Tailoring Continuous Word Representations for Dependency Parsing

Mohit Bansal **Kevin Gimpel** **Karen Livescu**
Toyota Technological Institute at Chicago, IL 60637, USA
{mbansal, kgimpel, klivescu}@ttic.edu

Abstract

Word representations have proven useful for many NLP tasks, e.g., Brown clusters as features in dependency parsing (Koo et al., 2008). In this paper, we investigate the use of *continuous* word representations as features for dependency parsing. We compare several popular embeddings to Brown clusters, via multiple types of features, in both news and web domains. We find that all embeddings yield significant parsing gains, including some recent ones that can be trained in a fraction of the time of others. Explicitly tailoring the representations for the task leads to further improvements. Moreover, an ensemble of all representations achieves the best results, suggesting their complementarity.

1 Introduction

Word representations derived from unlabeled text have proven useful for many NLP tasks, e.g., part-of-speech (POS) tagging (Huang et al., 2014), named entity recognition (Miller et al., 2004), chunking (Turian et al., 2010), and syntactic parsing (Koo et al., 2008; Finkel et al., 2008; Täckström et al., 2012). Most word representations fall into one of two categories. **Discrete** representations consist of memberships in a (possibly hierarchical) hard clustering of words, e.g., via k-means or the Brown et al. (1992) algorithm. **Continuous** representations (or distributed representations or embeddings) consist of low-dimensional, real-valued vectors for each word, typically induced via neural language models (Bengio et al., 2003; Mnih and Hinton, 2007) or spectral methods (Deerwester et al., 1990; Dhillon et al., 2011).

Koo et al. (2008) found improvement on in-domain dependency parsing using features based on discrete Brown clusters. In this paper, we experiment with parsing features derived from continuous representations. We find that simple attempts based on discretization of individual word vector dimensions do not improve parsing. We see gains only after first performing a hierarchical clustering of the continuous word vectors and then using features based on the hierarchy.

We compare several types of continuous representations, including those made available by other researchers (Turian et al., 2010; Collobert et al., 2011; Huang et al., 2012), and embeddings we have trained using the approach of Mikolov et al. (2013a), which is orders of magnitude faster than the others. The representations exhibit different characteristics, which we demonstrate using both intrinsic metrics and extrinsic parsing evaluation. We report significant improvements over our baseline on both the Penn Treebank (PTB; Marcus et al., 1993) and the English Web treebank (Petrov and McDonald, 2012).

While all embeddings yield some parsing improvements, we find larger gains by tailoring them to capture similarity in terms of context within syntactic parses. To this end, we use two simple modifications to the models of Mikolov et al. (2013a): a smaller context window, and conditioning on syntactic context (dependency links and labels). Interestingly, the Brown clusters of Koo et al. (2008) prove to be difficult to beat, but we find that our syntactic tailoring can lead to embeddings that match the parsing performance of Brown (on all test sets) in a fraction of the training time. Finally, a simple parser ensemble on all the representations achieves the best results, suggesting their complementarity for dependency parsing.

2 Continuous Word Representations

There are many ways to train continuous representations; in this paper, we are primarily interested in neural language models (Bengio et al., 2003), which use neural networks and local context to learn word vectors. Several researchers have made their trained representations publicly avail-

Representation	Source	Corpus	Types, Tokens	V	D	Time
BROWN	Koo et al. (2008)	BLLIP	317K, 43M	316,710	–	2.5 days†
SENNA	Collobert et al. (2011)	Wikipedia	8.3M, 1.8B	130,000	50	2 months*
TURIAN	Turian et al. (2010)	RCV1	269K, 37M	268,810	50	few weeks*
HUANG	Huang et al. (2012)	Wikipedia	8.3M, 1.8B	100,232	50	—
CBOW, SKIP, SKIP$_{\text{DEP}}$	Mikolov et al. (2013a)	BLLIP	317K, 43M	316,697	100	2-4 mins.†

Table 1: Details of word representations used, including datasets, vocabulary size V, and dimensionality D. Continuous representations require an additional 4 hours to run hierarchical clustering to generate features (§3.2). RCV1 = Reuters Corpus, Volume 1. $*$ = time reported by authors. $\dagger$ = run by us on a 3.50 GHz desktop, using a single thread.

able, which we use directly in our experiments. In particular, we use the SENNA embeddings of Collobert et al. (2011); the scaled TURIAN embeddings (C&W) of Turian et al. (2010); and the HUANG global-context, single-prototype embeddings of Huang et al. (2012). We also use the BROWN clusters trained by Koo et al. (2008). Details are given in Table 1.

