Attacking Parsing Bottlenecks with Unlabeled Data and Relevant Factorizations

Emily Pitler
Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
epitler@seas.upenn.edu

Abstract

Prepositions and *conjunctions* are two of the largest remaining bottlenecks in parsing. Across various existing parsers, these two categories have the lowest accuracies, and mistakes made have consequences for downstream applications. Prepositions and conjunctions are often assumed to depend on lexical dependencies for correct resolution. As lexical statistics based on the training set only are sparse, unlabeled data can help ameliorate this sparsity problem. By including unlabeled data features into a factorization of the problem which matches the representation of prepositions and conjunctions, we achieve a new state-of-the-art for English dependencies with 93.55% correct attachments on the current standard. Furthermore, conjunctions are attached with an accuracy of 90.8%, and prepositions with an accuracy of 87.4%.

1 Introduction

Prepositions and *conjunctions* are two large remaining bottlenecks in parsing. Across various existing parsers, these two categories have the lowest accuracies, and mistakes made on these have consequences for downstream applications. Machine translation is sensitive to parsing errors involving prepositions and conjunctions, because in some languages different attachment decisions in the parse of the source language sentence produce different translations. Preposition attachment mistakes are particularly bad when translating into Japanese (Schwartz et al., 2003) which uses a different post-position for different attachments; conjunction mistakes can cause word ordering mistakes when translating into Chinese (Huang, 1983).

Prepositions and conjunctions are often assumed to depend on *lexical dependencies* for correct resolution (Jurafsky and Martin, 2008). However, lexical statistics based on the training set only are typically sparse and have only a small effect on overall parsing performance (Gildea, 2001). *Unlabeled data* can help ameliorate this sparsity problem. Backing off to cluster membership features (Koo et al., 2008) or by using association statistics from a larger corpus, such as the web (Bansal and Klein, 2011; Zhou et al., 2011), have both improved parsing.

Unlabeled data has been shown to improve the accuracy of conjunctions within complex noun phrases (Pitler et al., 2010; Bergsma et al., 2011). However, it has so far been less effective within full parsing — while first-order web-scale counts noticeably improved overall parsing in Bansal and Klein (2011), the accuracy on conjunctions actually decreased when the web-scale features were added (Table 4 in that paper).

In this paper we show that unlabeled data *can* help prepositions and conjunctions, *provided that the dependency representation is compatible with how the parsing problem is decomposed for learning and inference*. By incorporating unlabeled data into factorizations which capture the relevant dependencies for prepositions and conjunctions, we produce a parser for English which has an unlabeled attachment accuracy of 93.5%, over an 18% reduction in error over the best previously published parser (Bansal and Klein, 2011) on the current standard for dependency parsing. The best model for conjunctions attaches them with 90.8% accuracy (42.5% reduction in error over MSTParser), and the best model for prepositions with 87.4% accuracy (18.2% reduction in error over MSTParser).

We describe the dependency representations of prepositions and conjunctions in Section 2. We discuss the implications of these representations for how learning and inference for parsing are decomposed (Section 3) and how unlabeled data may be used (Section 4). We then present experiments exploring the connection between representation, factorization, and unlabeled data in Sections 5 and 6.

2 Dependency Representations

A dependency tree is a rooted, directed tree (or arborescence), in which the vertices are the words in the sentence plus an artificial root node, and each edge (h, m) represents a directed dependency relation from the head h to the modifier m. Throughout this section, we will use Y to denote a particular parse tree, and $(h, m) \in Y$ to denote a particular edge in Y.

The Wall Street Journal Penn Treebank (PTB) (Marcus et al., 1993) contains parsed constituency trees (where each sentence is represented as a context-free-grammar derivation). Dependency parsing requires a conversion from these constituency trees to dependency trees. The Treebank constituency trees left noun phrases (NPs) flat, although there have been subsequent projects which annotate the internal structure of noun phrases (Vadas and Curran, 2007; Weischedel et al., 2011). The presence or absence of these noun phrase internal annotations interacts with constituency-to-dependency conversion program in ways which have effects on conjunctions and prepositions.

