Semantic Dependency Parsing using the Commerce_buy and Commerce_sell frames of FrameNet

Jorge Mallol
Department of Computer Science
Lund University
jmallol@gmail.com

Dan Thorin
Department of Computer Science
Lund University
dan.thorin@gmail.com

January 14, 2008

Abstract

This project focuses on two semantic frames of FrameNet: Commerce_buy and Commerce_sell. Each has a set of lexical units connected to them, where in this project attention is given only to the category verbs. The task of the project is to identify the frame elements, given a sentence in the context of either buying or selling goods. These frame elements include for instance the buyer, the seller, the traded goods, the means of payment and so on. One of our goals is to reach a good frame element identification score, but the main objective is to figure out which of the features are the most important in order to get an acceptable result, that is as close as possible to the state of the art.

1 Credits

The project was originally motivated by work made at the Microsoft Development Center in Copenhagen and at the University of Copenhagen, Denmark, where they run two research projects in a similar area of semantic dependency parsing (Andersen, Elsborg, Henglein, Grue Simonsen & Stefansen, 2007 and Ib Nielsen, Grue Simonsen and Friis Larsen, 2007). The topic for the project was proposed by Pierre Nugues, professor at the Department of Computer Science of Lund University, Sweden. Helpful directions and guidance during the course of the work was given by Richard Johansson, a PhD student at Lund University.

2 Introduction

2.1 Dependency parsing

The underlying mechanism used to resolve the different frame elements of a sentence is the concept of dependency parsing. In this concept the main verb, usually referred to as the target, is considered to be the pivotal element of the sentence, from which all the various relations within the sentence derive. The two grammatical functions closest related to the main verb are the subject and the object of the sentence. These in turn might have additional grammatical functions tied to them, which in one way or another further determines the adjacent details of the frame elements. Figure 1 gives a graphical example of the dependency connections between words in a sentence.

Figure 1. The semantic hierarchy of a simple sentence.

The dependency paradigm is nowadays often favored over the phrase-structure grammar, which uses a tree structure to parse the sentence. The reason for adopting dependency parsing instead of the more traditional, and still widely popular phrase-structure grammar, is the general superiority in terms of efficiency (Nugues, 2006), even though more recent work have shown that the paradigm difference isn't that huge after all (Johansson & Nugues, 2007). Dependency parsing is being used

in a diverse set of recent applications related to text parsing.

The area of dependency parsing has been refined and improved by shared competitions like CoNLL-X (CoNLL-X, 2006), where participants implemented multi-lingual algorithms for parsing. These algorithms were then trained on different corpora in a number of different natural languages and tested to see how well they corresponded to a gold standard of the current corpus. The aim, of course, was to obtain the highest possible correspondence for each language, where the best had a score of over 90% on one of the languages, Japanese.

2.2 FrameNet

FrameNet is a frame semantic project carried out by the International Computer Science Institute at Berkeley, California. It's an online lexical database for English based on the idea that each word, or lexical unit, in our vocabulary is tied to a concept of meaning, a semantic frame. The internal relations of the semantic frame need to be known to us, if we want to be able to understand the meaning of a certain word. Some words in FrameNet are tied to the same semantic frame which makes up a grid of intertwining meanings and relations, connections referred to as valences. Currently FrameNet holds over 10000 lexical units which are connected to a total of over 825 semantic frames. Each frame has a number of frame elements, for instance *buyer, goods, seller* and so on.

2.3 Machine learning techniques

The adopted method for the project was machine learning techniques, which is a large sub field of artificial intelligence. Machine learning techniques can be used in a large number of applications ranging from search engines, medical diagnosis, computerized object recognition and robot locomotion, to natural language processing in this case. The machine learning procedure was carried out using Weka, a java based freeware from the University of Waikato, New Zeeland.

2.4 Focus and goal

In this project we focused on the semantic frames Commerce_buy and Commerce_sell. They hold the lexical verb units buy and purchase, and retail, sell and vend, respectively. All entities of these verbs, regardless of tense in our input corpus, are being assimilated to its corresponding frame.

The project's core purpose is to automatically be able to detect the frame elements in the context of buy and sell. These frame elements could for instance be *Buyer*, *Seller*, *Goods*, *Money*, *Place*, etcetera. In order to make this detection possible we make use of various features so that the typical structure of the text can be tracked and trained. These features could be for instance the part of speech of current word, grammar function of current word or the part of speech of closest semantic dependency bond.

