Simple Semi-Supervised Learning for Prepositional Phrase Attachment

Gregory F. Coppola, Alexandra Birch, Tejaswini Deoskar and Mark Steedman
School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK
g.f.coppola@sms.ed.ac.uk
{abmayne, tdeoskar, steedman}@inf.ed.ac.uk

Abstract

Prepositional phrase attachment is an important subproblem of parsing, performance on which suffers from limited availability of labelled data. We present a semi-supervised approach. We show that a discriminative lexical model trained from labelled data, and a generative lexical model learned via Expectation Maximization from unlabelled data can be combined in a product model to yield a PP-attachment model which is better than either is alone, and which outperforms the modern parser of Petrov and Klein (2007) by a significant margin. We show that, when learning from unlabelled data, it can be beneficial to model the generation of modifiers of a head collectively, rather than individually. Finally, we suggest that our pair of models will be interesting to combine using new techniques for discriminatively constraining EM.

1 Introduction

Labelled data for NLP tasks will always be in short supply. Thus, a statistical parser trained with labelled data alone will always be troubled by unseen events—primarily when parsing out-of-domain data, or when faced with rare events from in-domain data. Thus, a major focus of current work is the use of cheap, abundant *unlabelled data* to improve state-of-the-art parser performance.

We focus on an important sub-problem of parsing—prepositional phrase attachment—and demonstrate a successful semi-supervised learning strategy. We show that, using a mix of labelled and unlabelled data we can improve *both* the in-domain and out-of-domain performance of a prepositional phrase attachment classifier.

Prepositional phrase attachment, for us, is the decision as to which heads a series of prepositional phrases of the form [$_{PP}$ prep NP] modify, as in, e.g.,

He ate a salad [$_{PP}$ with a fork] [$_{PP}$ of plastic]

Prepositional phrase attachment is an important sub-problem of parsing in and of itself. Structural heuristics perform poorly (cf., Collins and Brooks, 1995), and so lexical knowledge is crucial.

Moreover, the highly lexicalized nature of prepositional phrase attachment makes it a kind of microcosm of the general problem of learning dependency structure, and so acts as a computationally less-demanding testing ground on which to try out learning techniques. We have endeavoured to approach the problem with a strategy that might be likely to generalize: a mix of generative and discriminative lexical models, trained using techniques that have worked for parsers.

The main contributions of this paper are:

- We compare the performance on the prepositional phrase attachment task of natural lexicalized dependency parsing strategies, to the popular semi-lexicalized model of Petrov and Klein (2007), and show that a lexical is more effective for this problem.

- We show that a discriminative lexical model trained from labelled data and a generative lexical model learned through Expectation Maximization on unlabelled data can perform better in a *product model* than either does alone, yielding a significant improvement over our baseline reference, the parser of Petrov and Klein (2007).

- We show that, in this case, when learning from unlabelled data, a strategy of generating all modifiers of a head *collectively* works better than generating them individually.

2 Prior Work

Work on the topic of prepositional phrase attachment typically views the problem as a binary classification task. Given a 4-tuple,

$$\langle \mathsf{verb}, \mathsf{noun}_1, \mathsf{prep}, \mathsf{noun}_2 \rangle$$

the task is to decide whether the attachment of the $[_{\mathrm{PP}}\ \mathsf{prep}\ \mathsf{NP}]$ prepositional phrase characterized by $\langle \mathsf{prep}, \mathsf{noun}_2 \rangle$ is to verb or to noun_1.

Human performance on this prepositional phrase attachment task has been estimated by Ratnaparkhi et al. (1994). They found that treebanking experts given the 4-tuple binary decision task choose the correct attachment 88.2% of the time. And, when then given the full context for the same examples, they choose the "correct" attachment (i.e. the same attachment that is given by the Penn Treebank) 93.2% of the time.

Approaches to training from labelled data include rule-based (Brill and Resnik, 1994), maximum-entropy (Ratnaparkhi et al., 1994), and a generative "backed-off" model (Collins and Brooks, 1995).

The state-of-the-art is an approach by Stetina and Nagao (1997). They replace each noun and verb by a WordNet sense using a custom word sense disambiguation algorithm. Then, they train a decision tree on labelled WSJ data. Their method achieves essentially a human level of accuracy on this task: 88.1%. Toutanova et al. (2004) achieve a comparable result with a method that integrates word-sense disambiguation into the generative attachment model.

There is also a variety of work that makes use of unlabelled data to learn prepositional phrase attachment. An early example in this category is Hindle and Rooth (1991). They estimate the probability that a given preposition modifies a given head using an iterative process with heterogeneous steps. Ratnaparkhi (1998) uses deterministic heuristics.

The state-of-the-art in this area is due to Pantel and Lin (2000). They, use a homespun iterative algorithm learning algorithm which bears a resemblance to EM, but it does not seem to learn a generative model. One interesting feature of this approach is that the attachment decision for a given word is allowed to make use of the statistics collected for *similar words*, which helps to the sparsity that occurs even in a large, unlabelled corpus. Their performance on the binary classification task is 84.5%.

We are aware of one case that has used a mix of labelled and unlabelled data: Volk (2002) uses a back-off strategy in which information from labelled data is used when conclusive, and information from unlabelled data otherwise. Performance on the same task on a NEGRA-based data set lags behind the others, at 81.0%.

Finally, Atterer and Schütze (2007) argue that an experimental setup that evaluates a prepositional phrase attachment with possible attachments given by an "oracle," rather than an actual parser, may make the problem appear easier than it really is. This is a good point. But, for the purposes of comparing learning techniques, we feel that the typical oracle task is better suited, as it avoids introducing the noise of parser mistakes.

