published as: S. Linckels, Christoph Meinel: A Simple Solution for an Intelligent Librarian System;
Proceedings IADIS International Conference of Applied Computing (AC2005);
Lisbon, Portugal, Vol. I, pp. 495-503.

A SIMPLE SOLUTION FOR AN INTELLIGENT LIBRARIAN SYSTEM

Serge Linckels, Christoph Meinel
Hasso-Plattner-Institut für Softwaresystemtechnik, University of Potsdam
D-14440 Potsdam, Germany
{linckels, Meinel}@hpi.uni-potsdam.de

ABSTRACT

In this paper, we describe a method to retrieve documents based on the semantics of a user's question, rather than on keywords. It is based on domain ontology and on RDF (*Resource Description Framework*) described knowledge. The user enters his question in natural language, which is then converted into an interpreted sentence of semantically relevant words. Each word is classified in a hierarchy of concepts in order to compute the exact semantics and the degree of importance for each of the used words. The interpreted user question is then mapped to a general assertion. Finally, a semantic query is generated, according to the mapped general assertion and the object values derived from the user question. Hence, only semantically relevant documents (resources) are found. This semantic retrieval method was implemented in an e-learning tool called CHESt (*Computer History Expert System*). We also present in this paper the results drawn from experiments with this tool, which was used as a complement in common school courses.

KEYWORDS

Semantics, information retrieval, multimedia, knowledge base, ontology, e-learning.

1. INTRODUCTION

With the increasing amount of information and knowledge, people are faced more and more with two major problems: the organization of the information to be stored and the search for relevant information. A long time ago, this problematic already left the pure field of databases and has become a general problem of the society ever since, at least with the appearance of the World Wide Web. The problem is not that the answer to the user's question doesn't exist; the problem is to find *the* appropriated information among an ocean of potential documents. It is the famous problem of the needle in the haystack. Today, using a search engine on the web is often the exercise of entering some keywords and then filtering manually 95% of *noise* from the resulting list of links. The problem of finding pertinent and correct information has become one of the main challenges in computer science in the last years. With our method, the awkward work of filtering results is shifted from the user side to the search engine side. This improved view is illustrated with the example of the librarian's problem.

1.1 The Librarian's Problem

Let us suppose that Paul wants to learn about the invention of the transistor. He goes to a library and asks the librarian: "I want to know who invented the transistor?" The librarian perfectly understood Paul's question and knows where to find the right book. He also understood that Paul does not want all available books in the library that explain how a transistor works or those which illustrate in detail the lives of its inventor(s). It is evident for the librarian that Paul only wants one pertinent document in which he will find the answer to his question. In conclusion, we can make the following statements:

For the client:

- Paul formulates his question in natural language (NL).
- Paul has no knowledge about the internal organization of the books in the library.
- Paul does not know what he is looking for in particular; he gave no book title to the librarian.

In Proceedings of the IADIS International Conference of Applied Computing 2005 (IADIS AC2005), 22-25 February 2005, Lisbon, Portugal, Vol. I, pp. 495-503.

For the librarian:

- He is able to understand the client's question (the language and the sense).
- He does not know the answer to the client's question.
- He controls the internal organization of the library.
- From all the existing books in the library, he finds the one(s) that best fit(s) the needs of the client.

It is obvious that the larger the library is, the more books will be potentially pertinent. If Paul wants to be sure that he will only get a very short list of relevant books, then he should go to a specialized library. There, the potential amount of documents is far smaller but the chance to find pertinent results is higher. Visiting specialized libraries also reduces the risk of ambiguity. If Paul asked in a general library for a book about golf, the librarian would have the choice between several possible interpretations: a sport, a German car, a geographical position. However, if Paul was in a library about sports, the context would clarify the question.

1.2 Overview

In this paper we describe a method that allows in a deterministic way to find semantic matching documents in a domain ontology. Furthermore, the user can formulate his question in NL. Our method was used in a multimedia e-learning tool CHESt (*Computer History Expert System*) which was firstly presented in [8]. This improvement allows to replace the keyword search engine of CHESt by a powerful semantic search engine.

