A Puristic Approach for Joint Dependency Parsing and Semantic Role Labeling

Alexander Volokh
LT-lab, DFKI
66123 Saarbrücken, Germany
Alexander.Volokh@dfki.de

Günter Neumann
LT-lab, DFKI
66123 Saarbrücken, Germany
neumann@dfki.de

Abstract

We present a puristic approach for combining dependency parsing and semantic role labeling. In a first step, a data-driven strict incremental deterministic parser is used to compute a single syntactic dependency structure using a MEM trained on the syntactic part of the CoNLL 2008 training corpus. In a second step, a cascade of MEMs is used to identify predicates, and, for each found predicate, to identify its arguments and their types. All the MEMs used here are trained only with labeled data from the CoNLL 2008 corpus. We participated in the closed challenge, and obtained a labeled macro F1 for WSJ+Brown of 19.93 (20.13 on WSJ only, 18.14 on Brown). For the syntactic dependencies we got similar bad results (WSJ+Brown=16.25, WSJ= 16.22, Brown=16.47), as well as for the semantic dependencies (WSJ+Brown=22.36, WSJ=22.86, Brown=17.94). The current results of the experiments suggest that our risky puristic approach of following a strict incremental parsing approach together with the closed data-driven perspective of a joined syntactic and semantic labeling was actually too optimistic and eventually too puristic.

The CoNLL 2008 shared task on joint parsing of syntactic and semantic dependencies (cf. Surdeanu, 2008) offered to us an opportunity to initiate, implement and test new ideas on large-scale data-driven incremental dependency parsing. The topic and papers of the ACL-2004 workshop "Incremental Parsing: Bringing Engineering and Cognition Together" (accessible at http://aclweb.org/anthology-new/W/W04/#0300) present a good recent overview into the field of incremental processing from both an engineering and cognitive point of view.

Our particular interest is the exploration and development of strict incremental deterministic strategies as a means for fast data-driven dependency parsing of large-scale online natural language processing. By strict incremental processing we mean, that the parser receives a stream of words w_1 to w_n word by word in left to right order, and that the parser only has information about the current word w_i, and the previous words w_1 to w_{i-1}.[1] By deterministic processing we mean that the parser has to decide immediately and uniquely whether and how to integrate the newly observed word w_i with the already constructed (partial) dependency structure without the possibility of revising its decision at later stages. The strategy is data-driven in the sense that the parsing decisions are made on basis of a statistical language model, which is trained on the syntactic part of the CoNLL 2008 training corpus. The whole parsing strategy is based on Nivre (2007), but modifies it in several ways, see sec. 2 for details.

Note that there are other approaches of incremental deterministic dependency parsing that assume that the complete input string of a sentence is already given before parsing starts and that this additional right contextual information is also used as a feature source for language modeling, e.g., Nivre (2007).

In light of the CoNLL 2008 shared task, this actually means that, e.g., part-of-speech tagging and lemmatization has already been performed

[1] Note that in a truly strict incremental processing regime the input to the NLP system is actually a stream of signals where even the sentence segmentation is not known in advance. Since in our current system, the parser receives a sentence as given input, we are less strict as we could be.

CoNLL 2008: Proceedings of the 12th Conference on Computational Natural Language Learning, pages 213–217
Manchester, August 2008

for the complete sentence before incremental parsing starts, so that this richer source of information is available for defining the feature space. Since, important word-based information especially for a dependency analysis is already known for the whole sentence before parsing starts, and actually heavily used during parsing, one might wonder, what the benefit of such a weak incremental parsing approach is compared to a non-incremental approach. Since, we thought that such an incremental processing perspective is a bit too wide (especially when considering the rich input of the CoNLL 2008 shared task), we wanted to explore a strict incremental strategy.

Semantic role labeling is considered as a post-process that is applied on the output of the syntactic parser. Following Hacioglu (2004), we consider the labeling of semantic roles as a classification problem of dependency relations into one of several semantic roles. However, instead of post-processing a dependency tree firstly into a sequence of relations, as done by Hacioglu (2004), we apply a cascade of statistical models on the unmodified dependency tree in order to identify predicates, and, for each found predicate, to identify its arguments and their types. All the language models used here are trained only with labeled data from the CoNLL 2008 corpus; cf. sec. 3 for more details.

Both, the syntactic parser and the semantic classifier are language independent in the sense that only information contained in the given training corpus is used (e.g., PoS tags, dependency labels, information about direction etc.), but no language specific features, e.g., no PropBank frames nor any other external language and knowledge specific sources.

