Automatic Reading and Learning from Text

Ana Oliveira
Instituto Superior de Engenharia de Coimbra
Quinta da Nora, Coimbra, Portugal
aalves@isec.pt

Francisco Câmara Pereira
Departamento de Engenharia Informática da Universidade de Coimbra
Pinhal de Marrocos, Coimbra, Portugal
camara@dei.uc.pt

Amílcar Cardoso
Departamento de Engenharia Informática da Universidade de Coimbra
Pinhal de Marrocos, Coimbra, Portugal
amilcar@dei.uc.pt

Keys:
Natural Language Processing, Text Analysis, Case-based Reasoning, Dialogue, Concept Maps

Technical Area:
Information Retrieval

Abstract

This paper presents our more recent research on the area of text reading and understanding and knowledge extraction. More specifically, we give an overview of TextStorm and Clouds: two modules for the construction of concept maps. The first one deals with the task of extracting relations between concepts from a text file, while the latter concentrates on completing these relations and extrapolating rules about the knowledge in hand. This is a hybrid framework that applies two different areas of Artificial Intelligence: Natural Language Processing (in TextStorm) and Machine Learning (in Clouds). We show an example and draw conclusions.

1. Introduction

This paper presents a framework that aims at extracting concept maps [Novak and Gowin, 84] from Natural Language texts. This representation, the concept maps, is a simple subtype of semantic networks that consists of graphs in which nodes are concepts and arc are relations. In this simplified semantic network there is no additional features for arcs (i.e., no inheritance or equal arcs, e.g. “isa” or “ako”). This work is composed of two different autonomous modules: TextStorm, a text reader that extracts concept relations through the parsing process; Clouds, a concept map constructor that uses Machine Learning inspired algorithms to complete the map through dialogue with the user.

Seen from the point of view of a user that wants to build a concept map about a given domain, he/she must give TextStorm a file with relevant facts about it, and then answer Clouds’ questions. In the end, he/she will get a concept map and a set of learned rules about the information involved.

As shown in [omitted reference], these concept maps are very important to the project Dr. Divago, a system that uses metaphor mappings [Veale and Keane, 93] to search for solutions in a multi-domain environment. Apart from this central motivation of TextStorm and Clouds, we think this project can add promising perspectives on text analysis and understanding. Furthermore, concept maps are considered by some educational psychologists [Novak and Gowin, 84] as a very important tool to improve learning. If this were so, wouldn't it be interesting to have a teacher contrasting his own concept maps with his students'? With this system, it is possible to have several users having a dialogue with a computer about a domain, starting with a common initial Knowledge Base (read and converted by TextStorm).

The system is at the moment on a stable development version, and we will show later in this paper, in section 4, an example of its current performance. Previously, we will present TextStorm in detail, after which an overview of Clouds will be done (sections 2 and 3). The final sections (5 and 6) will be dedicated to a discussion of the limitations and conclusions.

2. TextStorm

One important part of the process of understanding a text consists on apprehending its underlying interrelations of concepts. Furthermore, a tool able to extract automatically such information will fit the needs of several knowledge dependent systems, like those oriented towards translation, intelligent search, concept learning, ontology extraction and others. Among those related to concept learning, we can find Clouds, a concept map building tool which will be explained later. TextStorm was initially thought about as a simple add-on Natural Language interface to facilitate user's work with Clouds.

Soon after the beginning of its development, and since we didn't find any similar work (at least, in which respects to its applicability to our project), we decided to invest on this work as a potentially autonomous module. In fact, in the initial survey phase, prior to TextStorm development, we mostly found applications and projects that relied almost entirely on extra-textual information (like lexical databases) to generate a model of understanding (e.g. some tools of summarization and statistics-based text analysis). We believe some of the difficulties in this area may be softened through an equal concentration of efforts on the text itself. We refer to the work at [Hahn, Klenner & Schattinger, 96] and [Ciravegna et al, 99] as interesting projects towards this goal.

In the work of [Hahn, Klenner & Schattinger, 96], beyond the information underlying in a text, they suggest using a suitable knowledge base of a specific domain. Thus, new concepts are inferred not only from text, but also from an initial set of other domain concepts.

