Improving cross-domain dependency parsing with dependency-derived clusters

Jostein Lien **Erik Velldal** **Lilja Øvrelid**

Department of Informatics
University of Oslo, Norway
{josteili, erikve, liljao}@ifi.uio.no

Abstract

This paper describes a semi-supervised approach to improving statistical dependency parsing using dependency-based word clusters. After applying a baseline parser to unlabeled text, clusters are induced using K-means with word features based on the dependency structures. The parser is then re-trained using information about the clusters, yielding improved parsing accuracy on a range of different data sets, including WSJ and the English Web Treebank. We report improved results using both in-domain and out-of-domain data, and also include a comparison with using n-gram–based Brown clustering.

1 Introduction

Several recent studies have attempted to improve dependency parsers by including information about word clusters into their statistical parsing models. This is typically motivated by at least two concerns, both of which relate to the shortage of labeled training data. As argued by Koo et al. (2008), the lexicalized statistics important to disambiguation in parsing are often sparse, and modeling relationships on a more general level than the words themselves may therefore be helpful. The other motivation is domain adaptation, attempting to leverage a parsing model for use on data from a new domain. By including information about word clusters estimated from unlabeled in-domain data, one can hope to reduce the loss in performance expected from using a parser trained on an out-of-domain treebank.

While previous approaches have typically relied on the n-gram–based Brown clustering (Brown et al., 1992), this paper instead describes experiments using dependency-based word clusters formed using the generic clustering algorithm K-means. After applying a baseline dependency parser to unlabeled text, K-means is applied to form word clusters with features based on the dependency structures produced by the parser. The parser is then re-trained using features that record information about the dependency-derived clusters, thereby introducing an element of self-training. The re-trained parser obtains improved parsing accuracy on a range of different data sets, including the five web domains of the English Web Treebank (EWT) (Bies et al., 2012) and the Wall Street Journal (WSJ) portion of the Penn Treebank (PTB) (Marcus et al., 1993). We document improvements using both in-domain and out-of-domain data, and also when compared to using Brown clusters. All our parsing experiments use MaltParser (Nivre et al., 2007), a data-driven transition-based dependency parser.

The rest of the paper is structured as follows. Section 2 provides an overview of previous work. Section 3 details the data sets we use, including comments on the pre-processing. Section 4 then describes the experimental set-up, while the actual experiments and results are described in Section 5. A summary with thoughts about future directions is provided in Section 6.

2 Related work

The task of assigning word-to-word relations is at the core of dependency parsing, and statistics regarding relations between different words in the training data therefore provide vital information. These lexical statistics are, however, often sparse, and there exists a growing body of work which examines various strategies for generalizing over the

distributions of words and using different kinds of lexical categories. This section reviews relevant previous work in this direction based on the use of word clusters.

Several prior studies on using word clusters for improving statistical parsers have relied on the Brown algorithm (Brown et al., 1992) to produce the clusters. The Brown algorithm produces a hierarchical clustering, with each node in the tree corresponding to a pairwise merge operation. The criterion for merging clusters in the Brown algorithm is to minimize the decrease in the likelihood of a given corpus according to a class-based bigram language model. As with any hierarchical clustering method, the result is actually a set of nested partitions, and in order to produce a final set of flat clusters, a cut must somehow be defined on the tree (i.e., selecting all nodes at a certain depth from the root and collapsing all nodes below them).

One of the reports on using Brown clusters is the study presented by Koo et al. (2008). In experiments with dependency parsing of PTB and the Prague Dependency Treebank (PDT) (Hajič, 1998), Koo et al. (2008) showed substantial performance gains for both English and Czech when incorporating cluster-based features in their discriminative learner (averaged perceptron). The English word clusters were derived from the BLLIP corpus (Charniak et al., 2000), which contains roughly 30 million words of Wall Street Journal text (and overlaps with the Penn Treebank). Czech word clusters were derived from the raw text section of the PDT 1.0, reported to contain about 39 million words of newswire text. In both cases the clustering is performed on data overlapping with what is used for parsing.

