# Erasmus scholarship

Implementation of an ontology mapping algorithm
(Internship at University Oporto from Oct-2006 to Feb-2007)

*Final report*

Universidade
do Porto
Faculdade de
Engenharia

Bernd Schneiders, 2006/2007
Umwelt-Campus Birkenfeld
Applied computer science

UMWELT-CAMPUS BIRKENFELD
Fachhochschule Trier
University of Applied Sciences

Supervisor: Professor Eugénio Costa Oliveira

## Table of Content

# 1 Introduction

This final report is developed from an internship (16. weeks) project at the NIAD&R (Distributed Artificial Intelligence & Robotics Group) at University of Oporto. Aim of this project was reimplementing an ontology mapping algorithm to the still in developing Virtual Institution platform by NIAD&R.

First this document contains a short overview of related work, about MAS´s and the platform. Next, the most important part, is about ontology mapping, structure, how the algorithm works and about the protocol. The final chapter is about some experiments.

# 2 Related work

This work is based on parts of an algorithm which was developed by [1] and [2]. They developed an ontology algorithm which based on a Multi-Agent-System written in JADE in an B2B environment.

This reimplementation of the algorithm is integrated in Virtual Institution MAS platform of [3].

# 3 MAS

A Multi-Agent-System (MAS) is a software platform which contains several similar or different types of software agents. The platform provide a system for communication and management. In our platform the JADE framework is used for developing a MAS.

## *3.1 JADE*

JADE (Java Agent DEvelopment Framework) is a software framework fully implemented in Java language. It simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA specification. The framework is released under the open source LGPL licence.

The agent platform can be distributed across machines that do not need to share the same operating system. Only one Java application and, therefore, only one Java Virtual Machine (JVM), is executed on each host. Each JVM is basically a container of agents that provides a complete runtime environment for agent execution and allows several agents to run concurrently on the same host.[6]

## *3.2 Platform*

The platform where the ontology mapping algorithm is implemented is written in Java and using the JADE framework. Figure 1 show the structure and represents a Virtual Institution that facilitates a Virtual Enterprise life cycle. It comprise several different services like Negotiation Mediator, Notary, Contract Monitoring and a Ontology Mapping service.