Ciravegna and colleagues propose to extract, from the parsing process of text files, relations between inherent constituents. They represent these relations in a Quasi Logical Form (QLF) and summarize the relevant information in normalized templates that are adaptable to different user needs.

On its hand, TextStorm is a Natural Language tool that extracts binary predicates[1] from a text file using syntactic and discourse knowledge. These predicates will feed another module, named Clouds, that infers knowledge based on machine learning.

[1] These predicates have the common Prolog form: *functor(Argument 1, Argument 2)*

TextStorm doesn’t need any preview knowledge about the discussed domain. It receives a text as initial base of the information extraction. After TextStorm tagged a text file using the external lexicon (WordNet base), it builds predicates that map relations between two concepts from parsing of sentences. Its practical goal is to be able to extract from utterances like “Cows, as well as rabbits, eat only vegetables, while humans eat also meat”, the predicates {eat(cow, vegetables), eat(rabbit, vegetables), eat(human, vegetables), eat(human, meat)} which will form its concept map. As we will show later, then Clouds will be able to produce rules like “isa(X, vegetarian):-eat(X, vegetables)”[2]. Although we feel the limit of 2 arguments per predicate as highly restrictive, we imposed ourselves this constraint as a development condition.

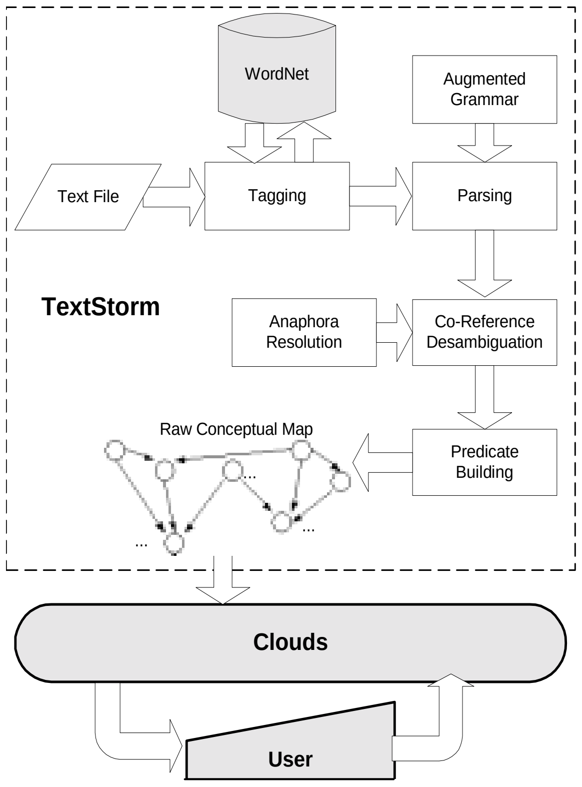


Figure 2.1 -TextStorm Architecture

[2] These rules have the common Prolog form *Conclusion:- premisse1, premisse2, ..., premisseN*

As the above figure suggests, the program includes a tagging module to find all parts of speech to which a word may belong. For this task, it accesses the WordNet [Miller et al, 1993] database. Then, the text is parsed using an augmented grammar.

A fact about two concepts specifies an existent relation between them. Generally, this relation is identified by the main verb in a sentence. And where can we find the binary predicate concepts? Generally, the subject represents a concept whose relation, or property, is passed in a sentence. Thus, an object, or qualifier, that is in the Verbal Phrase shall be considered as the second concept in a binary predicate. For example, in the sentence “Jupiter is a big planet”, there is a relationship between the concept ‘Jupiter’ and ‘planet’: isa. This originates the predicate “isa(jupiter, planet)”. Furthermore, ‘Jupiter’ has the ‘big’ property, leading to the predicate ”property(Jupiter, big)”.

Sometimes, finding concepts in a dependent sentence isn’t clear for an automatic tool. If some ambiguity arises in this process, TextStorm shall apply Anaphora Resolution and/or Context-Dependent Analysis at the Co-Reference Disambiguation Module. Traditional text disambiguation through Anaphora Resolution is essentially founded on a model of Anaphora Resolution based on History Lists [Allen 1995]. A history list is a list of discourse entities generated from the preceding sentences, where the most recent is given much more importance than the first sentence that was analyzed. Internally, there is a history list that is supplied with new predicates produced in the previous iteration.