2. ONTOLOGY PRELIMINARIES

In this section, we review how domain ontologies are described. Ontologies are formal and consensual specifications of conceptualizations that provide a shared understanding of a domain, an understanding that can be communicated across people and application systems [4]. Domain ontologies capture the knowledge valid for a particular type of domain for example computer history, biology. Organizing knowledge in domains rather than dealing with a large "World Ontology" allows more efficient semantic retrieval. A domain ontology is composed of a common language and a concept taxonomy. Both are explained in this section.

2.1 Domain Language

Most people that are not expert users of modern information retrieval systems are not able to formulate their request in a machine optimized way, for example by combining search terms with Boolean operators. Furthermore, it is possible that they do not use the right domain expressions or make spelling errors.

Our method allows the user to freely formulate his question in NL. It uses a dictionary to process the words from the user's question, even if they contain spelling or grammatical errors. We think that a large scale dictionary like *WordNet* is not the best possible solution for a domain ontology. First of all, different meanings for the same word are possible. Hence, information retrieval systems must set the different interpretations in a context to find the best match. Secondly, large scale dictionaries often lack specific domain expressions. Thus, we propose either to use an existing domain specific dictionary or to create a dictionary of its own. In our implementation, we converted the documents (slides) with a special tool to pure text. Then we extracted some 3500 different words which were used as domain dictionary. Words that the user will use and that are not found in this dictionary are logged and added later.

2.2 Concept Taxonomy

Based on the work of [1], we organize a domain ontology as a hierarchy of concepts (HC). HC's are structures having the explicit purpose of organizing/classifying some kind of data (such as documents). HC's are widely used in many applications, for example web-directories like Yahoo!. Figure 1 shows an example of a simplified HC for computer history. A document describing for example the transistor would be placed in the concept "EComponent" (electronic component), which is a specification of "Hardware". On the one

hand, the more detailed the HC is, the more exact the system can classify an object. On the other hand, a very detailed HC reduces the tolerance for the user question, so that it must be very well and precisely formulated.

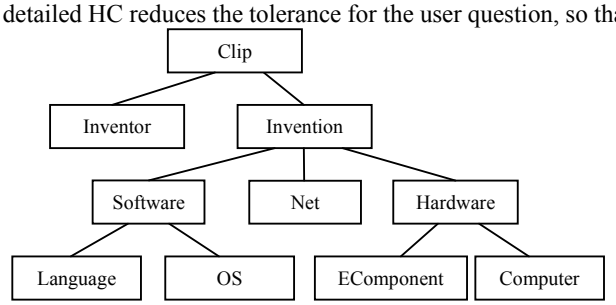

Figure 1: Example of a HC for computer history.

Concept: **Operating System**
s = `chest:OS` *T* = { Windows, Linux, MsDOS, VMS } type = object

Figure 2: Example of a node in a HC.

3. SEMANTIC QUERY PREPROCESSING

In this section, we formally explain how our semantic search method tries to understand the user question. We first declare the domain ontology components (language and taxonomy). Then the meaning of the NL question is detailed.

3.1 Preparing the Domain Ontology

The user can formulate his question by using any possible word of his language. The domain language may or may not contain all these possible words. In fact, it is a subset of the language that the user will use to formulate his question.

Definition 1 (domain language): *Let L be the set of all existing words over a certain alphabet so that* $L \subseteq \Sigma^*$. *A domain language* L_H *is the set of all words that are known in a HC H (we will call them the well-known-words, wkw in the remaining part of our paper) so that* $L_H \subseteq L$.

A domain language can contain verbs, nouns, articles as well as names, numbers, etc. Furthermore, the same domain language can contain words in different languages. The domain language gives no semantic or other description to its elements. It is just a set of stored words. The semantics are attached to each word by classification in a concept hierarchy.

Definition 2 (concept hierarchy): *A concept hierarchy* $H = (V, E, v_0)$ *is a rooted and oriented tree where each node, except the root-node* (v_0), *has exactly one parent. E is the set of all edges and V is the set of all nodes with* $V = \{ (s, T) \mid s \in S \}$ *where s is a unique label and T a set of wkw associated to a node,* $T = \{ t \mid t \in L_H \}$.