Koo et al. (2008) experiment with different feature configurations, extending the baseline feature sets of McDonald et al. (2005; Carreras (2007), but only generate cluster-based features for the top N=800 most frequent words in the corpus, and set the Brown algorithm to only recover at most 1,000 distinct clusters. Koo et al. (2008) reports relative error reductions of up to 14% for unlabeled parsing of PTB when adding cluster features to their baseline parser. Looking at learning curves, Koo et al. (2008) show that the use of word clusters can also be used to compensate for reduced training data for the parser.

Candito and Seddah (2010) apply Brown clusters in the context of statistical constituent parsing for French, experimenting with creating clusters of lemmas and PoS-tagged lemmas. The clusters themselves are created from the L'Est Républicain corpus (using up to 1,000 clusters), comprising 125 million words of news text, and cluster-based features are then added to the Berkeley PCFG parser with latent annotations (Petrov et al., 2006), before parsing the French Treebank (Abeillé et al., 2003). Candito and Seddah (2010) analyze the results with respect to word frequency and find improvements in performance for all strata; unseen or rare words, as well as medium- to high-frequency words. Adding PoS-information to the lemmas also appeared beneficial, though depending on the quality of the tagger.

Øvrelid and Skjærholt (2012) apply Brown clusters to improve dependency parsing of English web data using MaltParser. Augmenting a WSJ-trained parser with Brown clusters – using the cluster labels of Turian et al. (2010) computed for the Reuters corpus – is shown to improve parsing accuracy on a range of web texts, including the Twitter and user forum data from the web 2.0 data sets described by Foster et al. (2011) and web data from various sources in the OntoNotes corpus, release 4 (Weischedel et al., 2011). In the experiments of Øvrelid and Skjærholt (2012), cluster information was found to be more beneficial for parsing with automatically assigned PoS tags (using SVMTool), while less so when using gold PoS tags. Improvements were also more pronounced for the web data than on WSJ. Experimenting with different tree cut-offs, producing different numbers of clusters, Øvrelid and Skjærholt (2012) found that using a smaller number of large and general clusters (100–320) worked better than using a higher number of smaller and more fine-grained clusters (experimenting with up to 3200 clusters).

As an alternative to the above approaches using n-gram-based Brown clusters, the current paper documents experiments with using syntactically informed clusters instead, generated with a generic clustering algorithm. One previous study following a related line of investigation is that of Sagae and Gordon (2009) who also used parsed data for creating syntactically informed clusters. The clustering is there performed by applying the general method of (average-link) hierarchical agglomerative clustering to the 5,000 most frequent words of

the BLLIP WSJ corpus, containing approximately 30 million words of WSJ news articles, parsed with the Charniak (2000) parser. The features used for the clustering encode phrase-structure tree paths that include direction information and non-terminal node labels, but does not include lexical information or part-of-speech tags. The clusters are then added as features in a data-driven transition-based dependency parser which is again used to identify predicate-argument dependencies extracted from the HPSG Treebank developed by Miyao et al. (2004) comprising the standard PTB WSJ sections. The pipeline described by Sagae and Tsujii (2008) thus include several layers of cross-framework interactions. Cutting the cluster hierarchy to include 600 clusters was shown to given the highest F-score, significantly improving the accuracy of the predicate-argument dependency parser.

The goal of Sagae and Gordon (2009) is to improve the accuracy of a fast dependency parser by using a corpus which has previously been automatically annotated using a more accurate but slower phrase-structure parser. In our experiments we seek to improve a baseline dependency parser by using clusters formed directly on the basis of the annotations of the baseline parser itself, without the complexity of involving a second parser. Using the method of K-means we will define a flat partition directly, without the need to cut the tree formed by a hierarchical method. While Sagae and Gordon (2009) focus on cross-framework leveraging, all testing is for in-domain models only, like for Koo et al. (2008), whereas the current paper will also investigate the benefit of dependency-based word clusters for porting a parser to new domains and text-types. The following section presents the various data sets we use, before moving on to describe the experimental set-up and the results.

3 Data sets

We experiment with using several different data sets, both for forming the word clusters and for evaluating the re-trained cluster-informed parser. We describe the data sets as well as the relevant pre-processing steps below.

