The AI-KU System at the SPMRL 2013 Shared Task : Unsupervised Features for Dependency Parsing

Volkan Cirik **Husnu Sensoy**
Artificial Intelligence Laboratory
Koç University, İstanbul, Turkey
{vcirik,hsensoy}@ku.edu.tr

Abstract

We propose the use of the word categories and embeddings induced from raw text as auxiliary features in dependency parsing. To induce word features, we make use of contextual, morphologic and orthographic properties of the words. To exploit the contextual information, we make use of substitute words, the most likely substitutes for target words, generated by using a statistical language model. We generate morphologic and orthographic properties of word types in an unsupervised manner. We use a co-occurrence model with these properties to embed words onto a 25-dimensional unit sphere. The AI-KU system shows improvements for some of the languages it is trained on for the first Shared Task of Statistical Parsing of Morphologically Rich Languages.

1 Introduction

For the first shared task of Workshop on Statistical Parsing of Morphologically Rich Languages (Seddah et al., 2013), we propose to use unsupervised features as auxillary features for dependency parsing.

We induce the unsupervised features using contextual, morphological and orthographic properties of the words. We use possible substitutes of the target word which are generated by a statistical language model to exploit the contextual information. We induce morphological features with a HMM-based model (Creutz and Lagus, 2005). We combine contextual, morphological and orthographic features of co-occurring words within the co-occurrence data embedding framework (Maron et al., 2010). The framework embeds word types sharing similar context, morphological and orthographic properties closely on a 25-dimensional sphere. Thus, it provides the word embeddings on a 25 dimensional sphere. We conduct experiments using these word embeddings with MaltParser (Nivre et al., 2007) and MaltOptimizer (Ballesteros and Nivre, 2012). In addition to CONLL features (Buchholz and Marsi, 2006a), they are added as additional features and the parsers are configured such that they are able to exploit these additional features. As a first step we use real valued word embeddings as they are. Secondly, we discretize the real valued word embeddings. Finally, we cluster them and find fine-grained word categories for word types.

Our experiments show that, the AI-KU system leads to better results than the baseline experiments for some languages. We claim that with the correct parameter settings, these unsupervised features could be useful for dependency parsing.

In the following sections, we introduce the related work, the algorithm, experiments, results and provide a conclusion.

2 Related Work

The features extracted from unlabeled corpora are already used for all major NLP tasks. Early studies mainly use clustering based representations (especially Brown clustering (Brown et al., 1992)) to obtain those features. Miller et al. (2004; Freitag (2004) utilized Brown Clusters to improve Named Entity Recognition (NER) performance whereas Biemann et al. (2007) used them for NER, Word Sense Disambiguation(WSD), and chunking. Ushioda (1996) extended Brown Clustering to cluster

Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically Rich Languages, pages 78–85,
Seattle, Washington, USA, 18 October 2013.

not only words but also phrases using hierarcical clustering and uses them to improve supervised part-of-speech (PoS) tagging. More recently, Brown Clusters are used for Chinese word segmentation and NER (Liang, 2005).

Just like other tasks, clustering based representations are used to improve parser performance. Koo et al. (2008; Suzuki et al. (2009) improved dependency parsing by using Brown clusters. While Candito and Seddah (2010; Candito and Crabbé (2009) improved PCFG parsing by using them and Goldberg et al. (2009) improved PCFG parser for Hebrew by using HMM generated features. More recently Socher et al. (2010) used word embeddings computed using method explained in (Collobert and Weston, 2008) for syntactic parsing.

3 Algorithm

In this section, the general flow of the algorithm will be presented. First, we explain how we generate the substitute vectors. Then, we explain the induction procedure of morphological features. In the following subsection, we explain how we use substitute vectors and morphological features and generate word embeddings. The same flow is followed for all languages we work on.

3.1 Substitute Vectors

A target word's substitute vector is represented by the vocabulary of words and their corresponding probabilities of occurring in the position of the target word.

> (1) " Nobody **thought** you could just inject DNA into someone 's body and they would just suck it up."

Probability	Substitute Word
0.123	thought
0.091	knew
0.064	felt
0.062	said
0.052	believed
0.037	wish

Table 1: Substitute Vector for "thought" in above sentence.

