Training MST Parsers from Partially Annotated Corpora

Shinsuke MORI †1, DANIEL FLANNERY †2, YUSUKE MIYAO ‡3 and GRAHAM NEUBIG ‡2

We introduce a maximum spanning tree (MST) dependency parser that can be trained from partially annotated corpora, allowing us to maximize the use of available linguistic resources and reduce the costs of preparing new training data. This is especially important for domain adaptation in a real-world situation. Experiments on Japanese dependency parsing show that this approach allows for rapid training and achieves accuracy comparable to parsers trained on fully annotated data.

1. Introduction

Parsing is one of the fundamental building blocks of natural language processing, with applications ranging from machine translation1) to information extraction2). However, while statistical parsers achieve higher and higher accuracies on in-domain text, the creation of data to train these parsers is labor-intensive, which becomes a bottleneck for smaller languages. In addition, it is also a well known fact that accuracy plummets when tested on sentences of a different domain than the training corpus3),4), so in-domain data must be annotated to make up for this weakness.

In this paper, we propose a maximum spanning tree (MST) parser that helps ameliorate these problems by allowing for the efficient development of training data. This is done through a combination of a novel parsing method and an efficient corpus annotation strategy. For corpus construction, we use partial annotation5),6), which allows an annotator to skip annotation of unnecessary edges, focusing their efforts only on the ones that will provide the maximal gains of accuracy. In the parser, we make the assumption that the score of each edge is independent of the other edges in the dependency tree, which allows for simple training using either fully or partially annotated data.

Second, we perform experiments in both domain adaptation and small-data settings, and find that partial annotation allows for greater performance gains with comparable amounts of annotated data.

2. Pointwise estimation for dependency parsing

This work follows the standard setting of recent work on dependency parsing9). Given as input a sequence of words, \( w = (w_1, w_2, \ldots, w_n) \), the goal is to output a dependency
tree \( d = (d_1, d_2, \ldots, d_n) \), where \( d_i \equiv j \) when the head of \( w_i \) is \( w_j \). We assume that \( d_i = 0 \) for some word \( w_i \) in a sentence, which indicates that \( w_i \) is the head of the sentence.

The parsing model we pursue in this paper is McDonald et al.'s edge-factored model. A score, \( \sigma(d_i) \), is assigned to each edge (i.e. dependency) \( d_i \), and the parsing is to find a dependency tree, \( \hat{d} \), that maximizes the sum of the scores of all the edges.

\[
\hat{d} = \arg \max_d \sum_{d \in \mathcal{D}} \sigma(d).
\]

It has been known that, given \( \sigma(d) \) for all possible dependencies in a sentence, \( \hat{d} \) can be computed by the maximum spanning tree algorithm such as Chu-Liu/Edmonds' algorithm.

An important difference from McDonald et al.'s \cite{mcdonald2008} is in the estimation of \( \sigma(d) \). McDonald et al. \cite{mcdonald2008} applied a perceptron-like algorithm that optimizes a score of entire dependency trees. However, we stick to pointwise estimation: \( \sigma(d_i) \) is estimated for each \( w_i \) independently. A variety of machine learning-based classifiers can be applied to the estimation of \( \sigma(d) \), because it is essentially a n-class classification problem. In the experiments, we estimate a log-linear model \( p(d_i = j) \equiv p(j|w, i) \), and \( \sigma(d_i) \) is defined as log probability \( \log p(d_i) \). It should be noted that the probability depends only on \( i, j \), and the input \( w \), which assure that \( p(d_i) \) is estimated independently for each \( w_i \). Because parameter estimation does not involve computing \( \hat{d} \), we do not apply the maximum spanning tree algorithm in training.

Our current implementation uses features on the distance between a dependent word and its candidate head, the surface forms of the dependent/head words and their surrounding words (up to three words before/after the dependent/head words), and the parts-of-speech of the dependent/head words.

Pointwise estimation rather than structured estimation might hurt parsing accuracy. However, our method can enjoy greater flexibility, which allows for training from partially annotated corpora as will be described in Section 3.

In the experiments, we target Japanese parsing. Because Japanese is a head-final language, we assume \( d_i > i \) for all \( i \neq n \) and \( d_n = 0 \). This assumption reduces the maximum spanning tree algorithm to a simpler algorithm: for each word we select the dependency with the maximum score. This never creates a loop of dependencies, and a recursive process as in Chu-Liu/Edmonds' algorithm is not necessary.

