyword was added to the ad.

Based on these statistics we can compute several indicators for each ad. The first is the ratio between the number of automatically suggested keywords and the total number of keyword associated with the ad, including those directly typed in by the user. Let $\int_{user}(i)$ be the number of keywords associated with ad a_i directly typed in by the user, and $\int_{auto}(i)$ be the number of keywords suggested by either $\mathcal{F}_L$ or $\mathcal{F}_N$ (only one of the suggestion function is used for each ad). Then, the *automatic suggestion ratio* for ad a_i is given by:

$$\mathcal{S}_r(i) = \frac{\int_{auto}(i)}{\int_{user}(i) + \int_{auto}(i)} \tag{11.5}$$

Values of $\mathcal{S}_r(i)$ will range from 0 to 1. Values higher than 0.5 mean that users are accepting more keywords suggested by the automatic suggestion system than the one that they are typing in. For each suggestion function, a global performance figure, $avg(\mathcal{S}_r)$, can be obtained by averaging $\mathcal{S}_r(i)$ over all ads with keywords suggested by that function. The suggestion function – $\mathcal{F}_L$ or $\mathcal{F}_N$ – that has higher $avg(\mathcal{S}_r)$ can be seen as more *useful* for the advertiser.

Data regarding the position at which the suggested keyword was ranked in the list of suggestions, $r(k_{ij})$, will help to test if ranking procedure is compatible with implicit preferences of the advertiser. If ranking procedure is efficient, top ranked keyword suggestions should be chosen more frequently than the others. For each ad a_i we can compute:

- *average suggestion rank*, $\mathcal{R}(i)$: the average of rank of the suggested keywords selected, $r(k_{ij})$;

- $\mathcal{T}_{@1}(i)$: the fraction of suggested keywords at rank 1 selected by the user;

- $\mathcal{T}_{@10}(i)$: the fraction of suggested keywords up to rank 10 selected by the user;

Again, global performance figures can be obtained for each suggestion function by averaging the previous statistics, $\mathcal{R}(i)$, $\mathcal{T}_{@1}(i)$ and $\mathcal{T}_{@10}(i)$, over all ads with keywords suggested by that system, to obtain $avg(\mathcal{R})$, $avg(\mathcal{T}_{@1})$ and $avg(\mathcal{T}_{@10})$, respectively.

Finally, the number of *iterations until selection*, $i(k_{ij})$, will allow us to evaluate the period during which the suggestion system is capable of presenting *novel* useful keywords that are still relevant to the advertiser. For each ad a_i we can compute $\mathcal{I}(i)$, the average on number of the iterations at which the suggested keywords were chosen (always larger that one). A global performance value for each of the suggestion function can again be computed by averaging $\mathcal{I}(i)$ over all ads with keywords suggested by the system at stake to obtain $avg(\mathcal{I})$.

11.4.2 Impact on Revenue

We also wish to compare the impact of using the new suggestion function $\mathcal{F}_N$ on indicators associated with the revenue that ads are capable of generating. We have the following run-time information available for all keywords, k_{ij}, associated with an ad a_i:

- $\#_{imp}(i, j)$: the number of times that ad a_i was selected for being printed in a web page as a result of the bid associated with keyword $k_{i,j}$;
- $\#_{clk}(i, j)$: the number of clicks on a_i after being printed on a web page as a results of a bid on keyword $k_{i,j}$

For each suggestion function we can compute the following set of statistics regarding the generated keywords:

- *average keyword printability*, $avg(\mathcal{P}_k)$, the average number of ad prints that were made as result of a bid placed on a suggested keyword;
- *average keyword clickability*, $avg(\mathcal{C}_k)$, the average number of clicks made on ads that were printed as result of a bid placed on a suggested keyword;
- *keyword printabilty efficiency*, $\epsilon(\mathcal{P}_k)$, the fraction of suggested keywords that lead the corresponding to ad being printed;
- *keyword clickability efficiency*, $\epsilon(\mathcal{C}_k)$, the fraction of suggested keywords that lead the corresponding ad being clicked;

We can also compute statistics that reflect the overall impact of the suggestion systems on the *ads*. Based on counts $\#_{imp}(i, j)$ and $\#_{clk}(i, j)$ regarding automatically suggested keywords *only*, we can compute:

- *average ad printability*, $avg(\mathcal{P}_a)$: the average number of times an ad is printed as result of an automatically suggested keyword;

- *average ad clickability*, $avg(\mathcal{C}_a)$: average number of times an ad is clicked as a result of an automatically suggested tag;
- *average click through rate*, $avg(CTR)$: ratio between $avg(\mathcal{C}_a)$ and $avg(\mathcal{P}_a)$

From all these statistics, clickability related ones – $avg(\mathcal{C}_k)$, $\epsilon(\mathcal{C}_k)$ and specially $avg(\mathcal{C}_a)$ – are the ones that best reflect the impact of the suggestion system on the revenues of the broker. However, printability statistics – $avg(\mathcal{P}_k)$, $\epsilon(\mathcal{P}_k)$ and specially $avg(\mathcal{P}_a)$ – are also extremely important because they reflect whether suggested keywords help the broker in choosing content-targeted ads instead of printing default ads, which are content-agnostic. As explained in the introduction, the incapability for choosing appropriate content specific ads to be printed due to the absence of keywords provided by the advertisers is one of the main problems that brokers face.

11.5 Experimental Set-Up

For computing keyword synonymy by the process described in section 11.3.1, we used a set of 84,180 ads, compiled over a period of about 5 years, during which the web interface used by advertisers only had the legacy suggestion function available. In average, these ads have 14.14 keywords, but 63% only have one keyword associated and almost 70% have 5 or less keywords associated. Ads with more that 75 keywords (2022) were ignored to avoid *catch-all* ads that have long lists of keywords, most of them semantically unrelated. Ads with duplicate lists of keywords (8434) were also ignored since we considered that they do not bring useful synonym information. For the set of valid ads, we compiled all keyword co-occurrence pairs. We obtained co-occurrence information for a set of 122,099 keywords.

Each of these keywords was represented by a feature vector whose components are the values of the co-occurrence with other keywords (see Equation 11.2). Vectors components were then weighted by Mutual Information [CH90] (see section 3.4.3). Next, all-against-all vector comparison using the cosine metric (see section 3.5.1) was performed in order to obtain the keyword synonymy graph, $\mathcal{G}_s$. For each node (i.e. keyword) we kept only the top 100 closest nodes (i.e. most similar keywords), in order to simplify the link graph.

Given the synonymy graph, $\mathcal{G}_s$, and the ranking procedure described in section 11.3.2 we set up the new keyword suggested function, $\mathcal{F}_N$ running side by side with the existing legacy function $\mathcal{F}_L$ in the web platform dedicated to advertisers. The legacy function $\mathcal{F}_L$ combines three methods for suggesting keywords from the list keywords directly typed in by the advertiser:

1. using the OpenOffice thesaurus for Portuguese[7] to find related words;

[7]Available through `ptopenthesaurus.caixamagica.pt`

2. selecting from the query logs of a commercial web search engine the most frequent search queries that lexically include the user defined keywords (this tends to generate a very high number of suggestions);

3. select those keywords from the ads already in ads database that lexically include user defined keywords.

When starting a new campaign publishing session, one (and only one) of the two suggestion function, either $\mathcal{F}_N$ or $\mathcal{F}_L$, is activated with 50% probability. The activated suggestion function is then used throughout the *entire* publishing session, so that *all* keywords suggested for a given ad come either from $\mathcal{F}_N$ or from $\mathcal{F}_L$.

We kept this configuration running for a period of 15 weeks. Unfortunately, most of advertisers already have predefined lists of keywords for associating with their ads, so only a small subset of them (approx. 5%) actually uses automatic keyword suggestion. Therefore, we only compared 192 ads for which at least one keyword was suggested by either one of the two suggestion function available: 69 ads have keywords suggested by $\mathcal{F}_N$ and 132 ads have keywords suggested by $\mathcal{F}_L$.

11.6 Results and Analysis

We will present statistics considering two possible analysis scenarios: (1) including *all* ads, even those that can be considered *catch-all ads*[8] ($\mathcal{F}_N^{all}$ and $\mathcal{F}_L^{all}$), and (2) by including only ads with no more than 75 keywords ($\mathcal{F}_N^{75}$ and $\mathcal{F}_L^{75}$). Table 11.1 presents some statistics about the ads analyzed in both scenarios, for each suggestion function. We present (i) the number of ads that use each suggestion function, $\#_{ads}$, (ii) the average and median number of keywords associated with each ad, $avg(\#_{kwrd})$ and $med(\#_{kwrd})$, and (iii) the average and median number of keywords *automatically suggested* by the active suggestion system, $avg(\#_{kwrd}^{sug})$ and $med(\#_{kwrd}^{sug})$. Scenario 1 includes about 20-35% more ads than Scenario 2, i.e. it includes those ads that can be considered *catch-all* ads. Also, in both scenarios advertisers tend to select (per ad) more keywords suggested by $\mathcal{F}_N$ than by $\mathcal{F}_L$, even in Scenario 2 in which the average number of keywords per ad is similar.

However, there are less ads with keywords suggested by $\mathcal{F}_N$ (69) than there are ads with keywords by $\mathcal{F}_L$ (132). One possible explanation for this is the inability of $\mathcal{F}_N$ to suggest keywords as a response to "unknown" keywords input by the advertisers (i.e. to keywords that have not been found in ads used for generating the link graph). If the first and only keyword directly provided by the advertiser is one of such unknown keywords, no suggestions can be made, which will leave the advertiser frustrated, making them possibly quit the session. When

[8] *Catch-all ads are those ads to which advertisers have more or less indiscriminately associated hundreds or thousands of keywords in order to cover a wider range of possible contexts.*

restarting the procedure later, there are chances of $\mathcal{F}_L$ being chosen as the active suggestion function, which does not suffer from this problem so severely.