A node represents a concept. The words that refer to this concept are regrouped in *T*. We assume that each set of words T_i is semantically related to the concept that the node v_i represents. Figure 2 illustrates this idea with the example of the concept "operating system". Here, words like "Windows" or "Linux" refer to the same concept. Of course, a certain term, for example "Ada" could refer to different concepts; Ada is the name of a programming language but also the name of a person (Ada Lovelace).

Not all words in L_H can be affected to a concept. Only words that are semantically relevant are classified. In general, nouns and verbs are best indicators of the sense of a question [7]. The difference between words that are semantically irrelevant and words that are not contained in L_H is that for the second ones, the system has absolutely no idea, if they are relevant or not.

Definition 3 (label): *A label is a unique identifier in a HC so that for a given label s the system can find the corresponding concept and vice versa. S is the set of all existing labels in a HC.*

Technically, the kind of labels used depends on the encoding framework. We used RDF (*Resource Description Framework*) and RDF Schema to describe all documents (resources) with metadata (see section 4.1). Thus, a label is a namespace prefix (for example `chest`) and a local name (for example `OS`). Together they form the label of a node (for example `chest:OS`). A label also allows to make a statement about the type of a word. We make the difference between predicates (in general verbs) and objects (in general nouns or persons).

Definition 4 (classification): *A document $d \in D$ is classified under a concept k, if d is about k and there isn't a more specific concept k' under which d could be classified.*

In certain cases, a document can be classified in more than one concept. For example the document introducing the ARPA is classified in a concept named "Net" but also in a concept named "Inventor".

3.2 Semantic Interpretation

The interpretation of a user question is divided into two steps: the interpretation of a single word of the question and the interpretation of the whole sentence.

Definition 5 (user question): *A user question q is a sentence that the user formulates in NL, so that $q \subseteq L^*$ and which is composed of words, so that $q = \{w_1,...,w_n\}$ with $n \geq 1$ and $w_k \in L$.*

Definition 6 (word interpretation): *The meaning of each word $w_k \in L$ is made explicit in a HC with the function wi(w) which returns a set of interpretations C_w, so that $C_w = \{ (v_i, x, \sigma) \mid x\, ft\, (vi) : \pi\, (w, x) \geq \varepsilon \}$.*

The function $ft(v_i)$ returns the set of wkw T_i associated to the node v_i. The function $\pi\,(a, b)$ quantifies the similarity of two given words ($a, b \in L$) using a logic W, so that $\pi\,(a, b) = \sigma$.

The choice of W depends on how expressive one wants to be in the approximation of the meaning of the concepts and on the complexity of the NLP techniques used to process words. For example, W could be based on Bayesian Logic to compute the probability that a is similar to b. W could be Boolean Logic to compute if a equals b. We used in our implementation a derivation of the *edit distance* (also known as *Levenshtein distance*). The Levenshtein distance is a measure of the similarity between two strings. The distance is the number of deletions, insertions, or substitutions required to transform a into b. The greater the Levenshtein distance, the more different the strings are. The accepted tolerance is given by ε so that a and b are accepted as similar if their difference is not greater than ε. A given word w is supposed to be semantically irrelevant if it is not found in L_H, or if no similar word with an acceptable tolerance was found in L_H. This heuristic is not needed when supposing that the domain dictionary L_H is exhaustive so that $L_H = L$. In that case the function π could be reduced to a simple equal-test.

w	***wi(w)* = C_w**	**Number of interpretations**
the	{}	0
Windows	{ (v_α,"Windows",0) }	1
Ada	{ (v_β,"Ada",0), (v_χ,"Ada",0) }	2

Figure 3: Example of word interpretations.

Normally, a given word has or has not one semantic interpretation in a given HC. If the word is semantically not important (for example "the"), C is an empty set. If the word is semantically important, C contains its interpretation(s). In rare cases, C can contain several different interpretations, see figure 3 for some examples. Only more precise knowledge about the context of the user question can clarify which interpretation is appropriated. This phase of contextualization is included in the sentence interpretation (see definition 7).

A general problem in the processing of NL are names, especially first names. First names are of no semantic importance but can lead to ambiguity. For example, the name "Bill" can be misinterpreted; it can be a popular American first name, a US-banknote, a statement of money, etc. We suggest to consider all first names as semantically unimportant in LH.

