Incorporating Semi-supervised Features into Discontinuous Easy-first Constituent Parsing

Yannick Versley
Department of Computational Linguistics
University of Heidelberg
versley@cl.uni-heidelberg.de

Abstract

This paper describes our entry for the discontinuous constituent parsing track using EAFI, a parser for easy-first parsing of discontinuous constituents,[1] as well as the work we undertook to adapt it to multiple languages as well as make use of the unlabeled data that was provided as part of the SPMRL shared task 2014.

1 Introduction

The SPMRL shared task 2014 (Seddah et al., 2014) augments the 2013 shared task dataset – dependency and constituent trees for several languages, including discontinuous constituent trees for Swedish and German – with unlabeled data that allows for semisupervised parsing approaches.

For the shared task entry, we added provisions to EaFi that make it easier to use it in new languages (section 2.1), by using the provided dependency data to construct a head table automatically, and use refinements to the feature model based on the Universal POS tagset mapping (Petrov et al., 2012) for Swedish and German.

Section 3.1 explains the use of word clusters for semi-supervised parsing, and finally section 3.2 discusses how we incorporated a dependency bigram model into the parser.

2 Easy-first parsing of discontinuous constituents

The EAFI parser uses the easy-first parsing approach of Goldberg and Elhalad (2010) for discontinuous constituent parsing. It starts with the sequence of terminals with word forms, lemmas, and part-of-speech tags, and progressively applies the parsing action that has been classified as most certain. Because the classifier only uses features from a small window around the action, the number of feature vectors that have to be computed and scored is linear in the number of words, contrary to approaches that perform parsing based on a dynamic programming approach.

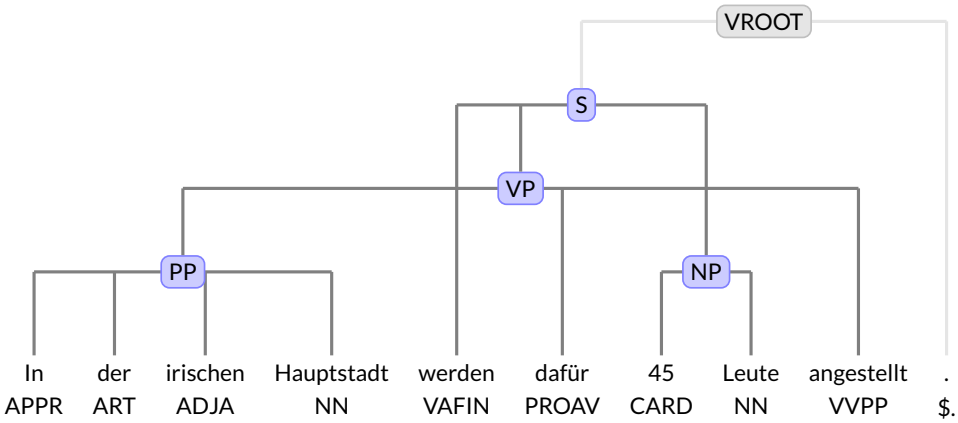
By using a *swap* action similar to the online reordering approach of Nivre et al. (2009), EAFI is able to perform discontinuous constituent parsing in sub-quadratic time, with an actual time consumption being close to linear.

In order to learn the classifier for the next action, the training component of EAFI runs the parsing process until the first error (early stopping, cf. Collins and Roark, 2004). The feature vectors of the erroneous action and of the highest-scoring action are used to perform a regularized AdaGrad update (Duchi et al., 2011).

2.1 Multilingual Adaptations

The EaFi parser uses two components that are language-specific and not contained in the treebank that is used for training: the first is a head table that is used to induce the head (pre-)terminal of a constituent,

[1] EaFi will be released as part of the PyTree tool suite at http://bitbucket.org/yannick/pytree after the requisite cleanups.



"*In the Irish capital, 45 people were hired for it.*"

Figure 1: A discontinuous constituent tree. The VP node has block-degree 3.

and the second is a list of "*special*" part-of-speech tags where the parser augments the POS values in features with the word forms.

We induce a **head table** by interposing the dependency data in CoNLL format with the constituency trees. Based on this, the *actual head* of a phrase is the a preterminal that has a governor outside the phrase. A constituent may have multiple heads when constituency and dependency criteria do not match exactly; In cases such as coordination constructions or appositions, the head determination always follows the rules of the dependency scheme. The *head constituent* is the constituent that has the head (preterminal) as part of its yield.