Below, we describe embeddings that we train ourselves (§2.1), aiming to make them more useful for parsing via smaller context windows (§2.1.1) and conditioning on syntactic context (§2.1.2). We then compare the representations using two intrinsic metrics (§2.2).

2.1 Syntactically-tailored Representations

We train word embeddings using the continuous bag-of-words (CBOW) and skip-gram (SKIP) models described in Mikolov et al. (2013a; 2013b) as implemented in the open-source toolkit `word2vec`. These models avoid hidden layers in the neural network and hence can be trained in only minutes, compared to days or even weeks for the others, as shown in Table 1.[1] We adapt these embeddings to be more useful for dependency parsing in two ways, described next.

2.1.1 Smaller Context Windows

The CBOW model learns vectors to predict a word given its set of surrounding context words in a window of size w. The SKIP model learns embeddings to predict each individual surrounding word given one particular word, using an analogous window size w. We find that w affects the embeddings substantially: with large w, words group with others that are topically-related; with small w, grouped words tend to share the same POS tag. We discuss this further in the intrinsic evaluation presented in §2.2.

[1]We train both models on BLLIP (LDC2000T43) with PTB removed, the same corpus used by Koo et al. (2008) to train their BROWN clusters. We created a special vector for unknown words by averaging the vectors for the 50K least frequent words; we did not use this vector for the SKIP$_{\text{DEP}}$ (§2.1.2) setting because it performs slightly better without it.

2.1.2 Syntactic Context

We expect embeddings to help dependency parsing the most when words that have similar parents and children are close in the embedding space. To target this type of similarity, we train the SKIP model on *dependency context* instead of the linear context in raw text. When ordinarily training SKIP embeddings, words v' are drawn from the neighborhood of a target word v, and the sum of log-probabilities of each v' given v is maximized. We propose to instead choose v' from the set containing the grandparent, parent, and children words of v in an automatic dependency parse.

A simple way to implement this idea is to train the original SKIP model on a corpus of dependency links and labels. For this, we parse the BLLIP corpus (minus PTB) using our baseline dependency parser, then build a corpus in which each line contains a single child word c, its parent word p, its grandparent g, and the dependency label ℓ of the $\langle c, p\rangle$ link:

"$\ell_{\text{<L>}}$ $g_{\text{<G>}}$ p c $\ell_{\text{<L>}}$",

that is, both the dependency label and grandparent word are subscripted with a special token to avoid collision with words.[2] We train the SKIP model on this corpus of tuples with window size $w = 1$, denoting the result SKIP$_{\text{DEP}}$. Note that this approach needs a parsed corpus, but there also already exist such resources (Napoles et al., 2012; Goldberg and Orwant, 2013).

2.2 Intrinsic Evaluation of Representations

Short of running end-to-end parsing experiments, how can we choose which representations to use for parsing tasks? Several methods have been proposed for intrinsic evaluation of word representa-

[2]We use a subscript on g so that it will be treated differently from c when considering the context of p. We removed all $g_{\text{<G>}}$ from the vocabulary after training. We also tried adding information about POS tags. This increases M-1 (§2.2), but harms parsing performance, likely because the embeddings become too tag-like. Similar ideas have been used for clustering (Sagae and Gordon, 2009; Haffari et al., 2011; Grave et al., 2013), semantic space models (Padó and Lapata, 2007), and topic modeling (Boyd-Graber and Blei, 2008).