The shared task[1] on parsing English web data hosted by the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL 2012) provided both unlabeled and labeled data for the five different domains from the English Web Treebank (EWT): weblogs, emails, question-answers, newsgroups, and reviews (Petrov and McDonald, 2012). In addition to the web texts, the SANCL data also contains the WSJ portion of the OntoNotes corpus, release 4.0 (Weischedel et al., 2011). (The OntoNotes version of WSJ differs slightly from the original PTB in terms of tokenization and noun-phrases analysis in certain places.) For the shared task, the data for weblogs and emails were used for development testing, while answers, newsgroups, and reviews were reserved for held-out testing. We will be following that same structure here.

The SANCL data comprises both labeled and unlabeled data. The labeled web data, corresponding to the EWT,[2] is what we will be using for our parser evaluations in addition to WSJ sections 22 (dev.) and 23 (held-out). The unlabeled SANCL data will be used for clustering, in addition to the newswire collection of the Reuters Corpus Volume I (RCV1) (Lewis et al., 2004). All the unlabeled data sets are summarized in Table 1. Note that the SANCL data is provided pre-segmented, tokenized and converted to Stanford dependencies in the CoNLL06/07 data format. For Reuters we segmented and tokenized the data using NLTK (Bird et al., 2009).

Various other pre-processing steps are applied to the unlabeled data prior to clustering. First, PoS-tagging is performed using SVMTool (Giménez and Màrquez, 2004) (using version 1.3.1 with the pre-trained WSJ model and the following options: '`-S LRL -T 0`'). Note that we are clustering lemmas rather than word forms, and lemmatization is performed using the NLTK WordNet lemmatizer (Bird et al., 2009). Finally, a baseline configuration of MaltParser is applied using the parse model of Foster et al. (2011) and Øvrelid and Skjærholt (2012) – more information about the parser and the feature set is provided in Section 4.2.

In Section 5.4 we also compare results to Øvrelid and Skjærholt (2012) for using Brown clusters rather than K-means with dependency features. For this comparison we use some additional data

[1] `https://sites.google.com/site/sancl2012/home/shared-task`

[2] Despite the separation into development and test domain, the SANCL data still defines development and test splits for the labeled data in all five domains, but we will simply merge all the labeled data for each domain.

	Reuters	Weblogs	Emails	Answers	Newsgroups	Reviews
Sentences	12,515,901	524,834	1,194,172	27,274	1,000,000	1,965,350
Tokens	217,635,636	10,356,138	17,046,119	424,292	18,424,049	29,288,947

Table 1: The number of sentences and tokens in the unlabeled corpora used for clustering.

	WSJ 02-21	WSJ 22	WSJ 23	Weblogs	Emails	Answers	Newsgr.	Reviews
Sentences	30,060	1,336	1,640	4,060	4,900	6,976	4,782	7,627
Tokens	731,678	32,092	39,590	88,762	57,807	108,006	86098	111,182

Table 2: The number of sentences and tokens in the labeled corpora used for parsing.

from the OntoNotes corpus: general English web data (Eng.: 71,500 tokens) and a larger set of sentences originally selected to improve sense coverage in the corpus (Sel.: 279,000 tokens).

4 Experimental set-up

In this section we present the set-up of the experimental process. We start by describing the set-up for the clustering before turning to how the parser is trained and applied. The actual experiments and results are the provided in Section 5.

4.1 K-means word clustering with dependency features

We experiment with forming word clusters of lemmas in several different data sets: Reuters and the unlabeled SANCL data for five different web domains. The web data is clustered per-domain as well as all together. To run clustering on a given corpus, we first extract the 50,000 most frequent lemmas, only considering verbs, nouns and adjectives.[3] The next step – after the initial pre-processing with the SVMTool PoS tagger and MaltParser – is to record features for the various lemma occurrences across the corpus.

