# Combining Clustering Approaches for Semi-Supervised Parsing: the BASQUE_TEAM system in the SPRML'2014 Shared Task

**Iakes Goenaga, Nerea Ezeiza**
IXA NLP Group
Faculty of Computer Science
Univ. of the Basque Country UPV/EHU
iakesg@gmail.com, n.ezeiza@ehu.es

**Koldo Gojenola**
IXA NLP Group
Technical School of Engineering, Bilbao
Univ. of the Basque Country UPV/EHU
koldo.gojenola@ehu.es

## Abstract

This paper presents a dependency parsing system, presented as BASQUE_TEAM at the SPMRL'2014 Shared Task, based on the combination of different clustering approaches. We create new features applying clustering methods to automatically annotated large corpora. Once these new features are calculated, we add them to the base features in order to create a series of analyzers using two freely available and state-of-the-art dependency parsers, MaltParser and Mate. Finally, we will combine previously obtained results using a voting approach.

## 1 Introduction

In this paper we will try the use of a a semi-supervised approach to parsing morphologically-rich languages, using a large unannotated corpus as a source to create new features, which have been shown effective in several NLP tasks. In most cases the new features are derived from entire words without any separation between the lemmas and morphemes. This line of research can find sparsity problems in some languages. For that reason, we will experiment with the separation between the parts of the words into lemmas and morphemes. Taking these ideas into consideration and following Goenaga et al. (2013), we will work on three different approaches:

- We will experiment the effect of using new features derived from Brown clusters (Brown et al., 1992). In order to create these clusters we will make use of large unnanotated corpora. Each word of the corpora has been divided into lemma and suffix for the purpose of avoiding sparsity.

- Following Bansal et al. (2014), we believe that continuous representations (or distributed representations or embeddings) of the words could help us to obtain better results in dependency parsing. As we have done to create Brown clusters, we have made use of large unnanotated corpora where the words are separated into lemmas and suffixes. After that, we have created word clusters taking into account the cosine similarity of the word embeddings using the K-means algorithm (Hartigan and Wong, 1979). We will use these new features (word clusters) in the FEATS column with the intention of providing complementary semantic information to the parsers. In that way they have a wider information of each sentence and they will be able to use this knowledge for the purpose of obtaining better parsing results.

- Finally, we will experiment the combination of the different parsers with a voting approach (Hall et al., 2010) using the MaltBlender tool[1].

All of the experiments will be performed on automatically predicted *POS* and morphosyntactic data, taking the tags given in the Shared Task data, that is, we will not made use of any specifically trained morphological tagger.

In the rest of this paper we will first present the related work in section 2 followed by the resources we have used to carry out our experiments in section 3. We include a study of the contribution of the new features derived from Brown clusters to parsing in section 4 and the effect of the new features derived

[1]http://w3.msi.vxu.se/users/jni/blend/

from K-means clusters based on word embeddings on the individual parsers in section 5. The final results of the best parser combinations are showed in section 6 and the main conclusions of the work in section 7.

## 2 Related Work

The use of clustering methods in NLP tasks has been increasing in the last years because they have been shown effective in several tasks. Miller et al. (2004) presented a technique for augmenting annotated training data with hierarchical word clusters automatically derived from a large unannotated corpus. They evaluated their technique for named-entity tagging and demonstrated that it requires only 13% as much annotated data to achieve the same level of performance compared with a state-of-the-art name finder.

The experiments carried out by Koo et al. (2008) demonstrated that word clusters can be quite effective in dependency parsing applications. They presented a semi-supervised method for training dependency parsers introducing features that incorporate word clusters derived from a large unannotated corpus. They made experiments on the Penn Treebank and Prague Dependency Treebank, and they showed that the use of cluster-based features yielded substantial gains in performance. They improved the baseline accuracy by 1.14% and 1% for English and Czech, respectively.