From these *observed* head constituents, we then derive a head table in the format used by RPARSE (Maier, 2010) and DISCODOP (van Cranenburgh, 2012) by finding a priorization of the head constituent labels that fits the actual heads maximally well.

For each constituent label, we start with a *candidate set* of all labels of head constituents, and infer a priorization of these constituent labels that fits the observed head constituents:

1. Look for phrases where two daughter constituent labels from the candidate set occur. These are called *conflicts* because the assigned head would depend on the order in the head rule.

   The *score* of a label is the number of *wins* (where this label and another candidate co-occur and this label has the actual head) minus the number of *losses* (where this label and another candidate co-occur and the other candidate is the actual head constituent).

2. The label from the candidate set with the highest score is appended to the rule. To decide on right-to-left or left-to-right precedence, look at conflicts between two instances of this label and count the number of conflicts that have been resolved towards the right/left constituent of those cases.

   Remove the label from the candidate set, and all conflicts that contain this candidate. If any labels remain in the candidate set, start again at (1.)

For the list of **special categories**, our intuition is that these will be most useful in cases such as PP attachment (which motivated their treatment as a special case in the case of Goldberg and Elhalad (2010), and possibly conjunctions).

We use the *Universal Tagset Mapping* of Petrov et al. (2012) where it is available to make a three-way split between open-class POS tags, closed-class POS tags, and punctuation.

- In the case of tagsets that have a Universal POS tag mapping, and tags that are mapped to ADP (adpositions) and CONJ (conjunctions) are included in the closed-class tags. Tags that have a universal POS mapping as . (punctuation) count as punctuation.

- In the case where no such mapping is available (e.g. Polish), we look at the count of types and tokens.

  If a tag has more instances containing punctuation than those containing containing a letter, and the number of tokens that contain a letter is less than 5, this tag is treated as punctuation.

  If a tag has more than 100 occurrences, while it only occurs with less than 40 different word forms, it is treated as a closed-class tag.

2.2 General tuning

EaFi uses online learning with a hash kernel to realize the learning of parameters – in particular, AdaGrad updates (Duchi et al., 2011) with forward-backward splitting (FOBOS) for L1 regularization (Duchi and Singer, 2009). Several parameters influence the performance of the parser:

- The size of the weight vector – because a hash kernel is used, collisions of the hash function on the available dimension can have a negative impact on the performance.

  We use a 400MB weight vector, which still allows training on modestly-sized machines, but leaves room for more features than the 80MB weight vector used by Versley (2014).

- The size of the regularization parameter. As FOBOS does not modify the weights between updates, smaller parameters seem to work better than larger ones. Goldberg (2013) suggests a value of $\lambda = \frac{0.05}{|D|}$, due to Alexandre Passos, where $|D|$ is the number of decisions per epoch.

  In our case, a value of $\lambda = \frac{0.001}{N}$ (for N the number of sentences in the training set) works considerably better than the $\lambda = \frac{0.1}{N}$ that was used in the initial results reported by Versley (2014).

3 Integrating semi-supervised features

3.1 Using word clusters

Augmenting word forms with word clusters is univerally recognized as a straightforward way to improve the generalization performance of a parser. In discriminative parsers such as the dependency parser of Koo et al. (2008), features that use surface forms are complemented by duplicated features where the word forms are (wholly or in part) replaced by clusters. A discriminative framework also allows to use both clusters and reduced clusters.

Candito and Seddah (2010) have shown that word clusters can productively be incorporated into a generative parser such as the Berkeley Parser, which uses a PCFG with latent annotations (PCFG-LA). In their case, they augment the clusters with suffixes to improve the parser's ability to assign the correct part-of-speech tags.

As EAFI uses discriminative parsing, we followed Koo et al. in providing duplicates of features where word form features are replaced by features using clusters.[2]

For all bigrams m, n (both bigrams, and the skip bigrams $n_{-1}n_2$ and n_0n_2), the supervised model already includes the combinations

$$\mathrm{W}m\mathrm{W}n\ \mathrm{W}m\mathrm{C}n\ \mathrm{C}m\mathrm{W}n\ \mathrm{W}m\mathrm{W}n$$

of words and the category.