Representation	SIM	M-1
BROWN	–	**89.3**
SENNA	49.8	85.2
TURIAN	29.5	87.2
HUANG	**62.6**	78.1
CBOW, $w = 2$	34.7	84.8
SKIP, $w = 1$	37.8	86.6
SKIP, $w = 2$	43.1	85.8
SKIP, $w = 5$	44.4	81.1
SKIP, $w = 10$	44.6	71.5
SKIP_{DEP}	34.6	88.3

Table 2: Intrinsic evaluation of representations. SIM column has Spearman's $\rho \times 100$ for 353-pair word similarity dataset. M-1 is our unsupervised POS tagging metric. For BROWN, M-1 is simply many-to-one accuracy of the clusters. Best score in each column is bold.

tions; we discuss two here:

Word similarity (SIM): One widely-used evaluation compares distances in the continuous space to human judgments of word similarity using the 353-pair dataset of Finkelstein et al. (2002). We compute cosine similarity between the two vectors in each word pair, then order the word pairs by similarity and compute Spearman's rank correlation coefficient (ρ) with the gold similarities. Embeddings with high ρ capture similarity in terms of paraphrase and topical relationships.

Clustering-based tagging accuracy (M-1): Intuitively, we expect embeddings to help parsing the most if they can tell us when two words are similar *syntactically*. To this end, we use a metric based on unsupervised evaluation of POS taggers. We perform clustering and map each cluster to one POS tag so as to maximize tagging accuracy, where multiple clusters can map to the same tag. We cluster vectors corresponding to the tokens in PTB WSJ sections 00-21.[3]

Table 2 shows these metrics for representations used in this paper. The BROWN clusters have the highest M-1, indicating high cluster purity in terms of POS tags. The HUANG embeddings have the highest SIM score but low M-1, presumably because they were trained with global context, making them more tuned to capture topical similarity. We compare several values for the window size (w) used when training the SKIP embeddings, finding that small w leads to higher M-1 and lower SIM. Table 3 shows examples of clusters obtained by clustering SKIP embeddings of $w = 1$ versus $w = 10$, and we see that the former correspond closely to POS tags, while the latter are much more topically-coherent and contain mixed POS tags.[4] For parsing experiments, we choose $w = 2$ for CBOW and $w = 1$ for SKIP. Finally, our SKIP_{DEP} embeddings, trained with syntactic context and $w = 1$ (§2.1.2), achieve the highest M-1 of all continuous representations. In §4, we will relate these intrinsic metrics to extrinsic parsing performance.

[3]For clustering, we use k-means with $k = 1000$ and initialize by placing centroids on the 1000 most-frequent words.

w	Example clusters
1	[Mr., Mrs., Ms., Prof., ...], [Jeffrey, Dan, Robert, Peter, ...], [Johnson, Collins, Schmidt, Freedman, ...], [Portugal, Iran, Cuba, Ecuador, ...], [CST, 4:30, 9-10:30, CDT, ...], [his, your, her, its, ...], [truly, wildly, politically, financially, ...]
10	[takeoff, altitude, airport, carry-on, airplane, flown, landings, ...], [health-insurance, clinic, physician, doctor, medical, health-care, ...], [financing, equity, investors, firms, stock, fund, market, ...]

Table 3: Example clusters for SKIP embeddings with window size $w = 1$ (syntactic) and $w = 10$ (topical).

3 Dependency Parsing Features

We now discuss the features that we add to our baseline dependency parser (second-order MSTParser; McDonald and Pereira, 2006) based on discrete and continuous representations.

3.1 Brown Cluster Features

We start by replicating the features of Koo et al. (2008) using their BROWN clusters; each word is represented by a 0-1 bit string indicating the path from the root to the leaf in the binary merge tree. We follow Koo et al. in adding cluster versions of the first- and second-order features in MSTParser, using bit string prefixes of the head, argument, sibling, intermediate words, etc., to augment or replace the POS and lexical identity information. We tried various sets of prefix lengths on the development set and found the best setting to use prefixes of length 4, 6, 8, and 12.[5]

3.2 Continuous Representation Features

We tried two kinds of indicator features:

Bucket features: For both parent and child vectors in a potential dependency, we fire one indicator feature per dimension of each embedding

[4]A similar effect, when changing distributional context window sizes, was found by Lin and Wu (2009).