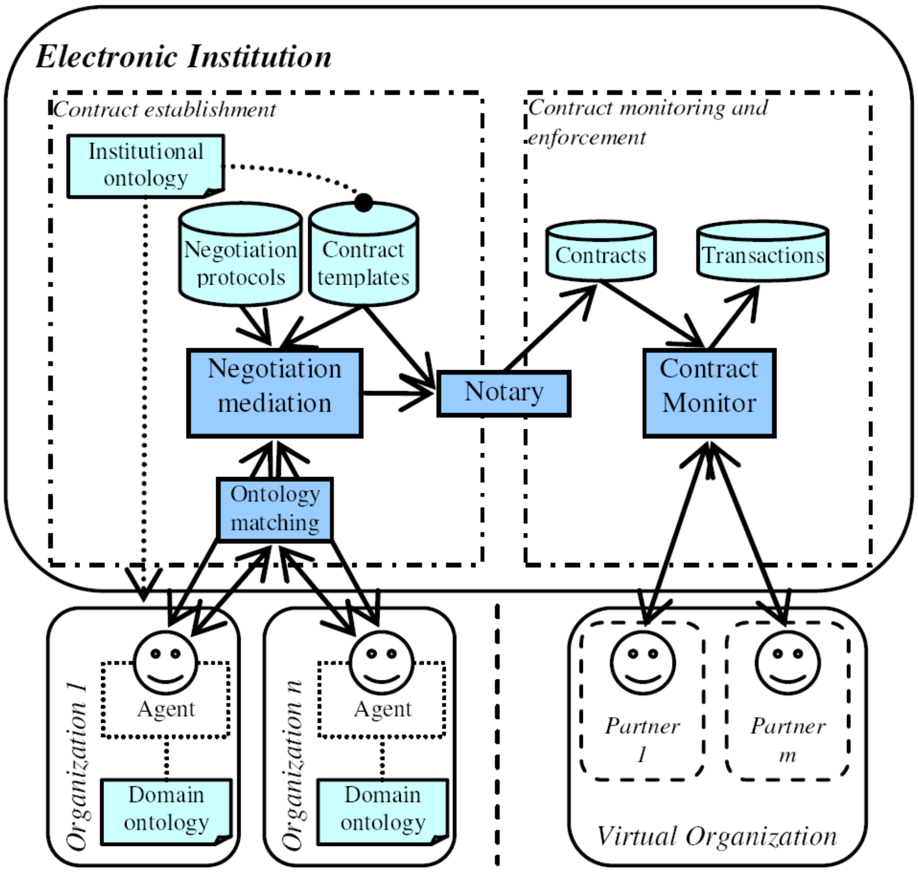


*Figure 1: Scheme of MAS platform*

## 4 Ontology Mapping

Ontology mapping is the process of finding correspondences between the concepts of two ontologies. If two concepts correspond, then they mean the same thing or closely related things [Dou et al., 2003].

### *4.1 Introduction*

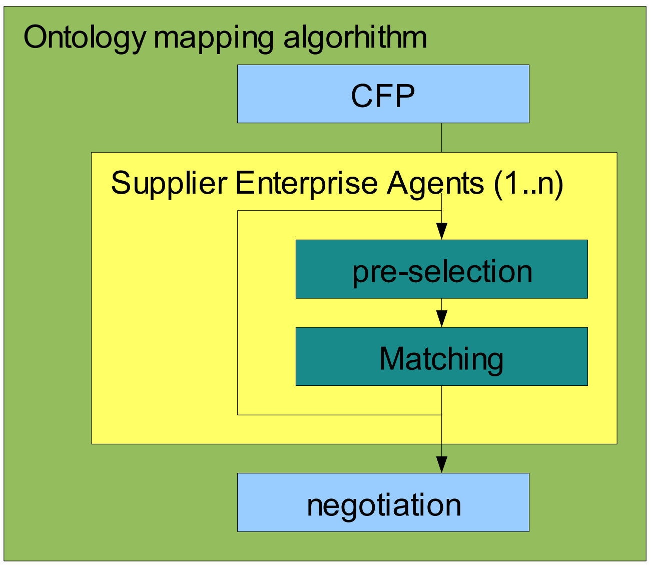
There are several ontology definitions that have evolved in the last decades. However, there is a consensus among the ontology community of researchers about their role: they provide a common understanding regarding some domain knowledge. Diverse research communities, such as knowledge engineering, database and software engineering, use ontologies for many different purposes: natural language processing, electronic commerce, knowledge management, semantic Web, etc. [1]

Ontologies generally consist of individuals, classes, attributes and relations.

- Individuals represent an instance of an object with all attributes an their values. (e.g. in our e-commerce scenario it represents an instance of a motor named 'motor01')
- Classes or concepts conceptualise objects which describe common properties. These classes could consist of main classes and subclasses. 'car' as main class and 'wheel' as a subclass of a car.
- Attributes describe characteristics of a class (e.g. 'number_of_cylinder' of the class 'motor')
- Relations represent relationships between concepts (e.g. a Motor 'has_cylinderblock')

If two software agents want to negotiate about a product, they have to understand each other. There are two ways to solve this problem. The first simple approach for negotiation is to use same ontologies for both agents, customer and supplier agent to represent their products. Each item, supplier and customer, need to abide by the exact specification of the used ontology. It is recommend to use for every product group (e.g. CD, wheel, etc.) a separate ontology. This approach is only practicable by simple products, it is more difficult to represent products like a car with a public ontology.
A more realistic approach is that every agent use his own domain ontology. Only the use of an institutional ontology is predetermined. This institutional ontology is required to facilitate a negotiation. Institutional ontologies contains e.g. price, deliver time etc. Developer/user of enterprise agents could now adapt the domain ontology to the product specification or requirements.
On this approach it occur a new problem the so-called heterogeneity problem. It is more difficult for an agent to determine if the own and the ontology of the in negotiation involved agent represent the same item. This heterogeneity problem could be solved by an ontology mapping algorithm. [1]

The ontology mapping algorithm is separated into two parts. First part is the pre-selection algorithm. This algorithm allows a reduction of items to handle. The second part, the matching algorithm, calculates a similarity between two ontologies.