Chen et al. (2006) presented an application of a spectral clustering technique to unsupervised relation extraction. It works by calculating eigenvectors (a special set of vectors associated with a linear system of equations that are sometimes also known as characteristic vectors, proper vectors, or latent vectors) of an adjacency graph's Laplacian to recover a submanifold of data from a high dimensional space, and then performing cluster number estimation on a transformed space defined by the first few eigenvectors. They claim that spectral clustering doesn't need to provide the number of clusters by users in advance and it may help to find non-convex clusters. After several experiments they concluded that the use of spectral clustering outperforms other clustering methods such as Hasegawa et al. (2004)'s and K-means.

Kim et al. (2014) divide the words in morphemes in a semantic role labeler for Korean, an agglutinative language with a rich morphology. Their SRL system obtained the best SRL performance reported to date for an agglutinative language.

Although the use of word vectors in NLP tasks is relatively new, in the last years some authors have developed interesting ideas that can be used in several research areas. One of the most active authors who has experimented with word vectors is Mikolov (Mikolov et al., 2013c; Mikolov et al., 2013b; Mikolov et al., 2013a). Many different types of models were proposed for estimating continuous representations of words. However, Mikolov et al. focus on distributed representations of words learned by neural networks, as it was previously shown that they perform significantly better than others for preserving linear regularities among words. They proposed two new model architectures for learning distributed representations of words that try to minimize computational complexity: Continuous Bag-of-Words Model (CBOW) and Continuous Skip-gram Model (Mikolov et al., 2013a). The major difference between them it is that the CBOW architecture tries to predict the current word based on the context whereas the Skip-gram architecture tries to maximize the classification of a word based on another word in the same sentence. More precisely, it uses each current word as an input to a log-linear classifier with a continuous projection layer, and predicts words within a certain range before and after the current word. Including these two models, the neural network is able to create word vectors within minutes.

Finally, Andreas and Klein (2014) investigates a variety of ways in which word embeddings might augment a constituency parser with a discrete state space: by connecting out-of-vocabulary words to known ones, by encouraging common behavior among related in-vocabulary words, and by directly providing features for the lexicon. Their results show that unsupervised word embeddings do contain some syntactically useful information, but this information is redundant with what the model is able to determine by itself from only a small amount of labeled training data. They added that their results for constituency parsing are extremely sensitive to training conditions, and not nearly as accessible as they seem to be in dependency parsers. Thus, they conclude that their results suggest that word embeddings can be useful for dependency parsing because they provide a level of syntactic abstraction which is

| | **Basque** | **French** | **German** | **Hungarian** | **Swedish** |
|---|---|---|---|---|---|
| **Baselines** | | | | | |
| MaltOptimizer | 80.0 | 79.9 | 87.6 | 77.2 | 73.4 |
| Mate | 83.0 | 84.2 | 91.0 | 82.8 | 76.7 |
| **Brown Clusters** | | | | | |
| MaltOptimizer | 80.5 | 80.1 | 87.7 | 77.5 | 73.4 |
| Mate | 83.4 | 84.3 | 91.1 | 82.7 | 77.2 |

Table 1: Testing the effect of morphological Brown clusters on MaltParser and Mate.

explicitly annotated in constituency parses.

## 3 Resources

This section will describe the main resources that have been used in the experiments. Subsection 3.1 will describe the languages we have used in our experiments, subsection 3.2 will explain the parsers we use, while subsection 3.3 will present briefly the *MaltBlender* tool and parser combination. Finally, we include a short description of the method we used to divide the words into lemmas and suffixes in subsection 3.4.

### 3.1 Selected Languages

Although the *SPMRL'2014 Shared Task* (Seddah et al., 2014) offers the opportunity to parse nine morphologically rich languages, to carry out our experiments we have selected five of them. Taking into account that Basque (Aduriz et al., 2003), German (Seeker and Kuhn, 2012) and Hungarian (Vincze et al., 2010) have a rich suffix variety, we decided to apply morphologically-aware clusters to these languages. On the other hand, we employ the *classic* method to create clusters for French (Abeillé et al., 2003) and Swedish (Nivre et al., 2006). In that way, we will be able to compare the results.