Table 1 illustrates the substitute vector of "thought" in (1). There is a row for each word in the vocabulary. For instance, probability of "knew" occurring in the position of "thought" is 9.1% in this context.

To calculate these probabilities, as described in (Yatbaz et al., 2012), a 4-gram language model is built with SRILM (Stolcke, 2002) on the corpora of the target languages. For French, Hungarian, Polish and Swedish we used Europarl Corpus[1](Koehn, 2005). For German, CONLL-X German Corpus is used (Buchholz and Marsi, 2006b). For Hebrew, we combined HaAretz and Arutz 7 corpora of MILA[2](Itai and Wintner, 2008). For the tokens seen less than 5 times we replace them with an unknown tag to handle unseen words in training and test data. We should note that these corpora are not provided to the other participants.

To estimate probabilities of lexical substitutes, for every token in our datasets, we use three tokens each on the left and the right side of the token as a context. Using Fastsubs (Yuret, 2012) we generated top 100 most likely substitute words. Top 100 substitute probabilities are then normalized to represent a proper probability distribution.

We should emphasize that a substitute vector is a function of the context and does not depend on the target word.

3.2 Morphological Features

In order to generate unsupervised word features, the second set of features that we use are morphological and orthographic features.

The orthographic feature set used is similar to the one defined in (Berg-Kirkpatrick et al.,2010)

INITIAL-CAP	Capitalized words with the exception of sentence initial words.
NUMBER	The token starts with a digit.
CONTAINS-HYPHEN	Lowercase words with an internal hyphen.
INITIAL-APOSTROPHE	Tokens that start with an apostrophe.

The morpological features are obtained using the unlabeled corpora that are used for the generation

[1]http://www.statmt.org/europarl/

[2]http://www.mila.cs.technion.ac.il

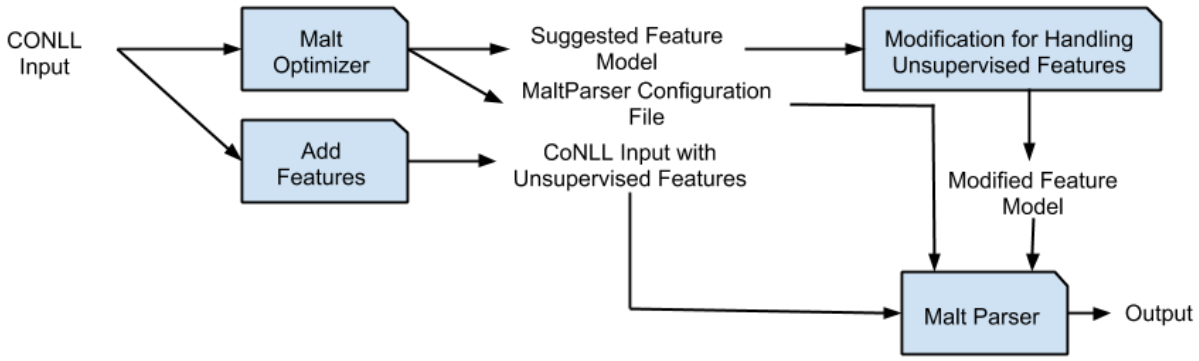

Figure 1: The Flow of The Modification for Handling New Features

of substitute vectors, using Morfessor defined in (Creutz and Lagus, 2005). We will only give a brief sketch of the model used. Morfessor splits each word into morphemes (word itself may also be a morpheme) which can be categorized under four groups, namely prefix, stem, suffix, non-morpheme. The model is defined as *a maximum a posteriori* (MAP) estimate which maximizes the lexicon (set of morphemes) over the corpus.

The maximization problem is solved by using a greedy algorithm that iteratively splits and merges morphemes, then re-segments corpus using Viterbi algorithm and reestimates probabilities until convergence. Finally, a final merge step takes place to remove all non-morphemes.

3.3 Co-occurence Embedding

For a pair of categorical variables, the Spherical Co-occurrence Data Embedding (S-CODE) framework (Maron et al., 2010) represents each of their values on a sphere such that frequently co-occurring values are positioned closely on this sphere.