Definition 7 (sentence interpretation): *The meaning of a user question q is computed in a HC by the function si(q), which returns a set Φ of word interpretations, so that*

$$\Phi = \{ \phi \mid \phi \in wi(w_k),\ w_k \in p(q),\ wi(w_k) \neq \varnothing \}.$$

Every word w_k from the user question q is interpreted using the function $wi(w_k)$. The function $si(q)$ filters all semantically unimportant words (noise) from the question q; these are words whose semantic interpretation C is an empty set. Only the semantically relevant words are kept. The function $p(q)$ performs a lexical analysis (see definition 8).

The structure of Φ can be represented like this: $\Phi = \{ O_1, O_2,...,O_n, P_1, P_2,..., P_m \}$ where O_i is an object and P_j is a predicate, n is the number of different identified objects with $n \geq 0$ and m the number of different identified predicates with $m \geq 0$. Experiments showed that the number of objects is on average 2 and the number of predicates is on average 1. This is explained by the fact that users normally formulate short and

precise questions of the basic linguistic structure: subject - verb - object. The functions $o(\Phi)$ and $p(\Phi)$ return the number of objects and the number of predicates in Φ respectively (see figure 4 for examples).

q_1 = "Who invented the very first transistor?"
$p(q_1)$ = "who invented the very first transistor"

q_2 = "Was the Mark I invented in 1939, or in 1940???"
$p(q_2)$ = "was the mark i invented in 1939 or in 1940"

q_3 = "Did H. Aiken once met the German K. Zuse?"
$p(q_3)$ ="did h aiken once met the German k zuse"

Underlined words will be detected as semantically important. For each question $o(\Phi) = 1$ and $p(\Phi) = 1$.

Figure 4: Example of sentence interpretation.

Definition 8 (lexical analysis): *The function p(q) performs a lexical analysis over the question q, which is the process of converting a stream of characters into a stream of words [2], so that* $p:L \rightarrow L^*$ *and* $p(q) = q'$.

One of the major objectives of lexical analysis is the identification of the words in the text. This is a very important step, because it has a direct impact on the quality and the complexity of the semantic interpretation. The lexical analysis performs the following operations.

- Punctuations are replaced by a space.
- Umlauts are converted (for example ö → oe).
- All characters are transformed into lower cases.
- Special characters are striped.
- Double spaces and spaces at the end and at the beginning of the question are removed.
- Processing numbers.
- Processing hyphens.

The last two operations are the most difficult. First, numbers are the biggest problem in lexical analysis, because it has to be decided if a number is semantically important or not. If the retrieval system must make a difference between for example the old computers "Mark 1" and "Mark 2", then the numbers are important. If this difference is not relevant, then the numbers have no semantic importance and can be removed. Another problem are Roman numbers, for example "Mark I" and "Mark II". Only an exhaustive linguistic analysis of the structure of the question reveals if an "I" represents an article or a number, if "II" represents a spelling error or a number. Numbers in words should not be removed, especially not if the domain ontology is about computers. For example, the word "PDP1" or "B2B" must be considered as axioms. Second, it has to be decided whether hyphens are replaced by a space or whether the hyphenated word represents an axiom and cannot be separated. For example, "Ms-DOS" is not necessary the same as "Ms DOS". "Ms-DOS" is the name of an operating system, but "Ms" can be interpreted as the abbreviation of a company and "DOS" as the abbreviation for old operating systems in general.

4. SEMANTIC RETRIEVAL

In this section we will explain, how the knowledge base is queried in order to find relevant documents. First of all, we explain how the knowledge base must be structured semantically. Secondly, we will define a set of general assertions to regroup the user questions. Then, we will describe how a question is semantically interpreted to retrieve relevant documents. Finally, we will present a simple algorithm, how semantic queries are generated.

4.1 Structuring the knowledge Base

Most retrieval systems work by indexing important words in the documents. However, we assume that the documents in the knowledge base can be of any form (for example hypertext, video). Therefore, the search for pertinent documents cannot be done by processing the documents' content, because it is not guaranteed

that it can be accessed easily. Furthermore, we are not interested in the documents' content but only in the semantics of the document (what it is about). Thus, the document should be described with metadata which explain the meaning of its content.