[5]See Koo et al. (2008) for the exact feature templates. They used the full string in place of the length-12 prefixes, but that setting worked slightly worse for us. Note that the baseline parser used by Koo et al. (2008) is different from the second-order MSTParser that we use here; their parser allows grandparent interactions in addition to the sibling interactions in ours. We use their clusters, available at `http://people.csail.mit.edu/maestro/papers/bllip-clusters.gz`.

vector, where the feature consists of the dimension index d and a bucketed version of the embedding value in that dimension, i.e., $bucket_k(E_{vd})$ for word index v and dimension d, where E is the $V \times D$ embedding matrix.[6] We also tried standard conjunction variants of this feature consisting of the bucket values of both the head and argument along with their POS-tag or word information, and the attachment distance and direction.[7]

Cluster bit string features: To take into account all dimensions simultaneously, we perform agglomerative hierarchical clustering of the embedding vectors. We use Ward's minimum variance algorithm (Ward, 1963) for cluster distance and the Euclidean metric for vector distance (via MATLAB's `linkage` function with {*method=ward, metric=euclidean*}). Next, we fire features on the hierarchical clustering bit strings using templates identical to those for BROWN, except that we use longer prefixes as our clustering hierarchies tend to be deeper.[8]

4 Parsing Experiments

Setup: We use the publicly-available MSTParser for all experiments, specifically its second-order projective model.[9] We remove all features that occur only once in the training data. For WSJ parsing, we use the standard train(02-21)/dev(22)/test(23) split and apply the NP bracketing patch by Vadas and Curran (2007). For Web parsing, we still train on WSJ 02-21, but test on the five Web domains (answers, email, newsgroup, reviews, and weblog) of the 'English Web Treebank' (LDC2012T13), splitting each domain in half (in original order) for the development and test sets.[10] For both treebanks, we convert from constituent to dependency format using `pennconverter` (Johansson and Nugues, 2007), and generate POS tags using the MXPOST tagger (Ratnaparkhi, 1996). To evaluate, we use unlabeled attachment score (UAS).[11] We report statistical significance ($p < 0.01$, 100K samples) using the bootstrap test (Efron and Tibshirani, 1994).

System	Dev	Test
Baseline	92.38	91.95
BROWN	93.18	92.69
SENNA (Buckets)	92.64	92.04
SENNA (Bit strings)	92.88	92.30
HUANG (Buckets)	92.44	91.86
HUANG (Bit strings)	92.55	92.36
CBOW (Buckets)	92.57	91.93
CBOW (Bit strings)	93.06	92.53

Table 4: Bucket vs. bit string features (UAS on WSJ).

System	Dev	Test
Baseline	92.38	91.95
BROWN	93.18	**92.69**
SENNA	92.88	92.30
TURIAN	92.84	92.26
HUANG	92.55	92.36
CBOW	93.06	92.53
SKIP	92.94	92.29
SKIP_{DEP}	**93.33**	**92.69**
Ensemble Results		
ALL – BROWN	93.46	92.90
ALL	93.54	92.98

Table 5: Full results with bit string features (UAS on WSJ).

Comparing bucket and bit string features: In Table 4, we find that bucket features based on individual embedding dimensions do not lead to improvements in test accuracy, while bit string features generally do. This is likely because individual embedding dimensions rarely correspond to interpretable or useful distinctions among words, whereas the hierarchical bit strings take into account all dimensions of the representations simultaneously. Their prefixes also naturally define features at multiple levels of granularity.

WSJ results: Table 5 shows our main WSJ results. Although BROWN yields one of the highest individual gains, we also achieve statistically significant gains over the baseline from all embeddings. The CBOW embeddings perform as well as BROWN (i.e., no statistically significant difference) but are orders of magnitude faster to train. Finally, the syntactically-trained SKIP_{DEP} embeddings are statistically indistinguishable from BROWN and CBOW, and significantly better than all other embeddings. This suggests that targeting the similarity captured by syntactic context is useful for dependency parsing.

[6] Our bucketing function $bucket_k(x)$ converts the real value x to its closest multiple of k. We choose a k value of around 1/5th of the embedding's absolute range.

[7] We initially experimented directly with real-valued features (instead of bucketed indicator features) and similar conjunction variants, but these did not perform well.

[8] We use prefixes of length 4, 6, 8, 12, 16, 20, and full-length, again tuned on the development set.

[9] We use the recommended MSTParser settings: training-k:5 iters:10 loss-type:nopunc decode-type:proj

[10] Our setup is different from SANCL 2012 (Petrov and McDonald, 2012) because the exact splits and test data were only available to participants.