### 3.2 Parsers

We have made use of MaltParser (Nivre et al., 2007b) and Mate (Bohnet and Nivre, 2012), two state of the art dependency parsers[2] representing the dominant approaches in data-driven dependency parsing, and that have been successfully applied to typologically different languages and treebanks.
MaltParser is a representative of local, greedy, transition-based dependency parsing models, where the parser obtains deterministically a dependency tree in a single pass over the input using two data structures: a stack of partially analyzed items and the remaining input sequence. To determine the best action at each step, the parser uses history-based feature models and discriminative machine learning. The specification of the learning configuration can include any kind of information (such as word-form, lemma, category, subcategory or morphological features). We will use one of its latest versions (MaltParser version 1.7).

To fine-tune Maltparser we have used MaltOptimizer (Ballesteros and Nivre, 2012a; Ballesteros and Nivre, 2012b). This tool is an interactive system that first performs an analysis of the training set in order to select a suitable starting point for optimization and then guides the user through the optimization of parsing algorithm, feature model, and learning algorithm. Empirical evaluation on data from the CoNLL 2006 and 2007 shared tasks on dependency parsing shows that MaltOptimizer consistently improves over the baseline of default settings and sometimes even surpasses the result of manual optimization.
The Mate parser (Bohnet and Nivre, 2012) is a development of the algorithms described in (Carreras, 2007; Johansson and Nugues, 2008). It basically adopts the second order maximum spanning tree (MST) dependency parsing algorithm. However, second-order non-projective MST parsing is NP-hard, as shown in (McDonald and Pereira, 2006). To solve this Bohnet and Nivre (2012) adopts the approximate algorithm designed by McDonald and Pereira (2006) that is based on the exact *O(*$n^3$*)* second-order projective Eisner algorithm. The approximation works by rst nding the highest scoring projective parse.

[2]Due to time constraints, we did not have enough time to experiment with other options such as the MST parser or the EasyFirst parser.

It then rearranges edges in the tree, one at a time, as long as such rearrangements increase the overall score and do not violate the tree constraint.

### 3.3 Parser Combinations

In order to combine single parsers we have made use of the MaltBlender tool. It was used for the first time for the ten languages in the multilingual track of the CoNLL 2007 shared task on dependency parsing(Hall et al., 2010). Sagae and Lavie (2006) presented a framework for combining the output of several different parsers to produce results that are superior to those of each of the individual parsers. This is done in a two stage process they called *reparsing*. In the first stage, *m* different parsers analyze an input sentence, each producing a syntactic structure. When *m* parsers each output a set of dependencies (forming *m* dependency structures) for a given sentence containing *n* words, the dependencies can be combined in a simple word-by-word voting scheme, where each parser votes for the head of each of the *n* words in the sentence, and the head with most votes is assigned to each word. This very simple scheme guarantees that the final set of dependencies will have as many votes as possible, but it does not guarantee that the final voted set of dependencies will be a well-formed dependency tree. In fact, the resulting graph may not even be connected. Instead, if the sentence based on the output of the *m* parsers is *reparsed*, the number of votes for a well-formed dependency structure can be maximized. Therefore, once the *m* initial dependency structures to be combined are created, the first step is to build a graph where each word in the sentence is a node. Then, weighted directed edges are created between the nodes corresponding to words for which dependencies are obtained from each of the initial structures. In cases where more than one dependency structure indicates that an edge should be created, the corresponding weights are simply added. As long as at least one of the *m* initial structures is a well-formed dependency structure, the directed graph created this way will be connected.

Once this graph is created, in the second stage, a parsing algorithm is applied to the original sentence, taking into account the analyses produced by each parser in the first stage. Finding the optimal dependency structure given the set of weighted dependencies is simply a matter of finding the maximum spanning tree (MST) for the directed weighted graph, which can be done using the Chu-Liu/Edmonds directed MST algorithm (Chu and Liu, 1965; Edmonds, 1967). The maximum spanning tree maximizes the votes for dependencies given the constraint that the resulting structure must be a tree. If projectivity (no crossing branches) is desired, Eisner (1996) dynamic programming algorithm (similar to CYK) for dependency parsing can be used instead.