Ontology mapping algorhithm
CFP
Supplier Enterprise Agents (1..n)
pre-selection
Matching
negotiation
*Figure 2: Ontology mapping algorithm*

## 4.2 Pre-selection

The pre-selection is a simple way to reduce the amount of considered items. After CFP the negotiation mediator only considers items of supplier enterprise agents which are in price range of proposed item. Price range is between 75% and 125%, e.g. it is not expected that a motor is in price range of a battery.

## 4.3 Matching item

After pre-selection the remain items have to be compared to calculate a similarity. First item name, description and the attributes by N-Grams algorithm. Then the item name matching is performed by the semantic similarity algorithm of LCH. Last a final result is calculated with a weighted consideration of partial results.

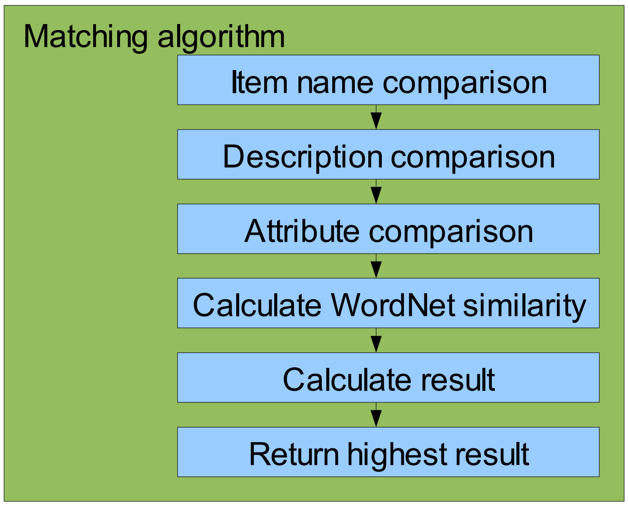


*Figure 3: Matching algorithm*

### 4.3.1 N-Grams

The N-Grams algorithm facilitates a calculation of similarity between two strings.

The first step is to normalize both strings, i.e. delete all stop-words like “a”, “around” or “that”, then replace all irregular like '_' or '?' with ' '. Next separate both strings into so-called grams. These grams are sub-strings of length n.

In the next step the algorithm compares all sub-strings of first string with all sub-strings of second string. If two sub-strings are equal a counter will be increment.

Last a value of similarity will be calculate:

$$value = \frac{\text{count of matches}}{\text{number of sub-strings}}$$

The value is in range of 0.0 to 1.0. A value of 0.0 (i.e. no sub-string match with another sub-string) is the worst case. The best case is a value of 1.0 (i.e. all sub-string of first string match with sub-stings of second string).

It is important to divide 'count of matches' by 'number of sub-string' of the shortest string, otherwise the value could be higher then one.

Another part of this algorithm is to insert at the begin and end of the string spaces. This is important to match sub-strings with the first and last character of a string.

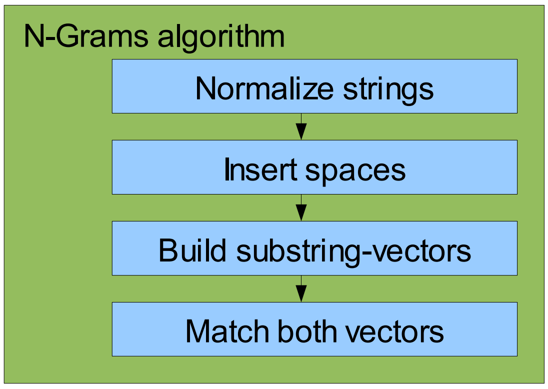


*Figure 4: N-Grams algorithm*

```
N-Grams(3, 2) => 3=number of letters, 2=number of spaces

String1='frontseat' with spaces='  frontseat  '
String2='backseat'  with spaces='  backseat  '

Grams string1='  f', ' fr', 'fro', 'ron', 'ont', 'nts',
              'tse', 'sea', 'eat', 'at ', 't  '
Grams string2='  b', ' ba', 'bac', 'ack', 'cks', 'cks',
              'sea', 'eat', 'at ', 't  '

count of matches=4
number of sub-strings=10

value=0.4
```

*Table 1: Example N-Grams*

In our e-commerce scenario the best parameters for N-Grams algorithm are 3 for number of letters (i.e. length of sub-strings) and 2 for number of spaces [1].

### 4.3.2 WordNet

WordNet is a free lexical database which contains semantic and lexical relations between words.