Table 11.1: Statistics about the ads compared under both scenarios.

	Scenario 1		Scenario 2	
	$\mathcal{F}_L^{all}$	$\mathcal{F}_N^{all}$	$\mathcal{F}_L^{75}$	$\mathcal{F}_N^{75}$
$\#_{ads}$	132	69	103	51
$avg(\#_{kwrd})$	51.3	93.2	27.7	27.8
$med(\#_{kwrd})$	6	11	4	8
$avg(\#_{kwrd}^{sug})$	19.1	32.8	7.5	11.3
$med(\#_{kwrd}^{sug})$	3	8	2	6

11.6.1 Method Usefulness

Table 11.2 shows statistics related with how advertisers use suggestion functions. Specifically, it presents values regarding suggestion ratio, $avg(\mathcal{S}_r)$, average suggestion rank, $avg(\mathcal{R})$, the ratio of keywords at rank 1 and up to rank 10 selected by the user, $avg(\mathcal{T}_{@1})$ and $avg(\mathcal{T}_{@10})$, and the average number of the iteration at which suggested keywords were selected, $avg(\mathcal{I})$.

Table 11.2: Statistic regarding advertiser's use of the suggestion function.

	Scenario 1		Scenario 2	
	$\mathcal{F}_L^{all}$	$\mathcal{F}_N^{all}$	$\mathcal{F}_L^{75}$	$\mathcal{F}_N^{75}$
$avg(\mathcal{S}_r)$	**0.37**	0.35	0.27	**0.40**
$avg(\mathcal{R})$	154.9	**129.1**	65.1	**27.6**
$avg(\mathcal{T}_{@1})$	0.09	**0.14**	0.06	**0.32**
$avg(\mathcal{T}_{@10})$	0.17	**0.20**	0.22	**0.49**
$avg(\mathcal{I})$	1.04	**1.32**	1.14	**1.34**

There are significant differences between both scenarios, which confirms that there are in fact good reasons for considering these two separate scenarios. Except for one case, $\mathcal{F}_N$ scores better in all indicators. Statistics $avg(\mathcal{R})$, $avg(\mathcal{T}_{@1})$ and $avg(\mathcal{T}_{@10})$ show that advertisers tend to pick suggestions ranked higher when using $\mathcal{F}_N$ than when using $\mathcal{F}_L$. This is specially so in Scenario 2, where comparison is not biased by the very long lists of keywords associated with catch-all ads. The higher value of $avg(\mathcal{I})$ shows that advertisers requests suggestion more often when using $\mathcal{F}_N$. Statistic $avg(\mathcal{S}_r)$ relates the number of suggested keywords selected by the user with the one he/she directly types and shows a different behaviour.

In Scenario 1, both $\mathcal{F}_N$ and $\mathcal{F}_L$ have very similar and relatively high values for $avg(\mathcal{S}_r)$, which is not surprising since advertisers wishing to create *catch-all* ads will tend to pick as many words as possible more or less indiscriminately. However, in Scenario 2, where advertisers are expected to be more selective, one can see that $avg(\mathcal{S}_r)$ increases for $\mathcal{F}_N$ and drops significantly for $\mathcal{F}_L$. The average number of iterations, $avg(\mathcal{I})$, is always higher for $\mathcal{F}_N$, but this is probably related to the fact that the first iteration of $\mathcal{F}_N$ tends to generate a small number of suggestions if the advertiser has only given one seed keyword. Therefore, in such conditions advertisers will almost always have to request additional suggestion iterations.

11.6.2 Impact on Revenue

Table 11.3 presents statistics about the contribution of suggested keywords to the number of times ads are printed and clicked (i.e. all these statistics take into account *only* data coming from suggested keywords).

Table 11.3: Printability and clickability statistics per keyword suggested

	Scenario 1		Scenario 2	
	$\mathcal{F}_L^{all}$	$\mathcal{F}_N^{all}$	$\mathcal{F}_L^{75}$	$\mathcal{F}_N^{75}$
$avg(\mathcal{P}_k)$	6865.4	**13896**	1065.4	**2679.0**
$avg(\mathcal{C}_k)$	**10.0**	9.7	0.80	**2.35**
$\epsilon(\mathcal{P}_k)$	0.17	**0.18**	0.11	**0.22**
$\epsilon(\mathcal{C}_k)$	0.071	**0.08**	0.048	**0.10**

There are several significant facts in these statistics. First, there is a very large difference in the number of average prints and clicks generated by suggested keywords between both scenarios. This basically suggests that there are a few ads in Scenario 1 that generate an enormous amount of prints and clicks, and that are not included in Scenario 2. We manually verified the presence of a few of such "outliers". The second interesting fact is that the number of prints generated by keywords suggested by $\mathcal{F}_N$ is always much higher. For Scenario 2, such difference is more than 150%. As for the average number of clicks per ad due to suggested keywords, $avg(\mathcal{C}_k)$, the situation is a bit different. While in Scenario 1, $avg(\mathcal{C}_k)$, is quite similar for both $\mathcal{F}_N$ and $\mathcal{F}_L$, in Scenario 2 $\mathcal{F}_N$ scores almost three times as much. If we look at keyword printabilty and clickability efficiencies (i.e. the fraction of suggested keywords that actually generate a print and then a click), $\epsilon(\mathcal{P}_k)$ and $\epsilon(\mathcal{C}_k)$, in all cases keywords suggested by $\mathcal{F}_N$ score higher. Again, in Scenario 2, $\mathcal{F}_N$, outperform keywords suggested by $\mathcal{F}_L$ at least 100%.

Results presented in Table 11.4 summarize the impact of the suggested keywords on the overall printability and clickability of ads. Statistics $avg(\mathcal{P}_a)$ and

$avg(\mathcal{C}_a)$ indicate, respectively, the average number of times that ads are printed and clicked due to a bid on any of their suggested keywords.

Table 11.4: Printability and clickability statistics for ads based on suggested keywords only.

	Scenario 1		Scenario 2	
	$\mathcal{F}_L^{all}$	$\mathcal{F}_N^{all}$	$\mathcal{F}_L^{75}$	$\mathcal{F}_N^{75}$
$avg(\mathcal{P}_a)$	131,279	**450,713**	7,995	**30,309**
$avg(\mathcal{C}_a)$	191.5	**312.1**	5.99	**26.59**
$avg(CTR)$ ($\times 10^{-3}$)	**1.45**	0.69	0.74	**0.87**

Again, in both scenarios these printability and clickability statistics, $avg(\mathcal{P}_a)$ and $avg(\mathcal{C}_a)$, are higher for ads with suggestions from $\mathcal{F}_N$. In Scenario 2, ads with suggestions from $\mathcal{F}_N$ are printed and clicked approximately *4 times more* then ads with keywords suggestions coming from $\mathcal{F}_L$. The click-through rate is also about 17% higher for $\mathcal{F}_N$ ads. However, although in Scenario 1 ads with keywords from $\mathcal{F}_N$ do generate more prints and clicks than those with keywords from $\mathcal{F}_L$, the click-through rate of the last is much higher than of the first. The reason for this is related to a small set of atypical ads. Figure 11.3 compares click-through rates (CTR) at several values for the threshold on the maximum number of keywords for an ad to be considered valid. It shows that CTR values are higher for ads with keywords from $\mathcal{F}_N$ when the threshold is lower than 100 keywords, except for one case (threshold=50). It also clearly shows a peak for $\mathcal{F}_L$ when threshold is set 200, as a result of a few outliers found in that range. Given these results,

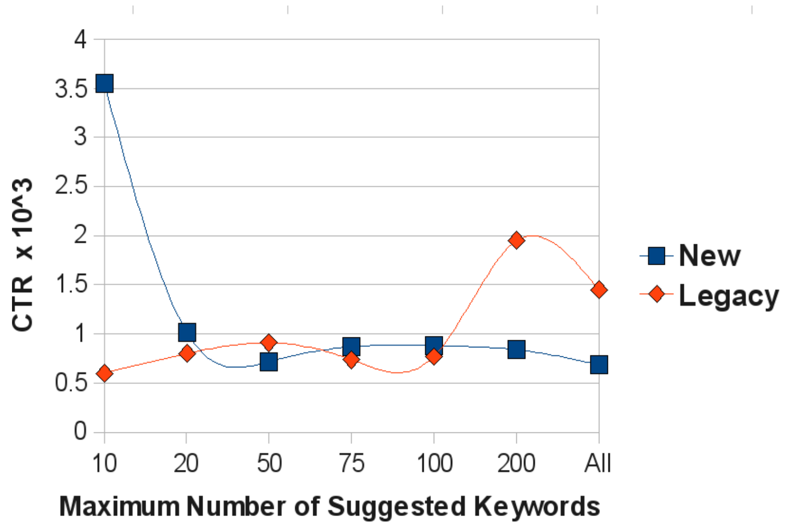

Figure 11.3: CTR for ads with keyword suggested by $\mathcal{F}_N$ and $\mathcal{F}_L$, at several thresholds on the maximum number of keywords.

and excluding the influence of the outliers which we believe would be much less significant if more data was available, it is quite reasonable to say that $\mathcal{F}_N$ can almost always lead to ads with better CTR values than those obtained by ads with keywords suggested by $\mathcal{F}_L$. In any case, from an absolute point of view, keywords from $\mathcal{F}_N$ generate more prints and clicks in both scenarios.