For each kind K of clusters, we additionally include combination of category and the cluster of the respective head word:

$$\mathrm{C}m\mathrm{K}n\ \mathrm{K}m\mathrm{C}n\ \mathrm{C}m\mathrm{K}m\mathrm{C}n\ \mathrm{C}m\mathrm{C}n\mathrm{K}m\ \mathrm{C}m\mathrm{K}m\mathrm{C}n\mathrm{K}n$$

We made experiments with the original clusters and with the clusters shortened to 6 bits and 4 bits, respectively, in which the full clusters performed best. The final model combines features using the full clusters with features using the 6-bit cluster prefixes.

[2]Thanks to Djamé Seddah for providing providing Brown clusters for these languages.

	F_1	EX	UAS	NP	PP	VP
original EAFI	76.35	41.35	87.02	75.1	83.2	57.9
+ multilingual adaptation	76.70	41.93	87.12	75.5	83.1	58.5
+ 400MB weights	76.65	41.79	87.11	75.4	83.3	58.6
+ l1=0.001	80.60	48.43	89.75	79.8	86.9	65.0

Table 1: Parser parameters (German only, dev set, gold preprocessing, $\ell \leq 70$)

3.2 Using a Dependency Bigram Language Model

For models with a generative component, self-training (as in McClosky et al., 2006) can provide tangible benefits. Indeed, Suzuki et al. (2009) show that it is possible to reach improvements in dependency parsing beyond what is possible with word clustering when combining a discriminative model that uses word clusters with an ensemble of generative models that are used as features.

While the approach of Suzuki et al. works with a dynamic programming model of parsing, Zhu et al. (2013) show that it is also possible to use lexical dependency statistics learned from a large corpus to improve a state-of-the-art shift-reduce parser for constituents.

Similar to Zhu et al., we first extract pairs of words linked by a dependency edge, and a weighting function is applied to these co-occurrence counts: in the case of *Raw*, the raw counts are used; in the case of *L1*, the counts are simply normalized over each head words, and in the case of *LL*, the G^2 likelohood ratio statistic of Dunning (1993) is used.

The scored weights for dependents of a head word that were seen in the corpus are then discretized into quantiles, with the top-10% of seen dependents being labeled `HI` (high association), the top-30% scored dependents being labeled as `MI` (medium asssociation), and the remaining seen dependents being labeled as `LO` (low association), with unseen dependents receiving a `NO` (no association label). Through this mechanism, any potential dependent-head pair can be given one label from the set of `NO`/`LO`/`MI`/`HI`.

In an edge-factored dynamic programming parser, it would be sufficient to simply attach these labels as features to single edges; In the easy-first model, we also need to include features concerning *concurrent* edges that we exclude by joining the partial trees at positions 0 and 1. Hence, not only the pair $(0, 1)$ receives features based on the dependency bigram (both the bin label, and that label joined with head and dependent-candidate part-of-speech tags), but also the position pairs $(1, 2)$ and $(0, 2)$.

The bigram association strength feature is taken both by itself and paired with the POS tags of the words in question.

4 Experiments

Among the treebanks used in the SPMRL shared task, German and Swedish have discontinuous constituents – in this case, German has a large number of them (about ten thousand on the five thousand sentences of the test set), while Swedish only has very few (only fifty discontinuous phrases in the 600 sentences of the test set).

Based on prior experiments, learning on the larger German dataset was run for 15 epochs, whereas training on the Swedish dataset was run for 30 epochs.

5 Results and Discussion

Table 1 shows how the adaptations to the purely supervised part of EAFI influence the results based on results for German gold tags. In particular, the data-driven head table and special POS tags have a slight positive effect. Increasing the size of the weight vector does not seem to have strong effect, which implies that the existing weight vector is sufficient for the feature set used in the experiments. However, a different setting for the regularization constant yields a rather large difference (almost +4%), indicating that the previous setting was suboptimal.

In tables 2 and 3, we find the supervised initial results together with experiments regarding the use of clusters and their granularity, and the use of features based on the bigram language model.