The Perl program *WordNet::Similarity* use the WordNet database to calculate a semantic similarity between  two words. It facilitates different measure methods to calculate semantic similarity. The best method in our e-commerce scenario is LCH [1]. LCH calculates a taxonomic path length between two words.

### 4.3.3 Matching Description

Matching description is based on N-Grams algorithm. The idea behind this matching is that different experts which are describing a item will use similar words in the description.

As an example we would have the description of “Engine” as “engine is a motor that converts thermal energy to mechanical work” and “Motor” as “motor is a machine that converts other forms of energy into mechanical energy and so imparts motion”. Taken out the stop-words we would have the description of “Engine” as “engine motor converts thermal energy mechanical work” compared with the description of “Motor” as “motor machine converts forms energy mechanical imparts motion”. [5]

### 4.3.4 Matching Attributes

The idea behind matching attributes is that two experts which create a concept probably use equal object-types for same attributes(e.g. object-type for number of wheels is typically integer and not float). Also they use similar names for attributes.

First part of algorithm is to separate all attributes of a concept by equal attribute-type. Next step is to build for each attribute-type a separate string. These strings contains the names of each attribute of certain attribute-types.

| Concept 1 | Concept 2 |
|---|---|
| int number_of_wheels | Int numberOfWheels |
| int horse_power | int hp |
| string color | string color |
| bool has_aircondition | bool has_ac |
| Ints =“number_of_wheels | Ints =“numberOfWheels hp” |
| horse_power” | strings=“color” |
| strings=“color” | bools =“has_ac” |
| bools =“has_air_condition” | |

*Table 2: Attribute matching*

The final part of this algorithm is to compare the created attribute-strings of two concepts. For this comparison the N-Grams algorithm is used.

After using N-Grams on each attribute-string an average of all types is calculated. The final result is in range of 0.0 (not similar) to 1.0 (similar)

### 4.3.5 Matching item name

To calculate similarity of item name based on WordNet, the process is divided into two parts. The first part is integrated in the platform. This part is written in Java. To communicate with the second part, a Perl script, the TCP/IP protocol is used. Every time the matching item-name algorithm establish a new connection.

The Perl script provides a TCP/IP server. This server receive the request from and send the result to the client. After receiving a request, the server call the WordNet::Similarity module which use the WordNet::QueryData perl module to access the WordNet database.

The reason why a client/server based system is used, is that initialising the WordNet database need some seconds. In this approach the initiating of the database is needed only one time at start. A request base on client/server approach require only some milliseconds.