We consider two such mapping regimes here:

1. PTB trees $\rightarrow$ *Penn2Malt*[1] $\rightarrow$ Dependencies

2. PTB trees patched with NP-internal annotations (Vadas and Curran, 2007) $\rightarrow$ *pennconverter*[2] $\rightarrow$ Dependencies

[1] `http://w3.msi.vxu.se/~nivre/research/Penn2Malt.html`

[2] Johansson and Nugues (2007) `http://nlp.cs.lth.se/software/treebank_converter/`

Regime (1) is very commonly done in papers which report dependency parsing experiments (e.g., (McDonald and Pereira, 2006; Nivre et al., 2007; Zhang and Clark, 2008; Huang and Sagae, 2010; Koo and Collins, 2010)). *Penn2Malt* uses the head finding table from Yamada and Matsumoto (2003).

Regime (2) is based on the recommendations of the two converter tools; as of the date of this writing, the *Penn2Malt* website says: "Penn2Malt has been superseded by the more sophisticated pennconverter, which we strongly recommend". The *pennconverter* website "strongly recommends" patching the Treebank with the NP annotations of Vadas and Curran (2007). A version of *pennconverter* was used to prepare the data for the CoNLL Shared Tasks of 2007-2009, so the trees produced by Regime 2 are similar (but not identical)[3] to these shared tasks. As far as we are aware, Bansal and Klein (2011) is the only published work which uses both steps in Regime (2).

The dependency representations produced by Regime 2 are designed to be more useful for extracting semantics (Johansson and Nugues, 2007). The parsing attachment accuracy of MALTPARSER (Nivre et al., 2007) was lower using *pennconverter* than *Penn2Malt*, but using the output of MALTPARSER under the new format parses produces a much better semantic role labeler than using its output with *Penn2Malt* (Johansson and Nugues, 2007).

Figures 1 and 2 show how conjunctions and prepositions, respectively, are represented after the two different conversion processes. *These differences are not rare*–70.7% of conjunctions and 5.2% of prepositions in the development set have a different parent under the two conversion types. These representational differences have serious implications for how well various factorizations will be able to capture these two phenomena.

3 Implications of Representations on the Scope of Factorization

Parsing requires a) learning to score potential parse trees, and b) given a particular scoring function, finding the highest scoring tree according to that function. The number of potential trees for a sen-

[3] The CoNLL data does not include the NP annotations; it does include annotations of named entities (Weischedel and Brunstein, 2005) so had some internal NP edges.

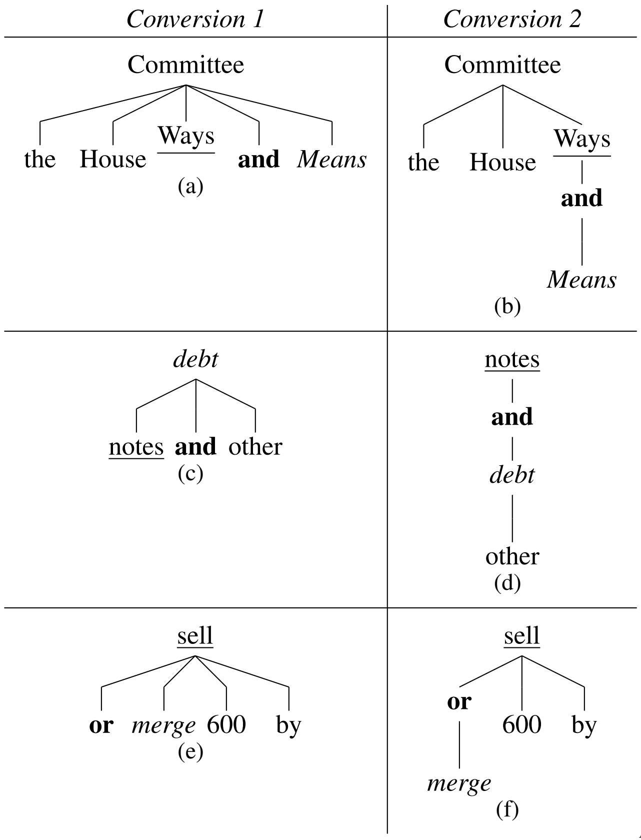

Figure 1: Examples of conjunctions: *the House Ways and Means Committee*, *notes and other debt*, and *sell or merge 600 by*. The conjunction is bolded, the left conjunct (in the linear order of the sentence) is underlined, and the right conjunct is italicized.