3 Background

3.1 The Prediction Task

We treat prepositional phrase attachment as a structured prediction task, rather than as a binary decision. The input to the prediction procedure will be a (prepositional phrase) *attachment problem*, a string matching either the regular expression (1) or (2).

(1) verb baseNP (prep baseNP)*

(2) baseNP (prep baseNP)*

For example, some attachment problems are:

(3) sought man from Germany with expertise

(4) man from Germany with expertise

Though not indicated above, we assume that POS tags are given as part of the problem, and need not be predicted.

Our goal is to create a prediction procedure which, given an attachment problem, $\mathbf{x}$, will return a *derivation*, $\mathbf{d}$, which is a parse for $\mathbf{x}$ using the following mini CFG grammar whose initial symbol is ROOT:

(5) a. $\textsc{root} \rightarrow \mathsf{VP}$

b. $\textsc{root} \rightarrow \mathsf{NP}$

c. $\mathsf{VP} \rightarrow \mathsf{verb}_{\mathrm{H}}\ \mathsf{NP}\ \mathsf{PP}^*$

d. $\mathsf{NP} \rightarrow \mathsf{baseNP}_{\mathrm{H}}\ \mathsf{PP}^*$

e. $\mathsf{PP} \rightarrow \mathsf{prep}_{\mathrm{H}}\ \mathsf{NP}$

The head of each XP is indicated with a subscripted H. All siblings to a head are called its *modifiers*.

Given a full parse tree in the style of Marcus et al. (1993), *attachment problem-derivation* pairs were extracted using a TGrep-like functional program, which is described in Appendix A.[1]

3.2 Scoring Performance

When evaluating a prediction procedure, we will give it a series of attachment problems and ask for the derivations. In most cases, the score we will focus on is what we can call the *binary decision score*, i.e., the percentage of the time in which the first prepositional phrase following a verb–direct-object pair is attached correctly. In this case, we are reporting the same score as is typically reported on this task, so as to avoid introducing a new metric.

To be clear, then, when scoring, in this way,

$$[_{\text{VP}} \text{ set } [_{\text{NP}} \text{ rate } [_{\text{PP}} \text{ on } [_{\text{NP}} \text{ refund}]] \,] \, [_{\text{PP}} \text{ at } [_{\text{NP}} \text{ 5 percent}]]]]$$

we only ask where [PP on [NP refund]] attaches, and ignore the attachment decision of [PP at [NP 5 percent]].

One might thus ask what the point of bothering with the whole derivation is if we are typically only intending to score a binary decision. Well, our model in §5.2 makes each attachment decision independently, and in this example from the WSJ development test set, incorrectly attaches both *on refund* and *at 5 percent* to the verb. In contrast, our model of §5.3 attaches prepositional phrases *collectively*, and rejects the derivation where *on refund* and *at 5 percent* both modify *set*. Thus, though we only score one decision in a derivation, that decision can be influenced by others, and so it does make a difference to work at the level of derivations.

3.3 Data Sets

The traditional semi-supervised learning task involves learning from two sets. The first is a set of pairs, $\{(\mathbf{x}_i, \mathbf{d}_i)\}$, of data points along with their labels. The second is a (typically much larger) set of data points alone, $\{\mathbf{x}_j\}$ (i.e. without labels). In our case, the data points are attachment problems and the labels, which are structured, are derivations.

Our experiment is interested in performance across domains. Thus, we also distinguish between in-domain labelled data, some of which we will allow ourselves to use to train model parameters, and out-of-domain labelled data, which we will not use to train model parameters.[2]

Our source of labelled data is the Penn Treebank (Marcus et al., 1993). We use sections 0-22 of the WSJ portion of the Penn Treebank for training and development and sections $23, 24$ are left for final evaluation. We variously use i) sections 2-21 as labelled training data, with sections $0, 1, 22$ as a development test set, or ii) sections $0, 1, 5$-22 as labelled data with sections 2-4 as held-out set for parameter tuning.

We split up the Brown portion of the treebank similarly to Gildea (2001)—i.e. we split it into 10 sections such that the s'th sentence is assigned to section $s \mod 10$. We then use sections 0-2 development test set, 4-6 for tuning, and sections 7-9 are left for final evaluation. Our divisions of the Penn Treebank are chosen to resemble the canonical training-test split for parsing, but we use more sections for testing, to obtain more reliable test scores, as there are far fewer decisions to test on in each section in our task, when compared to parsing.

Our source of unlabelled data is the New York Times portion of the GigaWord corpus (Graff et al., 2005). These sentences are parsed automatically using the generative semi-lexicalized parser of Petrov and Klein (2007). We did some filtering of these unlabelled sentences, removing sentences with quotations (as quoted material can be ungrammatical) and sentences over 40 words in length (to increase the chance that each automatic parse used was reasonable).

We use these automatically extracted parses only to identify *attachment problems* ($\mathbf{x}$), which we will then treat as unlabelled. That is, while there is possibly information in the *derivations* (the $\mathbf{d}$) that the parser is coming up with, we did not use this.

In terms of size, our WSJ2-21 set has $29,750$ examples. The GigaWord set has $8,038,001$ examples.

[1] Our data sets extracted from the Penn Treebank will be available on request to those with the relevant license(s) to use Penn Treebank data.

[2] However, we do appeal to the labels of a portion of the out-of-domain Brown data once, in order to fix a single experimental parameter, which is the number of iterations of EM to run on the unlabelled data, cf. note 6.

3.4 Baseline

Much past work has tested on the 4-tuple, binary decision data set of Ratnaparkhi et al. (1994). This data does not have all of the information required by our approach, and is based on a preliminary version of the Penn Treebank (version 0.75), which is incomplete and difficult to work with. Thus, we could not compare our work directly with past work.