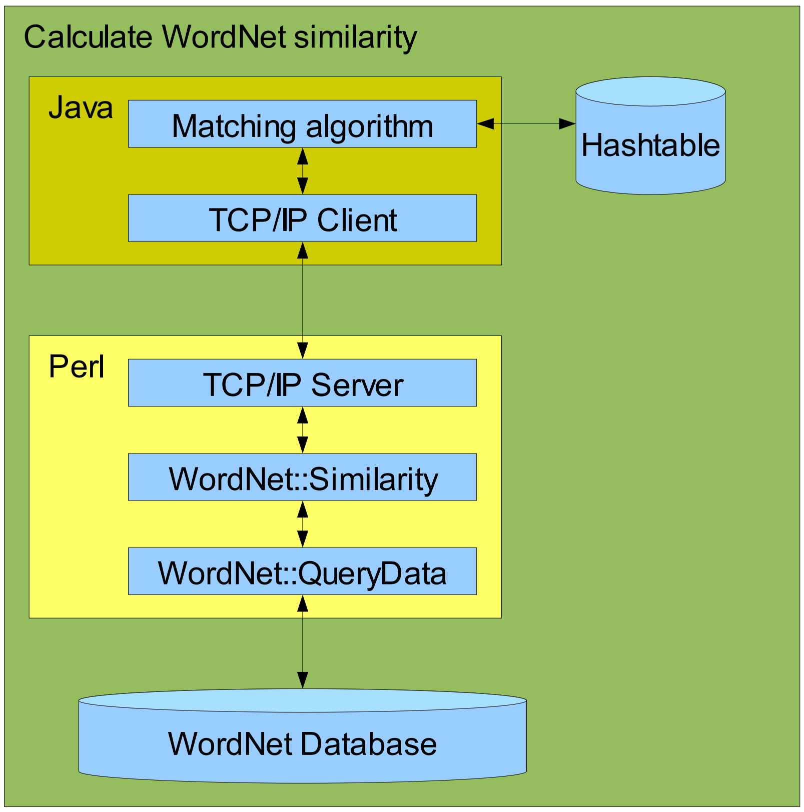


*Figure 5: Calculate WordNet similarity*

### 4.3.6 Final matching result

Last step of this algorithm is to calculate a final result for every concerned item and return the item with the highest result. The finial result is calculated of partial results of item name, description, attribute comparison by N-Grams and item name comparison.

## *4.4 Protocol*

After initiating negotiation by a REQUEST of a Customer-Enterprise-Agent to Negotiation-Mediator service a Negotiation-Handler will be created for negotiating. The ontology mapping service try to find corresponding items between customer and supplier. After matching, the service return if a corresponding item was found or not. The Negotiation-Handler can now include or exclude the supplier form negotiation process.