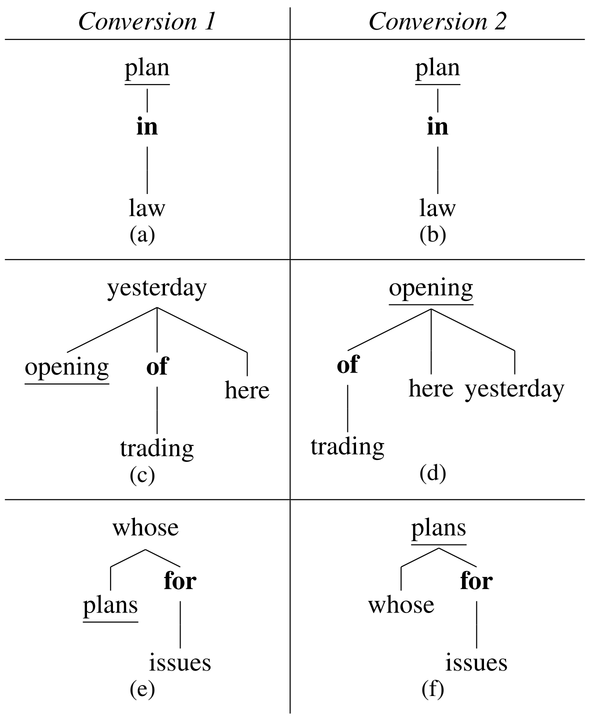

Figure 2: Examples of prepositions: *plan in the S&L bailout law*, *opening of trading here yesterday*, and *whose plans for major rights issues*. The preposition is bolded and the (semantic) head is underlined.

tence is exponential, so parsing is made tractable by decomposing the problem into a set of local substructures which can be combined using dynamic programming. Four possible factorizations are: single edges (edge-based), pairs of edges which share a parent (siblings), pairs of edges where the child of one is the parent of the other (grandparents), and triples of edges where the child of one is the parent of two others (grandparent+sibling). In this section, we discuss these factorizations and their relevance to conjunction and preposition representations.

3.1 Edge-based Scoring

One possible factorization corresponds to first-order parsing, in which the score of a parse tree Y decomposes completely across the edges in the tree:

$$S(Y) = \sum_{(h,m) \in Y} S(h,m) \qquad (1)$$

Conjunctions: Under Conversion 1, we can see three different representations of conjunctions in Figures 1(a), 1(c), and 1(e). Under edge-based scoring, the conjunction would be scored along with *neither* of its conjuncts in 1(a). In Figure 1(c), the conjunction is scored along with its right conjunct only; in figure 1(e) along with its left conjunct only. The inconsistency here is likely to make learning more difficult, as what is learned is split across these three cases. Furthermore, the conjunction is connected with an edge to either zero or one of its two arguments; at least one of the arguments is completely ignored in terms of scoring the conjunction.

In Figures 1(c) and 1(e), the words being conjoined are connected to *each other* by an edge. This overloads the meaning of an edge; an edge indicates both a head-modifier relationship and a conjunction relationship. For example, compare the two natural phrases *dogs and cats* and *really nice*. *dogs* and *cats* are a good pair to conjoin, but *cats* is not a good modifier for *dogs*, so there is a tension when scoring an edge like $(dogs, cats)$: it should get a high score when actually indicating a conjunction and low otherwise. $(nice, really)$ shows the opposite pattern–*really* is a good modifier for *nice*, but *nice* and *really* are not two words which should be conjoined. This may be partially compensated for by including features about the surrounding words (McDonald et al., 2005), but any feature templates which would be identical across the two contexts will be in tension.

In Figures 1(b), 1(d) and 1(f), the conjunction participates in a directed edge with each of the conjuncts. Thus, in edge-based scoring, at least under Conversion 2 neither of the conjuncts is being ignored; however, the factorization scores each edge independently, so how compatible these two conjuncts are with each other cannot be included in the scoring of a tree.