In order to evaluate our performance, then, we will compare our model against the performance on prepositional phrase attachment of the Berkeley parser (Petrov and Klein, 2007), which is popular, readily available, and essentially state-of-art among supervised parsing methods. And, as we said, this is precisely the technology that we use to process unlabelled data, so it makes sense that our model should improve upon this in order to be of any use.

So, we need to evaluate the prepositional phrase attachment performance of the Berkeley parser. What we do is parse the test sections of the Penn Treebank using this parser (which is trained on WSJ2-21). We run our functional program to extract problem-derivation pairs from the automatic parse. Suppose we extract the pair $(\mathbf{d}_a, \mathbf{x}_a)$ from the automatic parse. We compare this to the gold tree. If the gold tree contains the pair $(\mathbf{d}_g, \mathbf{x}_g)$, and $\mathbf{x}_a = \mathbf{x}_g$, then we score $\mathbf{d}_a$ with respect to $\mathbf{d}_g$. Otherwise, we do not score $\mathbf{d}_a$.

This means the parser is not penalized for failing to identify attachment problems in the gold parse. And, this should be a favourable comparison for the parser as it is evaluated on the examples it knows most about, i.e. those for which it can identify the location of an attachment problem.

We supply the parser with gold POS tags to maximize the chance that it will find each attachment problem. In this way, we were able to assess the performance of the Berkeley parser on $\frac{3184}{3475} = 91.6\%$ of the test examples in the WSJ set, and $\frac{3091}{3509} = 88\%$ in the Brown (both dev. test and final test). Our models are tested on all examples.

An important parameter for the Berkeley parser is the number of *split-merge iterations* done during training (cf. Petrov and Klein, 2007). The documentation suggests 6 is better for parsing the WSJ, while 5 is better for parsing other English. We tried both. The results are shown in Table 1. We will use whichever parameterization did better on each data set as the baseline on that data set.[3]

	WSJ		Brown	
Parser	Dev.	Test	Dev.	Test
Berkeley 5 SM	**85.3**	**83.0**	82.4	81.1
Berkeley 6 SM	84.6	**83.0**	**83.3**	**82.7**

Table 1: Performance of the Berkeley parser on the prepositional phrase attachment task. The best scores on each data set will be our baseline.

3.5 Reduction of Open-Class Words

In all experiments, all nouns and verbs were replaced by more general forms. If applicable, nouns were replaced by their NER label, either *person*, *place* or *organization*, using the NER classifier of Finkel et al. (2005). All numeral strings of two or four digits were replaced with a symbol representing *year*, and all other numeral strings were replaced with a symbol representing *numeric value*.

A word not reduced in either of these ways was replaced by its stem using the stemmer designed by Minnen et al. (2001).[4]

Finally, this reduced form is paired with the category of the word $c \in \{\text{NOUN}, \text{VERB}\}$ to distinguish uses of words that can either be nouns or verbs. We find that these reductions improve performance slightly and also reduce the size of the generative probability table.

4 A Discriminative Model from Labelled Data

4.1 The Model

As noted, we have access to one set $\{(\mathbf{x}_i, \mathbf{d}_i)\}$ of labelled examples. We begin by discriminatively training two conditional models on this set.

Our model uses two types of features: i) structural, and ii) lexical.

The structural features exploit two characteristics of prepositional phrase attachment that are often noted in the literature. First, a prepositional phrase headed by the preposition *of* almost always attaches to the nearest available attachment site to its left. So, one feature fires whenever this is

[3] The performance of this parser depends on a random seed used to initialize the training parser. Our 6-iteration grammar was downloaded from the authors' web site. Our 5-iteration grammar is the one that resulted from our first run of the training process.

[4] We use the Java reimplementation in the Stanford NLP API, http://nlp.stanford.edu/software/index.shtml.

not the case. Second, prepositional phrases almost never attach to pronouns. So, another feature fires whenever some prepositional phrase attaches to a pronoun.

The first model, denoted $p_{\mathbf{D}_1}(\mathbf{d} \mid \mathbf{x}; \theta_{\mathbf{D}_1})$, uses these and lexical features of the form

$$\langle \mathbf{head}, \mathbf{prep} \rangle$$

Here, $\mathbf{head}$ is either a noun or a verb, and $\mathbf{prep}$ is a preposition. The feature $\langle \mathbf{head}, \mathbf{prep} \rangle$ is active in derivation $\mathbf{d}$ iff (a prepositional phrase headed by) $\mathbf{prep}$ modifies $\mathbf{head}$ in $\mathbf{d}$.

Our second model, $p_{\mathbf{D}_2}(\mathbf{d} \mid \mathbf{x}; \theta_{\mathbf{D}_2})$, has all features mentioned above, and also those of the form

$$\langle \mathbf{head}, \mathbf{prep}, \mathbf{noun}_{\text{inside}} \rangle$$

Here, $\mathbf{head}$ and $\mathbf{prep}$ are as before, and $\mathbf{noun}_{\text{inside}}$ is the head of the noun phrase inside the prepositional phrase headed by $\mathbf{prep}$.

For example, in

$$[_{\text{NP}} \text{ salad } [_{\text{PP}} \text{ with } [_{\text{NP}} \text{ dressing}]]]$$

The active features are $\langle$salad, with$\rangle$ and $\langle$salad, with, dressing$\rangle$.

This latter type of feature represents straightforward specialization of the concept used by higher-order dependencies for parsing, especially Carreras (2007), to the problem of prepositional phrases.

Our estimates of $\theta_{\mathbf{D}_1}$ and $\theta_{\mathbf{D}_2}$ are arrived at using structured perceptron training (Collins, 2002). The models trained on all examples in our training section (i.e. those of type NP and of type VP). We pick the number of perceptron training iterations for each model by maximizing performance on a held-out set, using our parameter tuning split described in §3.3. We only use feature instances that have occurred at least twice in training.