We will now analyse these values taking into account statistics concerning all ads submitted by advertisers during the test period. We will only analyse ads with no more than 75 words. At this threshold, there are 2684 ads with no keywords automatically suggested by any of the two systems. As it is possible to confirm, only a small fraction of ads – 5.42% – have automatically suggested keywords. Notably, ads with no suggested keywords have much inferior average CTR values (0.15×10^{-3}), which shows the usefulness of keyword suggestion systems. During the analysis period, ads with no suggested keywords were printed 3,591,028,887 times and generated 551,864 clicks. We will start by assuming that only the new suggestion method was available and that all the 154 ads with suggested keywords had received the suggestions from the new suggestion method. We can thus admit that they would all have the same average printability and clickability statistics per ad that are seen in Table 11.4 for $\mathcal{F}_N$ (i.e. $avg(\mathcal{P}_a) = 30,309$ and $avg(\mathcal{C}_a) = 26.59$). In such case, these ads would generate 2,298,301 additional prints and 2,120 more clicks. Globally, this represents relative increases of only 0.06% prints but a 0.38% increase in clicks. Although modest at this stage, these numbers could become much more important when the fraction of advertisers using the keyword suggestion mechanism become higher (currently only around 5.0%).

11.7 Conclusion

We presented a keyword suggestion mechanism that mines information from a database of previously known ads in order to infer *local synonymy* information between keywords. The method exploits the fact that keywords previously assigned to ads have already gone through a relevancy selection procedure made by previous advertisers, and uses synonymy information to perform *relevant* (and *non-obvious*) suggestions, while automatically performing implicit sense-disambiguation. We performed *on-line* evaluation of our system by comparing it against a legacy keyword suggestion system. Both systems were exposed to real advertisers during 15 weeks. As far as we know, this is the first study of this type that is reported. We proposed a set of novel performance measures for such an experimental setting. Using these measures we showed that keywords suggested by our system outperform keywords suggested by the legacy system in several parameters related to printability and clickability. Moreover, we showed that ads with keywords suggested by our system are printed more often and clicked more frequently than those with keywords suggested by the legacy system, and that they also tend to have higher CTR values.

Part V

Conclusions

Chapter 12

Conclusions

12.1 Overview

The central question addressed in this thesis was:

How can we decide whether, in a specific context, and according to a pre-defined notion of similarity, the referents of two lexical items are "similar" or not?

The motivation for this work has been the belief, drawn from practical experience, that language processing applications can only be *robust* in *real-world* scenarios if they possess the capability for calculating the semantic similarity between lexical items. We believe that this capability is particularly relevant for language applications operating on the Web, which, due to the large variability of contexts at stake, is an extremely harsh text processing environment. Throughout this thesis, we presented several practical cases that required the computation of a certain notion of similarity for solving a language processing problem. In part I, we addressed mainly background issues. In chapter 1, we presented a conceptual framework that unifies several concepts related to similarity, namely *content similarity*, *type similarity*, *functional similarity*, and also *ambiguity*. This encompassing framework is one of the main contributions of this thesis. In chapter 2, we described the main mathematical tool used in this thesis, namely the Vector Space Model. We compiled and organized a vast amount of information about the Vector Space Model from a wide variety of sources. We proceeded in chapter 3 to discuss methodological issues related to evaluation, and we presented information about *methods*, *metrics* and *gold-standard* resources for the evaluation of semantic-related procedures. Such a comprehensive systematization of issues related to evaluation is another contribution of this thesis. In part II, we investigated questions related to "traditional" lexical items and to *content similarity*. In chapter 5, we focused on the task of automatically finding verb *synonyms* in Portuguese, using a Vector Space Model approach. By performing an *exploration* of several parameters of the VSM, we obtained a clear notion of the feature information that is required for computing verb synonymy. Then, in chapter 6, we turned our attention to *paraphrase*, motivated by the need to identify references to entities made using *job titles*. We presented an *unsupervised* method for inferring job title paraphrasing rules, which allows us to expand a set of known job titles for a given entity with several new paraphrases. Since a very significant portion of the content of the Web is directly or indirectly related to *entities*, in part III we addressed similarity questions related to entities and to their *names*. In chapter 7, we presented a data-driven method for expanding a set of entities with others of the same *type*. The method uses information about how names co-occur in *coordination structures* to detect *type similarity* between the corresponding entities. Next, in chapter 8, we focused on the problem of name *ambiguity*, and we propose a clustering approach to disambiguate mentions on the Web. The scale at which we attempt to disambiguate names, i.e. the Web, not only raised several issues regarding computational scalability, but it also exposed the limitations of naive definitions of the named-entity disambiguation task itself. Finally, in part IV we turned our attention to *Web-specific* lexical items, namely *Labels*, *Tags* and *Keywords associated to Web ads*. We studied different methods for obtaining functionally similar items in three different scenarios. In chapter 9, we studied a solution to the problems that arise from assigning labels to news items in a inconsistent fashion (by editors or journalists), especially those that interfere with subsequent news classification procedures. We investigated which features can be extracted from text to support the inference of *functional similarity* (including *content similarity*) between tags, in order to reduce fragmentation problems. In chapter 10, we presented a method that expands the tag description of media objects published in Web 2.0 sites, by mining *functionally similar* tags in *community-edited* knowledge resources. Last, in chapter 11 we proposed a method for suggesting keywords to Web advertisers using the notion of *local synonyms*. In this case, evaluation was performed by running our keyword suggestion method *in parallel* with a legacy suggestion method and comparing the results.

12.2 Answer to Research Questions

In chapter 1 of this thesis, we raised four research questions, which we now answer based on the conclusions we reached through each experiment presented in this thesis.

1. Can we provide a unified conceptualization that embraces all sub-concepts of similarity found between text elements, and, at the same time, provides solutions to practical questions imposed by the Web?

In chapter 2, we showed that, starting from the definition of some basic concepts, such as *Lexical Item* and *Dynamic Lexicon* (see section 2.2), and from a generic definition for Semantic Similarity (See Equation 2.4), it is possible to formulate additional definitions for several other more specialized concepts related to similarity, namely *Content Similarity* (section 2.3.1), *Type Similarity* (section 2.3.2) and *Functional Similarity* (section 2.3.3). Furthermore, we showed that it is possible to formulate other semantic concepts, such as *ambiguity*, *on top* of the definition of semantic similarity (section 2.4). In parts II, III and IV we presented several approaches for solving problems involving different situations regarding similarity (i.e. Content Similarity, Type Similarity, and Functional Similarity). These problems were related not only to *traditional lexical units*, such as *verbs* and *noun-phrases* (part II), but also with *names of entities* (part III) and *Web-specific lexical units*, such as *topic labels*, *Web 2.0 tags* and *web ads keywords* (part IV). As shown throughout this thesis, the formalization we proposed allows straightforward computational implementations. Since it explicitly includes the role of the *Context* in the definitions of similarity, our framework promotes *data-driven* approaches to solving practical problems. In fact, for solving all problems addressed in this thesis, we followed *data-driven* approaches. This comes as a direct consequence of the encompassing formalization for similarity that we proposed. Thus, the answer to this question is *positive*.

2. How does one deal with web-specific lexical items, such as tags or topic labels? What notions of similarity can be applied to them?

In the parts II, III and IV of this thesis we focused on problems that were motivated by applications intended to operate on the Web, or over web-derived data. These problems arise from the need to deal with a wide variety of situations that can be found on Web data, which challenge the *robustness* of language processing applications, from the point of view of both *recall* and *precision*. In part IV, however, we explicitly focused on *Web-specific* lexical items: *topic labels*, *Web 2.0 tags* and *trigger keywords for web ads*. We saw that, in most cases, the problems being tackled were generically related to *functional similarity*, due to the functional nature of this type of lexical items. We also saw that, in some situations, notions of *content-similarity* are applicable (e.g. synonymous topic labels, or alternative spellings for a technical tag) but in all cases such content similarity is bound to a notion of *local semantics*, which might not always be compatible with the semantics that govern traditional lexical items: a specific similarity relation in traditional text environments might not hold in certain web-specific contexts, and vice-versa. In part III we addressed issues related to *names* of entities, including not only *people*, *organizations* and *places*, but also other types of entities which, despite not being *exclusive* of the Web, have found a prolific ecosystem there. We showed how the notion of *type similarity* is diffuse, and how it needs to be tuned to specific contexts. Also, we showed how problems related to name *ambiguity* are dramatically different on the Web. More specifically, we demonstrated that, because the number of *facets* that each entity might have is dependent on the broadness of the data being analysed (e.g. a few domains vs. a significant fraction of the Web), two mentions of the same entity can be seen as similar enough or not, depending on the size of the text universe being disambiguated. This shows that similarity is dependent not only on local semantics, but also on the number of local semantic structures that co-exist in a given text environment. In conclusion, while traditional notions of similarity can be applied to web-specific items, this is only possible if local contexts are taken into account, and these might be governed by different (and possibly incompatible semantics), and might fluctuate with the size of the text environment at stake. These dynamics provide a justification for our work in developing methods for (re-)computing similarity in different semantic contexts.

3. Which information sources can be used for grounding the computation of similarity?

One of the goals of this thesis has been to develop methods for computing the similarity between lexical items of different types that, on the one hand, take into account contextual information, and, on the other, involve the least human intervention possible. We opted for *never* using pre-existing lexical-semantic resources, such as WordNet-like resources, for computing similarity, not only because they are not easily available most of the time, but also because this contradicts the prerequisite of taking into account the *locality* of the semantics. Instead, all the information needed by the methods we proposed is either directly extractable from the text environment *itself*, or is derived from other *community-edited* resources available on the Web (or resources compiled in an *unsupervised* way). We followed a data-driven approach in all the experiments presented in this thesis. The sources of *feature information* used for describing lexical items were diverse. In some cases, we used distributional statistics regarding the co-occurrence of lexical items:

- with *n-gram contexts* (chapter 5);
- with other words in the scope of *coordination structures* (chapter 7); and also
- with *names* found inside the same document (chapter 8)
- with *similar (keyword) items* (chapter 11).