### 3.4 Morphological Analysis

Taking into account time constraints we have created a very simple method to divide the words into lemmas and suffixes, taking the word form and the lemma of each word and looking if the lemma is included as it is in the word form. In case it is, the word form is divided into lemma and the rest (suffix). Otherwise, it only returns the lemma.

## 4 Applying Morphological Brown Clustering to Parsing

Brown clustering is a well-known clustering method that has been successfully applied to several NLP tasks such as Dependency Parsing (Haffari et al., 2011; Koo et al., 2008), Named Entity Recognition (Turian et al., 2010) or Question Answering (Momtazi and Klakow, 2009). This algorithm (Brown et al., 1992) is a bottom-up agglomerative clustering algorithm which classifies each word into a cluster. The primary motivation of this co-occurrence based algorithm is to learn the class based language model. Since this technique is co-occurrence based, it is able to extract the classes that have the flavor of either syntactically based groupings or semantically based grouping, depending on the nature of the underlying statistics. In order to provide a wider introduction of the algorithm, we will describe it briefly below.

When the algorithm starts working, each word in the vocabulary is considered to be in its own distinct cluster. After that, the algorithm repeatedly merges the pair of clusters which causes the smallest decrease in the likelihood of the text corpus, according to a class-based bi-gram language model defined on the word clusters. Following the pairwise merge operations, we can obtain a hierarchical clustering of the words, which can be represented as a binary tree. Within this tree, each word has a unique bit string that is easily identifiable following its path from the root. When the clusters (bit strings) are created by the algorithm we can play with their bit longitude if we want to provide different generalization degrees to the parsers. Koo et al. (2008) use different prefixes of the Brown cluster hierarchy to produce clusterings of varying granularity. They noticed that it is an important decision to select the proper prefix lengths for the dependency parsing task because after using the prefix lengths proposed in the Miller et al. (2004) work (between 12 and 20 bits) they obtained poor results. After experimenting with many different bit-string configurations, they decided to use short bit-string prefixes (e.g., 4-6 bits). According to these suggestions we decided to try our system using three bit-string prefixes (4, 5 and 6 bits) with the goal of determining which configuration is the best. After several experiments we decided that the most suitable prefix longitude was of 4 bits, being this longitude which provides the highest level of generalisation among the three bit-strings.

Although Brown clusters derived from words without any separation between lemmas and suffixes can bring us good new features we can use to make better parsers, our intention is to go a little further. Instead of taking the words as they are, we have divided them in lemmas and suffixes with the aim of avoiding the sparsity that can appear in some morphologically rich languages such as Basque or Hungarian. We describe our methodology to create new features based on Brown clusters below.

Before applying the Brown clustering algorithm to the words contained in the corpus we divide them in lemmas and suffixes. If we take this example for Basque:

*[Donostira] [noa] [gaur gauean].*

*[to Donostia][I'm going][tonight].*

We would apply Brown clustering algorithm after converting the text above in this one:

*[Donostia] [ra] [noa] [gaur] [gau] [ean].*

*[Donostia] [to] [I'm going] [today] [night] [at].*

When the algorithm finishes all the lemmas and suffixes are classified in many different clusters (we have created 800 clusters for each language). However, we noticed that many suffixes are classified in the same cluster. Therefore, in order to create features derived from suffixes we decided to use the suffixes instead of bit-strings provided by Brown clusters. Following with the previous example, let's imagine the algorithm assigns the following clusters for each element:

*Donostia [10100] ra [1000] noa [10000000] gaur [1010101] gau [111111] ean [1000].*

Having said that, we have created three new features for each word we have to analyze syntactically. If we would find the word *Donostiara* in our sentences to parse, we would create the following features:

- 4 bit-string prefix of the lemma: ClusterS (Cluster short) = 1010
- Full bit-string of the lemma: ClusterF (Cluster full) = 10100
- The suffix of the entire word: Suffix = ra

We have to say we have applied Morphological Brown Clustering only to Basque, German and Hungarian. For French and Swedish we have used the *classic* Brown clustering creating two new features for each word (ClusterS and ClusterF) instead of creating three. In addition, we have made use of a maximum 50 million words corpora for each language, due to the fact that the Brown clustering algorithm is time consuming. For all of the experiments in this section, we used Liang (2005)'s implementation of the Brown algorithm to obtain the necessary word clusters.