Prepositions: For all of the examples in Figure 2, there is a directed edge from the head of the phrase that the preposition modifies to the preposition. Differences in head finding rules account for the differences in preposition representations. In the second example, the first conversion scheme chooses *yesterday* as the head of the overall NP, resulting in the edge *yesterday→ of*, while the second conversion scheme ignores temporal phrases when finding the head, resulting in the more semantically meaningful *opening→of*. Similarly, in the third example, the preposition *for* attaches to the pronoun *whose* in the first conversion scheme, while it attaches to the noun *plans* in the second.

With edge-based scoring, the object is not accessible when scoring where the preposition should attach, and PP-attachment is known to depend on the object of the preposition (Hindle and Rooth, 1993).

3.2 Sibling Scoring

Another alternative factorization is to score *siblings* as well as parent-child edges (McDonald and Pereira, 2006). Scores decompose as:

$$S(Y) = \sum_{\left\{ (h,m,s) \;\middle|\; \begin{array}{l} (h,m) \in Y, (h,s) \in Y, \\ (m,s) \in Sib(Y) \end{array} \right\}} S(h,m,s) \qquad (2)$$

where $Sib(Y)$ is the set containing *ordered* and *adjacent* sibling pairs in Y: if $(m,s) \in Sib(Y)$, there must exist a shared parent h such that $(h,m) \in Y$ and $(h,s) \in Y$, m and s must be on the same side of h, m must be closer to h than s in the linear order of the sentence, and there must not exist any other children of h in between m and s.

Under this factorization, two of the three examples in Conversion 1 (and none of the examples in Conversion 2) in Figure 1 now include the conjunction and both conjuncts in the same score (Figures 1(c) and 1(e)). The scoring for head-modifier dependencies and conjunction dependencies are again being overloaded: $(debt, notes, and)$ and $(debt, and, other)$ are both sibling parts in Figure 1(c), yet only one of them represents a conjunction. The position of the conjunction in the sibling is not enough to determine whether one is scoring a true conjunction relation or just the conjunction and a different sibling; in 1(c) the conjunction is on the right of its sibling argument, while in 1(e) the conjunction is on the left.

For none of the other preposition or conjunction examples does a sibling factorization bring more of the arguments into the scope of what is scored along with the preposition/conjunction. Sibling scoring may have some benefit in that prepositions/conjunctions should have only one argument, so for prepositions (under both conversions) and conjunctions (under Conversion 2), the model can learn to disprefer the existence of any siblings and thus enforce choosing a single child.

3.3 Grandparent Scoring

Another alternative over pairs of edges scores grandparents instead of siblings, with factorization:

$$S(Y) = \sum_{\left\{ (h,m,c) \;\middle|\; (h,m) \in Y, (m,c) \in Y \right\}} S(h,m,c) \qquad (3)$$

Under Conversion 2, we would expect this factorization to perform much better on conjunctions and prepositions than edge-based or sibling-based factorizations. Both conjunctions and prepositions are consistently represented by exactly one grandparent relation (with one relevant argument as the grandparent, the preposition/conjunction as the parent, and the other argument as the child), so this is the first factorization that has allowed the compatibility of the two arguments to affect the attachment of the preposition/conjunction.

Under Conversion 1, this factorization is particularly appropriate for prepositions, but would be unlikely to help conjunctions, which have no children.

3.4 Grandparent-Sibling Scoring

A further widening of the factorization takes grandparents and siblings simultaneously:

$$S(Y) = \sum_{\left\{ (g,h,m,s) \;\middle|\; \begin{array}{l} (g,h) \in Y, (h,m) \in Y, \\ (h,s) \in Y, (m,s) \in Sib(Y) \end{array} \right\}} S(g,h,m,s) \quad (4)$$

For projective parsing, dynamic programming for this factorization was derived in Koo and Collins (2010) (Model 1 in that paper), and for non-projective parsing, dual decomposition was used for this factorization in Koo et al. (2010).

This factorization should combine all the benefits of the sibling and grandparent factorizations described above–for Conversion 1, sibling scoring may help conjunctions and grandparent scoring may help prepositions, and for Conversion 2, grandparent scoring should help both, while sibling scoring may or may not add some additional gains.