The features we use for the K-means clustering record information about the target lemma, its head, its leftmost / rightmost siblings, and its leftmost / rightmost dependents. The *siblings* of a target are defined as the tokens having the same head as the target. The *dependents* of a target are all the tokens having the target as the head. For both these notions, the leftmost or rightmost token corresponds to the one furthest to the left or right in the sentence, respectively. The clustering features record the following information for each lemma token:

- PoS
- Dependency label
- PoS of head
- Dependency label of head
- Lemma of head
- PoS leftmost/rightmost sibling
- Dependency label of leftmost/rightmost sibling
- Lemma of leftmost/rightmost sibling
- PoS leftmost/rightmost dependent
- Dependency label of leftmost/rightmost dependent
- Lemma of leftmost/rightmost dependent

The actual clustering is performed using the K-means implementation of the Python-based toolkit *scikit-learn* (Pedregosa et al., 2011), using its *mini-batch* version of the algorithm – an alternative online implementation optimized for large samples (Sculley, 2010). We perform clustering for K (i.e., the pre-defined number of clusters) set to 10, 50 and 100 (using higher values for K failed due to memory constraints).

4.2 Parser set-up

As said, our experiments our based on MaltParser (Nivre et al., 2007) (v. 1.7.2), a system for data-driven dependency parsing which is based on a deterministic parsing strategy in combination with treebank-induced classifiers for predicting parse transitions. It supports a rich feature representation of the parse history and can easily be extended to take additional features into account. We choose to use MaltParser primarily due to its easily extendable feature model which facilitates experimentation with additional features during parsing. As our baseline parser, we use the parse model described by Foster et al. (2011) and Øvrelid and

[3] More specifically, we only consider lemmas with a PoS tag from one of the sets {NN, NNS, NNP, NNPS}, {VB, VBD, VBG, VBN, VBP, VBZ} or {JJ, JJR, JJZ}.

Feature set	Feature templates
Baseline	S_0p, S_1p, S_2p, S_3p, L_0p, L_1p, L_2p, I_0p, $S_{0l}p$, $S_{0r}p$, $S_{1r}p$, $S_{0l}d$, $S_{1r}d$, S_0w, S_1w, S_2w, L_0w, L_1w, $S_{0l}w$, $S_{1r}w$, S_0pS_1p, S_0wL_0w, S_0pS_0w, S_1pS_1w, L_0pL_0w, $S_{1r}dS_{0l}d$, $S_{1r}pS_{1l}p$, $S_0pS_1pL_0p$, $S_0pS_1pS_2p$, $S_0pL_0pL_1p$, $L_0pL_1pL_2p$, $L_1pL_2pL_3p$, $S_0pL_0pI_0p$, $S_1pS_{1l}dS_{1r}d$
PoS simple	+ S_0l, S_1l, S_2l, S_3l, L_0l, L_1l, L_2l, I_0l, $S_{0l}l$, $S_{0r}l$, $S_{1r}l$
Form simple	+ S_0l, S_1l, S_2l, L_0l, L_1l, $S_{0l}l$, $S_{1r}l$
Form all	+ S_0l, S_1l, S_2l, L_0l, L_1l, $S_{0l}l$, $S_{1r}l$, S_0lL_0l, S_0pS_0l, S_1pS_1l, L_0pL_0l,

Table 3: Feature models for the baseline and the re-trained parser, where p = PoS-tag, w = word form, d = dependency label in the graph constructed so far (if any), and l = cluster label. MaltParser's stacklazy algorithm operates over three data structures: a stack (S) of partially processed tokens, a list (I) of nodes that have been on the stack, and a "lookahead" list (L) of nodes that have not been on the stack. We refer to the top of the stack using S_0 and subsequent nodes using S_1, S_2, etc., and the leftmost/rightmost dependent of S_0 with S_{0l}/S_{0r}.

Skjærholt (2012). It employs the stacklazy algorithm (Nivre, 2009), along with the LIBLINEAR package (Fan et al., 2008) for inducing parse transition SVM classifiers.

4.2.1 Parser features

Table 3 describes the baseline feature set, along with three additional feature sets based on the models described in Øvrelid and Skjærholt (2012) and that in various ways include information about cluster labels: *PoS simple*, *Form simple* and *Form all*. These augmented feature sets are constructed by copying the full baseline feature set (*all*) or only the features that pertain to a single token (*simple*) and involve either the PoS-tag or the word form respectively. (Note that a *PoS all* feature set was also tried but proved to be too large for practical experimentation.) Preliminary experimentation on the development sets showed that the *Form all* feature model consistently outperformed the other two cluster-based feature set, so we will only be reporting results for this feature set in the paper, in addition to the baseline.