The complete system has been designed and implemented from scratch after the announcement of the CoNLL 2008 shared task. The main goal of our participation was therefore actually on being able to create some initial software implementation and baseline experimentations as a starting point for further research in the area of data-driven incremental deterministic parsing.

In the rest of this brief report, we will describe some more details of the syntactic and semantic component in the next two sections, followed by a description and discussion of the achieved results.

1 Syntactic Parsing

Our syntactic dependency parser is a variant of the incremental non-projective dependency parser described in Nivre (2007). Nivres' parser is incremental in the sense, that although the complete list of words of a sentence is known, construction of the dependency tree is performed strictly from left to right. It uses Treebank-induced classifiers to deterministically predict the actions of the parser. The classifiers are trained using support vector machines (SVM). A further interesting property of the parser is its capability to derive (a subset of) non-projective structures directly. The core idea here is to exploit a function *permissible(i, j, d)* that returns true if and only if the dependency links $i \rightarrow j$ and $j \rightarrow i$ have a degree less than or equal to d given the dependency graph built so far. A degree $d=0$ gives strictly projective parsing, while setting $d=\infty$ gives unrestricted non-projective parsing; cf. Nivre (2007) for more details. The goal of this function is to restrict the call of a function *link(i, j)* which is a nondeterministic operation that adds the arc $i \rightarrow j$, the arc $j \rightarrow i$, or does nothing at all. Thus the smaller the value of d is the fewer links can be drawn.

The function *link(i, j)* is directed by a trained SVM classifier that takes as input the feature representation of the dependency tree built so far and the (complete) input $x = w_1, \ldots, w_n$ and outputs a decision for choosing exactly one of the three possible operations.

We have modified Nivres algorithm as follows:

1. Instead of using classifiers learned by SVM, we are using classifiers based on Maximum Entropy Models (MEMs), cf. (Manning and Schütze, 1999).[2]
2. Instead of using the complete input x, we only use the prefix from w_1 up to the current word w_i. In this way, we are able to model a stricter incremental processing regime.
3. We are using a subset of feature set described in Nivre (2007).[3] In particular, we had to discard all features from Nivre's set that refer to a word right to the current word in order to retain our

[2] We are using the opennlp.maxent package available via http://maxent.sourceforge.net/.

[3] We mean here all features that are explicitly described in Nivre (2007). He also mentions the use of some additional language specific features, but they are not further described, and, hence not known to us.

strict incremental behavior. Additionally, we added the following features:

a. Has j more children in the current dependency graph compared with the average number of children of element of same POS.
b. Analogously for node *i*
c. Distance between *i* and *j*

Although some results – for example Wang et al. (2006) – suggest that SVMs are actually more suitable for deterministic parsing strategies than MEMs, we used MEMs instead of SVM basically for practical reasons: 1) we already had hands-on experience with MEMs, 2) training time was much faster than SVM, and 3) the theoretical basis of MEMs should give us enough flexibility for testing with different sets of features.

Initial experiments applied on the same corpora as used by Nivre (2007), soon showed that our initial prototype is certainly not competitive in its current form. For example, our best result on the TIGER Treebank of German (Brants et al., 2002) is 53.6% (labeled accuracy), where Nivre reports 85.90%; cf. Volokh (2008) and sec. 4 for more details

Anyway, we decided to use it as a basis for the CoNLL 2008 shared task and to combine it with a component for semantic role labeling at least to get some indication of "what went wrong".

2 Semantic Role Labeling

On the one hand, it is clear that we should expect that our current version of the strict incremental deterministic parsing regime still returns too erroneous dependency analysis. On the other hand, we decided to apply semantic role labeling on the parser's output. Hence, the focus was set towards a robust strictly data-driven approach.

Semantic role labeling is modeled as a sequence of classifiers that follow the structure of predicates, i.e., firstly candidate predicates are identified and then the arguments are looked up.

Predicate and argument identification both proceed in two steps: first determine whether a word can be a predicate or argument (or not), and then, each found predicate (argument) is typed. More precisely, semantic role labeling receives the output of the syntactic parser and performs the following steps in that order:

1. Classify each word as being a predicate or not using a MEM-based classifier.
2. Assign to each predicate its reading. Currently, this is done on basis of the frequency readings as determined from the corpus (for unknown words, we simply assign the reading .01 to the lemma if the whole word was classified as a predicate).
3. For each predicate identified in a sentence, classify each word as argument for this predicate or not using a MEM-based classifier.
4. For each argument identified for each predicate, assign its semantic role using a MEM-based classifier.