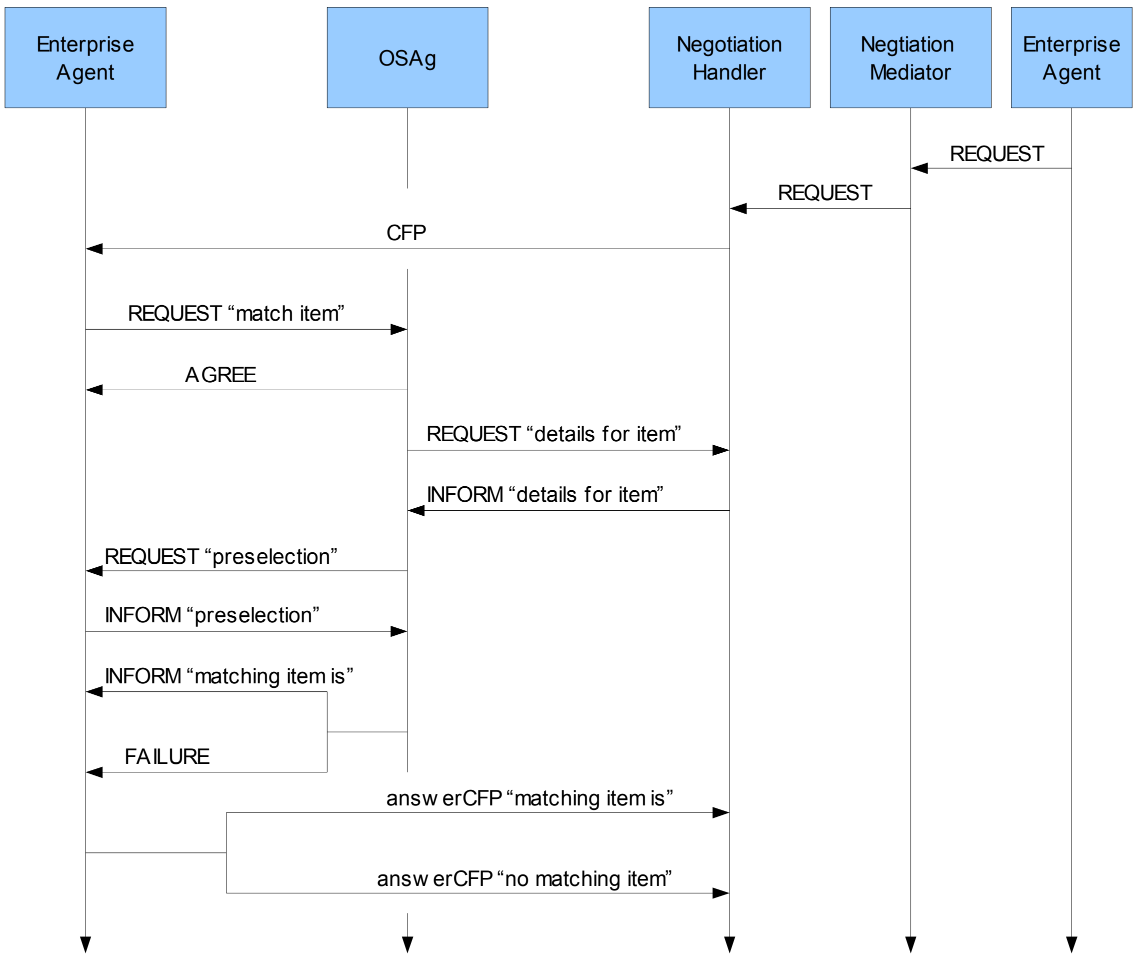


*Figure 6: Ontology mapping protocol*

### 4.4.1 CFP

The Initiator of negotiation is a CEAg (Customer-Enterprise-Agent). This agent send a REQUEST message to the NegMed (Negotiation Mediator). The NegMed creates a NegHandler (Negotiation Handler) which is controlling the negotiation process. NegHandler sends a CFP (Call For Proposal) to all SEAg(Supplier-Enterprise-Agent) in platform to initiate a ontology mapping  process with each SEAg. Aim of this process is to find all SEAg´s  which are providing the item of interest.

After receiving CFP the SEAg will send a REQUEST to OSAg (Ontology Service Agent) which contains as content a string "matching item <item_name>" (<item_name> is a place holder for item name). After OSAg receiving the REQUEST the OSAg reply with a AGREE message.

Content of CFP is the item name as a string (e.g. "engine")

### 4.4.2 Details for item

Next step is to get ontology which describes the item of interest. First the OSAg sends a REQUEST to NegHandler with message content “details for item <item_name>”. Now the NegHandler return with a INFORM message which contains all needed information about the used ontology of CEAg which describes the product. This content use the following structure:

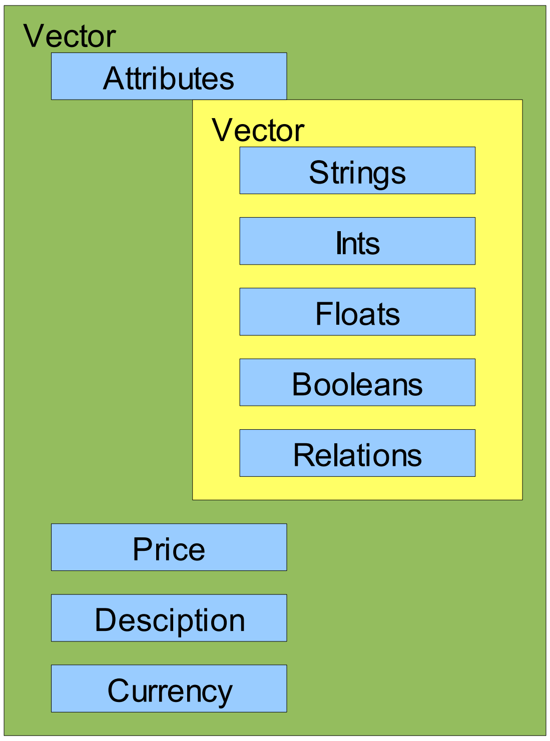


*Figure 7: Structure of details for item content*

whereat blue rectangles are of type string. Objects like “ints” or “float” are concatenated strings which containsq the names of attributes separated by a blank.

### 4.4.3 Pre-selection

For reducing the amount of to be considered item the OSAg request only a list of item from SEAg´s, which are in price range of proposed item. First the OSAg send a REQUEST with the content “preselection <price>” to each SEAg´s. Each SEAg´s answer with an INFORM message with the content like in Figure 7.

### 4.4.4 Result

After matching the pre-selected items with the considered item of CEAg the SEAg has two possible answers. First, if the item match with the CEAg item, the SEAg will call the answerCFP function with parameter *true*. Otherwise the SEAg will call the answerCFP function with *false*. This function tells the NegHandler about the matching result.

## 5 Experiments

Some experiment have been done. First N-Grams experiments by item name, description and attributes. Next semantic similarity by WordNet. Last matching different items and calculate a final result.

