Extending Dependency Treebanks with Good Sentences

Alexander Volokh
alexander.volokh@dfki.de
DFKI. Stuhlsatzenhausweg 3. 66123 Saarbrücken, Germany

Günter Neumann
neumann@dfki.de
DFKI. Stuhlsatzenhausweg 3. 66123 Saarbrücken, Germany

Abstract

For many resource-poor languages additional annotated data would be beneficial. However, annotation process is tedious and expensive. We propose a metric for selecting the most promising sentences for annotation. Annotating only good sentences saves time and would allow better results to be achieved even with a smaller amount of annotated data. We demonstrate how our method works on the example of parsing Finnish dependency treebank with MaltParser.

1 Introduction

Creation of annotated resources is a tedious and expensive work. Therefore for most languages resources like treebanks are rather small, despite the fact that learning curve experiments show that additional data would be beneficial and further improve performance of NLP applications.

In this paper we investigate whether it is possible to measure the usefulness of annotating and adding a sentence to an existing treebank and whether this value significantly differs across different sentences. We ascertain that it is indeed possible and show that a lot of time can be saved and much better results can be achieved, if one a) does not waste time on annotating *bad* sentences and b) concentrates on annotating *good* sentences. Finally, we construct a statistical model able to discriminate between those, evaluate it and discuss the results.

For our experiments we have used the Finnish treebank (Haverinen et al., 2010) and the dependency parser MaltParser (Nivre et al., 2007). Since we do not have any knowledge of the Finnish language and are not able to annotate sentences ourselves, we leave a portion of annotated sentences out of the treebank and add them in the process as if they were newly annotated by us. The reason for choosing Finnish is that it is a resource-poor language and its treebank is currently being annotated, so the work might be tested in a real-world application scenario, whereas the treebanks for languages that we speak, e.g. English or German, would lack this interesting property, because they are big and their development is completed.

2 Related Work

A similar work for dependency parsing exists in the area of active learning (Mirroshandel and Nasr, 2011). The work has exactly the same motivation of selecting only that material for annotation, which is beneficial for the performance of one's system and not wasting the time on the rest. They even propose not to annotate whole sentences manually, but rather process a sentence with a parser and only manually overwrite those dependencies which are actually useful.

The method of selecting good material proposed by Mirroshandel and Nasr is different from what we are proposing in this paper. Their idea is that those structures which are most error-prune require annotation, since obviously the model is missing the knowledge to process them. They then manually annotate the wrong parts, relearn the model and hope that the parser has learned to deal with the structures which went wrong before. Thus the most useful sentence is the one which contained the most mistakes prior to its manual annotation. On the other hand sentences which are already parsed correctly are not useful and do not require annotation.

Our proposal is that a sentence is useful and requires annotation if its inclusion to the training data improves the performance of the parser on the test data. This way the most useful sentence is the one whose annotation has the biggest positive effect on the performance of the parser on the test data.

There are also other works with the same motivation in the area of active learning for other domains, including other parsing areas, e.g. HPSG parsing (Baldridge and Osborne, 2008).

3 Sentence Usefulness

We define usefulness (su) of a sentence (s_i) as the influence of a sentence on the accuracy (acc) of a system for the test data (TE_m) when added to the training data (TR_n) of that system:

$$su(s_i) = acc(TR_n + s_i; TE_m) - acc(TR_n; TE_m)$$

For accuracy we use the metric called labeled attachment score (LAS), which represents the percentage of tokens for which both the parent and the dependency relation type were predicted correctly by the parser.

In order to compute a good estimate for sentence usefulness we use several training data sets and several test data sets and average the change in accuracies across all experiments.

For experiments described in this paper we have taken the Finnish treebank (6375 sentences) and split it into 2 training data sets (30% of the whole data each), 2 test data sets (10% each) and left the remaining 20% (1276 sentences) for experimenting. Thus for each of these 1276 sentences we have computed the accuracy for all (4) combinations and averaged over their number:

$su(s_i) = (acc(TR_1+s_i;TE_1) - acc(TR_1;TE_1) + acc(TR_1+s_i;TE_2) - acc(TR_1;TE_2) + acc(TR_2+s_i;TE_1) - acc(TR_n;TE_1) + acc(TR_2+s_i;TE_2) - acc(TR_2;TE_2)) / 4$

In order to test whether our sentence usefulness measure serves its purpose, namely as a selection criterion for sentences which should be added to the training data foremost, we have performed several experiments. We have computed accuracies for randomly selected *n* sentences and compared it to the accuracies when the best *n* sentences where selected according to our metric. E.g. when 20 sentences were randomly added to the training data the accuracy improved by 0.08% LAS on average, whereas with 20 best sentences the improvement was 0.3% LAS. For 50 random sentences the improvement was 0.13% LAS, whereas for the best 50 it was 0.48% LAS.