[11] We find similar improvements under labeled attachment score (LAS). We ignore punctuation : , " " . in our evaluation (Yamada and Matsumoto, 2003; McDonald et al., 2005).

System	ans	eml	nwg	rev	blog	Avg
Baseline	82.6	81.2	84.3	83.8	85.5	83.5
BROWN	83.4	81.7	**85.2**	84.5	**86.1**	84.2
SENNA	**83.7**	**81.9**	85.0	**85.0**	86.0	**84.3**
TURIAN	83.0	81.5	85.0	84.1	85.7	83.9
HUANG	83.1	81.8	85.1	84.7	85.9	84.1
CBOW	82.9	81.3	**85.2**	83.9	85.8	83.8
SKIP	83.1	81.1	84.7	84.1	85.4	83.7
SKIP_{DEP}	83.3	81.5	**85.2**	84.3	86.0	84.1
Ensemble Results						
ALL–BR	83.9	82.2	85.9	85.0	86.6	84.7
ALL	84.2	82.3	85.9	85.1	86.8	84.9

Table 6: Main UAS test results on Web treebanks. Here, ans=answers, eml=email, nwg=newsgroup, rev=reviews, blog=weblog, BR=BROWN, Avg=Macro-average.

Web results: Table 6 shows our main Web results.[12] Here, we see that the SENNA, BROWN, and SKIP_{DEP} embeddings perform the best on average (and are statistically indistinguishable, except SENNA vs. SKIP_{DEP} on the reviews domain). They yield statistically significant UAS improvements over the baseline across all domains, except weblog for SENNA (narrowly misses significance, p=0.014) and email for SKIP_{DEP}.[13]

Ensemble results: When analyzing errors, we see differences among the representations, e.g., BROWN does better at attaching proper nouns, prepositions, and conjunctions, while CBOW does better on plural common nouns and adverbs. This suggests that the representations might be complementary and could benefit from combination. To test this, we use a simple ensemble parser that chooses the highest voted parent for each argument.[14] As shown in the last two rows of Tables 5 and 6, this leads to substantial gains. The 'ALL – BROWN' ensemble combines votes from all non-BROWN continuous representations, and the 'ALL' ensemble also includes BROWN.

Characteristics of representations: We now relate the intrinsic metrics from §2.2 to parsing performance. The clearest correlation appears when comparing variations of a single model, e.g., for SKIP, the WSJ dev accuracies are 93.33 (SKIP_{DEP}), 92.94 ($w = 1$), 92.86 ($w = 5$), and 92.70 ($w = 10$), which matches the M-1 score order and is the reverse of the SIM score order.

[12] We report individual domain results and macro-average over domains. We do not tune any features/parameters on Web dev sets; we only show the test results for brevity.

[13] Note that SENNA and HUANG are trained on Wikipedia which may explain why they work better on Web parsing as compared to WSJ parsing.

[14] This does not guarantee a valid tree. Combining *features* from representations will allow training to weigh them appropriately and also guarantee a tree.

5 Related Work

In addition to work mentioned above, relevant work that uses discrete representations exists for POS tagging (Ritter et al., 2011; Owoputi et al., 2013), named entity recognition (Ratinov and Roth, 2009), supersense tagging (Grave et al., 2013), grammar induction (Spitkovsky et al., 2011), constituency parsing (Finkel et al., 2008), and dependency parsing (Tratz and Hovy, 2011). Continuous representations in NLP have been evaluated for their ability to capture syntactic and semantic word similarity (Huang et al., 2012; Mikolov et al., 2013a; Mikolov et al., 2013b) and used for tasks like semantic role labeling, part-of-speech tagging, NER, chunking, and sentiment classification (Turian et al., 2010; Collobert et al., 2011; Dhillon et al., 2012; Al-Rfou' et al., 2013).

For dependency parsing, Hisamoto et al. (2013) also used embedding features, but there are several differences between their work and ours. First, they use only one set of pre-trained embeddings (TURIAN) while we compare several and also train our own, tailored to the task. Second, their embedding features are simpler than ours, only using flat (non-hierarchical) cluster IDs and binary strings obtained via sign quantization ($\mathbb{1}[x > 0]$) of the vectors. They also compare to a first-order baseline and only evaluate on the Web treebanks.