We evaluate the parser outputs in terms of Labeled Attachment Score (LAS) – computed using the evaluation script[4] of the CoNLL-X shared task on multi-lingual dependency parsing – and compare them using Dan Bikel's Randomized Parsing Evaluation Comparator with $p \leq \alpha = 0.05$ considered statistically significant.

	PoS in training	
	Gold	Predicted
WSJ 22	81.54	**84.88**
WSJ 23	81.88	**84.79**

Table 4: The effect on LAS for training on gold vs. predicted PoS tags, when testing on predicted PoS tags.

4.2.2 Gold vs. predicted PoS tags

In preliminary experiments we assessed the effect of using gold standard versus automatically assigned part-of-speech-tags tags when training the parser, in both cases testing on automatically tagged text. (These experiments used the baseline MaltParser without cluster information and using only default parameter settings, including $C = 0.1$ for the SVM.) We trained two versions of the parser on WSJ sections 02–21 (from OntoNotes/SANCL) using (1) the gold PoS tags provided in the treebank and (2) replacing these with tags automatically predicted by SVMTool. We then applied the parsers to WSJ 22 and 23, for both parsers using SVMTool tags during testing. The results are shown in Table 4.2.2 and reveal that there is a clear advantage to training on predicted tags (all differences are statistically significant at $\alpha = 0.05$). For all parsing results reported elsewhere in this paper automatically predicted PoS tags are used in both training and testing.

5 Experiments and results

The development results reported below are obtained by; parsing unlabeled data using the baseline feature set trained on WSJ 02-21; computing lemma clusters from dependency features; retraining the parser on WSJ 02-21 with the augmented *Form all* feature set; and finally tuning the number of clusters (K) and the C parameter[5] of

[4] `http://ilk.uvt.nl/conll/software.html`

[5] The penalty factor C governs the trade-off between training error and margin. It can have a large impact on the resulting model and, in our case, parser performance. In our empirical tuning on the dev. set we first tested values in the interval $[2^{-7}, 2^{-6} \ldots 2^{6}, 2^{7}]$ and after identifying the appropriate neighborhood further fine-tuned the value using in-

	WSJ 22	Weblogs	Emails
Baseline	86.72	80.00	72.85
Reuters clusters	**86.79**	**80.34**	73.11
SANCL clusters, per-domain	n/a	80.20	**73.35**
SANCL clusters, all	n/a	80.26	73.27

Table 5: LAS results for the development data – WSJ 22 and the two SANCL test domains – comparing the baseline parser to parsers re-trained using word clusters from various sources of unlabeled data: the Reuters corpus, the unlabeled SANCL data for the respective domains, or clustering all the unlabeled SANCL data from all five domains together. (All data sets are PoS-tagged automatically using SVMTool.)

	Answers	Newsgroups	Reviews
Baseline	73.10	76.13	75.01
Reuters	**73.58**	**76.97**	75.43
SANCL per-domain	73.39	76.87	75.51
SANCL all	73.52	76.94	**75.53**

Table 6: Held-out LAS evaluation on the three SANCL test domains using the baseline parser compared to parsers re-trained with information about word clusters generated from various sources: the Reuters corpus, the unlabeled SANCL data for the respective test domains, or clustering all the unlabeled SANCL data from all five domains together. (All data sets are PoS-tagged automatically using SVMTool.)

the parser's SVM classifier to find the configuration with the highest LAS. The best configuration found for the development data is then applied to the held-out data.