Since the knowledge base in the beginning of the TextStorm’s session contains no data, the text concepts are formalized depending on the context where they are found. Thus it may be possible, in another session, that the same concepts have different interrelations inferring different knowledge.

By now, WordNet [Miller et al, 93] gives TextStorm only a lexical classification of the words in the parsing process, leaving out other WordNet information such as antonyms, hyponyms, hypernyms, meronyms, etc. I.e., Wordnet is just used to supply lexical verification of words present in sentences. Since, in real world, concepts in a text are not named every time by the same way, TextStorm uses synonymy semantic relationship from WordNet to identify the concepts that were already referred before with a different name. It is important to say that WordNet itself is organized according to synonymy: words are joined together in lots called synsets (essential structures of this database).

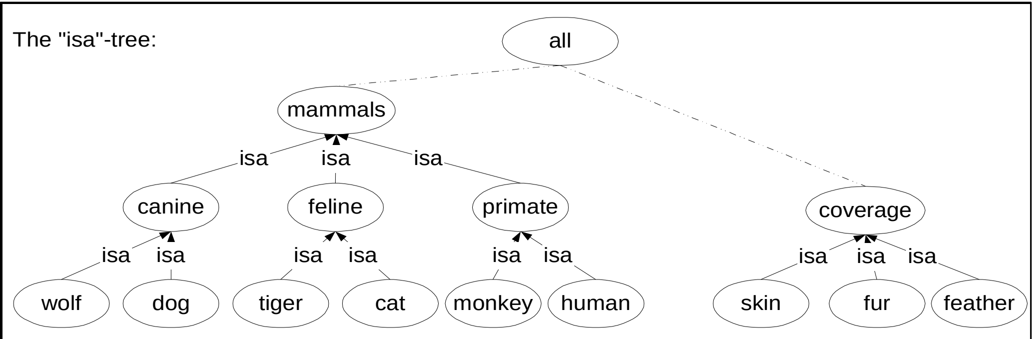
As will be seen in the section dedicated to experiments, the result of this whole process is a list of predicates that represent the concept interrelations TextStorm has detected. This list is then the input to Clouds, which, through dialogue with the user, will try to clarify as much important points as it can.

At present time, TextStorm only uses affirmative and declarative sentences. An interesting point would be to analyze negative sentences as negative examples of the knowledge database, as well as to include temporal reference to establish a non-monotonic database.

For a future improvement, we propose a mapping of n dimension predicates according to syntactic structures of related sentences.

3. Clouds

Integrated on a wider framework named Dr. Divago [omitted reference], Clouds [omitted reference] is responsible for the interactive construction of concept maps. As told before, a concept map in this system consists on a set of binary predicates that represent relations between concepts. Clouds was designed to accept any relation and concept the user inputs, building gradually in parallel the corresponding ontological “isa” tree and learning some particularities of the domain. It applies two different techniques of Inductive Learning in order to extract regularities on the relations and concepts of the map: a best current hypothesis [Mill 1843] based algorithm to learn the categories of the arguments of each relation; and an Inductive Logic Programming [Muggleton 1992] based algorithm to learn the contexts that are recurrent in each relation. The input for both algorithms consists on the binary relations of the concept map, each of them being a new isolated example (positive or negative) to the process of learning.



The examples:		Hypotesis
first positive example:	**have(tiger,fur)**	⇒have(tiger, fur).
second positive example:	have(cat, fur)	⇒generalization: have(feline, fur).
third positive example:	have(dog, fur)	⇒generalization: have(mammals, fur).
negative example:	~have(human, fur)	⇒specialization: have(canine, fur), have(feline, fur).

Figure 3.1 – Learning the relation “have”. An example of generalization and specialization in the Clouds categorization-learning algorithm.

It is expected that gradually in a concept map construction session, Clouds will get a more helpful performance by asking questions to the user about new concepts and new relations it suspects to exist.