Alternatively, we obtained information for describing items by *vectorizing documents* associated with those items (chapter 9). In other cases, we mined knowledge sources that were obtained using unsupervised methods (chapter 6), or which are supported by the efforts of a large community of users (chapter 10). For computing the values of the distinct similarity functions, we used such feature information in a variety of different ways. For example, for computing *content similarity*, we used information about n-gram co-occurrences (chapter 5) or *aligned* *data* compiled from the Web (chapter 6). For disambiguating names, we used information about name co-occurrence (chapter 8). *Type similarity* was computed using information about how items co-occur in coordination structures (chapter 7). And, depending on the specific scenario, we were able to compute *Functional Similarity* using (i) vectorized text sources (chapter 9), (ii) community-edited resources (chapter 10) or (iii) item co-occurrence information extracted from application logs (chapter 11). In summary, among other possible sources, information for computing similarity can be obtained, with little or no human effort, from *co-occurrence statistics* compiled in certain specific contexts (different similarity relations require different contexts), as well as from community-edited resources and from databases compiled from the Web using unsupervised methods.

4. How can we evaluate the results of procedures for computing similarity?

A great deal of the effort involved in this thesis was invested in devising *methods*, selecting *metrics* and searching for sources of *gold-standard information* for evaluating procedures for computing similarity (see chapter 3 for an overview of methods and metrics). As we developed our work, we found numerous difficulties in evaluation for which we proposed a variety of solutions. We believe that some of these solutions are among the most useful contributions of this thesis. The main difficulty was (and still is) the lack of solid gold-standard information, especially for the case of less traditional lexical items. Although there are some WordNet-like resources available containing gold-standard information for specific similarity relations (e.g. synonymy), these resources have usually significant *coverage* and *recall* gaps that do not allow a rigorous performance assessment. In fact, the lack of coverage and recall of the few existing lexical-semantic resources for Portuguese was one of the main motivations for this thesis. We showed, however, how such incomplete resources can still be used for performing *parameter exploration* (see chapter 5). For the specific case of similarity relations between *entities*, we confirmed that Wikipedia is a *direct* source of valuable gold-standard information, especially for *type similarity* (chapter 7). Also, by using information about the Web graph around Wikipedia, we were able to *compile* gold-standard information for named-entity disambiguation procedures. Another alternative for evaluating procedures related to Web 2.0 Tags consists in using diachronic information from Web 2.0 sites. The method involves extracting tag information from a specific Web 2.0 site (via a dedicated web-service or by crawling) at multiple points in time (e.g. every month). These snapshots provide us with information about how the *community of users* is *adding* or *revising* tag information by describing media items. These changes, especially when computed between two relatively distant points in time in order to stabilize possible fluctuations (e.g. six month), can be considered a gold-standard resource created by the "wisdom of the crowds " (see chapter 10). In any case, a more comprehensive evaluation may require combining the assessment made with the available gold-standard information with complementary evaluation procedures. These may include:

- Manual Evaluation: despite the disadvantages (inconsistency, too much effort, difficulty in reproducing results), manual evaluation is a good *complementary* option, especially when several judges participate. Manual evaluation should involve *simple decisions* (ideally *binary decisions*), in order to avoid a large number of potentially grey areas.

- Generate and Test Strategies: in some cases, complementary evaluation can be achieved by assuming that correct cases should *instantiate* on corpora (e.g. chapter 6). If the correct restrictions are enforced, this strategy is mostly conservative (i.e. incorrect cases will almost never instantiate at the risk of many correct cases also not instantiating). Therefore, it should allow us to find lower bounds for procedures under evaluation.

If no *absolute* performance standards exist, such as, for instance, when a totally new scenario is being considered, evaluation can be made in a *application-oriented* fashion, possibly comparing the results of the system under evaluation with a competing *legacy* system (see chapter 11). However, application-oriented evaluation may involve *high-level success criteria* for which it might be difficult to establish metrics. Defining the *fair* and *meaningful* metrics may be the most challenging issue for this type of evaluation scenario.

12.3 Main Contributions

The work presented in this thesis encompasses a number of *smaller practical problems*, for which, most of the time, we were able to devise the appropriate (practical) solutions. The work on these low-level problems was important, not only because it led us to several specific practical contributions (outlined in the Conclusions section of the previous chapters), but also because it allowed us to tackle separately different facets of more fundamental problems related to similarity. The result of this *bottom-up* approach can be summarized in three main high-level contributions whose support comes from the thesis as a whole:

1. An original *theoretical framework* that provides formalization for several concepts related to similarity (including *ambiguity*), and which provides support for practical approaches to the computation of similarity in *both* traditional and Web-based text environments.

2. A broad study of the *information sources* and *types of features* that can (or cannot) be used for supporting the computation of different similarity functions, in different text environments.

3. A systematic study of *methods*, *metrics* and potentially useful *gold-standard* resources for evaluating similarity-related functions. Also, we the proposed and described the implementation of several *original evaluation frameworks*,

which make use of different strategies for obtaining gold-standard information from public resources, and use different validation methodologies to assess the correctness of results.

12.4 Future Directions

The work presented in this thesis touches on many different facets of a large body of problems related to *similarity*. The experiments presented here helped us to solve some of our initial questions, but have definitely raised further and deeper questions. While some of these new question are specific to each particular problem tackled, others are more generic, and are fundamentally related to this complex concept that we call "similarity". Additionally, since most of our research has been motivated by *practical* questions raised during the development of different information extraction applications, the solutions we found allowed us to resolve certain bottlenecks in the development of such applications. Obviously, this led us to find *other bottlenecks* in the same application, some of a different nature, but also directly or indirectly related to similarity. Thus, and despite several points of contact, we divide the possibilities for future work into two groups. First, we consider *High-Level Improvements in Previous Work*. We will not discuss *low-level* improvements here, since this has been done already in the publications associated with each chapter. Instead, we focus on *high-level* issues that could lead to global improvements in *all* the experiments presented in this thesis. Second, we present *New Lines of Research*. These include challenges that were not addressed in this thesis but which, in many cases, come as the natural *result* of the work presented. In both cases we have already begun research on some of these lines of work, either in collaboration established with individual researchers, or by participating in larger projects (e.g. [SCS$^+$09]).

12.4.1 High-Level Improvements in Previous Work

Supervised Strategies & Feature Learning: Practically all the methods presented in this thesis used *unsupervised approaches*. We have usually started with a *hypothesis* about a specific set of features that, according to some *intuition* about language, could potentially be useful for computing a certain similarity function, and we have proceeded by applying one *off-the-shelf* unsupervised method to compute the desired similarity function based on such a set of features. This procedure usually involves some work on experimenting with different sets of features, but it took us to interesting results. However, as "language" becomes more idiosyncratic, and the corresponding notions of similarity become more complex, good intuitions become harder to come by. In such scenarios, choosing the best features for grounding the (unsupervised) similarity computation procedures tends to become more a matter of *art* than a matter of *science*. In fact, in this thesis we saw that some of the features of the exeperiments were found to be *insufficient* or even *inadequate*. Two lines of research can be followed in order to tackle this problem. The first consists of using *supervised approaches* whenever it is not too complex to obtain valid training data. Items in the training set can be described using as many features as possible, if the classification algorithm is capable of coping with high-dimensional feature spaces. For example, supervised approaches could be tried for finding *verb synonyms* (chapter 5). A training set, consisting of positive and negative examples of *synonym pairs*, could be generated using information from a thesaurus. Even if it is incomplete, the thesaurus should provide enough information for training a classifier to decide whether two words are synonyms (or antonyms) or not. The set of features may include distributional information taken from very large corpora, such as *n-gram* and *word co-occurrence* frequencies. A similar approach could eventually be attempted for finding *type similar* entities (chapter 7). In this case, the training data could be obtained from the *lists of entities* found in Wikipedia. Similar situations are possible for other cases. The second line of research, which could be supported by the previous one, is the *learning of features*. Instead of using *as many features as possible* to train classifiers (and let the classification algorithms deal with such a high number of features) we could try to learn the best features for classification, for each case. This would allow us to build more *compact* and more *efficient* classifiers that could eventually be used in *real-time* processing.

Exploring Inverse Relations and Scope: Most of the work developed in this thesis is concerned with computing the similarity between items that could be considered to be *at the same level* in terms of *polarity* and *scope* (e.g. synonyms or co-hyponyms). We did not explore other semantic relations that were also found to be relevant in many of the situations that we addressed. For example, from chapter 5 we concluded that being able to identify *antonyms* is extremely important (and difficult) in order to correctly find synonyms. Antonyms can be seen as an instantiation of content-similarity relation with *inverse polarity*. It seems relatively simple to accommodate antonymy in the framework presented in chapter 2, by considering that similarity functions can return *negative* values, instead of only *positive* values. From a practical point of view, many questions remain. Since antonyms and synonyms tend to share the same *distributional features*, distributional approaches such as the one we followed for synonymy are not sufficient. Thus, one interesting line of work consists in researching efficient complementary methods for differentiating the two opposite cases. Also, from chapters 6, 10, 11 and, *more obviously*, 9, we confirmed that the ability to identify *hypernyms/hyponyms* and the "IS-A" (*instantiation*) relations is very relevant in many practical situations. The notion of *type-similarity* that we formalized in chapter 2 does not explicitly distinguish the *scope* of the relation at stake, i.e. it does not separate cases of co-hyponymy from cases of hypernymy/hyponymy, which also share (partial) type-similarity. One obvious research question is related to the introduction of the *scope* parameter in our formal framework. For that, should we *specialize* the type similarity relation in several different functions that explicitly quantify co-hyponymy, hypernymy/hyponymy and the instantiation relation? How can we *grade* the relation of hypernymy/hyponymy to account for scope? Practical research questions include developing methods for quantifying these relations between all types of lexical items considered in this thesis (i.e. not just “traditional” words).

12.4.2 New Lines of Research

Robust Pre-Processing of User-Generated Contents: One of the most promising information sources for many different types of information extraction application is *user-generated content*, more specifically, *opinionated* or *personal* media such as *blogs*, *micro-blogs* (e.g. Twitter) and *user comments* (e.g. on on-line news papers). However, user-generated contents pose several *low-level* language processing challenges. User-generated contents:

- usually contain *misspelled* or *unknown* words or acronyms;
- are populated with *media-specific* symbols (e.g emoticons) or vocabulary (e.g. “lol”) that are not always lexicalized in a dictionary;
- tend to suffer from inconsistent use of capitalization, and non-standard punctuation;
- usually have *word tokenzation* problems.