We used RDF(S) to describe every document in the knowledge base with metadata. There should be enough metadata to allow classification of the documents in the domain taxonomy (see section 2.2). An example is presented in figure 5; it shows a multimedia resource in *RealMedia* format (`.rm`) that is about the invention of the transistor. The metadata describe that this resource is classified under the concept "EComponent" (electronic component), that it was invented in 1947 and that a resource exists for each of its creators.

```
<chest:EComponent rdf:about="…transistor.rm">
  <DC:title>Transistor</DC:title>
  <DC:date>1947</DC:date>
  <DC:creator rdf:resource="…shockley.rm" />
  <DC:creator rdf:resource="…bardeen.rm" />
  <DC:creator rdf:resource="…brattain.rm" />
</chest:EComponent>
```

Figure 5: Example of an RDF description for a resource.

The creation of the RDF description for the whole knowledge base is often a painful task if it is done manually. The automatic annotation and the information extraction are hot topics in computer science, especially in the field of web engineering [6]. This task is even harder when dealing with multimedia sequences [11]. In our implementation, we used templates to create the documents so that metadata could be found and extracted more easily. More than 85% of the RDF description was generated automatically.

4.2 Defining a Set of Domain Assertions

q_1 = "Who invented the **transistor**?"
q_2 = "Do you know the inventor(s) of the **transistor**?"
q_3 = "What were the names of the persons who built the **transistor**?"

The predicates "invented", "built" and "inventor" have the same semantic interpretation and refer to the same label ($\lambda_1 \equiv$ `dc:creator`):

$$wi(\text{"invented"}) = \ldots = wi(\text{"built"}) \rightarrow fs(wi(.)) = \texttt{dc:creator}$$

The only object found in all three questions is "transistor", thus the three sentences have the same interpretation and are mapped to the same general assertion:

$$si(q_1) = si(q_2) = si(q_3) \rightarrow g(si(.)) = \textit{An invention was invented by an inventor}$$

Figure 6: Example of an assertion mapping.

All systems that interact with humans have some concepts about their users [3]. The modeling of behavior can be useful in many ways, for example for defining the ontology taxonomy or the criteria of the lexical analysis. If the system is dedicated to a certain group of users, for example students in biology, then predications about the nature of the user questions can be made. In general, user questions can be categorized in general assertions. In consequence, a set of possible assertions can be created for each domain ontology to regroup all imaginable user questions. We learned that it is not useful to create a lot of assertions. As for the librarian's problem (see section 1.1), the system must not be able to understand the question to give the right answer, but it must be able to understand the sense of the question to find relevant documents in which the user can find the answer.

Definition 9 (domain assertions): *A set of domain assertions A_H is the categorization of few abstract statements for a given HC H, so that AH = { $a_1,\ldots,a_n$ } with $n \geq 1$.*

Based on empirical data, and the resulting behavior patterns of the involved users and their questions, we created only two general assertions in our implementation about computer history:

1) An invention was created by one (or more) inventor(s).
2) An invention is composed of one (or more) part(s).

Definition 10 (assertion mapping): *The assertion $a_i \in A_H$ to which a sentence interpretation Φ is mapped is given by the allocation function g(Φ), so that*

$$g(\Phi) = \begin{cases} a_1 & \exists x \in \Phi : fs(v \in x) = \lambda_1 \\ a_2 & \exists x \in \Phi : fs(v \in x) = \lambda_2 \\ \ldots & \\ \{\ \} & \text{otherwise} \end{cases}$$

The function $fs(v)$ returns the label of a node v. A user question can be mapped to an assertion if it contains a corresponding label (in our example λ_1 or λ_2). A complete example is given in figure 6. The mapping function can be improved by allowing that one question can be mapped to different assertions, thus $g(\Phi)$ returns a set of mapped assertions. This decision depends on the number and the nature of assertions, as well as on the behavior patterns about the users.

4.3 Finding Relevant Documents

The aim of every search engine is to find relevant documents. This means in our case to overlap the mapped assertion with the values extracted from the user question. The number of queries to generate depends on the number of relevant objects found in the user question.