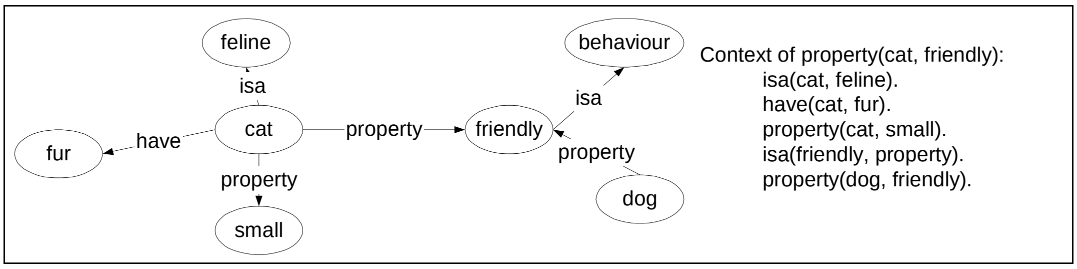
The Inductive Logic Programming based algorithm builds prolog-like rules that aim at describing relations in terms of its context. We consider the context of a relation to be the set of neighbour connections its arguments have, including the "isa" relations. An example of such a context is shown in figure 3.2.

After introducing the new relations (positive examples) "property(dog, small)" and "have(dog, fur)", the algorithm could conclude that

```
property(X, friendly):-property(X, small), have(X, fur).
```

This conclusion is directly drawn from the fact that both contexts of "property(cat, friendly)" and "property(dog, friendly)" have the relations "have(X, fur)" and "property(X, small)" (with X={dog, cat}).

Specialization is obviously made in the opposite direction: if a negative example is introduced, these rules are readapted to exclude it (by adding a new premise and/or by dividing it into new more specific rules).



```
Context of property(cat, friendly):
        isa(cat, feline).
        have(cat, fur).
        property(cat, small).
        isa(friendly, property).
        property(dog, friendly).
```

Figure 3.2. – The context of the relation "property(cat, friendly)"

This system is described in more detail in [omitted reference].

4. How does it work?

We will now show an example of a session, so that we can clear out some points about this work.

Suppose we are interested in building a concept map concerning biological kingdoms. We could go directly to Clouds, if we wanted to, and build it all from scratch, but some additional dedication and patience would be required, since a large set of primitive concepts would be necessary (e.g., What kinds of animals are there?). TextStorm's role on this framework is to bypass this step, becoming much easier and interesting to interact with Clouds. If we input, for example, the following text to TextStorm:

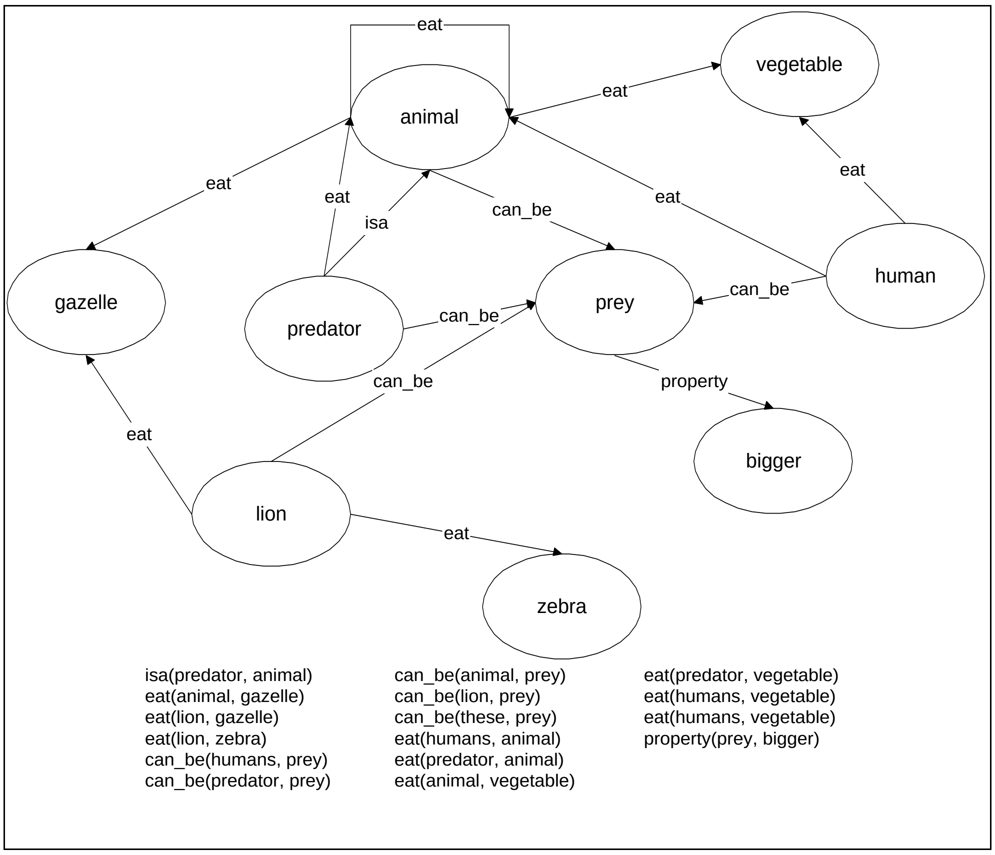
"*A predator is an animal that eats other animals. For example, Lions eat gazelles and zebras. These are the preys. Humans are predators, but they can be preys too.*