4 Using Unlabeled Data Effectively

Associations from unlabeled data have the potential to improve both conjunctions and prepositions. We predict that web counts which include both conjuncts (for conjunctions), or which include both the attachment site and the object of a preposition (for prepositions) will lead to the largest improvements.

For the phrase *dogs and cats*, edge-based counts would measure the associations between *dogs* and *and*, and *and* and *cats*, but never any web counts that include both *dogs* and *cats*. For the phrase *ate spaghetti with a fork*, edge-based scoring would not use any web counts involving both *ate* and *fork*.

We use *associations* rather than raw counts. The phrases *trading and transacting* versus *trading and what* provide an example of the difference between associations and counts. The phrase *trading and what* has a higher count than the phrase *trading and transacting*, but *trading and transacting* are more highly associated. In this paper, we use point-wise mutual information (PMI) to measure the strength of associations of words participating in potential conjunctions or prepositions.[4] For three words h, m, c, this is calculated with:

$$PMI(h, m, c) = \log \frac{P(h \;.\!*\; m \;.\!*\; c)}{P(h)P(m)P(c)} \quad (5)$$

[4]PMI can be unreliable when frequency counts are small (Church and Hanks, 1990), however the data used was thresholded, so all counts used are at least 10.

The probabilities are estimated using web-scale n-gram counts, which are looked up using the tools and web-scale n-grams described in Lin et al. (2010). Defining the joint probability using wildcards (rather than the exact sequence $h\ m\ c$) is crucially important, as determiners, adjectives, and other words may naturally intervene between the words of interest.

Approaches which cluster words (i.e., Koo et al. (2008)) are also designed to identify words which are semantically related. As manually labeled parsed data is sparse, this may help generalize across similar words. However, if edges are not connected to the semantic head, cluster-based methods may be less effective. For example, the choice of *yesterday* as the head of *opening of trading here yesterday* in Figure 2(c) or *whose* in 2(e) may make cluster-based features less useful than if the semantic heads were chosen (*opening* and *plans*, respectively).

5 Experiments

The previous section motivated the use of unlabeled data for attaching prepositions and conjunctions. We have also hypothesized that these features will be most effective when the *data representation* and the *learning representation* both capture relevant properties of prepositions and conjunctions. We predict that Conversion 2 and a factorization which includes grand-parent scoring will achieve the highest performance. In this section, we investigate the impact of unlabeled data on parsing accuracy using the two conversions and using each of the factorizations described in Section 3.1-3.4.

5.1 Unlabeled Data Feature Set

Clusters: We replicate the cluster-based features from Koo et al. (2008), which includes features over *all* edges (h, m), grand-parent triples (h, m, c), and parent sibling triples (h, m, s). The features were all derived from the publicly available clusters produced by running the Brown clustering algorithm (Brown et al., 1992) over the BLLIP corpus with the Penn Treebank sentences excluded.[5]

Preposition and conjunction-inspired features (motivated by Section 4) are described below:

[5]`people.csail.mit.edu/maestro/papers/bllip-clusters.gz`

Web Counts: For each set of words of interest, we compute the PMI between the words, and then include binary features for whether the mutual information is undefined, if it is negative, and whether it is greater than each positive integer.

For conjunctions, we only do this for triples of both conjunct and the conjunction (and if the conjunction is *and* or *or* and the two potential conjuncts are the same coarse grained part-of-speech). For prepositions, we consider only cases in which the parent is a noun or a verb and the child is a noun (this corresponds to the cases considered by Hindle and Rooth (1993) and others). Prepositions use association features to score both the triple (parent, preposition, child) and all pairs within that triple. The counts features are not used if all the words involved are stopwords. For the scope of this paper we use *only* the above counts related to prepositions and conjunctions.

5.2 Parser

We use the Model 1 version of *dpo*3, a state-of-the-art third-order dependency parser (Koo and Collins, 2010))[6]. We augment the feature set used with the web-counts-based features relevant to prepositions and conjunctions and the cluster-based features. The only other change to the parser's existing feature set was the addition of binary features for the part-of-speech tag of the child of the root node, alone and conjoined with the tags of its children. For further details about the parser, see Koo and Collins (2010).