Matching item name:

| Proposed item | Matching item | N-Grams result |
|---|---|---|
| 'frontseat' | 'Backseat' | 0.3333 |
| 'frontseat' | 'front_seat' | 1.0 |
| 'backseat' | 'front_seat' | 0.3333 |
| 'wheel' | 'Backseat' | 0.5555 |

Matching description:

| Proposed item | Matching item |
|---|---|
| `'a simple machine consisting of a circular frame with spokes'` | `'a handwheel that is used for steering'` |
| `'a simple machine consisting of a circular frame with spokes'` | `'a simple machine consisting of a circular frame with spokes'` |

`convert to:`

| Proposed item | Matching item | N-Grams result |
|---|---|---|
| `'simple machine consistin circular frame spokes'` | `'handwheel used steering'` | 0.0454 |
| `'simple machine consistin circular frame spokes'` | `'simple machine consistin circular frame spokes'` | 1.0 |

Matching Attributes:

| Proposed item | Matching item | N-Grams result |
|---|---|---|
| Strings: 'color' | Strings: 'color material' | 0.5714 |
| Floats: 'diameter price' | Float: 'diameter price' | 1.0 |
| Ints: 'quantity' | Ints: 'quantity number_of_screws' | 0.8888 |
| | | Total: 0.82 |

Matching item name (Semantic similarity):

| Proposed item | Matching item | WordNet similarity (LCH) result |
|---|---|---|
| 'Wheel' | 'wheel' | 1.01508 |
| 'backseat' | 'frontseat' | 'frontseat' not in WordNet database |
| 'backseat' | 'wheel' | 0.4348 |

In this experiment the OSAg is searching a corresponding item for 'backseat'. There are three items in stock (wheel, backseat and frontseat). After pre-selection by price two items are remaining (backseat and frontseat). There is no description and the items have the same attributes.

```
OSAg: Searching correspondent term for 'backseat' ...
preselected items for: backseat: 'Frontseat' 'Backseat'

OSAg: Pre-Selection proposed: 'Frontseat'
OSAg: N-Grams [item-name]: 0.3333333333333333
OSAg: N-Grams [descriptions]: n.a.
OSAg: N-Grams [attributes]: 1.0
* attr 1.0
ERROR:  not found in WordNet.
Canceling ... since (at least one of the) words not found in WordNet.
OSAg: 'backseat' or 'Frontseat' not found in WordNet
OSAg: Final result for Frontseat: 0.6666666666666666

OSAg: Pre-Selection proposed: 'Backseat'
OSAg: N-Grams [item-name]: 1.0
OSAg: N-Grams [descriptions]: n.a.
OSAg: N-Grams [attributes]: 1.0
OSAg: WordNet-Similarity ['backseat' - 'Backseat']: 1.0150877453695204

* attr 1.0
* WordNet 1.0150877453695204
OSAg: Final result for Backseat: 1.0075438726847603

HIGHEST RESULT: Backseat 1.0075438726847603
Matching item for 'backseat' found: 'Backseat'
```

After matching the item with the highest result is 'Backseat' and the OSAg found item 'Backseat' corresponding with 'backseat'.

More experiments with this algorithm have been done by [1] and [2].

## 6 Conclusion

At my internship I learnt a lot new things. First about software agents and their implementation in Multi-Agent-Systems. About Virtual Institutions, working in a team by integration a service in the platform and about ontologies, ontology mapping and associated algorithms.

## 7 Acknowledgement

I would like to thank Henrique Lopes Cardoso and Rui Neves for the good team work. And I would like to thank Prof. Dr. Norbert Kuhn and Prof. Eugénio Oliveira to make my intership possible.

## 8 References

[1] Andreia Macucelli, "Ontology-based Services for Agents Interoperability", *http://paginas.fe.up.pt/~eol/PUBLICATIONS/2006/Thesis_Malucelli_2006.pdf*, 2006

[2] Daniel Palzer, „Ontology-based Services in Multi-Agent Systems", *http://paginas.fe.up.pt/~eol/SOCRATES/Palzer/*, 2005

[3] Henrique Lopes Cardoso, "Electronic Institution: an E-contracting Platform for Virtual Organizations"

[4] Andreia Malucelli, Henrique Lopes Cardoso, Eugénio Oliveira, "Enriching a MAS Environment with Institutional Services", *http://paginas.fe.up.pt/~eol/PUBLICATIONS/2006/Livro_Malucelli_LCardoso_Oliveira.pdf*, 2006

[5] Andreia Malucelli Daniel Palzer Eugénio Oliveira, "Combining Ontologies and Agents to help in Solving the Heterogeneity Problem in E-Commerce Negotiations", 2005

[6] JADE, http://jade.tilab.com/