All these idiosyncrasies, which are not trivial to deal with, greatly complicate information extraction tasks. In fact, we have already encountered some of these problems during this thesis (see part IV). Thus, an important line of future work consists in developing methods for allowing robust processing of such contents. The methods should aim at:

1. achieving correct *tokenization*;
2. correcting *misspelled words* (in context), or recognizing *unknown* (yet valid) tokens;
3. correcting *inconsistent use of capitalization*; and
4. *normalizing* mentions to entities (see for example [JKMdR08]).

We are currently experimenting with the efficiency of several machine learning approaches for *tokenizing* Twitter messages. These approaches allow us to circumvent the problem of manually developing *medium-specific* tokenization rules, which are not only prone to errors, but are also very difficult to maintain.

Opinion Mining in User-Generated Contents: User-generated contents are a rich source of *opinions* about a wide variety of subjects, and are thus potentially useful for business and marketing intelligence applications. Besides the (low-level) challenges described in the previous paragraphs, performing *opinion mining* in user-generated content raises several problems, all of them leading to interesting research questions. One of the problems is to identify the target(s) of the opinion (e.g. a person, a product, etc.) in a certain opinionated text (e.g. a comment posted by a user in an on-line newspaper). This is not trivial since the correct target(s) may be mentioned *indirectly* (e.g. by one of many possible job descriptions if we are dealing with people), by *anaphoric reference* (e.g. "that man"), or by one of several possible *nicknames* that users assign to entities at any time in response to a particular event or psychological trace. Thus, researching methods for *resolving references* made to entities in user-generated contents (possibly resolving them by consulting an external knowledge source such as Wikipedia) is an extremely useful line of research, which we wish to follow. This line of work has connections with the work presented in chapters 6, 8 and 9. Another challenge in opinion-mining is identifying the *polarity* of the opinion about a subject expressed in a given user-generated text snippet. This is essentially a text classification problem, for which several approaches have been proposed, some based on sets of *manually developed rules* and other using traditional *machine learning techniques*. However, in any case, one important resource for determining polarity is the knowledge of the *prior-polarity* of words and expressions. Building such a lexicon involves assigning information regarding *prior polarity* to each word (or frequently used expression), as well as providing some additional information regarding restrictions and exceptions. This implies an immense amount of work that it is infeasible to perform manually on a large scale. We believe that a great deal of the work presented in this thesis can be directly applied in helping to obtain polarity information for all types of lexical items and expressions, especially by following semi-supervised approaches (e.g. chapters 7, 10 and 11). Alternative lines of work may include experimenting with methods for propagating polarity information taking into account information about synonymy, obtained from a combination of resources (e.g. several on-line thesaurus), or automatically inferred from text (such as we presented in chapters 5 and 9). Also, since machine-learning approaches to polarity detection require *annotated* training data, and since it is extremely difficult to perform such annotation manually, one important line of research is related to devising methods for automatically compiling training data. Some of the methods proposed so far focus on compiling information from sites that publish *product reviews* (e.g. Amazon), which usually match textual descriptions with explicit rating information posted by users (e.g. "3/5 stars"). However, the resulting training sets have the disadvantage of being specialized in a given domain (e.g. books) which makes them quite useless for training opinion classifiers for other domains. We are already investigating methods that allow the creation of more generic training sets.

More specifically we are trying to develop bootstrapping methods, which combine an initial step that uses manually developed patterns to identify an initial set of positive and negative sentences in a large corpus of opinionated text, with subsequent polarity propagation operation steps, where the polarity found for specific sentences is propagated to larger sections of text. Some initial work has already been presented [SCSdO09] and we wish to continue this line of research. Opinion mining of user-generated content raises further additional challenges to automatic methods: users often express their opinions *metaphorically* and *ironically*. The automatic treatment of these two phenomena is complex, and not much research has been done on these problems. Resolving metaphorical expressions might involve not only parsing the text correctly but also make use of *background knowledge*. For example, in many cases an opinion about a certain entity is expressed by comparison or association with some well-known historical or fictional entity (e.g. "this prime-minister is a real Hitler"). For assigning polarity in such situations it is crucial to possess information about the entity mentioned in the metaphor or comparison. We are currently experimenting with methods for automatically obtaining polarity information about a large number of entities (people, organizations, events) from Wikipedia using semi-supervised methods. The use of *irony* also raises complex problems, since opinions expressed ironically are usually opposite in polarity to any literal interpretation. Thus, detecting irony is fundamental, since it has a huge impact on the opinion-mining process. We have already performed some exploratory results regarding the identification of irony in user generated content [CSSdO09]. We identified several sets of lexical features that are typically associated with ironic comments. We wish to develop this line of research further by devising classifiers that use these features, as well as additional contextual information, in order to automatically detect ironic comments. We already know that this will be a very difficult task, since even humans have problems in detecting irony.

Automatic Authorship Attribution: Due to the simplicity of *copy-and-paste* text production strategies, and to the increasing amount of text contents freely available, the demand for systems capable of performing *authorship attribution* is greater today more than ever before. There are many practical situations where knowledge of whether a text can be attributed to a specific author is fundamental, not only in cases of possible *plagiarism*, but even cases involving legal action. Research in *forensic linguistics* has shown that texts from a given author exhibit consistency in several stylistic parameters, both within the same piece of text and between two distinct text pieces. The fundamental concept in authorship identification applications is *stylistic similarity*: two text excerpts from the same author should be stylistic similar, while excerpts from different authors should present differences in stylistic parameters. From a computational point of view, authorship attribution can be seen as a text classification problem. The main questions are related to the selection of the appropriate features, in order to capture relevant *stylistic* information. Such features should be immune to the influence of other text parameters that are supposed to be independent of authorship (e.g. the topic of the text). We have already begun some work on this issue [SSSG$^+$10]. We started by experimenting with the robustness of several potentially interesting stylistic features, which are relatively simple to extract from text and are practically omnipresent in all texts (e.g. counts on punctuation, POS n-grams, etc). Further work involves developing methods for extracting higher-level stylistic features from text and assessing their robustness. Additionally, since traditional authorship attribution techniques focus essentially on trying to establish the authorship of long (and relatively well-written) texts, an interesting line of work, which has not yet been explored, consists in adapting these techniques to micro-blogs (e.g. Twitter) and (short) comments posted to on-line newspapers.

Appendix A

Demonstrations for Chapter 8

Consider the set of $\mathcal{I}$ containing $|\mathcal{I}|$ *items* that belong to C classes c_1, c_2, c_3,... c_C. Let p_{ji} be the probability of an item (or element) e_j randomly picked from $\mathcal{I}$ belonging to class c_i: $P(e_j \in c_i) = p_{ji}$ with 1 < i < C.

Now consider the problem of sequentially comparing items in $\mathcal{I}$ (previously shuffled) in order to find items similar to the initial (target) item. If we randomly pick one item e_j from $\mathcal{I}$, we wish to estimate the number of additional items that we need to pick (without repetition) from $\mathcal{I}$ before we find another item that belongs to the same class. For a sufficiently large set of items the probabilities $P(e_j \in c_i)$ do not change significantly when we pick elements out of $\mathcal{I}$ without replacement, and we can consider two subsequent draws to be independent.

We can thus make $P(e_j \in c_i) = p_i$ and approximate this procedure by a *Bernoulli Process*. Therefore, for a given element of class c_i, the number of comparisons k_i needed for finding a similar item follows a *Geometric Distribution* with parameter, p_i. The expected value for k is $E(k_i) = \frac{1}{p_i}$. For C classes, the average number of comparisons is:

$$E(k) = \sum_{c=1}^{|C|} p_c \cdot E(k_c) = \sum_{c=1}^{|C|} p_c \cdot \frac{1}{p_c} = |C| \tag{A.1}$$

For sufficiently large $|\mathcal{I}|$, the number of classes will remain constant during almost the entire sampling process. Thus, the total number of comparisons for the $|\mathcal{I}|$ items is: $N_{comp} = |\mathcal{I}| \cdot |C|$

If we extend the previous item comparison procedure to find k_{pos} similar items to the target item,n we can model the process by a *Negative Binomial Distribution* (or *Pascal Distribution*) with parameters p_i and k_{pos}:

$$B_{neg}(k_i, k_{pos}) = \binom{k_i - 1}{k_{pos} - 1} \cdot p_i^{k_{pos}} \cdot (1 - p_i)^{k_i - k_{pos}} \tag{A.2}$$

In this case, the average number of comparisons made, given by the corresponding Expected Value is: $E_{B_{neg}}(k_i, k_{pos}) = k_{pos}/p_i$. The longest series of comparison will be made for the class with the lowest p_i, i.e. the small class. However, it leads us to an average number of comparisons when considering all the $|C|$ of classes of:

$$E_{comp}(k) = \sum_{c=1}^{|C|} p_c \cdot E_{B_{neg}}(k_c, k_{pos}) = k_{pos} \cdot |C| \tag{A.3}$$

For all $|\mathcal{I}|$ items we should thus have $N_{comp} = |\mathcal{I}| \cdot |C| \cdot k_{pos}$. If we now consider that there in a probability of p_{fn} of having a false negative when comparing two items, and that p_{fn} is constant and independent of classes, the p_i should be replaced by $p_i \cdot (1 - p_{fn})$, i.e. the probability of a random pick finding another item in class c_i has to be multiplied by the probability of not having a false negative.