If $\Phi(\mathbf{x}, \mathbf{d})$ is a feature vector characterizing $(\mathbf{x}, \mathbf{d})$, the perceptron algorithm will output a parameter vector $\theta_{\mathbf{D}_i}$, and the "score" assigned to a pair $(\mathbf{x}, \mathbf{d})$ under this interpretation will be $\theta_{\mathbf{D}_i} \cdot \Phi(\mathbf{x}, \mathbf{d})$, with the predicted derivation being the $\mathbf{d}$ with the highest score (in the case of any tie, we always choose attachment to the verb).

As we have suggested, we are interested in a conditional probability for $\mathbf{d}$ given $\mathbf{x}$, rather than just a linear "score". This is for use in a future section (i.e. §6). The natural way to achieve this is to interpret $\theta_{\mathbf{D}_i}$ as the parameters for a maximum entropy model, i.e.

$$p_{\mathbf{D}_i}(\mathbf{d} \mid \mathbf{x}; \theta_{\mathbf{D}_i}) = \frac{\exp\{\theta_{\mathbf{D}_i} \cdot \Phi(\mathbf{x}, \mathbf{d})\}}{\sum_{\mathbf{d}'} \exp\{\theta_{\mathbf{D}_i} \cdot \Phi(\mathbf{x}, \mathbf{d}')\}}$$

Fortunately, we will see that we never need actually compute the normalizing term in the denominator.

	WSJ		Brown	
Classifier	Dev.	Test	Dev.	Test
$p_{\mathbf{D}_2}$ (2nd-order)	**87.4**	**86.0**	**84.7**	**83.9**
$p_{\mathbf{D}_1}$ (1st-order)	86.2	**86.0**	**84.7**	83.0
Baseline	85.3	83.0	83.3	82.7

Table 2: Performance of the two discriminative classifiers.

4.2 Results

The performance of this model both in- and out-of-domain are shown in Table 2, along with the performance of the baseline Berkeley parser.

The results should be of interest to those interested the use of lexical features for parsing. The models that use lexical features outperform the semi-lexical model of Petrov and Klein (2007). Prepositional phrase attachment may be one area where lexicalized models are especially important.

We also see that the use of second-order features buys extra performance on some data sets, but not on others. A look at the dev. sets show that second-order features were active in 9 of the first 60 decisions in the WSJ, but in 0 of the first 60 in Brown. We speculate that use of word senses as features, taking after Stetina and Nagao (1997), might result in better generality across domains, but leave this to future work.

5 Two Generative Models Trained on Unlabelled Data

5.1 Common Model Structure

We now want to make use of our unlabelled data, $\{\mathbf{x}_j\}$. For this purpose, we turn to the Expectation Maximization algorithm (Dempster et al., 1977). Thus, we will estimate generative models of the data, each of the form $p_{\mathbf{G}_*}(\mathbf{x}, \mathbf{d}; \theta_{\mathbf{G}})$.

Our discriminatively trained classifier made use of both structural and lexical features. Our strategy will be to use our unlabelled examples to estimate just the *lexical* parameters.

It would seem impossible to expect that we could learn the structural features, e.g., that prepositional phrases do not attach to pronouns from unlabelled data. Nor do we need to do this, as this constraint can be encoded with little effort. What we do want to be able to estimate from unlabelled data is the strength of lexical relationships in arbitrary domains, which we could not hope to encode manually.

So, the question arises as to how to incorporate our knowledge of the structural constraints into the problem, and to constrain the EM process using these. Probably the most powerful and general purpose way to do this would be to use one of the new variants of EM that allows the specification of expectations for feature counts, such as Ganchev et al. (2010) or Druck and McCallum (2010). In this case we could specify that, e.g., we expect the average number of times we see a prepositional phrase attach to a pronoun to be 0, and the modified EM process would be encouraged or forced to converge to a solution that respects this expectation. This strategy is very general, but also much more expensive than ordinary EM, as each E-step involves an expensive optimization problem.

Here, we get by with a simpler, and much more computationally inexpensive strategy.

The structural constraints we have made use of are very robust. In our WSJ sections $2-21$, *of*-headed PPs attach to the nearest non-pronoun to their left in about 98.6% of cases. And, in 99.6% of cases, pronouns have nothing attaching to them.

Thus, we are willing to take a deterministic stance: we will assign 0 probability mass to derivations that violate our structural constraints. Let AD, intuitively the set of "admissible derivations," be the set of derivations that have: i) no attachment to a pronoun, unless there is no other choice, and ii) every *of*-pp attaching to its nearest available non-pronoun site, if it has one.

Let $\mathbf{1}_{d \in AD}$ be an indicator function, which is equal to 1 if $d \in AD$, and 0 otherwise. Then, the strategy is to let

$$p_{\mathbf{G}_*}(\mathbf{x}, \mathbf{d}) \;=\; p_{\mathbf{G}_*}(\mathbf{x}, \mathbf{d} \mid \mathbf{1}_{d \in AD}) \cdot \mathbf{1}_{d \in AD}$$

and to estimate $p_{\mathbf{G}_*}(\mathbf{x}, \mathbf{d} \mid \mathbf{1}_{d \in AD})$ from unlabelled data. The result is easily seen to be a probability distribution in which all weight is given to derivations in AD.

We will look at two model structures. In both cases, the data trained on will include all examples of type VP and NP from our GigaWord set. Examples with unambiguous attachment, such as [NP place [PP at table]], are included, so that parameters chosen must maximize likelihood over these examples as well. Also note that, since the number of derivations is never unmanageable, when summing over derivations, we do so exhaustively rather than use some form of dynamic programming such as, e.g., the inside-outside algorithm.

