Improving syntactical parsing using noun classes

Niklas Zechner
Department of Science
Lund University, Sweden
niklas.zechner@gmail.com

Abstract

This document gives a brief introduction to syntactic parsing, and describes a method for improving the accuracy of the parsing by using nouns classes. Several versions are tested, and some do lead to a significant improvement, whereas others decrease the accuracy. Some issues are examined which could cause this decrease.

1 Dependency parsing and syntactical parsing

This article deals with dependency parsing and syntactical parsing of natural written language. Dependency parsing means identifying how words relate to each other, for example that a particular noun is the argument to a particular verb, or an adjective to a noun. Syntactical parsing extends this to also include the nature of the relation, known as the function, for example whether the noun is the subject or object of the verb.

The basic parser algorithm we used was developed by Joakim Nivre. The idea is to transform the tree structure of dependency to a sequence of operations, and vice versa. The parser uses a stack and a queue, starting with the whole sentence in the queue and the stack empty. The four possible operations are:

- Shift, moves the first element in the queue to the stack
- Reduce, removes the top element from the stack
- Left arc, makes the top element of the stack a dependant of the first element in the queue, and removes the top element from the stack
- Right arc, makes the first element in the queue a dependant of the top element of the stack, and moves the first element in the queue to the stack

With these operations, any dependency tree can be described.

In the training step, the parser interprets the tree as a list of operations, and for each operation notes certain properties of the words in the stack and in the queue, known as features. If we are using functions, it also notes the function for each left arc or right arc operation. We let the parser do this with a large training corpus, and send the results to a machine learning program which builds a model. In the parsing step, the parser uses the model to predict the operations from the features, and thus construct the dependency tree.

2 Issues with the technique and a possible method for improvement

The technique of using statistics to determine syntactical roles is based on the assumption that the text in question has a relatively fixed word order. The extent to which this is true varies between languages and types of text. Isolating languages such as Chinese are very suitable for this type of analysis, whereas synthetic languages such as Latin are less suitable. (It is also worth noting that from the perspective of a simple statistical analysis, a more synthetic language will cause other difficulties. From equivalent texts, it will give a smaller corpus but a larger lexicon than a more isolating language. Any statistical analysis will be less effective, and particularly so if the analysis also relies on the actual word rather than the part of speech. For the most synthetic languages, analysing words as such would be meaningless, but that kind of languages are rare.) The languages on which this work is focused are Swedish and English. They could both be said to be somewhere in the middle on the scale, with English slightly more isolating, and both languages developing towards a higher degree of isolation. There is also in both these languages a clear difference between more and less formal text. Formal text is likely to have a higher degree of synthesis, and a more predictable word order (but, on the other hand, more complex sentence structures).

In general, grammar rules are flexible, including those of word order, in any language. Statistical analysis which only relies on word order can therefore never reach more than a certain accuracy; the probability of a word being correctly interpreted cannot be higher than the probability that the rule on which it is based applies. In English and Swedish, the general rule is that the subject comes before the verb it relates to, and the object

comes after; they are said to be SVO languages. But there are exceptions to this rule. From Swedish:

Vad äter musen?
Osten äter musen.

what eat-PRES mouse-DEF ?
cheese-DEF eat-PRES mouse-DEF .

This example would not be possible in English, but there are other examples where there are ambiguities. Consider the difference between the following:

the mouse eating cheese

the cheese-eating mouse

Some verbs are particularly prone to confusion:

"Hello", said the man.

Having an annotated corpus, we can easily check just how accurate the SVO rule is for each language. We simply count which fraction of subjects occur before their respective verbs, and similarly for objects. This method isn't quite "fair"; the rule is not supposed to act on that level, but it can hopefully give some indication of how firm the word order is.

For Swedish, we find that 78% of the subjects and 97% of the objects are in the expected place. For English, it is 95% and 98%.

There are also other cases where the parser goes wrong if it only looks at the parts of speech. A typical example is

They had tea in the kitchen.

Statistically, 'in the' is often followed by a time, such as 'afternoon'. Since 'kitchen' and 'afternoon' are both nouns, there is no way of telling which one is correct here, so the parser naturally assumes that it is a time. A similar example is

They ate a while.

Here, the parser only knows that 'while' is a noun, and therefore interprets it as the object, the thing being eaten. All these things could be solved if we give the parser the information of the actual word, but depending on the nature of the parser and the corpus this might not be practical. Some classifiers (such as the one used here) have difficulties dealing with large numbers of possible values. Also, unless the training corpus is extremely large, many words will occur so few times that they might be difficult to interpret.

One way to try to improve the parsing is by using noun classes. We divide the nouns into different groups on a semantic basis, effectively treating them as different parts of speech. There are several sets of noun classes that we could use. One possibility is to use animate versus inanimate nouns; 'animate' meaning words for things which can be the instigator of an action - essentially living beings. This should help identify the subject and object in the above examples, as we know that the subject of 'eat' must be animate, and so on. Another distinction is between concrete nouns (that is, physical objects) and abstract nouns. This could possibly solve the example with 'while', since we know that the object of 'ate' can only be a concrete noun. A problem here is that we don't have any information about the verb. Perhaps more effective then is to have a specific class for times; that would solve both the example with 'while' (by telling the parser that it is dealing with a time) and the example with 'in the kitchen' (by telling the parser that it is not dealing with a time).