3 Procedure

Two files that contained the pre-annotated corpora, an XML-file and a malt-file, were used as input data. These files held a poll of sentences and some grammatical information for every word as well as tagged frame elements for each sentence. This information is crucial for the extraction of features such as part of speech, grammar function, voice and so on. See table 1 below for the full list of the features that were extracted.

Previous previous part of speech
Previous part of speech
Part of speech
Next part of speech
Next next part of speech
First child part of speech
First child grammar function
First child word
Grammar function
Type of verb (sell/buy)
Voice (active/passive)
Frame element

Table 1. The features extracted from the data sets.

We use, as mentioned, two input files which contain the exact same set of sentences. In the XML-file, the target verb and all the frame elements are tagged in an XML tree structure manner.

He bought the car from his sister for $1000.

- **Target:** bought
- **Buyer:** He
- **Goods:** the car
- **Seller:** his sister

- **Money:** $1000

The malt-file contains the part of speech, the grammar function and a link to the closest semantically related word for each word in the file.

A java program was implemented to load the same sentence of both files into a joint data structure to deal with. This data structure makes the handling easier, and the features from the two input files were extracted from it.

Having extracted the features from the data set, an arff-file was produced as an output and used in Weka as training and test set. The arff-file contains all the features for each target element in each sentence. In order to make the arff-file work properly with Weka, we had to add headers to it, which the java program didn't accommodate for. This was done using a little Perl script provided by our supervisor.

We later evaluated our output in Weka to figure out which of the features were the most useful.

4 Results

After a little feature tinkering in Weka, we managed to get a peak score of 71.7% correctly classified instances. The features we used in this particular case were first child part of speech, first child grammar function, grammar function, type of verb and voice. Weka provides a number of different classifiers and we chose to go with the J48 tree classifier. This in combination with a 534 fold (the entire feature set) cross validation configuration gave us the best results.

The rest of the features; POS, previous previous POS, previous POS, next POS, next next POS and first child word, proved to be of less use. When all the different features were used at the same time, we reached a score of 66.5%.

To evaluate the impact of each of the useful features, we tried them out individually by removing them one at a time, while always leaving all of the less useful features out. We came to the conclusion that grammar function of the current word was by far the best feature, because if we left it out the score was impaired by more than 31 percent. The impact of the other features is shown in table 2.

Feature	**Impact on total score**
Grammar function	+ 31.3%
Voice	+ 6.0%
First child POS	+ 5.4%
Type of sentence	+ 3.2%
First child gr. function	+ 0.7%

Table 2. Individual feature impact.

5 Discussion

5.1 Verb chain issue and solution to it

A note has to be made about the algorithm of our implementation. There are a lot of different kinds of sentences, all with different levels of complexity. Therefore, we currently cannot deal with all of the frame elements from the input file.

In the process of extracting features we consider each frame element of each sentence separately. We look for the main word in the target element, which sometimes consists of a whole string of words. It is only after we have found the main word that we can extract features from it. To find the main word inside a frame element string we look at the dependency graph from the malt input file. We go through all the words of the frame element string to find the word that is directly linked to the target of the sentence (sell, sold, bought etcetera). If there isn't a direct link between the target and the main word of the frame element, then we normally leave this frame element out.

However, in order to be able to extract a bigger set of features from the input files, we implemented one exception. When the target consists of a verb chain, then all the links in the malt-file are pointing to the first verb of that verb chain. This is usually an auxiliary verb (as in "*was* buying"), and is not considered to be the main verb. This meant at first that we ruled out all of these instances. The following example describes the modification we implemented to avoid this feature extraction loss:

He was buying a car.

- **Target verb chain:** *was buying*

- **Buyer:** *He*
- **Goods:** *a car*

In this case the buyer, *he*, is directly linked to *was* in the verb chain and the same goes for the goods of the sentence; *a car*.

Thanks to our feature enhancing implementation, we now go through the verb chain to see if one of the words of the frame element string is semantically linked to one of the verbs in the verb chain. In the frame element string *a car*, which is the goods of the sentence, the word *car* is directly linked to *was*. Thus, the frame element is accepted and the features are extracted from it. This modification enabled us to render about 20% more feature lines up to a total of 534.

On the other hand, there are cases when there is no word in the frame element string that is linked to any of the verbs in the verb chain. In these cases, we will still discard that particular frame element from the feature set. One future improvement could be to go one step further down the semantic dependency line, and consider the child nodes of the first child too.