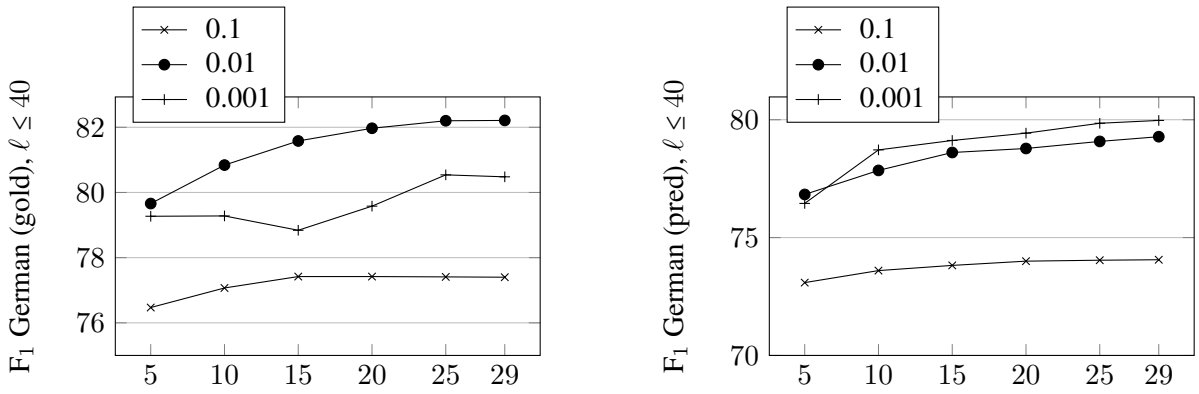


Figure 2: Influence of the regularizer on learning performance (yield F_1 versus epochs)

	F_1	EX	UAS	NP	PP	VP
German supervised	80.60	48.43	89.75	79.8	86.9	65.0
clusters (full)	**82.45**	**50.65**	**90.54**	81.3	87.9	67.4
clusters (6bit)	80.56	48.07	89.66	80.1	87.2	62.7
clusters (4bit)	80.81	47.95	89.83	79.9	87.1	64.2
clusters (full+6bit)	81.64	49.73	90.05	81.1	87.5	65.3
clust(full+6bit)+Bigram(*Raw*)	80.20	49.43	89.66	81.1	87.3	59.9
clust(full+6bit)+Bigram(*L1*)	80.60	49.09	89.62	81.3	87.1	60.4
clust(full+6bit)+Bigram(*LL*)	**82.20**	**50.86**	**90.31**	81.2	87.8	65.9
Swedish supervised	73.13	18.90	74.72	76.6	66.7	64.6
clusters (full)	**75.77**	**22.97**	**76.93**	78.3	71.1	66.0
clusters (6bit)	75.25	22.76	76.69	78.0	70.7	66.4
clusters (4bit)	74.98	21.95	76.17	78.0	70.7	66.6
clusters (full+6bit)	76.02	24.39	76.30	78.8	70.4	66.5
clust(full+6bit)+Bigram(*Raw*)	75.93	22.36	**77.18**	79.1	69.8	64.7
clust(full+6bit)+Bigram(*L1*)	76.12	**22.76**	77.08	78.9	71.2	66.8
clust(full+6bit)+Bigram(*LL*)	**76.37**	22.56	77.16	79.2	71.6	68.5

Table 2: Integration of semisupervised features (gold preprocessing)

Both for Swedish and for German, we see that adding cluster-based features improves the results considerably, with an increase of +1.3% in the case of German and of slightly more than +3.5% in Swedish for predicted tags and +1.7% and +2.6%, respectively, for gold tags.

We also see that the shortest version of the clusters (4bit) works less well than the others, while clusters shortened to 6-bit prefixes are relatively close to the results using full clusters.

6 Conclusions

In this paper, we have reported adaptations with the dual goal of, firstly, using the EAFI engine for parsing multiple languages by using existing dependency conversions and tagset mappings to provide head rules and lists of closed-class tags; secondly, of improving on these supervised learning results by incorporating features based on data from large corpora without manual annotation, namely Brown clusters and a dependency bigram language model.

Experimental results show that these two improvements are well-suited to improve the capabilities of the parser. At the same time, they demonstrate that techniques that are known from dependency parsing can also be harnessed to create parsers for discontinuous constituent structures that work better than existing parsers that are based on treebank LCFRS grammars, making it a practical solution for the parsing of discontinuous structures such as extraposition and scrambling.

References

Candito, Marie-Helene and Djamé Seddah. 2010. Parsing word clusters. In *Proceedings of the First Workshop on Statistical Parsing of Morphologically-Rich Languages (SPMRL 2010).*