Vegetarian animals are usually preys, while predators are obviously carnivores. Humans are omnivores, because they eat both animals and vegetables.

Another difference between those two groups is that predators are small and fast, while the preys are bigger and slower."

TextStorm analyzes each sentence to find inherent relations and concepts. In the first sentence ("A predator is an animal that eats other animals."), there is an ambiguity to define the subject of the second sub-sentence ("eats other animals"). Here, the program solved successfully this situation (concluding "eat(predator, animal"). The second sentence contains two objects connected with "and" ("gazelles" and "zebras"), which should supply the predicates "eat(lions, gazelles)” and “eat(lions, zebras)”. Keywords like *these* and *they* that refer to previous context (*anaphora*) generate several ambiguities (the word “These” in the third sentence can refer to “gazelles”, “zebras”, “lions”, two of them, etc.), so TextStorm postponed the correspondent choice to the user. In the fourth sentence, it extracted the correct relations “isa(humans, predator)” and “isa(humans, prey)”.

In the second paragraph, it started with an interesting choice (it “built” the concept “vegetarian_animals”, which is more specific than “animals”) and solved correctly an anaphora situation (“they” corresponds to “Humans”), although it didn’t process correctly the concept “both”. Its output to Clouds will therefore be the raw concept map in the figure 4.1.



```
isa(predator, animal)
eat(animal, gazelle)
eat(lion, gazelle)
eat(lion, zebra)
can_be(humans, prey)
can_be(predator, prey)
can_be(animal, prey)
can_be(lion, prey)
can_be(these, prey)
eat(humans, animal)
eat(predator, animal)
eat(animal, vegetable)
eat(predator, vegetable)
eat(humans, vegetable)
eat(humans, vegetable)
property(prey, bigger)
```

Figure 4.1. – Raw concept map generated by TextStorm

When Clouds receives this input, it will try to process each line as if it was an example given directly by the user (such as there in fig 3.1), therefore it will naturally start asking questions about missing knowledge that wasn’t acquired from text (e.g., "what is an animal"?). Remember that Clouds has an initial ontological "isa"-tree that contains some primitive generic concepts, so, in a first phase, it will need user's assistance to *attach* the new "isa" relations to the primitive tree.

After the establishment of the "isa"-tree, it will then proceed to extract conclusions from other relations. In the early example, we can see that there are few non-"isa" relations (as said above, some of them should be "property"), although there are sufficient "eat" relations to allow Clouds to start generalizing (it needs only two examples to make a rough generalization). It will then interact with the user, who is now free to introduce any new knowledge to the concept map (allowing Clouds to keep improving its performance).