### 3. Domain Adaptation for MST Parsing

Assuming that the cost of annotation corresponds roughly to the number of annotations performed, out of all possible annotations to have annotators perform for a target domain corpus we want to select the ones which provide the greatest benefit to accuracy when training. The high cost of annotation work is the primary motivation for this approach.

#### 3.1 Partial Annotation for a Parser

Before text can be annotated with dependencies for use in our system, it must first be tokenized and labeled with POS tags. \(^{\star2}\) We assume that the results of this tokenization and POS tagging are accurate enough that we need to manually annotate only the dependencies between the tokenized words.

In the context of dependency parsing, partial annotation refers to annotating only certain dependencies between words in a sentence. Dependencies which are assumed to

\(^{\star1}\) While we describe unlabeled dependency parsing for simplicity, it is trivial to extended it to labeled dependency parsing.

\(^{\star2}\) We take a language-independent approach that does not make any assumptions about the unit of tokenization or the meaning of tags used.
have little to no value for training are left unannotated. Figure 1 shows an example of a partially annotated sentence that can be used as training data by our system.

3.2 Estimating Edge Score from Partial Annotations

As explained in Section 2, edge scores, $\sigma(d_i)$, are estimated for each $w_i$ independently. This means that the estimation of $\sigma(d_i)$ requires only a gold dependency of $w_i$, and the other dependencies in a sentence are not necessary. This allows us to learn $\sigma(d_i)$ from partially annotated corpora. When training data includes a gold dependency that $w_i$ depends on $w_j$, a discriminative classifier like a log-linear model can be trained by regarding $d_i = j$ as a positive sample and $d_i = j'$ s.t. $j' \neq j$ as negative samples.

In the case of Japanese parsing, because $j > i$ for all $d_i = j$, negative samples are $d_i = j'$ s.t. $j' \neq j$ and $j' > i$. For example, from the partial annotation given in Figure 1, we can create a training instance for $w_2$, $\{ subj \}$, where the positive sample is $d_2 = 8$ and the negative samples are $d_2 = 3, 4, \ldots, 7, 9$.

3.3 Dependency Selection Criterion

The criterion we use to select words to annotate with their heads is based on the idea of additive smoothing\(^{(10)}\). The motivation for this idea is that it yields better performance than a simple maximum likelihood estimate when the size of the training corpus is small. The following is the procedure to annotate a corpus with $k$ dependencies.

1. Count the frequency of each word $f(w_i)$ in the training corpus.
2. Select $k$ words according to the following probability

$$\frac{\alpha + f(w)}{|W|\alpha + \sum_i f(w_i)},$$

where $W$ is the set of words appearing in the training corpus.
3. Annotate the selected words with their heads.

This criterion is very naive but is expected to work better than randomly selecting words to annotate. Theoretically speaking, it may be a good idea to annotate all dependencies in which the selected word appears, but for an annotator finding all of a word’s dependents is much more difficult than finding only its head. This is because a word has only one head, which for Japanese we assume always occurs to the right of that word in the sentence. In contrast, there could be multiple dependents.

3.4 Related Work

There has been a significant amount of work on how to utilize in-domain data to improve the accuracy of parsing. The majority of this work has focused on using unlabeled data in combination with self-training\(^{(11)}\)\(^{(12)}\) or other semi-supervised learning methods\(^{(13)}\)\(^{(14)}\). Roark and Bacchiani\(^{(11)}\) also present work on supervised domain adaptation, although this focuses on the utilization of an already-existing in-domain corpus.

There has also been some work on efficient annotation of data for parsing\(^{(16)}\)\(^{(17)}\). In particular Sassano and Kurohashi\(^{(18)}\) present a method for using partially annotated data with deterministic dependency parsers. In contrast, we present results for MST parsers, and demonstrate effectiveness in a domain adaptation scenario, where large amounts of labeled out-of-domain data are available.

4. Evaluation

As an evaluation of our parser, we measured parsing accuracies of several systems on test corpora in two domains: one is a general domain in which a fully annotated corpus annotated with word boundary and dependency information is available, and the other is a target domain assuming an adaptation situation in which only a partially annotated corpus is available for quick and low-cost domain adaptation.