5.1 Reuters clusters

Instantiating the clustering features described in Section 4.1 for the top 50k lemmas of Reuters resulted in a total of 1,673,744 feature types. Specifying a feature frequency cut-off of $>= 10$ brought this down to a more manageable set of 339,473 features. After running K-means for 10, 50 and 100 clusters and tuning the SVM penalty parameter of the parser, the best configuration for all the development data sets was found to be $K = 50$ and $C = 0.0625$. The results can be seen in Table 5, including the scores of the initial baseline parser.[6] Looking at the baseline scores, the results clearly demonstrates the difficulty in applying parsers to text outside the domain of the training data, combined with the added noise we can expect to find in web data text types compared to newswire text: There is a clear drop in performance for the web data compared to WSJ 22. While we see that the cluster-informed parser improves over the baseline across all data sets, we also see that the improvements are larger for the web data than for WSJ 22: For Weblogs and Emails the relative reductions of error rate (RER) are 1.7% and 0.96% respectively, compared to RER = 0.53% for WSJ. When applying these models to the held-out data, the gains of the cluster-informed models are even larger, as shown in Table 6, with error reductions of up to 3.52%. When testing on WSJ 23 (not included in the table), the baseline obtained LAS = 86.88, compared to 87.16 for the cluster model, amounting to RER = 2.13%. The differences in held-out performance where detected as statistically significant across all the data sets.

Note that one complication with respect to assessing the effect of K-means clustering is the fact that the algorithm is sensitive to the initial random seeding of the cluster centers. Using the mini-batch implementation in scikit-learn alleviates this problem to some degree in that it will compute a handful of different seedings and choose the one with the lowest inertia (i.e., within-cluster sum-of-squares) before starting the clustering. Still, repeated runs with the same parameters and the same input can generate different outputs. In order to quantify the extent of this effect we run K-means for $K = 100$ clusters 10 times on the Reuters data and parsed WSJ 22 with the re-trained parser (fixing the SVM parameter C to 0.0625). This resulted in mean and median LAS scores of 86.78 and 86.81 respectively, with a variance of 0.009 and a standard deviation of 0.095.

5.2 Per-domain SANCL clusters

While we already see improvements in parsing accuracy for the web data, one could expect to see even greater gains when using clusters generated from texts in the same domain that is to be parsed.

crements of 0.015: The best performance was typically found for $C = 0.0625$.

[6] For the baseline model, the best C value varied slightly across the different development sets, with $C = 0.0625$ for WSJ 22, 0.0775 for Weblogs, and 0.0925 for Emails. In subsequent held-out testing for the baseline we use the model trained with $C = 0.0625$ for the WSJ data and 0.0775 for the web data.

We therefore tried running K-means on the unlabeled SANCL data from respective test domains[7] (while still training the parser on WSJ like above). This means that, for example, the 4,060 sentences in the labeled Weblogs data is parsed using clusters generated for the 50K most frequent lemmas of the 524,834 sentences in the unlabeled Weblogs data. After empirically tuning the parameters, the highest LAS scores on the development sets were observed for the configuration of $K = 100$ and $C = 0.0625$. (As for all clustering results reported here we use the *Form all* feature set of Table 3.)

Development results are provided in Table 5 and held-out results in Table 6, see the row *SANCL per-domain*. Although the re-trained parser with per-domain clusters again significantly outperforms the baseline across all data sets, there is no clear advantage to using per-domain clusters compared to the Reuters cluster of the news domain. The parser using per-domain web clusters improves on the parser using Reuters clusters for two out of five domains: Emails (development) and Reviews (held-out). Interestingly, these are also the two domains with the largest unlabeled data sets, as shown in Table 1. At the same time, we see that the Reuters corpus is vastly larger than any of the unlabeled SANCL corpora. For our next round of experiments we therefore wanted to see whether we could compensate for this difference in size by clustering all the unlabeled SANCL data combined, while still hoping to see positive effects of using data closer to the test domain.

5.3 All-in-one SANCL clusters

The motivation of the experiments in this section is to see whether using word clusters generated from all the five unlabeled SANCL sections together yields better parsing performance than using clusters from each domain individually. Using a feature cut-off of ≥ 3, a total of 375,793 feature types are extracted for clustering the 50K most frequent noun/verb/adjective lemmas in the concatenated SANCL data. Using 100 clusters generated by K-means and setting $C = 0.0625$ for the SVM classifier in MaltParser, the results are shown as *SANCL all* in Tables 5–6.