Some of the questions it asks will be:

```
Clouds: Define animal with the predicate "isa"
User: isa(animal, living_entity).
Clouds: Define lions with the predicate "isa"
User: isa(lions, animal).
Clouds: Define gazelle with the predicate "isa"
User: isa(gazelle, animal).
Clouds: Complete the relation eat(carnivore,animal)
User: eat(lions, humans).
Clouds: Complete the relation eat(animal,animal)
User: eat(humans, gazelle).
```

Then, proceeding with the session, and after introducing the following relations (along with the necessary "isa" relations):

```
property(panther, black).
property(zebra, black).
property(zebra, white).
eat(gazelle, grass).
property(rabbit, white).
eat(rabbit, grass).
```

Clouds will arrive to the conclusions that:

```
ILP Algorithm:
        eat(A,gazelle):-isa(A,predator).
        eat(A,grass):-isa(A,prey) , isa(A,vegetarian_animals) , property(A,white).
        property(A,B):-isa(A,prey) , isa(B,color).

Categorization algorithm:
        [eat(predator,animal),eat(animal,animal),eat(animal,grass)]
```

And will ask questions like:

```
Clouds: Is it true that eat(zebra, grass)?
User: y.
Clouds: Complete the relation eat(panther, animal)?
User: eat(panther, zebra).
```

As this example has shown, the two modules have the complex task of processing Natural Language text and understanding its domain through Machine Learning. We think that TextStorm can facilitate the user's concept map construction, particularly if there is care in the choice of the text file.

We performed some tests with TextStorm using 21 small text files of size varying between 319 and 12 557 bytes. Theses were pertaining to 3 different types: 4 articles, 8 educative texts and 9 manuals:

articles	educatives	manuals
aerospace.txt	*antelope.txt*	*animalrights.txt*
cables.txt	*asteroids.txt*	*cmulisp.txt*
eclipse.txt	*cambodja*	*cockroach.txt*
sds.txt	*cicada.txt*	*domviolence.txt*
	culture.txt	*formz.txt*
	insectos.txt	*grafic.txt*
	otte.txt	*pci.txt*
	whirling.txt	*thesis.txt*
		win95.txt

The articles are characterized by representing complex information about a scientific domain, while the educative texts introduce a subject by a simple and direct way. Manuals are between these two types, since they are available for the general public.

TextStorm analyzed this set of texts and the results are shown in fig. 4.2. We mean correct predicates as those generated by TextStorm that were coherent with the respective input text. Examining the graph, we can conclude that TextStorm reached a correctness mean of 52%.

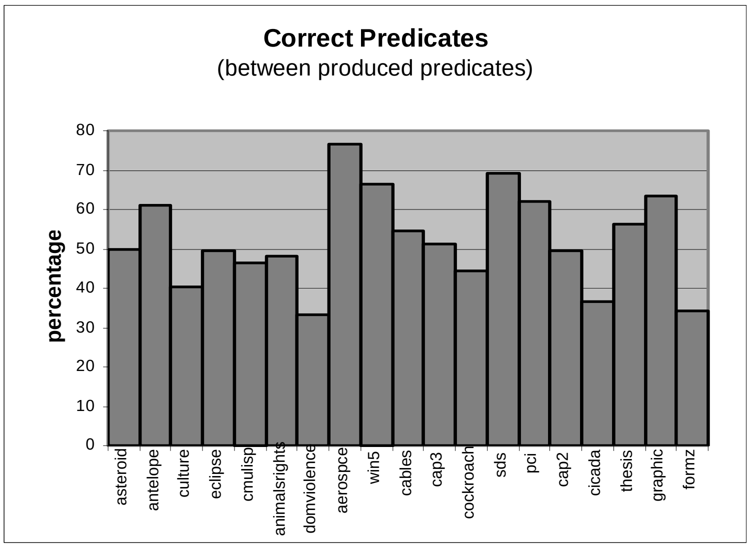


Figure 4.2. – Experiment results

The other predicates were not successfully extracted for some known reasons: part of them correspond to sentences where binary predicates are not the ideal representation, requiring predicates whit n-dimensions; the other part derives from the ambiguity of words themselves (e.g. the word “will” may be a verb or noun). Furthermore, as we could see, there are some current limitations in both modules (TextStorm and Clouds), which will be discussed later.