5.3 Experimental Set-up

Training was done on Section 2-21 of the Penn Treebank. Section 22 was used for development, and Section 23 for test. We use automatic part-of-speech tags for both training and testing (Ratnaparkhi, 1996). The set of potential edges was pruned using the marginals produced by a first-order parser trained using exponentiated gradient descent (Collins et al., 2008) as in Koo and Collins (2010). We train the full parser for 15 iterations of averaged perceptron training (Collins, 2002), choose the iteration with the best unlabeled attachment score (UAS) on the development set, and apply the model after that iteration to the test set.

[6]`http://groups.csail.mit.edu/nlp/dpo3/`

We also ran MSTParser (McDonald and Pereira, 2006), the Berkeley constituency parser (Petrov and Klein, 2007), and the unmodified dpo3 Model 1 (Koo and Collins, 2010) using Conversion 2 (the current recommendations) for comparison. Since the converted Penn Treebank now contains a few non-projective sentences, we ran both the projective and non-projective versions of the second order (sibling) MSTParser. The Berkeley parser was trained on the constituency trees of the PTB patched with Vadas and Curran (2007), and then the predicted parses were converted using pennconverter.

6 Results and Discussion

Table 1 shows the unlabeled attachment scores, complete sentence exact match accuracies, and the accuracies of conjunctions and prepositions under Conversion 2.[7] The incorporation of the unlabeled data features (clusters and web counts) into the *dpo3* parser yields a significantly better parser than *dpo3* alone (93.54 UAS versus 93.21)[8], and is more than a 1.5% improvement over MSTParser.

6.1 Impact of Factorization

In all four metrics (attachment of all non-punctuation tokens, sentence accuracy, prepositions, and conjunctions), there is no significant difference between the version of the parser which uses the grandparent and sibling factorization (Grand+Sib) and the version which uses just the grandparent factorization (Grand). A parser which uses only grandparents (referred to as Model 0 in Koo and Collins (2010)) may therefore be preferable, as it contains far fewer parameters than a third-order parser.

While the grandparent factorization and the sibling factorization (Sib) are both "second-order" parsers, scoring up to two edges (involving three words) simultaneously, their results are quite different, with the sibling factorization scoring much worse. This is particularly notable in the conjunction case, where the sibling model is over 5% absolute worse in accuracy than the grandparent model.

[7]As is standard for English dependency parsing, five punctuation symbols :, ,, “, ”, and . are excluded from the results (Yamada and Matsumoto, 2003).

[8]If the (deprecated) Conversion 1 is used, the new features improve the UAS of *dpo3* from 93.04 to 93.51.

Model	UAS	Exact Match	Conjunctions	Prepositions
MSTParser (proj)	91.96	38.9	84.0	84.2
MSTParser (non-proj)	91.98	38.7	83.8	84.6
Berkeley (converted)	90.98	36.0	85.6	84.3
dpo3 (Grand+Sib)	93.21	**44.8**	**89.6**	**86.9**
dpo3+Unlabeled (Edges)	93.12	43.6	85.3	**87.0**
dpo3+Unlabeled (Sib)	93.15	43.7	85.5	86.8
dpo3+Unlabeled (Grand)	**93.55**	**46.1**	**90.6**	**87.5**
dpo3+Unlabeled (Grand+Sib)	**93.54**	**46.0**	**90.8**	**87.4**
- Clusters	93.10	**45.0**	**90.5**	**87.5**
- Prep,Conj Counts	**93.52**	**45.8**	**89.9**	**87.1**

Table 1: Test set accuracies under Conversion 2 of unlabeled attachment scores, complete sentence exact match accuracies, conjunction accuracy, and preposition accuracy. Bolded items are the best in each column, or not significantly different from the best in that column (sign test, $p < .05$).

6.2 Impact of Unlabeled Data

The unlabeled data features improved the already state-of-the-art *dpo3* parser in UAS, complete sentence accuracy, conjunctions, and prepositions. However, because the sample sizes are much smaller for the latter three cases, only the UAS improvement is statistically significant.[9] Overall, the results in Table 1 show that while the inclusion of unlabeled data improves parser performance, increasing the size of factorization matters even more. Ablation experiments showed that cluster features have a larger impact on overall UAS, while count features have a larger impact on prepositions and conjunctions.