Definition 11 (question interpretation): *The interpretation $\mathcal{I}$ of a user question q in NL for a HC H and an allocation function g is written $\mathcal{I}\ [q]_g^H = R$ with R being the set of relevant documents, so that*

$$R = \{\ (d, \sigma) \mid d \in D, \sigma \geq \in\ \}.$$

$\mathcal{I}$ first interprets the user question q using the function $si(q)$, then maps it to an assertion using the function $g(si(q))$ and finally overlaps the mapped assertion with the values of the semantically important words from the user question q (see figure 7 for an example). Each relevant document d comes with a relevancy quantification σ that is expressed in a certain logic W (see section 3.2). This allows to rank them according to their pertinence if more documents were found.

> q = "Who invented the transistor?"
> $\mathcal{I}\ [q]_g^H \rightarrow$ The object in the concept of electronic components (`chest:EComponent`) which title (`dc:title`) is "transistor" was invented (`dc:creator`) by x. Find the resource(s) x.

Figure 7: Example of retrieval.

4.4 The algorithm SEMA

We express in the algorithm called SEMA how semantic queries can be generated for a HC about computer history (see figure 8). Based on the assumptions mentioned in section 3.2 about the statistical number of objects and predicates in a user question, we elaborated 4 rules for searching relevant documents. The function $ni(O_i)$ returns for a given object O_i the corresponding label s. We use RDQL formalism as query language.

- Rule 0 is fired if the user question contains no semantically known object, normally when it is out of the context of computer history (for example: "Who invented the penicillin?").
- Rule 1 is fired if the user question contains one object. This is the most common case (see figure 4).
- Rule 2 is fired if two objects are in the user question. Here, we improved the mapping function and elaborated three possible cases:
  - Both objects are inventors (for example: "Did Aiken and Zuse once meet?")
  - Both are inventions (for example: "Who built the Z1 and the Z2?").
  - Both are of different concepts (for example: "Did Zuse build the Z1?").
- Rule N is fired in more rare cases if there are three or more objects in the user question (for example: "Did Aiken or Zuse built the Z1 and Z2?").

```
SEMA(q)
 Φ ← si(q)
 λ1 ← dc:creator
 if o(Φ) = 0 then Rule0
 if o(Φ) = 1 then Rule1(O1)
 if o(Φ) = 2 then Rule2(O1, O2)
 if o(Φ) > 2 then RuleN(Φ)

Rule0:  // 0 objects in user question
 R = { }

Rule1(O1):  // 1 object in user question
 if g(Φ) = λ1 then R = SELECT ?x WHERE ?x; λ1; O1, ?x; rdf:type; ni(O1)
 else R = SELECT ?x WHERE ?x; ni(O1); O1

Rule2(O1, O2): // 2 objects in user question
 if ni(O1) ≡ inventor and ni(O2) ≡ inventor then
  R = SELECT ?x WHERE ?x; λ1; O1, ?x; λ1; O2, ?x; rdf:type; ni(O1)
 else if ni(O1) = invention and ni(O2) = invention then
  R = SELECT ?x WHERE O1; λ1; ?x, O2; λ1; ?x, ?x; rdf:type; ni(O1)
 else
  ensure: O1 = inventor and O2 = invention
  R = SELECT ?x WHERE ?x; λ1; O1, ?x = O2

RuleN(Φ): // more than 2 objects (rarely the case)
 M = Φ × Φ
 for each mi, mj ∈ M
  Rule2(mi, mj)
 next
```

Figure 8: SEMA algorithm.

5. RELATED WORK

We found two very promising projects that have several concepts in common with our method. The Institute of Computer Science form the FU-Berlin works on a *Semantic Web for Pathology* [10]. The project develops a prototypical system to process and retrieve pathologic image descriptions and diagnostic findings, especially in pulmonary diseases. Information should be continuously semantically annotated and ontologically linked, allowing content oriented searches, which is unlike today's data bases information. Expert knowledge contained in text and image descriptions will then be available and usable for other application scenarios. It can then be used for quality assurance, diagnostic support and differential diagnosis, for statistical evaluation, presentations and as teaching material. When compiling a report on diagnostic findings semantic information is automatically collected by computer linguistic analysis and entered into a semantic network, based on a pathology specific model. The system will provide this information for various different purposes. In addition, an integration component tries to integrate information from external knowledge sources. The system is technologically based on Semantic Web standards like XML, RDF and RuleML plus approved medical standards and patterns (UMLS, HL7, SCORM).