For step 1 the following features are used for word w_i: 1) word form, 2) word lemma, 3) POS, 4) dependency type, 5) number of dependent elements in subtree of w_i, 6) POS of parent, 7) dependency type of parent, 8) children or parent of word belong to prepositions, and 9) parent is predicate.

For step 3 the same features are used as in step 1, but 5) (for arguments the number of children is not important) and two additional features are used: 10) left/right of predicate (arguments are often to the right of its predicate), and 11) distance to predicate (arguments are not far from the predicate). Finally, for step 4 the same features are used as in step 1, but 5) and 9).

3 Experiments

As mentioned above, we started the development of the system from scratch with a very small team (actually only one programmer). Therefore we wanted to focus on certain aspects, totally abandoning our claims for achieving decent results for the others. One of our major goals was the construction of correct syntactic trees and the recognition of the predicate-argument structure - a subtask which mainly corresponds to the unlabeled accuracy. For that reason we reduced the scale of our experiments concerning such steps as dependency relation labeling, determining the correct reading for the predicates or the proper type of the arguments.

Unfortunately only the labeled accuracy was evaluated at this year's task, which was very frustrating in the end.

3.1 Syntactic Dependencies

For testing the strict incremental dependency parser we used the CoNLL 2008 shared task training and development set. Our best syntactic score that we could achieve on the development data was merely unlabeled attachment score (UAL) of 45.31%. However, as mentioned in sec. 2, we used a set of features proposed by Nivre,

which contains 5 features relying on the dependency types. Since we couldn't develop a good working module for this part of the task due to the lack of time, we couldn't exploit these features.

Note that for this experiment and all others reported below, we used the default settings of the opennlp MEM trainer. In particular this means that 100 iterations were used in all training runs and that for all experiments no tuning of parameters and smoothing was done, basically because we had no time left to exploit it in a sensible way. These parts will surely be revised and improved in the future.

3.2 Semantic Dependencies

As we describe in the sec. 3 our semantic module consists of 4 steps. For the first step we achieve the F-score of 76.9%. Whereas the verb predicates are recognized very well (average score for every verb category is almost 90%), we do badly with the noun categories. Since our semantic module depends on the input produced by the syntactic parser, and is influenced by its errors, we also did a test assuming a 100% correct parse. In this scenario we could achieve the F-score of 79.4%.

We have completely neglected the second step of the semantic task. We didn't even try to do the feature engineering and to train a model for this assignment, basically because of time constraints. Neither did we try to include some information about the predicate-argument structure in order to do better on this part of the task. The simple assignment of the statistically most frequent reading for each predicate reduced the accuracy from 76.9% down to 69.3%. In case of perfect syntactic parse the result went down from 79.4% to 71.5%.

Unfortunately the evaluation software doesn't provide the differentiation between the unlabeled and labeled argument recognition, which corresponds to our third and fourth steps respectively. Whereas we put some effort on identifying the arguments, we didn't focus on their classification. Therefore the overall best labeled attachment score for our system is 29.38%, whereas the unlabeled score is 50.74%. Assuming the perfect parse the labeled score is 32.67% and the unlabeled score is 66.73%. In our further work we will try to reduce this great deviation between both results.

3.3 Runtime performance

One of the main strong sides of the strict incremental approach is its runtime performance.

Since we are restricted in our feature selection to the already seen space to the left of the current word, both the training and the application of our strategy are done fast.

The training of our MEMs for the syntactic part requires 62 minutes. The training of the models for our semantic components needs 31 minutes. The test run of our system for the test data from the Brown corpus (425 sentences with 7207 tokens) lasted 1 minute and 18 seconds. The application on the WSJ test data (2399 sentences with 57676 tokens) took 20 minutes and 42 seconds. The experiments have been performed on a computer with one Intel Pentium 1,86 Ghz processor and 1GB memory.

4 Results and Discussion

The results of running our current version on the CoNLL 2008 shared task test data were actually a knockdown blow. We participated in the closed challenge, and obtained for the complete problem a labeled macro F1 for WSJ+Brown of 19.93 (20.13 on WSJ only, 18.14 on Brown). For the syntactic dependencies we got similar bad results (WSJ+Brown = 16.25, WSJ = 16.22, Brown = 16.47), as well as for the semantic dependencies (WSJ+Brown = 22.36, WSJ = 22.86, Brown = 17.94).