Then all the above equations will change by a constant factor, giving:

$$N'_{comp} = \frac{|\mathcal{I}| \cdot |C| \cdot k_{pos}}{1 - p_{fn}} \tag{A.4}$$

Likewise, the expected value for longest series of comparisons will be given by performing the same substitution in Equation 10, and making $p_i = p_{min}$:

$$E_{ls} = \frac{k_{pos}}{p_{min} \cdot (1 - p_{fn})} \tag{A.5}$$

Bibliography

[ACDK08] Rema Ananthanarayanan, Vijil Chenthamarakshan, Prasad M Deshpande, and Raghuram Krishnapuram. Rule based synonyms for entity extraction from noisy text. In *AND '08: Proceedings of the second workshop on Analytics for noisy unstructured text data*, pages 31–38, New York, NY, USA, 2008. ACM.

[AGAV08] Enrique Amigó, Julio Gonzalo, Javier Artiles, and Felisa Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*, Volume 12(Number 4):461–486, August 2008.

[AH07] Vibhanshu Abhishek and Kartik Hosanagar. Keyword generation for search engine advertising using semantic similarity between terms. In *ICEC '07: Proceedings of the ninth international conference on Electronic commerce*, pages 89–94, New York, NY, USA, 2007. ACM.

[AHK01] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. On the surprising behavior of distance metrics in high dimensional spaces. In *ICDT '01: Proceedings of the 8th International Conference on Database Theory*, pages 420–434, London, UK, January 4–6 2001. Springer-Verlag.

[AL07] Sören Auer and Jens Lehmann. What have Innsbruck and Leipzig in common? Extracting semantics from Wiki content. In *ESWC '07: Proceedings of the 4th European conference on The Semantic Web*, pages 503–517, Berlin, Heidelberg, 2007. Springer-Verlag.

[AP94] J.J. Almeida and Ulisses Pinto. Jspell – um módulo para análise léxica genérica de linguagem natural. In *Actas do X Encontro da Associação Portuguesa de Linguística*, pages 1–15, Évora, 1994.

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[BB98a] Amit Bagga and Breck Baldwin. Algorithms for scoring coreference chains. In *Proceedings of the LREC 1998 Workshop on Linguistic Coreference*, pages 563–566, Granada, Spain, May 28-30 1998.

[BB98b] Amit Bagga and Breck Baldwin. Entity-based cross-document coreferencing using the vector space model. In *Proceedings of the 17th international conference on Computational linguistics*, pages 79–85, Morristown, NJ, USA, 1998. Association for Computational Linguistics.

[BBH$^+$01] Brigitte Bigi, Armelle Brun, Jean Paul Haton, Kamel Smaïli, and Imed Zitouni. A comparative study of topic identification on newspaper and e-mail. In *Proceedings of String Processing and Information Retrieval (SPIRE)*, pages 238–241, Laguna de San Rafael, Chile, Nov 12 - 16 2001.

[BM01] Regina Barzilay and Kathleen R. Mckeown. Extracting paraphrases from a parallel corpus. In *ACL '01: Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 50–57, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[BMEML08] T. Bertin-Mahieux, D. Eck, F. Maillet, and P. Lamere. Autotagger: A model for predicting social tags from acoustic features on large music databases. *Journal of New Music Research*, Volume 37, Issue 2:115 – 135, June 2008.

[BP06] Razvan Bunescu and Marius Pasca. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06)*, pages 9–16, 2006.

[BR73] G.L.M. Berry-Rogghe. *The Computer and Literary Studies*, chapter The Computation of Collocations and Their Relevance in Lexical Studies, pages 103–112. Edinburgh University Press, Edinburgh, 1973.

[Bro97] Andrei Z. Broder. On the resemblance and containment of documents. In *SEQUENCES '97: Proceedings of the Compression and*

Complexity of Sequences 1997 (1997), Positano, Salerno, Italy, June 11-June 13 1997.

[Car01] S.A. Caraballo. *Automatic Acquisition of a Hypernym-Labeled Noun Hierarchy from Text*. PhD thesis, Brown University, 2001.

[CBCL08] Chris Callison-Burch, Trevor Cohn, and Mirella Lapata. Parametric: an automatic evaluation metric for paraphrasing. In *COLING '08: Proceedings of the 22nd International Conference on Computational Linguistics*, pages 97–104, Morristown, NJ, USA, 2008. Association for Computational Linguistics.

[CC08] Òscar Celma and Pedro Cano. From hits to niches? Or how popular artists can bias music recommendation and discovery. In *NETFLIX '08: Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition*, pages 1–8, New York, NY, USA, 2008. ACM.

[CFLZ03] J. Carrasco, D. Fain, K. Lang, and L. Zhukov. Clustering of bipartite advertiser-keyword graph. In *Proc. International Conference on Data Mining (ICDM'03)*, Melbourne, Florida, November 2003.

[CG91] Kenneth W. Church and William. A. Gale. Concordances for parallel text. In *Proceedings of the Seventh Annual Conference of the UW Center for the New OED and Text Research*, pages 40–62, September 1991.

[CGL07] Fabio Calefato, Domenico Gendarmi, and Filippo Lanubile. Towards social semantic suggestive tagging. In *Proceedings of the 4th Italian Semantic Web Workshop*, Bari - Italy, 18-20 December 2007.

[CH90] Kenneth Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29, 1990.

[CL07] O. Celma and P. Lamere. Music recommendation tutorial. In *8th International Symposium on Music Information Retrieval*, Vienna, Austria, September 23-27 2007.

[CLR90] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The MIT Press and McGraw-Hill Book Company, 1990.

[CP04] Timothy Chklovski and Patrick Pantel. VerbOcean: Mining the Web for Fine-Grained Semantic Verb Relations. In *In Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-04)*, Barcelona, Spain, 2004.

[CSSdO09] Paula Carvalho, Luís Sarmento, Mário J. Silva, and Eugénio de Oliveira. Clues for detecting irony in user-generated contents: Oh...!! it's "so easy" ;-). In *TSA'09 - 1st International CIKM Workshop on Topic-Sentiment Analysis for Mass Opinion Measurement*, Hong Kong, Nov. 6 2009.

[Cuc07] Silviu Cucerzan. Large scale named entity disambiguation based on Wikipedia data. In *The EMNLP-CoNLL Joint Conference*, pages 708–716, Prague, Czech Republic, June 28-30 2007.

[Cur04] James Richard Curran. *From Distributional to Semantic Similarity*. PhD thesis, Institute for Communicating and Collaborative Systems, School of Informatics, University of Edinburgh, 2004.

[CW03] Scott Cederberg and Dominic Widdows. Using LSA and noun coordination information to improve the precision and recall of automatic hyponymy extraction. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pages 111–118, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[CXY08] Yifan Chen, Gui-Rong Xue, and Yong Yu. Advertising keyword suggestion based on concept hierarchy. In *WSDM '08: Proceedings of the international conference on Web search and web data mining*, pages 251–260, New York, NY, USA, 2008. ACM.

[DEG+03] Stephen Dill, Nadav Eiron, David Gibson, Daniel Gruhl, R. Guha, Anant Jhingran, Tapas Kanungo, Sridhar Rajagopalan, Andrew Tomkins, John A. Tomlin, and Jason Y. Zien. Semtag and seeker: bootstrapping the Semantic Web via automated semantic annotation. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 178–186, New York, NY, USA, 2003. ACM.

[DES05] Doug Downey, Oren Etzioni, and Stephen Soderland. A probabilistic model of redundancy in information extraction. In *IJCAI'05: Proceedings of the 19th international joint conference on Artificial intelligence*, pages 1034–1041, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.

[DG04] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI ' 04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA*, pages 137–150, 2004.

[DGG$^+$06] Daniel Delling, Marco Gaertler, Robert Görke, Zoran Nikoloski, and Dorothea Wagner. How to Evaluate Clustering Techniques. Technical Report 2006-24, ITI Wagner, Faculty of Informatics, Universität Karlsruhe (TH), 2006.

[DLP99] Ido Dagan, Lilian Lee, and Fernando Pereira. Similarity-based models of word cooccurrence probabilities. *Machine Learning*, 34:43–69, 1999.

[DMP$^+$04] G. Doddington, A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, and R. Weischedel. The Automatic Content Extraction (ACE) Program–Tasks, Data, and Evaluation. pages 837–840, 2004.

[dS00] B. Dias da Silva. Construção de um thesaurus electrónico para o português do brasil. In *Processamento Computacional do Português Escrito e Falado (PROPOR)*, volume 4, pages 1–10, Atibaia, SP, November, 19-22 2000. ICMC/USP.

[Dun93] Ted E. Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74, 1993.

[ES92] Ute Essen and Volker Steinbiss. Cooccurrence smoothing for stochastic language modeling. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, 161-164, San Francisco, CA, USA, March 1992.

[ES07] Noemie Elhadad and Komal Sutaria. Mining a lexicon of technical terms and lay equivalents. In *Biological, translational, and clinical language processing*, pages 49–56, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[Eve05] Stefen Evert. *The statistics of word cooccurrences : word pairs and collocations*. PhD thesis, Institut fur maschinelle Sprachverarbeitung, Universitat Stuttgart, 2005.

[FL07] Norbert Fuhr and Mounia Lalmas. Advances in XML retrieval: the INEX initiative. In *IWRIDL '06: Proceedings of the 2006 international workshop on Research issues in digital libraries*, pages 1–6, New York, NY, USA, 2007. ACM.

[GA04] Chung Heong Gooi and James Allan. Cross-document coreference on a large scale corpus. In *Proceedings of HLT-NAACL 2004*, pages 9–16, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics.

[GBM03] Roxana Girju, Adriana Badulescu, and Dan Moldovan. Learning semantic constraints for the automatic discovery of part-whole relations. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 1–8, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[GCY92] William A. Gale, Kenneth W. Church, and David Yarowsky. One sense per discourse. In *HLT '91: Proceedings of the workshop on Speech and Natural Language*, pages 233–237, Morristown, NJ, USA, 1992. Association for Computational Linguistics.