We will now look at two different ways to structure a model for $p_{\mathbf{G}_*}(\mathbf{x}, \mathbf{d} \mid \mathbf{1}_{d \in AD})$.

5.2 Individual Dependent Generation

The first model uses the same lexical features as our first-order discriminative model. We model a derivation as a series of sub-events, which will be draws from a pair of random variables, $(\mathbf{head}, \mathbf{prep})$. Intuitively, this corresponds to the event *the phrase headed by* $\mathbf{head}$ *generates a modifier headed by* $\mathbf{prep}$. Following the reasoning of Collins (1999, p. 46), we imagine that each head's final modifier is a special STOP symbol.

Thus, the derivation

(6) [VP ate [NP salad] [PP with [NPfork]]]

is modeled as the four events (eat, with), (eat, STOP), (salad, STOP), (fork, STOP).

Then, the probability of a problem-derivation pair, conditioned on $\mathbf{1}_{d \in AD}$, is estimated as

$$p_{\mathbf{G}_{Ind}}(\mathbf{x}, \mathbf{d} \mid \mathbf{1}_{d \in AD}; \theta_{\mathbf{G}_{Ind}}) = \prod_{h \in heads(\mathbf{d})} \prod_{p \in deps(\mathbf{head})} p_{\mathbf{G}_{Ind}}(\mathbf{prep} \mid \mathbf{head}; \theta_{\mathbf{G}_{Ind}})$$

Here $heads(\mathbf{d})$ is the set of head of NP and VP phrases in $\mathbf{d}$, and $deps(\mathbf{head})$ is the list of modifiers of $\mathbf{head}$ in $\mathbf{d}$, including the STOP symbol.

Even though our unlabelled training set was large, there were still nouns and verbs seen during test that were not seen during training. Thus, we created a GENERIC head for each category (one for VERB and one for NOUN). Each time an event $(\mathbf{head}, \mathbf{prep})$ was counted, we would also count $(\text{GENERIC}_{type(\mathbf{head})}, \mathbf{prep})$, where $type(\mathbf{head}) \in \{\text{VERB}, \text{NOUN}\}$. Thus, the generic head represents a kind of "average head." Then, if we needed the probability for some event $(\mathbf{head}', \mathbf{prep}')$ during test, and $\mathbf{head}'$ had not been seen during training, we would back off to the event $(\text{GENERIC}_{type(\mathbf{head}')}, \mathbf{prep}')$.

5.3 Collective Dependent Generation

The next strategy we try is to generate the collection of dependents for a head $\mathbf{head}$ *as a whole*. In particular, we generate each head's *multi-set* of dependents.[5]

In this case, if a verb has a direct object, the direct object is represented in the multi-set of the verbs modifiers. But, rather than put the word itself, each direct object is represented by a special symbol, DO.

So, in this model, (6) is modeled as three events, (eat, $\{\text{DO}, with\}$), (salad, $\{\}$), and (fork, $\{\}$).

Then, if $deps_{\mathbf{d}}(\mathbf{head})$ is the multi-set of modifiers of $\mathbf{head}$ in $\mathbf{d}$, our estimate of the probability of the problem-derivation pair, conditioned on $\mathbf{1}_{d \in AD}$, is

$$p_{\mathbf{G}_{Col}}(\mathbf{x}, \mathbf{d} \mid \mathbf{1}_{d \in AD}; \theta_{\mathbf{G}_{Col}}) = \prod_{\mathbf{head} \in heads(\mathbf{d})} p_{\mathbf{G}_{Col}}(deps_{\mathbf{d}}(\mathbf{head}) \mid \mathbf{head}; \theta_{\mathbf{G}_{Col}})$$

Unseen heads are handled in the same manner as in §5.2.

5.4 Initializing and Terminating the EM Process

We consider two cases for initializing the EM process. In one case, our initial guess at the posterior distribution is the uniform distribution: all derivations for a given $\mathbf{x}$ that are in AD are considered equally likely, with small random perturbations to break any symmetry (we found there to be essentially no difference from run to run).

In the second case, we make better use of our labelled data, and begin with the conditional distribution given by $p_{\mathbf{D}_1}$, the model with only first-order features, that we had estimated discriminatively from labelled data, i.e.

$$p_{\mathbf{G}_*}(\mathbf{d}|\mathbf{x}, \mathbf{1}_{d \in AD}; \emptyset) = p_{\mathbf{D}_1}(\mathbf{d}|\mathbf{x}; \theta_{\mathbf{D}_1})$$

Given that running EM to convergence can overfit the training data, to the detriment of performance (cf., Liang and Klein (2009)), we chose to run EM for a fixed number of iterations, with the number of iterations determined using a tuning test set. We picked the number of iterations to maximize performance of the best performing model (which turned out to be the collective dependent generation initialized with $p_{\mathbf{D}_1}$) on a Brown tuning set, sections 4-6.[6] The optimal number of iterations was found to be 4.

5.5 Results

The performance of these two model structures, under the two kinds of initialization methods, is shown in Table 3. Performance on the dev. test sets is plotted versus the number of iterations of the EM procedure. The fourth row is highlighted as this was determined to be the optimal number of iterations in the manner just described.

We see that the best performing method is that which generates the (multi-set of the) modifiers simultaneously, while initializing using the conditional distribution estimated from labelled data. It is also interesting to note that, models initialized with $p_{\mathbf{D}_1}$ get progressively better at parsing Brown but worse at parsing WSJ. In fact, after 6 iterations, performance at parsing the WSJ is similar for both models initialized with $p_{\mathbf{D}_1}$ and those initialized with the uniform distribution. This suggests to us that the information that we have from our valuable labelled data is being lost, and leads us to think that it may be profitable to incorporate this information more forcefully, using techniques for constraining EM with information, such as those of Ganchev et al. (2010) and Druck and McCallum (2010) already mentioned.