Concurrently, Andreas and Klein (2014) investigate the use of embeddings in constituent parsing. There are several differences: we work on dependency parsing, use clustering-based features, and tailor our embeddings to dependency-style syntax; their work additionally studies vocabulary expansion and relating in-vocabulary words via embeddings.

6 Conclusion

We showed that parsing features based on hierarchical bit strings work better than those based on discretized individual embedding values. While the Brown clusters prove to be well-suited to parsing, we are able to match their performance with our SKIP_{DEP} embeddings that train much faster. Finally, we found the various representations to be complementary, enabling a simple ensemble to perform best. Our SKIP_{DEP} embeddings and bit strings are available at `ttic.edu/bansal/data/syntacticEmbeddings.zip`.

Acknowledgments: We would like to thank the anonymous reviewers for their insightful comments.

References

Rami Al-Rfou', Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *Proceedings of CoNLL*.

Jacob Andreas and Dan Klein. 2014. How much do word embeddings encode about syntax? In *Proceedings of ACL*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, March.

Jordan L. Boyd-Graber and David M. Blei. 2008. Syntactic topic models. In *Proceedings of NIPS*.

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.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407.

Paramveer Dhillon, Dean P. Foster, and Lyle H. Ungar. 2011. Multi-view learning of word embeddings via CCA. In *Proceedings of NIPS*.

Paramveer Dhillon, Jordan Rodu, Dean P. Foster, and Lyle H. Ungar. 2012. Two Step CCA: A new spectral method for estimating vector models of words. In *Proceedings of ICML*.

Bradley Efron and Robert J. Tibshirani. 1994. *An introduction to the bootstrap*, volume 57. CRC press.

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *Proceedings of ACL*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

Yoav Goldberg and Jon Orwant. 2013. A dataset of syntactic-ngrams over time from a very large corpus of English books. In *Second Joint Conference on Lexical and Computational Semantics (* SEM)*, volume 1, pages 241–247.

Edouard Grave, Guillaume Obozinski, and Francis Bach. 2013. Hidden markov tree models for semantic class induction. In *Proceedings of CoNLL*.

Gholamreza Haffari, Marzieh Razavi, and Anoop Sarkar. 2011. An ensemble model that combines syntactic and semantic clustering for discriminative dependency parsing. In *Proceedings of ACL*.

Sorami Hisamoto, Kevin Duh, and Yuji Matsumoto. 2013. An empirical investigation of word representations for parsing the web. In *Proceedings of ANLP*.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving Word Representations via Global Context and Multiple Word Prototypes. In *Proceedings of ACL*.

Fei Huang, Arun Ahuja, Doug Downey, Yi Yang, Yuhong Guo, and Alexander Yates. 2014. Learning representations for weakly supervised natural language processing tasks. *Computational Linguistics*, 40(1).

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *16th Nordic Conference of Computational Linguistics*.

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

Dekang Lin and Xiaoyun Wu. 2009. Phrase clustering for discriminative learning. In *Proceedings of ACL-IJCNLP*.

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

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

Ryan T. McDonald, Koby Crammer, and Fernando C. Pereira. 2005. Spanning tree methods for discriminative training of dependency parsers. Technical Report MS-CIS-05-11, University of Pennsylvania.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of ICLR*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*.

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

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of ICML*.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, AKBC-WEKEX '12, pages 95–100, Stroudsburg, PA, USA. Association for Computational Linguistics.

Olutobi Owoputi, Brendan OConnor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of NAACL*.

Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

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)*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of CoNLL*.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of EMNLP*.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: An experimental study. In *Proceedings of EMNLP*.

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*.

Valentin I. Spitkovsky, Hiyan Alshawi, Angel X. Chang, and Daniel Jurafsky. 2011. Unsupervised dependency parsing without gold part-of-speech tags. In *Proceedings of EMNLP*.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of NAACL*.

Stephen Tratz and Eduard Hovy. 2011. A fast, accurate, non-projective, semantically-enriched parser. In *Proceedings of EMNLP*.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of ACL*.

David Vadas and James R. Curran. 2007. Adding noun phrase structure to the Penn Treebank. In *Proceedings of ACL*.

Joe H. Ward. 1963. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of International Conference on Parsing Technologies*.