The input of S-CODE are tuples of values of categorical variables. In our case, these are word tokens, their substitutes, morphological and orthograpic features. We construct the tuples by sampling substitute words using substitute vectors, their corresponding morphological and orthographic features of the tokens. On each row of the co-occurrence input, there are the target token, its substitute sampled from its substitute vector, morphological and orthographic features. Tokens having the similar substitutes, morphological and orthographic features will be closely located on the sphere at the end of this process. As in (Yatbaz et al., 2012), the dimension of the sphere is 25, in other words for each word type seen in the corpora we have a 25 dimensional vector[3].

4 Experiments

We conduct experiments using MaltParser (Nivre et al., 2007) and MaltOptimizer (Ballesteros and Nivre, 2012) with features provided in CONLL format and the additional unsupervised features that we generated with default settings of the parsers. To make use of additional features, we need to modify MaltParser accordingly. Figure 1 shows that how we use MaltOptimizer and MaltParser with new features. In order to handle auxiliary features, the feature model file is modified in two different ways. We handle new features with feature functions Input[0] and Stack[0][4]. We should note that other feature functions should also be experimented as a future work.

The following subsections explain the details of the experiments.

4.1 Experiment I

Our first approach was trying to use word embeddings as they are with the MaltParser. For each token in the training and the test set, we added the corresponding 25-dimensional word vector from the word embeddings file to the training and test sets. If the word type is not present in the word embeddings, then, we use the unknown word vector.

[3]The vectors can be downloaded here : https://github.com/wolet/sprml13-word-embeddings

[4]Thanks for Joakim Nivre for his suggestions on this

	Stack[0]			Input[0]		
	LAS	UAS	LaA	LAS	UAS	Labeled Accuracy
Real Valued Vectors	80.56	84.33	85.78	80.63	84.38	85.92
Binning, b=5	80.25	84.07	85.58	80.45	84.20	85.79
Binning, b=2	80.41	84.19	85.79	80.47	84.26	85.77
Clustering, k = 50	80.48	84.29	85.79	80.50	84.24	85.78
Clustering k = 300	80.49	84.23	85.83	80.58	84.31	85.82

	LAS	UAS	LaS
Baseline	80.36	84.11	85.72

Table 2: Results on German with MaltParser of Development Set with Default Settings

	Stack[0]			Input[0]		
	LAS	UAS	LaS	LAS	UAS	LaS
Real Valued Vectors	87.30	89.33	93.35	87.29	89.30	93.32
Binning, b =2	87.12	89.20	93.20	87.04	89.11	93.16
Clustering, k = 300	90.30	91.80	95.09	90.49	91.94	95.19

	LAS	UAS	LaS
Baseline	90.38	91.88	95.14

Table 3: Results on German with MaltOptimizer of Development Set

	Gold			Predicted		
	LAS	UAS	LaS	LAS	UAS	LaS
Best System	90.29	91.92	95.95	85.86	89.19	92.20
AI-KU 1	86.39	88.21	94.07	72.57	78.54	82.39
AI-KU 2	86.31	88.14	94.05	72.55	78.55	82.36
Baseline	85.71	87.50	93.70	79.00	83.35	87.73

	Predicted (Unofficial)		
	LAS	UAS	LaS
AI-KU 1	79.92	83.94	88.51
AI-KU 2	79.84	83.85	88.45

Table 4: Results on French

	Gold			Predicted		
	LAS	UAS	LaS	LAS	UAS	LaS
Best System	91.83	93.20	96.06	86.95	91.64	94.38
AI-KU 1	86.98	88.71	93.70	82.32	85.31	89.95
AI-KU 2	86.95	88.67	93.67	82.29	85.30	89.95
Baseline	86.96	87.67	93.67	82.75	85.38	90.15

	Predicted (Unofficial)		
	LAS	UAS	LaS
AI-KU 1	84.08	86.71	91.13
AI-KU 2	83.93	86.54	91.05

Table 5: Results on German

	Gold			Predicted		
	LAS	UAS	LaS	LAS	UAS	LaS
Best System	83.87	88.95	89.19	80.89	86.7	86.93
AI-KU 1	79.42	84.48	86.52	69.01	75.84	79.01
AI-KU 2	78.73	83.79	85.98	62.27	75.84	79.01
Baseline	80.03	84.9	86.97	73.01	79.89	81.28