In table 1 we present the results of MaltOptimizer and Mate after including the new features.

If we analyze the table, we notice that there is not a regular pattern in the results. Sometimes MaltOptimizer performs better using the new features while other times Mate does better. Among the used languages, Basque language has been the one that has obtained the biggest improvements. Using MaltOptimizer we have improved the baseline in 0.5 and using Mate in 0.4. It seems that Morphological Brown Clustering is a suitable method for Basque due to its rich suffix variety.

| | **Basque** | **French** | **German** | **Hungarian** | **Swedish** |
|---|---|---|---|---|---|
| **Baselines** | | | | | |
| MaltOptimizer | 80.0 | 79.9 | 87.6 | 77.2 | 73.4 |
| Mate | 83.0 | 84.2 | 91.0 | 82.8 | 76.7 |
| **K-means Clusters** | | | | | |
| MaltOptimizer | 80.1 | 79.9 | 87.7 | 77.4 | 73.4 |
| Mate | 82.9 | 84.2 | 91.1 | 82.6 | 77.2 |

Table 2: Testing the effect of K-means clusters on MaltParser and Mate.

| | **Basque** | **French** | **German** | **Hungarian** | **Swedish** |
|---|---|---|---|---|---|
| **MaltOptimizer baseline** | 80.0 | 79.9 | 87.6 | 77.2 | 73.4 |
| **Mate parser baseline** | 83.0 | 84.2 | 91.0 | 82.8 | 76.7 |
| **Parser combination** | 84.9 | 84.8 | 91.7 | 83.8 | 77.9 |

Table 3: Results of parser combinations

On the other hand, we have achieved an improvement of 0.5 for Swedish using Mate whereas we haven't improved the baseline using MaltOptimizer. For French we have obtained small increases of 0.2 (MaltOptimizer) and 0.1 (Mate), similar to German; an improvement of 0.1 using both parsers. Finally, if we observe the Hungarian results, we see an improvement of 0.3 using MaltOptimizer and a negative result using Mate (-0.1).

## 5 Applying Clusters of Morphological Word Embeddings

As Mikolov et al. (2013b) proposed, using word embeddings can find interesting semantic and syntactic similarities between the words in an easy way. We have made some experiments with a 150 million words corpus for Basque in order to validate the utility of the word vectors. For example, when we looked for the nearest (cosine distance) words to *Bilbao* it was surprising that the 3 nearest words were cities of the Basque country. In addition, the 40 nearest words were cities of the world. Following this way, we repeat the experiment with the word *aitaren* (father's) and we obtained an encouraging result: not only most of the nearest words were related to the word *aitaren* such as *amaren* (mother's), *arrebaren* (sister's), *anaiaren* (brother's), *amonaren* (grandmother's) and *aitonaren* (grandfather's) but most of the 40 nearest words had the same suffix (*-ren*).

Following these ideas, our intention is to provide a wider semantic and syntactic perspective to the parser introducing features derived from word embeddings. As we have done in the previous section, we decided to divide the words in lemmas and suffixes to avoid the sparsity present in some morphologically rich languages. In our experiments we have employed this methodology for Basque, German and Hungarian. On the other hand, we have applied the classic method for French and Swedish. In that way we will be able to compare the results of each methodology and measure their contribution to parsing.

Once we had the word vectors (we have used the unlabeled data derived from the automatically annotated data provided at the *SPMRL'2014 Shared Task*), we had to decide how to employ this information with the purpose of improving dependency parsing results. Therefore, we decided to apply a clustering algorithm (K-means) over the word vectors as an easy and effective way to introduce the resultant clusters (500 clusters for each language) to the parsers. For all of the experiments in this section, we used *word2vec* tool (Mikolov et al., 2013a) to obtain the necessary word clusters.