3 Method

We used a Java implementation of Nivre's parser written by Pierre Nugues, which was modified to include slightly different features, and also functions. For machine learning we used Weka, and the classifier J48. The corpuses we used were the Talbanken corpus for Swedish and the Penn Treebank corpus used in CONLL 2008 for English.

The parser originally looks at the two top words in the stack and in the queue. The result is reasonable, but not impressive. We try to add a third word, in the stack and in the queue, but there is little difference in the result. There are various other features we could add to the parser to improve the accuracy, but those are not the focus of this article.

Instead, the parser was rewritten to look at the specific functions for each relation, such as whether a noun is the subject of its head verb or the object, making it a syntactical parser rather than just a dependency parser.

We attempt to improve the parsing by involving classes. For the Swedish corpus, we settle on the following classes:

- things (concrete inanimate)
- animate
- abstract
- locations

A program goes through the words, and finds all nouns which occur at least ten times in the union of the training corpus and the test corpus. Each of these words is manually assigned a class. The corpuses are then updated, replacing the noun symbol with a symbol for the specific class of noun, effectively treating nouns as five different parts of speech (four for the classes, and one for the remaining words which have not been classified).

Next we try using the English corpus, first without classes. Then we add classes, this time for the words which occur at least 100 times (since it's a much bigger corpus). The results are encouraging, so we try again with all words which occur at least 25 times. We have now also added a fifth class, for times.

Since we have no information on the verbs, but are using the classes solely to predict which function each word is likely to have, an idea springs to mind: What if we simply use one class for each function, assigning the word to the class corresponding to the function which it most commonly has in the training set? This can easily be done automatically, so we can now assign a class to every noun, not just to the most common ones.

4 Results

	Total %	Nouns %	Others %
Swe, two words	81.11	86.04	79.71
Swe, three words	81.11	86.04	79.71
Swe, with functions	67.91	78.75	64.83
Swe, with classes	66.99	77.13	64.11
Eng, no classes	73.12	73.57	72.91
Eng, classes to 100	73.43	74.67	72.83
Eng, classes to 25	73.54	75.03	72.84
Eng, function-classes	72.56	73.98	71.88

5 Analysis

The original parser has a reasonable accuracy, but is far from what the best parsers can do. When we try adding a third word to the stack and queue, this changes the interpretation of about one percent of the words - some for the better and some for the worse - but oddly enough the net result is less than 0.01%. We see that the nouns are already considerably easier to assess than the rest of the words, which is certainly of interest for our purposes.

(It should be noted that the program used here to count the percentages is not the standard CONLL evaluation program, since a program was required to investigate the nouns specifically, and there is a minute difference in the results due to issues of punctuation, but this is not significant to the conclusions. The difference for the first part is 0.03%.)

With functions, the parser predicts much more information, so we naturally expect a much lower percentage. The result of 68% is not surprising.

Rather disappointingly we find that adding classes gives a lower accuracy. How can this be? We are giving more information to the parser, so we would expect the result to be better, or at least not worse than before. The answer must be that the parser has lost a vital piece of information: the fact that these classes have something in common, that they are all nouns. Not being able to draw information from the nouns outside the given class gives the same effects as having a smaller corpus. Presumably there is also a positive effect, but it is obscured by this negative effect. If the corpus was larger, we would probably see a different result: As the corpus grows larger, the accuracy continues to improve, but the improvement gets smaller, so at some point the consequence of effectively cutting the corpus in four parts (as far as the nouns are concerned) becomes smaller than the positive effect of adding classes. Another way to get around this problem is by doing the process in several steps. If we first parse the sentences without classes, and then post-process with classes, we should be able to make use of both pieces of information. It might also be possible to simply consider class a separate feature, rather than mixing it with part of speech. Currently this parser is not equipped for those things, so that will have to be left for future work.

We should also consider the distribution of classes. If one class is much smaller than the others, that class would be particularly difficult to parse. Looking at the classified nouns for this corpus, we find

4.7% things
20.4% animate
70.6% abstract
4.4% locations

The percentage of locations and animate nouns could probably be considered rather normal, since most texts are partly but not entirely about people, and locations are generally a less common category. What is noteworthy is the fact that the abstract words greatly outnumber the things. It is no surprise, since the text is a rather convoluted and bureaucratic one. One might question the annotators' choice of text here; although it is desirable to have a realistic text and not a constructed example text, there should be texts available which are natural but somewhat less complicated. It is after all common scientific practice to start with simpler examples and make

sure to master them before moving on to more complex ones.