[GH05] Zoubin Ghahramani and Katherine A. Heller. Bayesian sets. In *Advances in Neural Information Processing Systems 18 (NIPS)*, Vancouver, British Columbia, Canada, December 5-8 2005.

[GM08] Asela Gunawardana and Christopher Meek. Tied Boltzmann machines for cold start recommendations. In *Proceedings of the ACM Conference on Recommender Systems*, pages 19–26. ACM, 2008.

[GMM$^+$03] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering Data Streams: Theory and Practice. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):515–528, 2003.

[Har54] Zellig S. Harris. Distributional structure. *Word*, 10(23):146–162, 1954.

[HBV01] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17:107–145, 2001.

[HT73] John Hopcroft and Robert Tarjan. Algorithm 447: efficient algorithms for graph manipulation. *Commun. ACM*, 16(6):372–378, 1973.

[IM98] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, New York, NY, USA, 1998. ACM.

[iW06] Sabine Schulte im Walde. Experiments on the automatic induction of German semantic verb classes. *Computational Linguistics*, 32(2):159–194, 2006.

[JKMdR08] Valentin Jijkoun, Mahboob Alam Khalid, Maarten Marx, and Maarten de Rijke. Named entity normalization in user generated content. In *AND '08: Proceedings of the second workshop on Analytics for noisy unstructured text data*, pages 23–30, New York, NY, USA, 2008. ACM.

[JM06] Amruta Joshi and Rajeev Motwani. Keyword generation for search engine advertising. In *ICDMW '06: Proceedings of the Sixth IEEE International Conference on Data Mining - Workshops*, pages 490–496, Washington, DC, USA, 2006. IEEE Computer Society.

[Ken38] Maurice G. Kendall. A new measure of rank correlation. *Biometrika*, 30:81–93, 1938.

[Kro97] Robert Krovetz. Homonymy and polysemy in information retrieval. In *In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL-97*, pages 72–79, 1997.

[Lam08] Paul Lamere. Social tagging and Music Information Retrieval. In *Journal of New Music Research*, volume Volume 37, Issue 2, pages 101 – 114, June 2008.

[Lee99] Lillian Lee. Measures of distributional similarity. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 25–32, Morristown, NJ, USA, 1999. Association for Computational Linguistics.

[Lee01] Lillian Lee. On the Effectiveness of the Skew Divergence for Statistical Language Analysis. In *Artificial Intelligence and Statistics*, pages 65–72, 2001.

[Lin91] Jianhua Lin. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991.

[Lin98] Dekang Lin. An Information-Theoretic Definition of Similarity. In Jude W. Shavlik, editor, *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), Madison, Wisconson, USA, July 24-27, 1998*, pages 296–304. Morgan Kaufmann, 1998.

[LP01] Dekang Lin and Patrick Pantel. DIRT - discovery of inference rules from text. In *In Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 323–328, 2001.

[LP02] Dekang Lin and Patrick Pantel. Concept discovery from text. In *Proceedings of the 19th international conference on Computational linguistics*, pages 1–7, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[LvAD07] E. Law, L. von Ahn, and R. Dannenberg. Tagatune: a game for music and sound annotation. In *8th International Symposium on Music Information Retrieval*, pages 361–364, Vienna, September 23-27 2007.

[Mal05] Bradley Malin. Unsupervised name disambiguation via social network similarity. In *Workshop on Link Analysis, Counterterrorism, and Security in conjunction with the SIAM International Conference on Data Mining*, pages 93–102, 2005.

[ME08] Michael I. Mandel and Daniel P. W. Ellis. A Web-based game for collecting music metadata. *Journal of New Music Research*, 37(2):151–165, 2008.

[Mei07] Marina Meilă. Comparing clusterings—an information based distance. *J. Multivar. Anal.*, 98(5):873–895, 2007.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval.* Cambridge University Press, July 2008.

[MS04] Bruno Martins and Mário J. Silva. A Statistical Study of the WPT-03 Corpus. Technical Report TR 4-4, Departamento de Informática da Fac. Ciências de Universidade de Lisboa, Lisboa, May 2004.

[MY03] Gideon S. Mann and David Yarowsky. Unsupervised personal name disambiguation. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pages 33–40, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[Nab04] Daniel Naber. Openthesaurus: Building a thesaurus with a Web community. Technical report, http://www.openthesaurus.de/download/openthesaurus.pdf (acessed Nov 6th 2009), 2004.

[OSGS08] Hugo Gonçalo Oliveira, Diana Santos, Paulo Gomes, and Nuno Seco. Papel: A dictionary-based lexical ontology for Portuguese. In *PROPOR '08: Proceedings of the 8th international conference on Computational Processing of the Portuguese Language*, pages 31–40. Springer-Verlag, Berlin, Heidelberg, 2008.

[Pea02] Darren Pearce. A comparative evaluation of collocation extraction techniques. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, pages 651–658, Las Palmas, Canary Islands, Spain, May 2002.

[PL02] Patrick Pantel and Dekang Lin. Document clustering with committees. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 199–206, New York, NY, USA, 2002. ACM Press.

[PPBLRS03] A. Pons-Porrata, R. Berlanga-Llavori, and J. Ruiz-Shulcloper. Building a hierarchy of events and topics for newspaper digital libraries. In F. Sebastiani, editor, *Proceedings of* 25^{th} *European Conference on IR Research (ECIR'03)*, volume 2633, pages 588–596, Pisa, Italy, 2003. Springer-Verlag.

[PT08] Yoon-Joo Park and Alexander Tuzhilin. The long tail of recommender systems and how to leverage it. In *RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems*, pages 11–18, New York, NY, USA, 2008. ACM.

[PTL93] Fernando Pereira, Naftali Tishby, and Lillian Lee. Distributional clustering of english words. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, pages 183–190, Morristown, NJ, USA, 1993. Association for Computational Linguistics.

[RC98] Brian Roark and Eugene Charniak. Noun-phrase co-occurrence statistics for semiautomatic semantic lexicon construction. In *Proceedings of the 17th international conference on Computational linguistics*, pages 1110–1116, Morristown, NJ, USA, 1998. Association for Computational Linguistics.

[RD00] Philip Resnik and Mona Diab. Measuring Verb Similarity. Technical Report LAMP-TR-047, UMIACS-TR-2000-40,CS-TR-4149, University of Maryland, College Park, June 2000.

[RS97] Ellen Riloff and Jessica Shepherd. A Corpus-Based Approach for Building Semantic Lexicons. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 117–124, 1997.

[Sah06] Magnus Sahlgren. *The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high dimensional vector spaces.* Sics dissertation series 44, Stockholm University, Sweden, 2006.

[Sar06a] Luís Sarmento. BACO - A large database of text and co-occurrences. In Nicoletta Calzolari, Khalid Choukri, Aldo Gangemi, Bente Maegaard, Joseph Mariani, Jan Odjik, and Daniel Tapias, editors, *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC'2006)*, pages 1787–1790, Genoa, Italy, 22-28 May 2006.

[Sar06b] Luís Sarmento. A expansão de conjuntos de co-hipónimos a partir de colecções de grandes dimensões de texto em português. In *Actas de 1a Conferência em Metodologias de Investigação Científica*, Porto, Portugal, Janeiro 2006.

[Sar06c] Luís Sarmento. SIEMÊS - a named-entity recognizer for Portuguese relying on similarity rules. In *PROPOR 2006 - Encontro para o Processamento Computacional da Língua Portuguesa Escrita e Falada*, pages 31–40, ME - RJ / Itatiaia, Rio de Janeiro - Brasil, May, 13-17 2006.

[Sar07] Luís Sarmento. A first step to address biography generation as an iterative QA task. In Carol Peters, Paul Clough, Fredric C. Gey, Douglas W. Oard, Maximilian Stempfhuber, Bernardo Magnini, Maarten de Rijke, and Julio Gonzalo, editors, *7th Workshop of the Cross-Language Evaluation Forum, CLEF 2006. Alicante, Spain, September 2006. Revised Selected papers*, Lecture Notes in Computer Science. Springer, Berlin / Heidelberg, 2007.

[SCO09] Luís Sarmento, Paula Carvalho, and Eugénio Oliveira. Exploring the vector space model for finding verb synonyms in Portuguese. In *Proceedings of RANLP - Recent Advances in Natural Language Processing*, Borovets, Bulgaria, September 14-16 2009.

[SCS$^+$09] Mário J. Silva, Paula Carvalho, Luís Sarmento, Eugénio Oliveira, and Pedro Magalhães. The design of OPTIMISM, an opinion mining system for Portuguese politics. In *Proceedings of EPIA 2009: Encontro Português de Inteligência Artificial (local proceedings)*, Aveiro, Portugal, 12-15 October 2009.

[SCSdO09] Luís Sarmento, Paula Carvalho, Mário J. Silva, and Eugénio de Oliveira. Automatic creation of a reference corpus for political opinion mining in user-generated content. In *TSA'09 - 1st International CIKM Workshop on Topic-Sentiment Analysis for Mass Opinion Measurement*, Hong Kong, Nov. 6 2009.

[SG00] Alexander Strehl and Joydeep Ghosh. Value-based Customer Grouping from Large Retail Data-sets. In *Proc. SPIE Conference on Data Mining and Knowledge Discovery, Orlando*, volume 4057, pages 33–42. SPIE, April 2000.

[SGCO09] Luís Sarmento, Fabien Gouyon, Bruno Costa, and Eugénio Oliveira. Visualizing networks of music artists with rama. In *Proceedings of the International Conference on Web Information Systems and Technologies*, Lisbon, Portugal, 23-26 March 2009.

[SGM00] Alexander Strehl, Joydeep Ghosh, and Raymond Mooney. Impact of Similarity Measures on Web-page Clustering. In *Proceedings of the 17th National Conference on Artificial Intelligence: Workshop of Artificial Intelligence for Web Search (AAAI 2000), 30-31 July 2000, Austin, Texas, USA*, pages 58–64. AAAI, July 2000.