The university Blaise Pascal Clermont2 and the university Claude Bernard Lyon1 have recently published their work on an algorithm to automatically rank documents returned by a search engine in the WWW following a query in natural language [5]. Description Logic was used as a formal representation language for specifying documents and queries. The matching step consists in comparing the two terminologies obtained from a query and a document. Given two terminologies $\mathcal{T}_Q$ and $\mathcal{T}_D$ describing a query Q and a document D respectively, the goal is to find the elements in $\mathcal{T}_Q$ and $\mathcal{T}_D$ that match. This is done by a matching function that takes two terminologies as input and produces a one-to-one mapping between defined concepts

of the two terminologies that correspond semantically to each other. Finally, the documents are ranked according to the size of the extra information contained in the query and not in the documents. The extra information is calculated with the help of the difference operation between pairs of mapped elements. The proposed algorithm computes the difference between $\mathcal{ALE}$-concept descriptions. It takes into account linguistic relations (synonymy, hyponymy...) between concept names occurring in the two descriptions.

6. CONCLUSION AND FUTURE WORK

The development of an effective and well working semantic search engine has been one of the biggest challenges in the last years for many researchers. We entered this challenge by shifting it from the semantic web to a domain ontology problem. This simplifies the task, especially because it has a well described native knowledge base. In this way, we could concentrate on the semantic interpretation of NL questions. We described in this paper a formal representation of a deterministic semantic search method using domain ontology technology. The algorithm was implemented in an e-learning expert system about computer history called CHESt. We also presented experiences about the multimedia aspect of the interface and pedagogical advantages of such a semantic retrieval system.

Our current research aims to put into effect two main improvements: (i) a feedback and learning mechanism and (ii) the mapping from our domain ontology to other related ontologies. First of all, the method as it is presented in this paper lacks learning capacities. All queries are built from bottom up, without considering former queries and without feedback possibility from the user, for example the possibility to state whether the delivered documents provided an answer to his question. We think that using Description Logic as formal representation will offer the needed theoretical background to improve the inferring and reasoning capacities of the method [9]. Secondly, we assume that all data in the native knowledge base are well described with the same vocabulary and have the same structure. Hence, creating such a knowledge base is a big investment. One could imagine that mappings between the native and trusted knowledge bases could be established, so that a query could be forwarded to other knowledge bases or distributed to a grid of ontologies.

REFERENCES

[1] Bouquet P. et al., Semantic Coordination: A New Approach and an Application, in Proceedings of ISWC2003, Sanibel Island, FL, USA (2003).

[2] Baeza-Yates R., Ribeiro-Neto B., Modern Information Retrieval, Addison-Wesley, USA (1999).

[3] Carstensen K.-U. et al., Computerlinguistik und Sprachentechnologien, Spektrum, Heidelberg-Berlin, Germany (2001).

[4] Fensel D., Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce, Springer, Berlin-Heidelberg-New York (2004).

[5] Karam N. et al., Semantic Matching of Natural Language Web Queries, in Proceedings of ICWE 2004, Munich, Germany (2004).

[6] Kiryakov A. et al., Semantic Annotation, Indexing, and Retrieval, in Proceedings of ISWC2003, Sanibel Island, FL, USA (2003).

[7] Kupiec J., MURAX, A Robust Linguistic Approach For Question Answering Using An On-Line Encyclopedia, in Proceedings of ACM-SIGIR'93, Pittsburgh, PA, USA (1993).

[8] Linckels S., Meinel Ch., An Application of Semantics for an Educational Tool, in Proceedings of IADIS AC2004, Lisbon, Portugal (2004).

[9] Linckels S., Meinel Ch., A Simple Application of Description Logics for an Educational Tool, in Proceedings of IADIS AC2005, Lisbon, Portugal (2005).

[10] Tolksdorf R. et al., A Semantic Web for Pathology, http://www.inf.fu-berlin.de/inst/ag-nbi/research/swpatho/.

[11] Troncy R., Integrating Structure and Semantics into Audio-visual Documents, in Proceedings of ISWC2003, Sanibel Island, FL, USA (2003).