Table 6: Results on Hebrew

	Gold			Predicted		
	LAS	UAS	LaS	LAS	UAS	LaS
Best System	88.06	91.14	92.58	86.13	89.81	90.92
AI-KU 1	83.67	87.08	89.64	78.92	83.77	85.98
AI-KU 2	83.63	87.06	89.58	78.76	83.60	85.95
Baseline	83.14	86.56	89.20	79.63	83.71	85.89

	Predicted (Unofficial)		
	LAS	UAS	LaS
AI-KU 1	79.98	84.42	87.12
AI-KU 2	79.74	84.12	86.93

Table 7: Results on Hungarian

	Gold			Predicted		
	LAS	UAS	LaS	LAS	UAS	LaS
Best System	89.58	93.24	93.42	87.07	91.75	91.24
AI-KU 1	85.16	88.86	90.87	81.86	86.96	88.06
AI-KU 2	85.12	88.79	90.84	78.31	84.18	85.64
Baseline	80.49	86.41	86.94	79.89	85.80	86.24

Table 8: Results on Polish

	Gold			Predicted		
	LAS	UAS	LaS	LAS	UAS	LaS
Best System	83.97	89.11	87.63	82.13	88.06	85.93
AI-KU 1	78.87	85.19	83.44	76.35	83.30	81.37
AI-KU 2	78.57	85.12	83.25	76.35	83.24	81.35
Baseline	77.67	84.6	82.36	75.82	83.20	80.88

Table 9: Results on Swedish

	Gold			Predicted		
	Precision	Recall	F1	Precision	Recal	F1
Best System	99.41	99.38	99.39	81.68	79.97	80.81
AI-KU 1	99.41	99.38	99.39	74.47	71.51	72.96
AI-KU 2	99.38	99.36	99.37	74.34	71.51	72.89
MaltOptimizer Baseline	98.77	99.18	99.26	72.64	68.09	70.29

Table 10: Results of Multi Word Expressions on French

4.2 Experiment II

The second approach is discretizing the real valued vectors. For each dimension of word embeddings, we separate b equal sized bins. Then, for each vector's dimensions, we assign their corresponding bin numbers.

4.3 Experiment III

The third approach is clustering the word embeddings. We use a modified k-means algorithm (Arthur and Vassilvitskii, 2007). We experiment with varying number of clusters k.

For each token in training and test file, we use word type's cluster id as an auxiliary feature. Again, if the token is not in the word embeddings file, we used the unknown word's cluster id.

5 Results

In Table 2, the experiments on German with MaltParser without the optimization step are demonstrated. We use the default settings of the MaltParser as our baseline. We use training data consisting of 5000 sentences with gold tags as training set and the provided development data as test set.

When we use real valued word embeddings as an auxiliary feature, we observe slight improvement compared to MaltParser baseline. The large binning size results in worse results compared to baseline due to sparsity. Clustering again leads to some improvement compared to MaltParser baseline. We also observe that increasing the number of clusters result in better scores compared to smaller k.

In Table 3, the results on German with MaltOptimizer can be seen. As a baseline, again, we use training data consisting of 5000 sentences with gold tags as training set and the provided development data as test set. We use the baseline experiment's parsing algorithm, feature model and learning algorithm to experiment with word embedding, binning and clustering on MaltParser.

Unlike in Table 2, in Table 3 we observe that only the clustering experiment outperforms the baseline but not significantly. Since clustering is leads to best results, for all other languages, we apply the same optimization and clustering pipeline. The only difference is that when the MaltOptimizer suggests Stack Projective as the best algorithm, instead of Input[0] ve use Stack[0], Stack[1], Stack[2] as feature functions. The two systems of AI-KU only differ in these feature functions.

In Table 3-7, the results of the best system, baseline MaltOptimizer result and our two submitted systems can be seen. For Polish, our system outperfoms the MaltOptimizer baseline significantly. For the rest of the languages, our systems are not significantly better or worse than the baseline. We make an assumption that we need to find the optimum settings, for instance the number of clusters, for each language separately, instead of using the fixed settings for all languages.

For French, German, Hungarian the model trained on the data with gold features is mistakenly used for testing on the data with predicted features. To correct these, for those languages, we report the unofficial results that are obtained by training on predicted features.