However, though the model may be "losing" information relevant to parsing the WSJ, the noteworthy aspect is that it ends up being able to parse the Brown data better than our discriminatively trained parsers (both with accuracy of 84.7 on Brown dev. test), and thus makes a contribution to overall parsing accuracy.

Note that we show the performance of the various models on the development test set only here. We report results on a held-out set in the next section.

[5] We also experimented with lists and sets, finding multi-sets to work slightly better. To avoid losing the focus of this discussion, we will only discuss multi-sets.

[6] In this case, we are using the labels from our out-of-domain data during training. We feel that this is legitimate because these are only used to tune a single experimental parameter, the number of EM iterations. Tuning a single parameter requires only a small number of examples, that does not necessarily grow with the number of lexical parameters being estimated. It is the fact that we can learn lexical model parameters from unlabelled data that will ultimately save us from having to label the entire web.

	Initialized with $p_{\mathbf{D}_1}(\mathbf{d} \mid \mathbf{x}; \theta_{\mathbf{D}_1})$				Initialized to uniform distr.			
	Independent		Collective		Independent		Collective	
EM Its.	WSJ	Brown	WSJ	Brown	WSJ	Brown	WSJ	Brown
1	85.7	83.9	86.3	85.8	82.0	83.4	81.6	84.5
2	84.2	84.9	84.9	85.7	82.4	84.4	82.7	85.2
3	83.3	84.5	84.7	86.0	82.8	84.6	82.5	85.1
4	83.0	84.6	83.8	86.3	82.9	84.6	82.5	85.4
5	82.9	84.6	83.5	86.0	82.9	84.7	82.6	85.7
6	82.8	84.6	82.8	86.0	82.9	84.7	82.4	85.8

Table 3: Performance of the two generative models. Variables are: i) the model learned, independent vs. collective modifier generation (§5.2, 5.3), ii) the intial guess at a conditional distribution for hidden variables (§5.4), and iii) the number of iterations of EM. Score is the binary decision score (cf. §3.2).

6 A Combination of Models

6.1 The Combination

We now have two models of prepositional phrase attachment. One is estimated from labelled data, and one from unlabelled data. Each performs well in isolation. But, we find that the combination of the two in a *logarithmic opinion pool* framework works better than either does alone.

With roots in Bordley (1982), a logarithmic opinion pool, as defined in Smith et al. (2005), has the form

$$p_{\mathbf{lop}}(\mathbf{d}|\mathbf{x}) = \frac{1}{Z_{\mathbf{lop}}} \prod_{\alpha} p_{\alpha}(\mathbf{d}|\mathbf{x})^{w_a}$$

In related work, Hinton (1999) describes a *product of experts*, i.e. multiple models trained to work together, so that an unweighted product, will be sensible. Petrov (2010) has success with the unweighted product of the scores of several similar parsing models. Smith et al. (2005) adjust weights by maximizing the likelihood of a labelled dataset.

Here, we weight the component models so as to maximize performance on a held out set. Where i is either 1 or 2, let

$$p_{\mathbf{lop},i}(\mathbf{d} \mid \mathbf{x}; \theta_{\mathbf{D}_i}, \theta_{\mathbf{G}_{Col}}) = \frac{1}{Z_i} \cdot p_{\mathbf{D}_i}(\mathbf{d} \mid \mathbf{x}; \theta_{\mathbf{D}_i})^{k_i} \cdot p_{\mathbf{G}_{Col}}(\mathbf{d} \mid \mathbf{x}; \theta_{\mathbf{G}_{Col}})^{1-k_i}$$

for some $k_i \in [0, 1]$. That i can be 1 or 2 signifies that we will create two combinations, one for $p_{\mathbf{D}_1}$ and one for $p_{\mathbf{D}_2}$. The Z_i are normalizing factors. The conditional distribution $p_{\mathbf{G}_{Col}}(\mathbf{d} \mid \mathbf{x}; \theta_{\mathbf{G}_{Col}})$ is obtained from the joint, in theory, by normalizing. In practice, we never actually need to compute any normalizing factors.[7]

To estimate k_i, we first train $\theta_{\mathbf{D}_1}{}'$ and $\theta_{\mathbf{D}_2}{}'$ on our WSJ tuning training sections $(0, 1, 5\text{-}22)$. We then use $\theta_{\mathbf{D}_1}{}'$ to estimate $\theta_{\mathbf{G}_{Col}}{}'$. Finally, we choose the value for k_i that maximizes the performance of the model that combines $\theta_{\mathbf{D}_i}{}'$ with $\theta_{\mathbf{G}_{Col}}{}'$ on our WSJ sections 2-4, with a simple search over values $[0, .01, \cdots, .99, 1]$. The optimal values for the k_i were $k_1 = .70$ and $k_2 = .71$.

6.2 Results

Table 4 compares our two combined models against our individuals models, and our Berkeley parser baseline. We see that the combined models outperform their component models on all tasks. That is, the $\text{LOP}_{\text{1st-order}}$ model that combines $p_{\mathbf{D}_1}$ and $p_{\mathbf{G}_{Col}}$ outperforms both $p_{\mathbf{D}_1}$ and $p_{\mathbf{G}_{Col}}$. And, the $\text{LOP}_{\text{2nd-order}}$ model that combines $p_{\mathbf{D}_2}$ and $p_{\mathbf{G}_{Col}}$ outperforms both $p_{\mathbf{D}_2}$ and $p_{\mathbf{G}_{Col}}$. This all adds up to a significant improvement over the performance of the Berkeley parser. And, we see that when parsing the out-of-domain Brown data, the first-order model performs as well or slightly better than the second-order model.