We see that for all but one data set, the use of all-in-one SANCL clusters yield better results than per-domain clusters. The exception is the Emails (development) data, where the per-domain clusters still yields the highest LAS overall. At the same time, we see that the initial Reuters clusters still provide the highest score for three of the data sets, while the all-in-one SANCL model has the highest overall score for the (held-out) Reviews section. It is also worth noting that at 75 million tokens, the concatenated unlabeled SANCL data is still a third of the size of Reuters. When testing for statistical significance on the held-out data, none of the differences between the Reuters and SANCL runs are detected as being significant.

In sum, it is not possible to conclude anything about which data set provides the optimal source for generating the dependency-based word clusters for the parser, although it is clear that whichever data set is used, the re-trained parser with cluster features improves significantly on the baseline parser. For the final round of experiments, we investigate the use of the dependency-based clusters compared to n-gram Brown clusters as used in most previous studies.

5.4 Comparison to using Brown clusters

In this section, we report the results of parsing the English web data of the OntoNotes corpus as described in Section 3, in addition to the OntoNotes version of WSJ section 23, mirroring the data sets used by Øvrelid and Skjærholt (2012). The purpose is to compare the results obtained using our dependency-based clusters and the Brown clusters used by Øvrelid and Skjærholt (2012) and several previous studies. As to isolate the effect of the clustering approach as best as possible, we here use the same version of MaltParser as used by Øvrelid and Skjærholt (2012) (i.e., v.1.4.1). We otherwise apply the model configuration that was found to give the best results for the development experiments in Table 5, i.e., $K = 100$ and $C = 0.0625$, and apply models based on both the Reuters clusters and the all-in-one SANCL clusters.

The LAS results for the different models are compared in Table 7. It is important to note that while the scores for the dependency-based clusters represent strict held-out results, the results for Øvrelid and Skjærholt (2012) are to be regarded as development results: The scores of Øvrelid and Skjærholt (2012) are maximums after tuning the model parameters directly on the given data. The

[7] The frequency cut-off on the dependency features for the clustering was set to ≥ 2 for these runs. Note also that the vocabulary extracted for the unlabeled Answers data only comprises 22,227 lemmas, due to the smaller size of this data set.

	WSJ 23	Eng	Sel
Baseline, Øvrelid (2012)	86.24	76.99	74.84
Baseline	86.67	78.45	76.02
Brown, Øvrelid (2012)	86.67	78.30	75.82
Reuters	**86.98**	78.71	76.23
SANCL all	86.90	**78.79**	**76.30**

Table 7: Comparing LAS with Øvrelid and Skjærholt (2012), using data sets with automatic part-of-speech tags generated by SVMTool.

parameters include the number of clusters and the choice of feature set for the parser, corresponding to the various options listed in Table 3. In spite of this, we find that all the models using dependency-based clusters yield quite a bit higher LAS than the Brown-based models of Øvrelid and Skjærholt (2012). At the same time, even our baseline models perform on par with or better than the Brown models, so it is likely that other factors not accounted for are also affecting the results reported in Øvrelid and Skjærholt (2012). Note that the table include baseline results for both our own set-up and the scores provided in Øvrelid and Skjærholt (2012). Despite our efforts to replicate the set-up described by Øvrelid and Skjærholt (2012) we were not able to reproduce the results. The scores shown for our own baseline in Table 7 were produced using our tuned C parameters fpr the SVMs (though using the same version of the parser and tagger), but even when using the default parameters like reported by Øvrelid and Skjærholt (2012) our scores diverged.

6 Summary and future work

This paper has described a semi-supervised approach for improving a data-driven dependency parser using dependency-based clusters. The parser is first applied to a large corpus of unlabeled text, providing the input to K-means clustering of lemmas using features extracted from the dependency structures. The parser is the re-trained with new features that include information about the word clusters, thereby introducing an element of self-training. The cluster-informed parser is shown to improve significantly over the baseline on both in- and out-of-domain tests, including a wide range of web texts. For held-out tests on the web data the use of clusters yields error reductions of up to 3.52% relative to the baseline. The results of using our dependency-based clusters also compare favorably to previous studies using the n-gram based Brown clusters.