	F_1	EX	UAS	NP	PP	VP
German supervised	78.63	44.97	87.51	77.4	85.9	59.3
clusters (full)	**79.96**	**47.35**	**88.16**	79.1	86.7	61.2
clusters (6bit)	79.34	45.91	87.69	78.1	86.4	60.8
clusters (4bit)	78.59	44.53	87.45	77.2	85.6	59.6
clusters (full+6bit)	79.77	46.79	87.96	79.1	86.4	60.9
clust(full+6bit)+Bigram(*Raw*)	79.95	47.09	88.14	79.1	86.7	61.5
clust(full+6bit)+Bigram(*L1*)	**80.07**	**47.25**	88.12	79.2	86.7	61.8
clust(full+6bit)+Bigram(*LL*)	79.96	47.19	**88.20**	79.3	87.0	60.8
Swedish supervised	70.72	16.06	73.32	74.9	66.1	61.5
clusters (full)	74.26	18.90	75.78	77.0	69.7	65.1
clusters (6bit)	74.05	19.72	75.57	77.0	69.6	65.2
clusters (4bit)	72.06	17.89	74.50	76.1	66.9	62.9
clusters (full+6bit)	**74.39**	**20.33**	**76.06**	76.5	69.0	66.4
clust(full+6bit)+Bigram(*Raw*)	**74.46**	19.92	**76.24**	77.4	70.2	66.2
clust(full+6bit)+Bigram(*L1*)	74.19	20.12	75.82	77.1	70.4	65.7
clust(full+6bit)+Bigram(*LL*)	74.39	**20.33**	76.06	76.5	69.0	66.4

Table 3: Integration of semisupervised features (predicted tags&morph)

	F_1	EX	UAS	NP	PP	VP
German, gold preprocessing						
supervised ♢	73.53	38.47	85.54	74.5	82.8	56.1
clusters (full+6bit)	75.76	39.71	86.51	76.1	83.7	60.3
+Bigram (LL) ♡	76.46	41.05	86.94	76.4	84.4	60.5
German, predicted tags+morph						
supervised ♢	71.96	35.12	83.15	71.7	80.9	55.0
clusters (full+6bit)	73.35	35.98	83.80	73.1	82.1	55.4
+Bigram (LL) ♡	73.90	37.00	84.16	73.7	82.6	56.7
Swedish, gold preprocessing						
supervised ♢	81.16	31.73	80.81	83.2	79.4	77.0
clusters (full+6bit)	82.10	34.89	81.56	84.6	80.3	77.1
+Bigram(LL) ♡	82.49	34.89	81.37	84.7	79.2	79.2
Swedish, predicted tags+morph						
supervised ♢	79.17	28.42	79.63	82.6	76.1	73.1
clusters (full+6bit)	80.61	30.83	79.71	83.4	78.2	76.6
+Bigram(LL) ♡	80.58	30.68	80.14	83.6	78.5	76.7

Table 4: Results on the test set. Results from the official submission are marked with ♢ (supervised run) and ♡ (semi-supervised run).

Collins, Michael and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *ACL-04*.

Duchi, John, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12:2121–2159.

Duchi, John and Yoram Singer. 2009. Efficient learning with forward-backward splitting. In *Proceedings of Neural Information Processing Systems (NIPS 2009)*.

Dunning, Ted. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics* 19(1):61–74.

Goldberg, Yoav. 2013. Dynamic-oracle transition-based parsing with calibrated probabilistic output. In *Proceedings of IWPT 2013*.

Goldberg, Yoav and Michael Elhalad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Proceedings of NAACL-2010*.

Koo, Terry, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *ACL 2008*.

Maier, Wolfgang. 2010. Direct parsing of discontinuous constituents in German. In *Proceedings of the NAACL-HLT First Workshop on Statistical Parsing of Morphologically Rich Languages*.

McClosky, David, Eugene Charniak, and Mark Johnson. 2006. Reranking and self-training for parser adaptation. In *CoLing/ACL 2006*.

Nivre, Joakim, Marco Kuhlmann, and Johan Hall. 2009. An improved oracle for dependency parsing with online reordering. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT)*.

Petrov, Slav, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation*.

Seddah, Djamé, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clergerie. 2014. Overview of the SPMRL 2014 Shared Task on Parsing Morphologically Rich Languages. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactical Analysis of Non-Canonical Language*. Seattle, WA, pages 146–182.

Suzuki, Jun, Hideki Isozaki, Xavier Carreras, and Michael Collins. 2009. An empirical study of semi-supervised structured conditional models for dependency parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

van Cranenburgh, Andreas. 2012. Efficient parsing with linear context-free rewriting systems. In *EACL 2012*.

Versley, Yannick. 2014. Experiments with easy-first nonprojective constituent parsing. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Language*.

Zhu, Muhua, Jingbo Zhu, and Huizhen Wang. 2013. Improving shift-reduce constituency parsing with large-scale unlabeled data. *Natural Language Engineering* .