5. Anaphora Resolution using CBR

Before implementing our anaphora resolution approach, we were conscious that there are various techniques to solve part of the general problem [Allen 95, Hahn & Strube 96]. Then, after reflecting about how humans naturally find a co-reference in a text, we arrived to the conclusion that a paradigm such as case-based reasoning can be applied. Indeed anaphora resolution has lots of “special cases” that known techniques avoid solving. Part of this process depends on how the problem was solved in the past. It seems for us that we can apply Case-Based Reasoning (C.B.R.) on Anaphora Resolution.

We designed a first model of C.B.R. that aims to resolve some anaphora that holds just simple cases. Each case consists of a problem, problem context and a solution that was used to solve the problem. The *problem* is the co-reference (anaphora) to resolve (sometimes identified by a pronoun), while the *context* is represented by two sentences: the one in which the anaphora occurs and the immediately previous sentence. This forced proximity seems a limitation, but after analyzed several kinds of anaphora, when a co-reference is about a concept that is not in the late sentence (i.e., the immediately previous sentence may include a co-reference, too), first we resolve this previous sentence (if necessary). After resolved, this previous sentence will influence the resolution of the current co-reference. Finally the *solution* is the concept (antecedent) that was referred by anaphora (e.g. see fig. 5.1). In other words, CBR is applied sequentially on sentences; an identified solution can help resolution of subsequent cases.

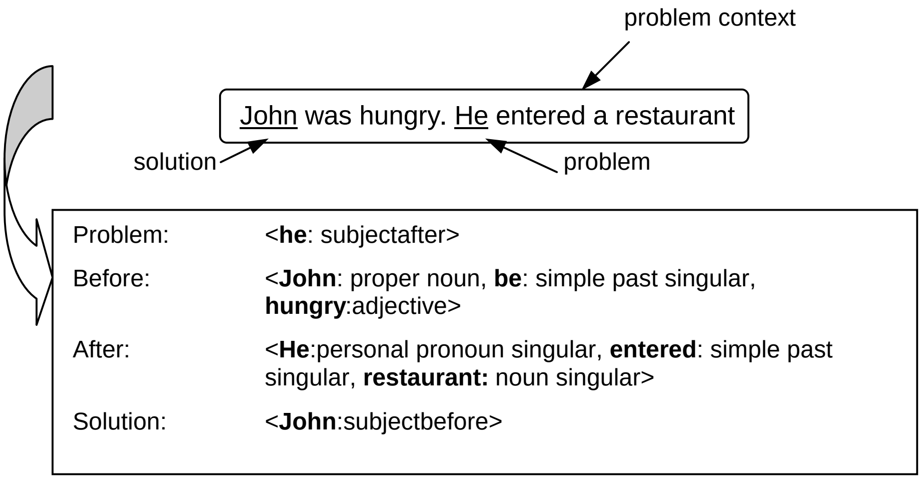

problem context
John was hungry. He entered a restaurant
solution
problem
Problem: <he: subjectafter>
Before: <John: proper noun, be: simple past singular, hungry:adjective>
After: <He:personal pronoun singular, entered: simple past singular, restaurant: noun singular>
Solution: <John:subjectbefore>

Figure 5.1. – An example of a case

Basically, the model extracts the syntactic and part-of-speech classification for main elements (subject, main verb or object) in the two sentences of a new case. Then, it searches for a similar case that was resolved in the past. The solution of this similar case is adapted to this new situation finding the word that has the same syntactic function. If any case was encountered, the manual completion of this new is required. This will increase the case base in a reliable way.

After some experiments, we concluded that case base size and CBR efficacy are intimately connected. After several text analyses we can reach a considerable case base. Thus, it is possible to have a dynamic method of anaphora resolution that is continually improved as used.

6. Limitations

By now, we think TextStorm will improve considerably its performance after adding the following capabilities:

- Complete integration with WordNet [Miller et al, 93]. If this program is able to perceive hyponyms and hypernyms, then it will avoid some extra work from the user on completing the isa-tree, WordNet may provide this information.