We have created two new features for each word in order to add them to the FEATS column of the parsers. Let's imagine we find the word *haientzako* (for them) in our sentences. We would create these two features:

- The cluster number of the word *haiek* (they). For example: ClusterWV (Cluster word vector) = 12
- The suffix of the entire word: Suffix = tzako

We have to say we have created these two features for Basque, German and Hungarian. For French and Swedish we have created only one new feature for each word (ClusterWV). Table 2 shows the results

|  | **Basque** | **French** | **German** | **Hungarian** | **Swedish** |
|---|---|---|---|---|---|
| **Basic Parser Combination** | 83.4 | 84.3 | 91.0 | 82.9 | 75.9 |
| **Enriched Parser Combination** | 84.9 | 84.8 | 91.7 | 83.8 | 77.9 |

Table 4: Comparative between basic and enriched parser combinations.

we have achieved with this approach. If we analyze the results on Basque, French and German, we can see we have not obtained any noticeable increase (maximum +0.1). On the other hand, we have improved the baseline for Hungarian in 0.2 using MaltOptimizer whereas using Mate we have achieved negative results. Finally, Swedish seems to be the most receptive language for this kind of information, improving the baseline in 0.5 using Mate.

## 6 Parser Combinations

Although in several cases the use of clusters does not give noticeable increases, we also tested their effect on parser combinations. Table 3 presents the result of combining the extended parsers with respect to the baselines (using all the features) obtained in individual parsers. The table shows that the Basque language has achieved the biggest increase. Parser combination in Basque helps with improvements of 1.9 and 4.9 with respect to the Mate and Maltparser baselines. Contrary to Basque, French is the language that has obtained the smallest increases in parser combination if we compare it with the Mate (highest) parser baseline. The combined system improves the Maltparser baseline by 4.9 and the Mate parser baseline by 0.6. Parser combination in German gives a 0.7 increase with respect to the best single parser (Mate, 91.02). Our system achieves a 6.6 increase for Hungarian with respect to Maltparser's baseline, while it improves the Mate parser's baseline by 1.0. Finally, if we focus on Swedish, the parser combination helps with a 4.5 increase respect to Maltparser and with a 1.2 respect to the Mate parser.

After examining the results, we consider interesting to include a comparative between parser combinations of basic parsers (MaltOptimizer's and Mate's baselines) and those we have presented above in order to measure the real contribution of the used clusters to parsing. Table 4 shows that Swedish is the most sensitive language to the proposed clusters (improvement of 2 points) whereas French language is the less sensitive (+0.5). Our system achieves a 1.5 increase for Basque with respect to basic parser combination, being this language the second language most sensitive between the five selected. On the other hand, we have achieved an improvement of 0.7 respect to basic parser combination for German. Even though this last result for German does not seem to be a noticeable increase, we have to take into account German baseline is very high (91.0) and beating this score is quite difficult. Furthermore, if we observe Hungarian results, we notice that our system improves the results obtained by the basic parsers in 0.9.

To summarize, we can say that the presented results suggest that the introduced variants contribute positively on parsing and they help to improve the scores obtained by the base parsers.

## 7 Conclusion and Future Work

We have presented a combined system that employs two different clustering approaches and different methods to create word clusters (*classic* vs lemma+suffix) in order to take advantage of the effect of these clusters on some parsers and languages. In general the improvements have been noticeable, specially for Basque and Swedish. We can point out some interesting avenues for research:

- Including new parsers with different levels of generalization derived from Brown clusters in order to enrich parser combinations. It would be interesting to include al least three levels of generalization, for example using 2, 4 and 6 bit-string prefixes.
- Experimenting different models for parser combination using new parsers. Several of the parser variants we have used give only slight modifications over the base algorithms, even though when combined they give significant increases. Widening the spectrum of parsers and adding new algorithms can imply an important boost in parser combination.

- Application to the rest of the languages of the *SPMRL 2014 Shared Task*: Korean, Hebrew, Arabic and Polish.
- Including new clustering approaches.

## Acknowledgements

This research was supported by the Department of Industry of the Basque Government (IT344-10, S PE11UN114), the University of the Basque Country (GIU09/19) and the Spanish Ministry of Science and Innovation (MICINN, TIN2010-20218).