There are several directions we wish to pursue in follow-up work. The experiments in this paper were based on the feature set described by Øvrelid and Skjærholt (2012). Further work will give priority to the design and experimentation with additional cluster-based features in the parser, preferably informed by an analysis of the parser errors. The clustering described above comprise a fairly large vocabulary of 50,000 lemmas. In future experiments we would like to gauge the trade-off between the vocabulary size N and the number of clusters K: Decreasing N would allow us to specify a higher K. Moreover, when inspecting the the word clusters many of them can be seen to be fairly specific to distinct parts-of-speech – unsurprisingly, given the feature templates described in Section 4.1. In further experiments we therefore plan on performing the clustering separately for lemmas of different parts-of-speech. This will also be beneficial in terms of scalability: Computational considerations otherwise enforce limitations on vocabulary size, the number of clusters, and the size of the feature space, but running multiple and separate K-means clusterings for different PoS classes means we can increase the number of total clusters used and the lexical coverage of the clusters.

References

Anne Abeillé, Lionel Clément, and François Toussenel, 2003. *Treebanks: Building and Using Parsed Corpora*, chapter Building a Treebank for French. Kluwer, Dordrecht.

Ann Bies, Justin Mott, Colin Warner, and Seth Kulick. 2012. English Web Treebank LDC2012T13.

Steven Bird, Edward Loper, and Ewan Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, December.

Marie Candito and Djame Seddah. 2010. Parsing word clusters. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 76–84, Los Angeles, CA.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings*

of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 957–961, Prague, Czech Republic.

Eugene Charniak, Don Blaheta, Niyu Ge, Keith Hall, John Hale, and Mark Johnson. 2000. Brown laboratory for linguistic information processing (BLLIP) 1987–89 WSJ corpus release 1 LDC2000T43.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 132–139, Seattle, WA.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, (9).

Jennifer Foster, Özlem Çetinoğlu, Joachim Wagner, Joseph Le Roux, Stephen Hogan, Joakim Nivre, Deirdre Hogan, and Josef van Genabith. 2011. #hardtoparse: POS tagging and parsing the twitterverse. In *Proceedings of the AAAI Workshop on Analysing Microtext*, pages 20–25, San Francisco, CA.

Jesús Giménez and Lluís Màrquez. 2004. SVMTool: A general POS tagger generator based on Support Vector Machines. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, Lisbon, Portugal.

Jan Hajič. 1998. Building a syntactically annotated corpus: The Prague Dependency Treebank. In Eva Hajičová, editor, *Issues of Valency and Meaning. Studies in Honor of Jarmila Panevová*, pages 12–19. Prague Karolinum, Charles University Press.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 595–603, Columbus, OH.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English. The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 91–98, Ann Arbor, MI.

Yusuke Miyao, Takashi Ninomiya, and Jun'ichi Tsujii. 2004. Corpus-oriented grammar development for acquiring a Head-driven Phrase Structure Grammar from the Penn Treebank. In *Proceedings of the 1st International Joint Conference on Natural Language Processing*, pages 684–693, Hainan Island, China.

Joakim Nivre, Johan Hall, Jens Nilsson, Gülsen Eryigit, Sandra Kübler, Marinov Svetoslav, Erwin Marsi, and Atanas Chanev. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the 47th Meeting of the Association for Computational Linguistics*, pages 351–359, Suntec, Singapore.

Lilja Øvrelid and Arne Skjærholt. 2012. Lexical categories for improved parsing of web data. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING)*, pages 903–912, Bombay, India.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia, July.

Kenji Sagae and Andrew S. Gordon. 2009. Clustering words by syntactic similarity improves dependency parsing of predicate-argument structures. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT)*, pages 192–201, Paris, France.

Kenji Sagae and Jun'ichi Tsujii. 2008. Shift-reduce dependency DAG parsing. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 753–760, Manchester.

David Sculley. 2010. Web-scale K-means clustering. In *Proceedings of the 19th International Conference on World Wide Web*, pages 1177–1178, Raleigh, NC.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Meeting of the Association for Computational Linguistics*, pages 384–394, Uppsala, Sweden.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston. 2011. OntoNotes release 4.0 LDC2011T03.