[SGO09] Luís Sarmento, Fabien Gouyon, and Eugénio Oliveira. Music artist tag propagation with wikipedia abstracts. In *Proceeding of the Workshop on Information Retrieval over Social Networks, European Conference on Information Retrieval (ECIR)*, Tolouse, April 2009.

[SJ01] Patrick Schone and Daniel Jurafsky. Is knowledge-free induction of multiword unit dictionary headwords a solved problem? In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, page 100–108, Pittsburgh, PA, 2001.

[SJB06] Claude St-Jacques and Caroline Barrière. Similarity Judgments: Philosophical, Psychological and Mathematical Investigations. In *Proceedings of the Workshop on Linguistic Distances*, pages 8–15, Sydney, Australia, July 2006. Association for Computational Linguistics.

[SJdRO07] Luís Sarmento, Valentin Jijkuon, Maarten de Rijke, and Eugénio Oliveira. "More like these": growing entity classes from seeds. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 959–962, New York, NY, USA, 2007. ACM.

[SJN04] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems (NIPS 2004)*, Vancouver, British Columbia, Canada, December 13-18 2004.

[SKOU09a] Luís Sarmento, Alexander Kehlenbeck, Eugénio Oliveira, and Lyle Ungar. An approach to Web-scale named-entity disambiguation. In *Proceedings of the International Conference on Machine Learning and Data Mining (MLDM) 2009*, LNAI, Leipzig, Germany, July 23-25 2009. Springer Verlag.

[SKOU09b] Luís Sarmento, Alexander Kehlenbeck, Eugénio Oliveira, and Lyle Ungar. Efficient clustering of Web-derived data sets. In *Proceedings of the International Conference on Machine Learning and Data Mining (MLDM) 2009*, LNAI, Leipzig, Germany, July 23-25 2009. Springer Verlag.

[SKPW08] Markus Schedl, Peter Knees, Tim Pohle, and Gerhard Widmer. Towards an automatically generated music information system via Web content mining. In *Proceedings of 30th European Conference on IR Research*, pages 585–590, Glasgow, UK, March 30-April 3 2008.

[SLC07] M. Sordo, C. Laurier, and O. Celma. Annotating music collections: How content-based similarity helps to propagate labels. In *8th International Symposium on Music Information Retrieval*, Vienna, Austria, September 23-27 2007.

[SM86] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, Inc., New York, NY, USA, 1986.

[Sma93] Frank Smadja. Retrieving collocations from text: Xtract. *Computational Linguistics*, 19(1):143–177, 1993.

[SN09] Luís Sarmento and Sérgio Nunes. Automatic extraction of quotes and topics from news feeds. In *Proceedings of DSIE'09 - 4th Doctoral Symposium on Informatics Engineering*, Porto, Portugal, February 5-6 2009.

[SNM08] Panagiotis Symeonidis, Alexandros Nanopoulos, and Yannis Manolopoulos. Tag recommendations based on tensor dimensionality reduction. In *Proceedings of the ACM Conference on Recommender Systems*, pages 43–50, 2008.

[SNTO09] Luís Sarmento, Sérgio Nunes, Jorge Teixeira, and Eugénio Oliveira. Propagating fine-grained topic labels in news snippets. In *Proceedings of the Workshop on Intelligent Analysis and Processing of Web News Content (held with WI/IAT 2009)*, Milan, Italy, September 2009.

[SPC06] Luís Sarmento, Ana Sofia Pinto, and Luís Cabral. REPENTINO - A Wide-Scope Gazetteer for Entity Recognition in Portuguese. In *PROPOR 2006 - Encontro para o Processamento Computacional da Língua Portuguesa Escrita e Falada*, pages 31–40, ME - RJ / Itatiaia, Rio de Janeiro - Brasil, May, 13-17 2006.

[SPUP02] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260, New York, NY, USA, 2002. ACM.

[SSCV06] Diana Santos, Nuno Seco, Nuno Cardoso, and Rui Vilela. HAREM: An advanced NER evaluation contest for Portuguese. In Nicoletta Calzolari, Khalid Choukri, Aldo Gangemi, Bente Maegaard, Joseph Mariani, Jan Odjik, and Daniel Tapias, editors, *Proceedings of the 5th International Conference on Language Resources and Evaluation, LREC 2006*, pages 1986–1991, Genoa, Italy, 22–28 May 2006. ELRA.

[SSLM02] Sreenivasa Sista, Richard Schwartz, Timothy R. Leek, and John Makhoul. An algorithm for unsupervised topic discovery from broadcast news stories. In *Proceedings of the second international conference on Human Language Technology Research*, pages 110–114, San Diego, California, 2002. Morgan Kaufmann Publishers Inc.

[SSS02] Yusuke Shinyama, Satoshi Sekine, and Kiyoshi Sudo. Automatic paraphrase acquisition from news articles. In *Proceedings of the second international conference on Human Language Technology Research*, pages 313–318, San Diego, California, 2002. Morgan Kaufmann Publishers Inc.

[SSSG$^+$10] Rui Sousa-Silva, Luís Sarmento, Tim Grant, Eugénio Oliveira, and Belinda Maia. Comparing sentence-level features for authorship analysis in Portuguese. In *Proceedings of International Conference on Computational Processing of Portuguese Language*, Porto Alegre - RS, Brazil, 27 to 30 of April 2010.

[STGO09] Luís Sarmento, Paulo Trezentos, João Pedro Gonçalves, and Eugénio Oliveira. Inferring local synonyms for improving keyword suggestion in an on-line advertisement system. In *ADKDD '09: Proceedings of the Third International Workshop on Data Mining and Audience Intelligence for Advertising*, pages 37–45, Paris, France, 2009. ACM.

[STO08] Luís Sarmento, Jorge Filipe Teixeira, and Eugénio Oliveira. Assessing the impact of thesaurus based expansion techniques in QA-centric IR. In Carol Peters, Tomas Deselaers, Nicola Ferro, Julio Gonzalo, Gareth J.F.Jones, Mikko Kurimo, Thomas Mandl, Anselmo Peñas, and Viviane Petras, editors, *Evaluating Systems for Multilingual and Multimodal Information Access 9th Workshop* *of the Cross-Language Evaluation Forum (Revised Selected Papers)*, Aarhus, Denmark, September 17-19 2008. Springer.

[Tak08] Koichi Takeuchi. Extraction of verb synonyms using co-clustering approach. In *ISUC '08: Proceedings of the 2008 Second International Symposium on Universal Communication*, pages 173–178, Osaka, Japan, December 15 - 16 2008. IEEE Computer Society.

[TBL08] Douglas Turnbull, Luke Barrington, and Gert Lanckriet. Five approaches to collecting tags for music. In *Proceedings of the 9th International Conference on Music Information Retrieval*, pages 225–230, Philadelphia, USA, 2008.

[TFK02] Aristomenis Thanapoulos, Nikos Fakotakis, and George Kokkinakis. Comparative Evaluation of Collocation Extraction Metrics. In *LREC 2002 - 3rd International Conference on Language Resources and Evaluation*, volume Vol. 2, pages 620–625, Las Palmas, Spain, May 29-31 2002.

[TKS04] Pang-Ning Tan, Vipin Kumar, and Jaideep Srivastava. Selecting the right objective measure for association analysis. *Inf. Syst.*, 29(4):293–313, 2004.

[TLBL07] D. Turnbull, R. Liu, L. Barrington, and G Lanckriet. Using games to collect semantic information about music. In *8th International Symposium on Music Information Retrieval*, Vienna, Austria, September 23-27 2007.

[TR02] Michael Thelen and Ellen Riloff. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 214–221, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[TSO10] Jorge Filipe Teixeira, Luís Sarmento, and Eugénio Oliveira. Comparing verb synonym resources for Portuguese. In *Proceedings of International Conference on Computational Processing of Portuguese Language*, Porto Alegre - RS, Brazil, 27 to 30 of April 2010.

[Tur01] Peter D. Turney. Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the 12th European Conference on Machine Learning*, volume Lecture Notes in Computer Science, 2167, pages 491–502, 2001.

[vAGKB07] L. von Ahn, S. Ginosar, M. Kedia, and M. Blum. Improving image search with phetch. In *IEEE International Conference on Acoustics,* *Speech and Signal Processing, 2007. ICASSP 2007.*, volume 4, pages IV–1209–IV–1212, April 2007.

[VPF91] Paola Velardi, Maria Teresa Pazienza, and Michela Fasolo. How to Encode Semantic Knowledge: A Method for Meaning Representation and Computer-Aided Acquisition. *Computational Linguistics*, 17(2):153–170, 1991.

[WD02] Dominic Widdows and Beate Dorow. A Graph Model for Unsupervised Lexical Acquisition. In *Procedings of 19th International Conference on Computational Linguistics (COOLING 19)*, pages 1093–1099, Taipei, 2002.

[WKPU08] Casey Whitelaw, Alex Kehlenbeck, Nemanja Petrovic, and Lyle Ungar. Web-scale named entity recognition. In *ACM 17th Conference on Information and Knowledge Management: CIKM 2008*. ACM Press, 2008.

[WWM04] Julie Weeds, David Weir, and Diana McCarthy. Characterising measures of lexical distributional similarity. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 1015, Morristown, NJ, USA, 2004. Association for Computational Linguistics.

[XFMS06] Zhichen Xu, Yun Fu, Jianchang Mao, and Difu Su. Towards the Semantic Web: Collaborative tag suggestions. In *WWW2006: Proceedings of the Collaborative Web Tagging Workshop*, Edinburgh, Scotland, may 2006.

[YE07] Alexander Yates and Oren Etzioni. Unsupervised resolution of objects and relations on the Web. In *Proceedings of NAACL HLT*, pages 121–130, Rochester, NY, April 2007.

[ZK01] Y. Zhao and G. Karypis. Criterion Functions for Document Clustering: Experiments and Analysis. Technical report, University of Minnesota, Minneapolis, 2001.