4 Usefulness Classes

Having shown that different sentences have different impact on the accuracy and that it is beneficial to add good ones first, we have investigated whether it is possible to automatically predict this value using a statistical classifier.

Basically, we are not interested in the exact sentence usefulness decimal value, but we rather want to know whether a sentence is worth annotating or not. Therefore we can group sentences into some discrete usefulness classes. The core problem is binary, since we want to discriminate between good sentences we want to annotate and the rest. However, the number of good sentences is small and the rest is big, which makes the problem very imbalanced and the prediction difficult. Therefore we have experimented with several different divisions and could achieve the best results when the data is split into four classes of equal number: 1 being the worst, 2 being slightly better, 3 – good, 4 – best. Even though that we are still interested only in the class 4, this partitioning allows us to better optimise the performance of the classifier. In cases when the decision is not very certain, usually the confusion is only between two classes (Keerthi et al., 2008). E.g. when two good classes (3 and 4) compete, a mistake is not that bad as e.g. in the case of a good and a bad class (1 and 4), where the risk of making a severe mistake is much higher. In the simple binary case one would not be able to differentiate between different types of mistakes that well.

5 Model

First, we have tried to represent each sentence as a feature vector and then learn the corresponding classes using linear SVMs (Lin et al., 2008; Keerthi et al., 2008). However, either because we were not able to come up with enough good sentence-level features or because the size of the training data was not big enough (we have used 20% of the data left for experimenting, which then again was randomly split into 90% for training and 10% for testing), the results were not satisfactory.

We have then decided to do the classification on the word level: for every word in a sentence of class *c*, we have constructed a feature vector and assigned it the class *c* in the training phase. E.g. for a class 3 sentence of length 10 we have constructed

10 feature vectors, one for each word, of the class 3. The feature templates which we used were:

sentence-level features:

length of the sentence, number of punctuation tokens, number of verbs, number of pronouns, number of conjunctions, number of adverbial modifiers

word-level features:

word form of the current word, word form of the next word, POS of the current word, POS of the next word, word length, dependency type, word novelty

These particular features were selected after some semi-automatic feature engineering, during which we have tried all kinds of different POS tags and dependency types as features.

In order to detect punctuation, verbs and pronouns we have simply used the Finnish POS tags: PUNCT, V and PRON respectively. For adverbial modifiers and dependency types in general we have parsed every sentence with MaltParser and used the label predicted by the parser (not the gold standard that we also had). The novelty of a word is a binary feature: for the given word form we look whether it already ever occurred in the training and test data or whether it is new.

In the test phase we would then predict a class for every word in a sentence and perform a voting procedure in order to infer the sentence usefulness class for the whole sentence. Given a set of votes $V = \{c_1, c_2, c_3, \ldots, c_n\}$, where $c_n \in \{1,2,3,4\}$, the voting procedure looks like that:

if $4 \in V$
  weight(1) = count(1)*2 in V
  weight(2) = count(2) in V
  weight(3) = count(3) in V
  weight(4) = count(4) in V
  return argmax_i(weight(i))
else
  return majority_voting(V)

So basically if V does not contain words classified as class 4, we simply perform majority voting for the remaining classes. Otherwise all votes for other classes are counted, whereas the votes for the worst class are always counted twice in order to avoid a severe mistake whenever there is a chance that a sentence might belong to class 1.

6 Evaluation

The evaluation of our model was not straightforward. The first metric which we have tried out was accuracy: the percentage of correctly classified classes. However, since we are not equally interested in all classes, but are rather interested in correctly finding the class 4, the metric was not very helpful.

That is why the second attempt was to compute precision and recall only for the class 4. However, as we have already implied in section 4, different mistakes have different impact on the result. A confusion between class 1 and class 4 is different from a confusion between classes 3 and 4. Precision and recall metrics are thus not helpful neither.

The next step was to compute a confusion matrix for all classes and try to minimise the number of actual instances of 1 and 2 predicted as 4 on the one hand and maximising the number of actual 4 predicted as 4 on the other hand.

In a real-world scenario the number of potential sentences which can be annotated is infinite and therefore the recall of the method might be arbitrary low and it will still find enough good candidates to annotate. It is merely important that the precision is high, such that the sentences which are annotated are really beneficial for the task. However, in our experiment the amount of data was small: in 20% of the sentences left for testing, i.e. 250 sentences, 57 belonged to class 4. Therefore we could not decrease recall because otherwise not enough sentences would have been found in order to clearly demonstrate that our approach outperforms the baseline of selecting sentences randomly.

We have run 10 tests randomly selecting 80% sentences for training and 20% for testing and averaged the performance on the confusion matrix:

In ~16% of all cases actual 1s were classified as 4s. In ~26% of all cases actual 2s were classified as 4s, in ~8% of all cases actual 3s were classified as 4s and in ~50% of all cases actual 4s were correctly classified as 4s.