Recall that we have used a new data set. In terms of past work on the Ratnaparkhi et al. (1994) data set, recall that the state-of-the-art using labelled data alone is Stetina and Nagao (1997), with

[7] To see this, note that

$$\begin{aligned} &\operatorname*{argmax}_{\mathbf{d}} \frac{\left(\frac{f(\mathbf{d})}{Z_1}\right)^a \cdot \left(\frac{g(\mathbf{d})}{Z_2}\right)^b}{Z_3} \\ =&\operatorname*{argmax}_{\mathbf{d}} \frac{1}{{Z_1}^a {Z_2}^b Z_3} f(\mathbf{d})^a g(\mathbf{d})^b \\ =&\operatorname*{argmax}_{\mathbf{d}} f(\mathbf{d})^a g(\mathbf{d})^b \end{aligned}$$

88.1% accuracy, and with unlabelled data alone is Pantel and Lin (2000), with 84.5% accuracy. Our results compare favourably to these, though, of course, the comparison is indirect.[8]

Furthermore, both of these other author's models make use of semantic resources such as WordNet and similar words lists to combat the sparsity of combinations that occurs even in large unlabelled samples. The limited utility of second-order features based on only the stems of the $\mathbf{noun}_{\text{inside}}$ suggests to us that features based on some more general semantic concept might generalize to other domains better. What is interesting in this regard is the strength of our first-order model, $\text{LOP}_{\text{1st-order}}$, which achieves performance approaching or passing the previous work without looking at $\mathbf{noun}_{\text{inside}}$. This suggests to us that collective dependent generation is a useful technique, which can presumably be combined with the semantic resources of the authors just mentioned to make an even better model.

Returning to the results, while the performance on the out-of-domain Brown corpus looks very comparable to that on the WSJ for both our system and the Berkeley parser, it actually seems that the Brown corpus is somewhat "easier", in the sense that it contains more examples that can be settled on the basis of our two fairly reliable structural heuristics.[9] Table 5 shows the performance on examples in which the structural heuristics do not apply. Here, we see that performance on the out-of-domain Brown corpus lags significantly behind performance on the WSJ.

Those interested in the automatic grammar discovery technique of the Berkeley parser will note the significant drop in performance of that parser on this subset of the data which implies that this parser had done essentially perfectly in cases where our structural heuristics did apply, meaning it must have *learned* equivalent heuristics to those that we encoded by hand, which is noteworthy.

Finally, though we have so far reported only the binary decision score (cf. §3.2) for each derivation, Table 6 shows a more general score: the percentage of correct attachments in any example

[8]While both the Ratnaparkhi et al. (1994) test set and our WSJ test set are from the essentially the same material, they are not exactly the same examples, as the Treebank has undergone reordering.

[9]On WSJ sections 0, 1, 22-24, structural features fire in 33.8% of examples, while on Brown sections 0-2, 7-9 structural features fire in 46.0% of examples.

	WSJ		Brown	
Model	Dev.	Test	Dev.	Test
$p_{\mathbf{G}_{Col}}$ (4 EM its.)	83.8	81.6	86.3	85.4
$\text{LOP}_{\text{2nd-order}}$	**88.9**	**86.9**	86.4	**86.2**
$p_{\mathbf{D}_2}$ (2nd-order)	87.4	86.0	84.7	83.9
$\text{LOP}_{\text{1st-order}}$	87.5	86.6	**86.6**	**86.2**
$p_{\mathbf{D}_1}$ (1st-order)	86.2	86.0	84.7	83.0
Baseline	85.3	83.0	83.3	82.7

Table 4: Performance of the combined model. Score is the binary decision score (cf. §3.2).

	WSJ		Brown	
Model	Dev.	Test	Dev.	Test
$\text{LOP}_{\text{2nd-order}}$	**84.1**	**80.3**	76.0	76.2
$\text{LOP}_{\text{1st-order}}$	82.3	79.8	**76.1**	**76.3**
Baseline	78.7	74.4	70.0	69.0

Table 5: Performance of the combined model on examples that cannot be settled by our two structural constraints. I.e., examples where i) the preposition is not *of*, and ii) the direct object is not a pronoun. Score is the binary decision score (cf. §3.2)

(whether headed by noun or verb) in which more than one attachment was possible. Here, we see that the general problem is, as one would expect, harder than the binary decision problem.

6.3 Analysis: What does Unlabelled Data Change?

At this point, we ask what difference the unlabelled data makes. To answer this question, we consider the nature of the disagreements between the $p_{\mathbf{D}_1}$ itself, and the $\text{LOP}_{\text{1st-order}}$ model that mixes $p_{\mathbf{D}_1}$ with $p_{\mathbf{G}_{Col}}$.

In Table 7, the *Count* column shows the number of agreements and disagreements on the development test sets. One possible outcome would have been few disagreements between models and that each of these would be won by the combined model. In fact, a significant number of disagree-

	WSJ		Brown	
Model	Dev.	Test	Dev.	Test
$\text{LOP}_{\text{2nd-order}}$	**85.8**	**84.8**	**83.7**	**84.0**
$\text{LOP}_{\text{1st-order}}$	84.5	84.2	83.4	83.9
Baseline	82.6	80.1	81.2	82.2

Table 6: Performance of the combined model. Score is the percentage of prepositional phrases attached correctly in all cases in which more than one attachment was possible.