However, there is no other available corpus in Swedish which serves our purpose, so we look at the English corpus. It has a higher overall percentage, but that is not something we should be overly concerned with. Part of the reason is of course that the corpus is bigger. That does not account for the whole difference; when using only a part of it, with the same size as the Swedish corpus, the result is still about two percentage points higher than for the Swedish. This corpus also comes from a different source, and of course there is some effect of changing languages - one is tempted to say that the above comparison of word order rules shows that English is easier to parse in this way than Swedish. Most of the effect is probably due to various language differences, and the different set of parts of speech. Either way, the results are not comparable for our purposes, but should be considered separately.

What might be of some interest is the fact that the nouns are far worse. This could be due to some unforeseen property of the language, but there is another possibility. The English corpus, unlike the Swedish, uses different tags for singular and plural nouns. It seems unlikely that this would have any significant positive effect, since the same noun can usually have the same functions regardless of its number - if one mouse can eat cheese, two mice can also eat cheese. But as we noticed before, arbitrary classes can have a negative effect on the result. The English corpus also contains quite a few proper nouns, which makes four classes for nouns.

A brief attempt at removing the number distinctions yields a significantly worse result. It might be that removing them and replacing with the classes would improve something. Perhaps there is in some sense an ideal number of classes - a few divisions is good, but too many ruins the classification. This needs to be investigated further.

In the next step we find perhaps the most enlightening results. First, we notice that the overall result is better with classes than without. Second, we see that it is only the nouns which have improved; the rest of the words actually have a slightly worse result than before. It seems that the classes have had the desired effect, making the nouns' functions easier to predict, whereas the negative effect can be seen in all the words. The function of the nouns themselves is connected to the classes, but the function of other words n relation to the nouns has little or no such connection. Therefore the prediction of functions for words of other parts of speech is not improved.

The improvement when classifying more words is expected. The slight change in the result for the non-nouns is too small to draw any conclusions from.

The distribution of the classes is now

7.9% things
27.7% animate
53.5% abstract
7.9% locations
3.0% times

It is possible that the smaller fraction of abstract nouns makes this corpus easier to interpret, and hopefully the inclusion of time words as a specific class also improved the results.

Finally, we look at the automatically generated classes. The result is somewhat ambiguous. For the non-nouns, we can see that the negative effect is bigger than before; there are more classes, making each class less common. For the nouns, the result is better than when we had no classes, but not as good as for the semantic classes we used before. The reason could be that while the positive effect is bigger than before, the negative effect has increased more. If we had an even bigger training corpus - quite a lot bigger - the negative effect might be negligible, and then we might see an improvement with this sort of classes.

6 Future work

There are other classifiers that might be better for large data sets.

There may be other noun classes that would be more effective.

Adding other features would no doubt improve the parsing. While this is nothing new, it could be interesting to see how the addition of classes affects a really good parser.

When an even bigger corpus becomes available, the function-classes might be worth testing. On the other hand, with a classifier which can handle the word itself as a feature, this type of class might be completely obsolete.

The process could be divided into steps, so that the classes are applied separately. This could work better with a different classifier, one that can output several answers and their probabilities. Other simple improvements would be post-processing to check that grammar rules are being followed; a sentence should not have more than one root, a verb should not have more than one subject, etc.

The use of noun classes would probably work very well together with certain types of valency lexicon. Lexicons such as FrameNet have complex systems of semantic categories, and would require a similarly complex system of classes to reach its full potential. For Swedish, The Lexin valency lexicon classifies verb arguments as “persons” and “things”; in other words, animate and inanimate. It should therefore work well together with this type of noun classes. If one prefers to base things on the training corpus alone (which would make it language-independent), it would certainly be possible to automatically classify the verbs according to their most common arguments.

7 References

Jan Einarsson. 1976. *Talbankens skriftspråkskonkordans*. Lund University, Department of Scandinavian Languages.

Mitchell P Marcus, Mary Ann Marcinkiewicz and Beatrice Santorini. 1993. *Building A Large Annotated Corpus Of English: The Penn Treebank*. University of Pennsylvania, Philadelphia.

Jens Nilsson, Johan Hall and Joakim Nivre. 2005. *MAMBA Meets TIGER: Reconstructing a Swedish Treebank from Antiquity*. Proceedings of the NODALIDA Special Session on Treebanks.

Jens Nilsson and Johan Hall. 2005. *Reconstruction of the Swedish Treebank Talbanken*. MSI report 05067. Växjö University, School of Mathematics and Systems Engineering.

Joakim Nivre. 2003. *An Efficient Algorithm for Projective Dependency Parsing*. Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03), 149-160. Nancy.

Joakim Nivre, Jens Nilsson and Johan Hall. 2006. *Talbanken05: A Swedish Treebank with Phrase Structure and Dependency Annotation*. Proceedings of the fifth international conference on Language Resources and Evaluation (LREC2006). Genoa.

Pierre Nugues. 2008. Nivre parser implemented in java. EDA171 course web. http://fileadmin.cs.lth.se/cs/Education/EDA171/Programs/parsing/Nivre.zip.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez and Joakim Nivre. 2008. *The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies*. Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008).