Thus this method allows to select good sentences significantly more often (namely twice as often) than when doing it randomly. Again, if we had more data available for testing, we could tune the precision at costs of recall, improving the performance even further. Our experiment described in the section 3 with gold-standard usefulness

classes, suggests that it is possible to gain accuracy up to four times faster when annotating sentences proposed by our method compared to random selection.

7 Discussion

We have presented an approach which allows to predict sentence usefulness and extend a treebank only with the most promising sentences for the given task. However, we suppose that the sentence usefulness is not a constant parameter, but rather changes with the growing size of the training data. Because of the limited size of data for experimenting we could not investigate whether our metric is still better than random when hundreds or thousands of sentences are added to the original corpus and/or whether the model has to be adapted and retrained on the new data, because in the course of annotation different units might become useful than it was the case prior to the treebank extension. Most probably the latter case is true and the model has to be regularly adopted to the changes in the treebank, however, we do not know whether it will be an easier or a harder task to predict new good candidates for annotation. On the one hand with the growing size it should become more difficult to find new beneficial sentences for the treebank, but on the other hand with more data available a more accurate model for prediction can be constructed.

Another interesting point is to analyse what actually makes some sentences better than other. It was important to make sure that longer sentences are not always automatically better, simply because they have more tokens and thus provide more information. So in one of experiments we have sorted all sentences according to their usefulness and looked at their length. The result was that even though the best sentences are on average slightly longer, there is still a huge amount of long sentences which were not useful and there are a lot of short sentences that were useful. In principle the approach might be further optimised by not only finding most promising but also short sentences, which do not require much time to annotate.

Unfortunately, we are not speakers of Finnish and we were not able to analyse neither the content of the most useful sentences nor their properties. It would be interesting to find out whether these sentences have certain linguistic structures or contain specific lexical entries. We will certainly try to cooperate with someone to perform this analysis in the future.

Neither could we investigate whether the useful sentences we are able to detect overlap with those predicted by the approach proposed by Mirroshandel and Nasr. The approaches are very different: they work in a bottom-up fashion by optimising the model for the sentences with poorest performance until the model performs well for most sentences and we work in a top-down fashion by optimising the model for sentences which improve the performance for as many other sentences as possible in the test data. Intuitively, the first and the latter sentence types are not the same and thus it would also be interesting to combine both approaches.

8 Conclusions

In this paper we have shown the necessity of extending annotated resources on the example of Finnish. Since the annotation process is expensive and tedious we have proposed an approach for selecting only the most promising sentences for annotation. We explain our sentence usefulness metric and how it can be predicted for new sentences. Our experiments show that it is possible to select good sentences significantly better than annotating random sentences. If gold standard values for sentence significance are used, the accuracy of the parser increases four times faster than when random sentences are added to the training data. With the values predicted by our model it is still twice as fast. We have also discussed some potential problems with the metric if the size of the initial treebank changes considerably. However, on the other hand it might become easier to accurately predict good sentences if more data becomes available.

Acknowledgements

The work presented here was partially supported by a research grant from the German Federal Ministry of Education and Research (BMBF) to the DFKI project Deependance (FKZ. 01IW11003) and partially funded by the EU project Excitement (FP7-IST). Additionally, we would like to thank Turku BioNLP group, in particular Filip Ginter and Katri Haverinen for their fantastic support. They have provided us with various versions of the Finnish treebank, replied numerous emails and even responded to some feature requests.

References

J. Baldridge and M. Osborne. 2008. *Active learning and logarithmic opinion pools for HPSG parse selection*. Natural Language Engineering Journal, 2, pp. 191-222.

Haverinen, K.; Viljanen, T.; Laippala, V.; Kohonen, S.; Ginter, F. & Salakoski, T. 2010. *Treebanking Finnish.* Proceedings of The Ninth International Workshop on Treebanks and Linguistic Theories (TLT9), pp. 79-90. 2010.

S. Sathiya Keerthi, S. Sundararajan, K.-W. Chang, C.-J. Hsieh, and C.-J. Lin. 2008. *A sequential dual coordinate method for large-scale multi-class linear SVMs*. KDD 2008.

C.-J. Lin, R.-E. Fan, K.-W. Chang, C.-J. Hsieh and X.-R. Wang. *LIBLINEAR: A library for large linear classification*. Journal of Machine Learning Research 9(2008), pp. 1871-1874.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gulsen Eryigit, Sandra Kubler, Svetoslav Marinov and Erwin Marsi. 2007. *MaltParser: A Language-Independent System for Data-Driven Dependency Parsing,* Natural Language Engineering Journal, 13, pp. 99-135.

Seyed Abolghasem Mirroshandel and Alexis Nasr. 2011. Active Learning for Dependency Parsing Using Partially Annotated Sentences. In Proceedings of the 12th International Conference on Parsing Technologies, pages 140–149, October 5-7, 2011, Dublin City University.