- Clarification of the ambiguities associated to qualifiers, so that it can produce “property” relations, or even more specific sub-classes (e.g. “color”, “weight”, “speed”, etc.);
- Generation of n-dimension predicates according to syntactic structures of related sentences in the input text. For this, we would need an exhaustive study of verb types (transitive accompanied by direct or indirect object, intransitive, linking, etc.) that WordNet doesn’t offer.

7. Conclusions

This paper presented a work centered on the idea of learning and understanding from domain independent text. It consists on a two-module framework, where TextStorm has the task of extracting as many concept interrelations as possible. Its output is directly fed into the second module, Clouds, which will use this knowledge to build a concept map with the help of the user. In this paper, we focussed mainly on TextStorm, a module that uses parsing of sentences to build binary predicates that represent domain knowledge from the text in hand. Binary predicates are composed employing parsing with an augmented grammar. The parameters of this grammar are the key factors in a sentence that represent relations: verbs and adjectives. The first one tells us which correlation exists between two concepts; and the second one brings us the notion of property.

From the example, we can conclude that by now it is able to extract an interesting set of relations, although there are many improvements to make. Furthermore, this system is already being useful to the Dr. Divago project, to which Clouds belong.

After the development of the first implementation, we have already decided the next milestones to fulfill, namely the complete connection to the WordNet [Miller et al, 93] framework and a more precise *anaphora* resolution. After the accomplishment of these tasks, we think TextStorm will be able to extract a reasonably high amount of knowledge from a text file.

Furthermore, we think TextStorm is not limited to its current application as a Clouds knowledge server, but also to tasks like text summarization, translation intelligent search in the World Wide Web (WWW), knowledge elicitation and automatic ontology extraction.

References

[Allen, 95] Allen, J.; *Natural Language Understanding,* The Benjamin/Cummings Publishing Company, Inc.; pp. 429-465. 1995

[Ciravegna et al. 1999] F. Ciravegna, A. Lavelli, N. Mana , L. Gilardoni, S. Mazza, J. Matiasek, W. Black, F. Rinaldi, D. Mowatt*; Classifying Texts Integrating Pattern Matching and Information Extraction.* Sixteenth International Joint Conference on Artificial Intelligence (IJCAI99), Stockholm, August, 1999

[Hahn , Klenner & Schnattinger 96] Hahn, U.; Klenner, M. & Schnattinger, K.; *Automatic Concept Acquisition from Real-World Texts*. Working Notes of the AAAI-96 Spring Symposium on 'Machine Learning in Information Access', Stanford, USA. 1996

[Hahn & Strube 96] Hahn, U. and Strube, M.; *ParseTalk about functional anaphora*. Advances in Artificial Intelligence. 11th Biennal Conference of Canadian Society for Computational Studies of Intelligence (AI'96), 1996.

[Novak and Gowin, 84] J. D. Novak and D. B. Gowin. Learning How To Learn, New York: Cambridge University Press. 1984.

[Mill, 1843] Mill, J. S.; *A System for Logic, Ratiocinative and Inductive: Being a Connected View of the Principles of Evidence, and Methods of Scientific Investigation*. J. W. Parker, London. 1843.

[Miller et al, 93] Miller, G. A.; Beckwith, R.; Fellbaum, C.; Gross, D. and Miller, K.; *Introduction to WordNet: An On-line Lexical Database;* Available at http://www.cogsci.princeton.edu/~wn/w3wn.html (Revised August 1993)

[Muggleton, 92] S. Muggleton, S.; Inductive Logic programming. Academic Press. 1992.

[omitted reference] Pereira, F., Oliveira, A. and Cardoso, A.; *Extracting Concept Maps with Clouds*. Argentine Symposium of Artificial Intelligence (ASAI 2000), Buenos Aires, Argentina, 2000.

[Veale and Keane, 1993] Veale, T. & M. T. Keane (1993) *A Connectionist Model of Semantic* Memory *for Metaphor Interpretation*, presented at the 1993 Workshop on Neural Architectures and Distributed AI, October 19-20, the Center for Neural Engineering, S.C. California

[Warren, 99] Warren, D.; *Programming in Tabled Prolog*; Department of Computer Science. Stony Brook, U.S.A. 1999