6.3 Comparison with Other Parsers

The resulting *dpo3*+Unlabeled parser is significantly better than both versions of MSTParser and the Berkeley parser converted to dependencies across all four evaluations. *dpo3*+Unlabeled has an UAS 1.5% higher than MSTParser, which has an UAS 1.0% higher than the converted constituency parser. The MSTParser uses sibling scoring, so it is unsurprising that it performs less well on the new conversion.

While the converted constituency parser is not as good on dependencies as MSTParser overall, note that it is over a percent and a half better than MSTParser on attaching conjunctions (85.6% versus 84.0%). Conjunction scope may benefit from parallelism and higher-level structure, which is easily accessible when joining two matching non-terminals in a context-free grammar, but much harder to determine in the local views of graph-based dependency parsers. The dependencies arising from the Berkeley constituency trees have higher conjunction accuracies than either the edge-based or sibling-based dpo3+Unlabeled parser. However, once grandparents are included in the factorization, the dpo3+Unlabeled is significantly better at attaching conjunctions than the constituency parser, attaching conjunctions with an accuracy over 90%. Therefore, some of the disadvantages of dependency parsing compared with constituency parsing can be compensated for with larger factorizations.

[9]There are 52,308 non-punctuation tokens in the test set, compared with 2416 sentences, 1373 conjunctions, and 5854 prepositions.

	Conjunctions	
Scoring	Conversion 1 (deprecated)	Conversion 2
Edge	86.3	85.3
Sib	**87.8**	85.5
Grand	**87.2**	**90.6**
Grand+Sib	**88.3**	**90.8**

Table 2: Unlabeled attachment accuracy for conjunctions. Bolded items are the best in each column, or not significantly different (sign test, $p < .05$).

6.4 Impact of Data Representation

Tables 2 and 3 show the results of the *dpo3*+Unlabeled parser for conjunctions and prepositions, respectively, under the two different conversions. The data representation has an impact on which factorizations perform best. Under Conversion 1, conjunctions are more accurate under a sibling parser than a grandparent parser, while the

	Prepositions	
Scoring	Conversion 1 (deprecated)	Conversion 2
Edge	87.4	**87.0**
Sib	87.5	86.8
Grand	**87.9**	**87.5**
Grand+Sib	**88.4**	**87.4**

Table 3: Unlabeled attachment accuracy for prepositions. Bolded items are the best in each column, or not significantly different (sign test, $p < .05$).

pattern is reversed for Conversion 2.

Conjunctions show a much stronger need for higher order factorizations than prepositions do. This is not too surprising, as prepositions have more of a selectional preference than conjunctions, and so the preposition itself is more informative about where it should attach. While prepositions do improve with larger factorizations, the improvement beyond edge-based is not significant for Conversion 2. One hypothesis for why Conversion 1 shows more of an improvement is that the wider scope leads to the semantic head being included; in Conversion 2, the semantic head is chosen as the parent of the preposition, so the wider scope is less necessary.

6.5 Preposition Error Analysis

Prepositions are *still* the largest source of errors in the dpo3+Unlabeled parser. We therefore analyze the errors made on the development set to determine whether the difficult remaining cases for parsers correspond to the Hindle and Rooth (1993) style PP-attachment classification task. In the PP-attachment classification task, the two choices for where the preposition attaches are the previous verb or the previous noun, and the preposition itself has a noun object. The ones that *do* attach to the preceeding noun or verb (not necessarily the preceeding word) and have a noun object (2323 prepositions) are attached by the dpo3+Unlabeled grandparent-scoring parser with 92.4% accuracy, while those that do not fit that categorization (1703 prepositions) have the correct parent only 82.7% of the time.

Local attachments are more accurate — prepositions are attached with 94.8% accuracy if the correct parent is the immediately preceeding word (2364 cases) and only 79.1% accuracy if it is not (1662 cases). The preference is not necessarily for low attachments though: the prepositions whose parent is not the preceeding word are attached more accurately if the parent is the root word (usually corresponding to the main verb) of the sentence (90.8%, 587 cases) than if the parent is lower in the tree (72.7%, 1075 cases).