For French, there is also another evaluation metric. It is about capturing the Multi Word Expressions(MWE). Table 10 reports the results of MWE and it shows that our system is significantly better than MaltOptimizer baseline.

6 Conclusion

We can speculate on these results in couple of ways. First, for all languages we used the same number of clusters. The optimum number of clusters may vary with the syntactic properties of these languages. Similarly, the optimum dimension of the word embeddings may vary with the languages. In addition, for co-occurence embedding and morphological induction we use the parameter settings of (Yatbaz et al., 2012) which is optimized for Part-of-Speech induction on Penn Treebank data. We suggest to find the optimum parameter settings for co-occurrence embedding and morphological induction as a future work.

We only experimented with simple feature functions, namely Input and Stack functions. Other configuration of these functions may lead to better results. Lastly, as a future direction, we propose to use real valued word embeddings and unsupervised word categories as auxiliary features in the training phase of the MaltOptimizer.

Acknowledgments

We would like to thank Joakim Nivre and Deniz Yuret for valuable suggestions and their support.

References

D. Arthur and S. Vassilvitskii. 2007. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics.

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

Chris Biemann, Claudio Giuliano, and Alfio Gliozzo. 2007. Unsupervised part-of-speech tagging supporting supervised methods. In *Proceedings of RANLP*, volume 7.

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.

S. Buchholz and E. Marsi. 2006a. CoNLL-X shared task on multilingual dependency parsing. SIGNLL.

Sabine Buchholz and Erwin Marsi. 2006b. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, pages 149–164, Stroudsburg, PA, USA. Association for Computational Linguistics.

Marie Candito and Benoît Crabbé. 2009. Improving generative statistical parsing with semi-supervised word clustering. In *Proceedings of the 11th International Conference on Parsing Technologies*, pages 138–141. Association for Computational Linguistics.

Marie Candito and Djamé Seddah. 2010. Parsing word clusters. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 76–84. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Mathias Creutz and Krista Lagus. 2005. Inducing the morphological lexicon of a natural language from unannotated text. In *Proceedings of AKRR'05, International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning*, pages 106–113, Espoo, Finland, June.

Dayne Freitag. 2004. Trained named entity recognition using distributional clusters. *In Proceedings of EMNLP*, pages 262–269.

Yoav Goldberg, Reut Tsarfaty, Meni Adler, and Michael Elhadad. 2009. Enhancing unlexicalized parsing performance using a wide coverage lexicon, fuzzy tag-set mapping, and em-hmm-based lexical probabilities. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 327–335. Association for Computational Linguistics.

Alon Itai and Shuly Wintner. 2008. Language resources for Hebrew. *Language Resources and Evaluation*, 42(1):75–98, March.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. Columbus, Ohio USA, June. ACL.

Percy Liang. 2005. Semi-supervised learning for natural language. Master's thesis, MIT, May.

Yariv Maron, Michael Lamar, and Elie Bienenstock. 2010. Sphere embedding: An application to part-of-speech induction. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1567–1575.

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

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

Djame Seddah, Reut Tsarfaty, Sandra Kubler, Marie Candito, Jinho Choi, Richard Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiorkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Wolinski, Alina Wroblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the spmrl 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the 4th Workshop on Statistical Parsing of Morphologically Rich Languages: Shared Task*, Seattle, WA.

Richard Socher, Christopher D Manning, and Andrew Y Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*.

Andreas Stolcke. 2002. Srilm-an extensible language modeling toolkit. In *Proceedings International Conference on Spoken Language Processing*, pages 257–286, November.

Jun Suzuki, 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: Volume 2-Volume 2*, pages 551–560. Association for Computational Linguistics.

Akira Ushioda. 1996. Hierarchical clustering of words and applications to nlp tasks. *In Proceedings of the Fourth Workshop on Very Large Corpora*, pages 28–41.

Mehmet Ali Yatbaz, Enis Sert, and Deniz Yuret. 2012. Learning syntactic categories using paradigmatic representations of word context. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 940–951, Jeju Island, Korea, July. Association for Computational Linguistics.

Deniz Yuret. 2012. Fastsubs: An efficient and exact procedure for finding the most likely lexical substitutes based on an n-gram language model. *Signal Processing Letters, IEEE*, 19(11):725–728, Nov.