We see at least the following two reasons for this disastrous result: On the one hand we focused on the construction of correct syntactic trees and the recognition of the predicate-argument structure which were only parts of the task. On the other hand we stuck to our strict incremental approach, which greatly restricted the scope of development of our system.

Whereas the labeling part, which was so far considerably neglected, will surely be improved in the future, the strict incremental strategy in its current form will probably have to be revised.

4.1 Post-evaluation experiments

We have already started beginning the improvement of our parsing system, and we briefly discuss our current findings. On the technical level we already found a software bug that at least partially might explain the unexpected high difference in performance between the results obtained for the development set and the test set. Correcting this error now yields an UAL of 53.45% and an LAL of 26.95% on the syntactic

part of the Brown test data which is a LAL-improvement of about 10%.

On the methodological level we are studying the effects of relaxing some of the assumptions of our strict incremental parsing strategy. In order to do so, we developed a separate model for predicting the unlabeled edges and a separate model for labeling them. In both cases we used the same features as described in sec. 2, but added features that used a right-context in order to take into account the PoS-tag of the N-next words viz. N=5 for the syntactic parser and N=3 for the labeling case. Using both models during parsing interleaved, we obtained UAL=65.17% and LAL=28.47% on the development set.

We assumed that the low LAL might have been caused by a too narrow syntactic context. In order to test this assumption, we decoupled the prediction of the unlabeled edges and their labeling, such that the determination of the edge labels is performed *after* the complete unlabeled dependency tree is computed. Labeling of the dependency edges is then simply performed by running through the constructed parse trees assigning each edge the most probable dependency type. This two-phase strategy achieved an LAL of 60.44% on the development set, which means an improvement of about 43%. Applying the two-phase parser on the WSJ test data resulted in UAL=65.22% and LAL=62.83%; applying it on the Brown test data resulted in UAL=66.50% and LAL=61.11%, respectively.

Of course, these results are far from being optimal. Thus, beside testing and improving our parser on the technical level, we will run further experiments for different context sizes, exploiting different settings of parameters of the classifier and feature values, and eventually testing other ML approaches. The focus here will be on the development of unlabeled edge models, because it seems that an improvement here is substantial for an overall improvement. For example, applying the decoupled edge labeling model directly on the given unlabeled dependency trees of the development set (i.e. we assume an UAL of 100%) gave as an LAL of 92.88%.

Beside this, we will also re-investigate interleaved strategies of unlabeled edge and edge labeling prediction as a basis for (mildly-) strict incremental parsing. Here, it might be useful to relax the strict linear control regime by exploring beam search strategies, e.g. along the lines of Collins and Roark (2004).

5 Conclusion

We have presented a puristic approach for joint dependency parsing and semantic role labeling. Since, the development of our approach has been started from scratch, we didn't manage to deal with all problems. Our focus was on setting up a workable backbone, and then on trying to do as much feature engineering as possible. Our bad results on the CoNLL 2008 suggest that our current strategy was a bit too optimistic and risky, and that the strict incremental deterministic parsing regime seemed to have failed in its current form. We are now in the process of analysis of "what went wrong", and have already indicated some issues in the paper.

Acknowledgement

The work presented here was partially supported by a research grant from the German Federal Ministry of Education, Science, Research and Technology (BMBF) to the DFKI project HyLaP, (FKZ: 01 IW F02). We thank the developers of the Opennlp.maxent software package.

References

Brants, Sabine, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER Treebank in Proceedings of the Workshop on Treebanks and Linguistic Theories Sozopol.

Collins, Michael, and Brian Roark. (2004). Incremental Parsing with the Perceptron Algorithm. ACL 2004.

Hacioglu, Kadri. 2004. Semantic Role Labeling Using Dependency Trees. *Coling 2004.*

Nivre, Joakim. 2007. Incremental Non-Projective Dependency Parsing. *NAACL-HLT 200).*

Manning, Christopher, and Hinrich Schutze. 1999. Foundations of statistical natural language processing. Cambridge, Mass.: MIT Press.

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

Volokh, Alexander. 2008. Datenbasiertes Dependenzparsing. *Bachelor Thesis, Saarland University.*

Wang, Mengqui, Kenji Sagae, and Teruko Mitamura. 2006. A Fast, Accurate Deterministic Parser for Chinese. ACL 2006.