7 Conclusion

Features derived from unlabeled data (clusters and web counts) significantly improve a state-of-the-art dependency parser for English. We showed how well various factorizations are able to take advantage of these unlabeled data features, focusing our analysis on conjunctions and prepositions. Including grandparents in the factorization increases the accuracy of conjunctions over 5% absolute over edge-based or sibling-based scoring. The representation of the data is extremely important for how the problem should be factored–under the old Penn2Malt dependency representation, a sibling parser was more accurate than a grandparent parser. As some important relationships were represented as siblings and some as grandparents, there was a need to develop third-order parsers which could exploit both simultaneously (Koo and Collins, 2010). Under the new pennconverter standard, a grandparent parser is significantly better than a sibling parser, and there is no significant improvement when including both.

Acknowledgments

I would like to thank Terry Koo for making the *dpo3* parser publically available and for his help with using the parser. I would also like to thank Mitch Marcus and Kenneth Church for useful discussions. This material is based upon work supported under a National Science Foundation Graduate Research Fellowship.

References

M. Bansal and D. Klein. 2011. Web-scale features for full-scale parsing. In *Proceedings of ACL*, pages 693–702.

S. Bergsma, D. Yarowsky, and K. Church. 2011. Using large monolingual and bilingual corpora to improve coordination disambiguation. In *Proceedings of ACL*, pages 1346–1355.

P.F. Brown, P.V. Desouza, R.L. Mercer, V.J.D. Pietra, and J.C. Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

K.W. Church and P. Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29.

M. Collins, A. Globerson, T. Koo, X. Carreras, and P.L. Bartlett. 2008. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *The Journal of Machine Learning Research*, 9:1775–1822.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*, pages 1–8.

D. Gildea. 2001. Corpus variation and parser performance. In *Proceedings of EMNLP*, pages 167–202.

D. Hindle and M. Rooth. 1993. Structural ambiguity and lexical relations. *Computational Linguistics*, 19(1):103–120.

L. Huang and K. Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of ACL*, pages 1077–1086.

X. Huang. 1983. Dealing with conjunctions in a machine translation environment. In *Proceedings of EACL*, pages 81–85.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*, pages 105–112.

D. Jurafsky and J.H. Martin. 2008. *Speech and language processing: an introduction to natural language processing, computational linguistics and speech recognition*. Prentice Hall.

T. Koo and M. Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of ACL*, pages 1–11.

T. Koo, X. Carreras, and M. Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL*, pages 595–603.

T. Koo, A.M. Rush, M. Collins, T. Jaakkola, and D. Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of EMNLP*, pages 1288–1298.

D. Lin, K. Church, H. Ji, S. Sekine, D. Yarowsky, S. Bergsma, K. Patil, E. Pitler, R. Lathbury, V. Rao, et al. 2010. New tools for web-scale n-grams. In *Proceedings of LREC*.

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

R. McDonald and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, pages 81–88.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL*, pages 91–98.

J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov, and E. Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of NAACL*, pages 404–411.

E. Pitler, S. Bergsma, D. Lin, and K. Church. 2010. Using web-scale n-grams to improve base np parsing performance. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 886–894.

A. Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of EMNLP*, pages 133–142.

L. Schwartz, T. Aikawa, and C. Quirk. 2003. Disambiguation of English PP attachment using multilingual aligned data. In *Proceedings of MT Summit IX*.

D. Vadas and J. Curran. 2007. Adding noun phrase structure to the Penn Treebank. In *ACL*, pages 240–247.

R. Weischedel and A. Brunstein. 2005. BBN pronoun coreference and entity type corpus. *Linguistic Data Consortium, Philadelphia*.

R. Weischedel, M. Palmer, M. Marcus, E. Hovy, S. Pradhan, L. Ramshaw, N. Xue, A. Taylor, J. Kaufman, M. Franchini, et al. 2011. Ontonotes release 4.0. *LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium*.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of International Workshop of Parsing Technologies*, pages 195–206.

Y. Zhang and S. Clark. 2008. A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search. In *Proceedings of EMNLP*, pages 562–571.

G. Zhou, J. Zhao, K. Liu, and L. Cai. 2011. Exploiting web-derived selectional preference to improve statistical dependency parsing. In *Proceedings of ACL*, pages 1556–1565.