## References

Anne Abeillé, Lionel Clément, and François Toussenel. 2003. Building a treebank for french. In Anne Abeillé, editor, *Treebanks*. Kluwer, Dordrecht.

I. Aduriz, M. J. Aranzabe, J. M. Arriola, A. Atutxa, A. Díaz de Ilarraza, A. Garmendia, and M. Oronoz. 2003. Construction of a Basque dependency treebank. pages 201–204.

Jacob Andreas and Dan Klein. 2014. How much do word embeddings encode about syntax? In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 822–827, Baltimore, Maryland, June. Association for Computational Linguistics.

Miguel Ballesteros and Joakim Nivre. 2012a. Maltoptimizer: A system for maltparser optimization. In *LREC*, pages 2757–2763.

Miguel Ballesteros and Joakim Nivre. 2012b. Maltoptimizer: an optimization tool for maltparser. In *Proceedings of the Demonstrations at the 13th Conference of the European Chaptr of the Association for Computational Linguistics*, pages 58–62. Association for Computational Linguistics.

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1455–1465. Association for Computational Linguistics.

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. *Comput. Linguist.*, 18(4):467–479, December.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *EMNLP-CoNLL*, pages 957–961.

Jinxiu Chen, Donghong Ji, Chew Lim Tan, and Zhengyu Niu. 2006. Unsupervised relation disambiguation using spectral clustering. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 89–96. Association for Computational Linguistics.

Yoeng-Jin Chu and Tseng-Hong Liu. 1965. On shortest arborescence of a directed graph. *Scientia Sinica*, 14(10):1396.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 71(4):233–240.

Jason M Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th conference on Computational linguistics*, pages 340–345. Association for Computational Linguistics.

Iakes Goenaga, Koldo Gojenola, and Nerea Ezeiza. 2013. Exploiting the contribution of morphological information to parsing: the basque team system in the sprml2013 shared task. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 61–67.

Gholamreza Haffari, Marzieh Razavi, and Anoop Sarkar. 2011. An ensemble model that combines syntactic and semantic clustering for discriminative dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 710–714. Association for Computational Linguistics.

Johan Hall, Jens Nilsson, and Joakim Nivre. 2010. Single malt or blended? a study in multilingual parser optimization. In *Trends in Parsing Technology*, pages 19–33. Springer.

John A Hartigan and Manchek A Wong. 1979. Algorithm as 136: A k-means clustering algorithm. *Applied statistics*, pages 100–108.

Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. 2004. Discovering relations among named entities from large corpora. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 415. Association for Computational Linguistics.

Richard Johansson and Pierre Nugues. 2008. Dependency-based syntactic-semantic analysis with propbank and nombank. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 183–187. Association for Computational Linguistics.

Young-Bum Kim, Heemoon Chae, Benjamin Snyder, and Yu-Seop Kim. 2014. Training a korean srl system with rich morphological features.

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

Percy Liang. 2005. *Semi-supervised learning for natural language*. Ph.D. thesis, Massachusetts Institute of Technology.

Ryan T McDonald and Fernando CN Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *EACL*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751. Citeseer.

Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In *Proceedings of HLT*, pages 337–342.

Saeedeh Momtazi and Dietrich Klakow. 2009. A word clustering approach for language model-based sentence retrieval in question answering systems. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1911–1914. ACM.

Joakim Nivre, Jens Nilsson, and Johan Hall. 2006. Talbanken05: A Swedish treebank with phrase structure and dependency annotation. In *Proceedings of LREC*, pages 1392–1395, Genoa, Italy.

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

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL*.

Djamé Seddah, Reut Tsarfaty, and Israel Rehovot. 2014. Introducing the spmrl 2014 shared task on parsing morphologically-rich languages. *SPMRL-SANCL 2014*, page 103.

Wolfgang Seeker and Jonas Kuhn. 2012. Making Ellipses Explicit in Dependency Conversion for a German Treebank. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pages 3132–3139, Istanbul, Turkey. European Language Resources Association (ELRA).

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

Veronika Vincze, Dóra Szauter, Attila Almási, György Móra, Zoltán Alexin, and János Csirik. 2010. Hungarian dependency treebank. In *LREC*.