5.2 Confusion matrix and future work

By analysing the confusion matrix (table 3) of the classification session, we found that almost a third of the errors made were due to a two way mix up between the *buyer* and the *goods*. Another notable error was the quite significant number of *seller* frame elements misclassified as *money*, which were responsible for a little more than ten percent of the total errors. We noticed that by adding the voice (active/passive tense) feature, we resolved quite a few of the *seller* elements being taken for *goods* during the classification. This is an expected turn of events since in the passive tense the *seller* element usually show up in the same position as the *goods* element does in the active tense of a sentence, thus being confused.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	<-classified as
96	24	0	0	3	0	0	0	0	2	0	1	0	0	a=Buyer
22	**189**	0	0	3	0	0	0	0	1	1	7	0	0	b=Goods
0	0	**4**	0	3	0	0	0	0	1	0	2	0	0	c=Manner
0	0	1	**0**	0	0	0	0	0	1	0	0	0	0	d=Means
2	5	1	0	**57**	0	0	0	0	3	0	2	0	0	e=Money
0	0	0	0	4	**2**	0	0	0	0	0	0	0	0	f=Place
0	0	0	0	0	1	**0**	0	0	0	0	0	0	0	g=Purpose
0	0	0	0	1	0	0	**0**	0	0	0	0	0	0	h=Purp.ofGoods
1	0	0	0	5	0	0	0	**0**	1	0	0	0	0	i=Rate
2	3	0	0	3	0	0	0	0	**6**	0	2	0	0	j=Recipient
0	1	0	0	1	0	0	0	0	0	**0**	0	0	0	k=Relay
7	9	2	0	16	1	0	0	0	2	0	**29**	0	0	l=Seller
0	0	0	0	2	0	0	0	0	0	0	0	**0**	0	m=Time
0	0	0	0	1	0	0	0	0	0	0	1	0	**0**	n=Unit

Table 3. The Weka confusion matrix.

So what could be the reason for mixing up the *buyer* and the *goods* for instance? Well, one possible reason could be due to errors made by the voice feature. A sentence that is written in passive voice is sometimes set as written in active voice, thus confusing the classifier to perform errors. In the XML-file used as input to the program, the following sentence is set as active by the voice feature, when in fact the sentence is of passive kind:

All ***<Goods>****tickets****</Goods>*** ***<Target>*** *purchased* ***</Target>*** ***<Seller>*** *at the Phoenix Cinema* ***</Seller>*** *are non-refundable and non-exchangable.*

The explanation for this is that the voice feature looks for a participle verb in the sentence (eg. purchased) and then looks for the closest auxiliary verb. In order to make the sentence passive, these could be any of be, been, is, are, was or were and no such verb is present in the given sentence. In the sentence

<Goods>*The drawing****</Goods>*** *was recently* ***<Target>*** *purchased* ***</Target>*** ***<Buyer>*** *by Colnaghi* ***</Buyer>*** ***<Money>*** *for $ 2,500* ***</Money>***

the auxiliary verb *was* is in fact present, but it isn't directly connected to the main verb, thus being ignored by the voice feature method.

It is small corrections like these that have to be done to the program in order to improve the total score. However, because of the rather small time frame given to the completion of the project, such modifications will have to be left out to possible further improvements in the future.

6 Conclusion

We have shown that the by far most useful feature is the grammar function, rendering a boost of more than 31% to the final score. The second best feature, the voice of the sentence, gave us a moderate improvement of 6% but it is quite possible that this feature could prove even more influential, given a little extra programming attention.

References

Jesper Andersen, Ebbe Elsborg, Fritz Henglein, Jakob Grue Simonsen and Christian Stefansen. 2007. *Compositional Specification of Commercial Contracts*, Department of Computer Science, University of Copenhagen, Denmark.

Morten Ib Nielsen, Jakob Grue Simonsen and Ken Friis Larsen. August 2007. *Tutorial on Modeling VAT rules using OWL-DL*, Department of Computer Science, University of Copenhagen, Denmark

Pierre M. Nugues. 2006. *An introduction to Language Processing with Perl and Prolog*, Springer-Verlag Berlin Heidelberg, Germany

Richard Johansson and Pierre Nugues. 2007. *Syntactic Representations Considered for Frame-semantic Analysis*. Department of Computer Science, Lunds Tekniska Högskola, Sweden

CoNLL-X. 2006. *URL: http://nextens.uvt.nl/~conll/*