4.1 Experimental Settings

In the experiments we used example sentences from a dictionary\(^{(10)}\) as the general domain data, and business newspaper articles (Nikkei), similar to the Wall Street Journal, as the target domain data. Their usages and specifications are shown in Table 1. All the sentences are segmented into words manually and all the words are annotated with

<table>
<thead>
<tr>
<th>ID</th>
<th>source</th>
<th>usage</th>
<th>#sentences</th>
<th>#words</th>
<th>#chars</th>
</tr>
</thead>
<tbody>
<tr>
<td>EHJ-train</td>
<td>example sentences</td>
<td>learning</td>
<td>11,700</td>
<td>145,925</td>
<td>197,941</td>
</tr>
<tr>
<td>EHJ-test</td>
<td>from a dictionary</td>
<td>test</td>
<td>1,300</td>
<td>16,348</td>
<td>22,207</td>
</tr>
<tr>
<td>NKN-train</td>
<td>newspaper</td>
<td>PA pool</td>
<td>9,023</td>
<td>263,427</td>
<td>398,570</td>
</tr>
<tr>
<td>NKN-test</td>
<td>articles</td>
<td>test</td>
<td>1,002</td>
<td>29,038</td>
<td>43,695</td>
</tr>
</tbody>
</table>

NKN-train is used as a partial annotation (PA) pool.
Table 2 Parsing Accuracy on EHJ-test.

<table>
<thead>
<tr>
<th>method</th>
<th>EHJ-test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Malt</td>
<td>96.63%</td>
</tr>
<tr>
<td>MST</td>
<td>96.67%</td>
</tr>
<tr>
<td>PW</td>
<td>96.83%</td>
</tr>
</tbody>
</table>

All systems were trained on EHJ-train.

For the first experiment, we measured the accuracy of each system on an in-domain test set when training on a fully annotated corpus. The results are shown in Table 2. Malt and MST have similar accuracy, but PW outperforms both of these systems. We also measured the training time and the parsing speed of each system. Table 3 shows the results. From this table, first we see that MST is much slower than Malt, as is well known. Our method, however, is much faster than MST and the parsing speed is approximately the same as the shift-reduce-based Malt.

Active learning has been shown to be effective at identifying informative examples to annotate for domain adaptation\(^\text{1,23,24}\), but since the system must be retrained after every set of annotations training time can become a bottleneck. Theoretically the training time of our method is proportional to the number of annotated dependencies. On this size of training corpus the training speed is fast enough that we can adopt an active learning framework in a real domain adaptation situation. This is one of the main advantages of our framework.

We performed a second experiment in the general domain to measure the impact of the training corpus size on parsing accuracy. To make smaller training corpora, we set a
fixed number of dependency annotations and then sequentially selected sentences from
EHIJ-train until the desired number of dependency annotations were collected. The
results are shown in Figure 2. Though PW achieves the highest accuracy when the full
training corpus is used, Malt has higher accuracy than both of the MST-based systems
when the training corpus is one-third or less the size of the full training corpus. It can
also be shown that both MST-based systems improve at a similar rate for all sizes of
training corpora.

We also tested two different annotation methods on training corpora of various sizes
in preparation for the domain adaptation experiment. We use PW as the baseline for
this comparison.

(1) **PW Additive Smoothing:** Our system, using the dependency selection crite-
ron we proposed in Section 3.3 to perform partial annotations. We set $\alpha = 0.5$.

(2) **PW Random:** Our system, using fully annotated sentences selected randomly
from the training data.

The results are shown in Figure 3. Comparing the different annotation methods for
PW, we see that the accuracy of PW Additive Smoothing is comparable to PW Ran-

dom and the baseline PW across the different sizes of training corpora. This shows that
the dependency selection criterion we proposed can make partial annotation a viable
alternative to full annotation methods.

### 4.3 Domain Adaptation with a Partially Annotated Training Corpus

One of the advantages of our parser is that it can be trained on a partially annotated
corpus. Thus for the domain adaptation experiment, we used EHIJ-train and a partially
annotated corpus built from newspaper data (NKN-train) according to the criterion
described in Section 3. For the partial annotation case, we set $\alpha = 0.005$.

The results are shown in Figure 4. When only 5k dependencies are annotated the
full annotation method gives higher parsing accuracy, but as the number of annotations
is increased the partial annotation method has higher accuracy. After 15k annota-
tions have been performed, the full annotation method requires almost 5k additional
annotations to match the accuracy of the partial annotation method.
5. Conclusion

We introduced an MST parser that evaluates the score for each edge in a dependency tree independently, which allows for the use of partially annotated corpora in training. When combined with a suitable strategy for selecting informative dependencies to annotate, training data can be prepared efficiently even in situations where data is limited, such as domain adaptation.

We evaluated state-of-the-art dependency parsers on a Japanese dependency parsing task, and found that our parser achieves accuracy comparable to that of a traditional MST parser. Additionally, the training and parsing speed of our parser is much faster than the traditional one, which allows it to be used for active learning in a real-world domain adaptation situation.

References