	WSJ Dev.		Brown Dev.	
Model correct	Count	$p_{\mathbf{D}_1}$ neutral	Count	$p_{\mathbf{D}_1}$ neutral
Both	1748	134	1396	227
$\text{LOP}_{\text{1st-order}}$	117	29	105	26
$p_{\mathbf{D}_1}$	87	9	72	12
Neither	176	37	161	49

Table 7: Analysis of the agreements and disagreements between the first-order discriminative classifier, $p_{\mathbf{D}_1}$, and the combined model, $\text{LOP}_{\text{1st-order}}$. The second row describes examples where $\text{LOP}_{\text{1st-order}}$ was correct but $p_{\mathbf{D}_1}$ was wrong, and the third row describes the converse situation. The *Count* column gives the number of examples in each category. The $p_{\mathbf{D}_1}$*neutral* column gives the number of examples in each category in which no features in the $p_{\mathbf{D}_1}$ model were active (the strategy is to pick attachment to verb in this case).

ments go each way, but the large majority are won by the combined model.

The $p_{\mathbf{D}_1}$ *neutral* category counts the number of times that none of $p_{\mathbf{D}_1}$'s features fire. These are the examples in which $p_{\mathbf{D}_1}$ has no opinion whatsoever, and so the combined model should have an advantage, if the unlabelled data is contributing useful information. We see that this is indeed the case. When no features fire for $p_{\mathbf{D}_1}$, the strategy, as noted, is to choose attachment to the verb, which should still lead to a large number of correct responses. Thus, it makes sense that some disagreements are won by $p_{\mathbf{D}_1}$, even when none of its features fire.

7 Conclusion and Future Work

We have shown that supervised techniques based on lexical dependency parsing outperform the semi-lexicalized strategy of Petrov and Klein (2007).

We have demonstrated that a properly chosen pair of models, one trained discriminatively from labelled data, and one trained generatively from unlabelled data, can be combined in a product model to yield a model better than either is alone.

We have shown that, when learning from unlabelled data, it may be preferable to generate dependents collectively.

Finally, we have introduced a pair of models which we think will be interesting to combine using the new methods for constraining EM, e.g., a la Ganchev et al. (2010) or Druck and McCallum (2010).

Acknowledgements

We thank Ioannis Konstas, Micha Elsner, and the reviewers for helpful suggestions. This work was supported by the Scottish Informatics and Computer Science Alliance and the ERC Advanced Fellowship 249520 GRAMPLUS.

References

Michaela Atterer and Hinrich Schütze. 2007. Prepositional phrase attachment without oracles. *Computational Linguistics*, 33(4):469–476.

R. F. Bordley. 1982. A multiplicative formula for aggregating probability assessments. *Management Science*, 28:1137–1148.

Eric Brill and Philip Resnik. 1994. A rule-based approach to prepositional phrase attachment disambiguation. In *COLING*, pages 1198–1204.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 957–961.

Michael Collins and James Brooks. 1995. Prepositional phrase attachment through a backed-off model. *CoRR*.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *EMNLP*, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B*, 39.

Gregory Druck and Andrew McCallum. 2010. High-performance semi-supervised learning using discriminatively constrained generative models. In *ICML*, pages 319–326.

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *ACL*.

Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11:2001–2049.

Daniel Gildea. 2001. Corpus variation and parser performance. In Lillian Lee and Donna Harman, editors, *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, EMNLP '01, pages 167–202, Stroudsburg. Association for Computational Linguistics.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2005. *English Gigaword Second Edition*. Linguistic Data Consortium: Philadelphia.

Donald Hindle and Mats Rooth. 1991. Structural ambiguity and lexical relations. In *ACL'91*, pages 229–236.

Geoffrey Hinton. 1999. Product of experts. In *ICANN*.

Percy Liang and Dan Klein. 2009. Online em for unsupervised models. In *HLT-NAACL*, pages 611–619.

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

Guido Minnen, John Carroll, and Darren Pearce. 2001. Applied morphological processing of english. *Nat. Lang. Eng.*, 7:207–223, September.

Patrick Pantel and Dekang Lin. 2000. An unsupervised approach to prepositional phrase attachment using contextually similar words. In *ACL*.

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

Slav Petrov. 2010. Products of random latent variable grammars. In *HLT-NAACL*, pages 19–27.

Adwait Ratnaparkhi, Jeffrey C. Reynar, and Salim Roukos. 1994. A maximum entropy model for prepositional phrase attachment. In *HLT*.

Adwait Ratnaparkhi. 1998. Statistical models for unsupervised prepositional phrase attachement. In *COLING-ACL*, pages 1079–1085.

Andrew Smith, Trevor Cohn, and Miles Osborne. 2005. Logarithmic opinion pools for conditional random fields. In *ACL*.

J. Stetina and M. Nagao. 1997. Corpus based pp attachment ambiguity resolution with a semantic dictionary. In *Natural Language Processing Pacific Rim Symposium*.

Kristina Toutanova, Christopher D. Manning, and Andrew Y. Ng. 2004. Learning random walk models for inducing word dependency distributions. In *ICML*.

Martin Volk. 2002. Combining unsupervised and supervised methods for pp attachment disambiguation. In *Proceedings of the 19th international conference on Computational linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Appendix A: Extracting Examples from Gold Trees

In this section, we describe the functional program used to extract examples from gold trees.

A *base noun phrase* is phrase labelled NP, which does not dominate any other phrases. A base noun phrase is identified with its head-word. Head words are found using the Java reimplementation of Collins (1999) head-finder in the Stanford NLP API.

Noun phrase examples extracted are those that match

$$[_{S} \ldots [_{NP_1} \text{BaseNP}_H \text{ (prep baseNP)}_i^* \ldots] \ldots]$$

NP_1 must be immediately dominated by an S, baseNP_H must be the left-most descendant of NP_1, and all (prep baseNP)_i substrings must be dominated by NP_1 as well.

Verb phrase examples are those that match

$$[_{VP_1} \text{verb}_H \text{ (PRT) (ADVP) (baseNP) (prep baseNP)}_i^+ \ldots]$$

VP_1 can occur in any environment. If there is a particle (PRT), this is appended to the verb. The adverbial phrase (ADVP) is ignored.