RC23121 (W0402-128) February 26, 2004
Computer Science

IBM Research Report

An EM Algorithm for History-Based Statistical Parsers

Xiaoqiang Luo, Salim Roukos, Todd Ward
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Min Tang
Spoken Language Systems Group
MIT Laboratory for Computer Science
Cambridge, MA 02139

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

An EM Algorithm for History-based Statistical Parsers

Xiaoqiang Luo† and **Min Tang**‡
†1101 Kitchawan Road, Rte 134
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598, USA

Salim Roukos† and **Todd Ward**†
‡Spoken Language Systems Group
MIT Laboratory for Computer Science
Cambridge, Massachusetts 02139, USA

{xiaoluo,toddward,roukos}@us.ibm.com mtang@sls.lcs.mit.edu

Abstract

This paper aims to improve statistical parsing by making use of partially-labelled data from a different domain. The labeled part of a parse tree is regarded as "observation" and the unlabelled part as missing information. The expectation-maximization (EM) algorithm is employed to infer missing information. Nested parser states are used to implement the E-step efficiently. We train a series of model on the UPenn Chinese treebank, and use a POS-tagged corpus from Peking University (PKU) as the EM learning data. We observe a parsing error (measured by equal-weighted label F-measure) reduction by as much as about one-third when the seed model is under-trained. The usefulness of PKU data, as expected, decreases as the seed model is trained with more labeled data.

1 Introduction

Statistical approaches (Jelinek et al., 1994; Ratnaparkhi, 1997; Collins, 1997; Charniak, 2000) have been very successful in parsing natural language. The success, however, relies on the availability of a large corpus with human annotation – the Wall Street Journal (WSJ) Penn treebank (Marcus et al., 1993). For most other languages or tasks, it is difficult to get a corpus in WSJ's size. On the other hand, we can often find similar genre of text that are labeled in different styles, for different purposes, or by different institutions. Costliness of acquiring labeled data makes it desirable to either reduce the amount of labeled data needed to train a model, or make better use of existing data. One example of the former scheme is active learning (Thompson et al., 1999; Hwa, 2000; Tang et al., 2002), which selects the most "useful" sentences for annotation. The paper explores an orthogonal venue, namely, use of partially-labeled data to improve statistical parsing. In particular, we study a class of problems where a limited amount of labeled data (i.e., *in-domain*) is available, and we want to benefit from labeled data in one or more different corpora (i.e., *cross-domain*).

The very first question is: can we find similar and labeled text? The answer is often yes. For example, English Penn treebank is a corpus of one million word Wall Street Journal articles that are fully parsed by human. We also have access to an AP news wire corpus that are fully parsed by human, albeit in a different style than WSJ. Both the WSJ Penn treebank and AP corpus are general news articles. UPenn also released a similar Chinese treebank (CTB) (Xia et al., 2000), and Beijing University provides a POS-tagged Chinese corpus[1], part of which is publicly available (PKU corpus hereafter). Both UPenn Chinese treebank (CTB) and PKU are general news articles. Presumably, the use of cross-domain data will be most beneficial when the amount of training data is small, as is the case of CTB.

More often than not, human labeling for a different corpus/domain can not be used directly in the domain we are interested in, as some information may be missing, or annotation style may be different, or a totally different label set may be used. *The goal of this study is to take advantage of partial information provided in cross-domain annotation*. We formulate it as a missing-data problem where partial information contained in cross-domain annotation is treated as "observation" while structure not annotated as "missing" information. The general set-up is that a seed model is learned from in-domain labeled data. Expectation-maximization (EM) (Dempster et al., 1977) is employed to infer missing structures for cross-domain with partial labels, which, together with in-domain data, is used to retrain the model.

The proposed algorithm is implemented in the framework of maximum entropy parser (Ratnaparkhi, 1997).

[1] See http://icl.pku.edu.cn/Introduction/corpustagging.htm

For the sake of completeness, a quick review is included in Section 2.1. A few examples of parse trees with missing structures are shown in this section too. In Section 2, an instance of EM algorithm is developed to handle missing information from cross-domain data. Nested parser states are used to implement the algorithm efficiently, which is covered in Section 3. Experimental results are reported in Section 4. Related work is discussed in Section 5.

2 EM algorithm for History-based Parsers

2.1 Derivation and Missing Information

A history-based statistical parser computes $P(T|S)$, the conditional probability of a parse tree T given a sentence S. A history-based parser typically converts T into a unique sequence of parse actions, i.e., $T = a_1, a_2, \cdots, a_{N_T}$, where N_T is the number of actions needed to *derive* T. Conversely, a valid parse sequence reconstructs a unique parse tree. Therefore, a parse tree T and its sequential representation are equivalent. The order by which a parse tree T is converted into the equivalent representation $T = a_1, a_2, \cdots, a_{N_T}$ is dubbed as *derivation*.

Since

$$P(T|S) = P(a_1, \cdots, a_{N_T}|S) \quad (1)$$

$$= \prod_{i=1}^{N_T} P(a_i|S, a_1^{(i-1)}), \quad (2)$$

the problem reduces to computing $P(a_i|S, a_1^{(i-1)})$. In the maximum entropy parser (Ratnaparkhi, 1997), a parse T is decomposed deterministically into a sequence of `tag`, `chunk`, `extend` and `reduce` (i.e., checking whether to close a constituent) actions. an exponential model is adopted to model $P(a_i|S, a_1^{(i-1)})$. Description of training algorithm of the exponential model is beyond the scope of the paper. Interested readers are referred to (Ratnaparkhi, 1997; Berger et al., 1996). Instead, to put our work in perspective, we will show by an example how a parse tree is decomposed as a sequential parse actions and how an unspecified tree structure corresponds to missing parse actions.

Figure 1 is a parse tree with its numbered parse actions $\{a_i\}_{i=1}^{N_T}$. There are four types of actions: `tag` (action 1 to 4), `chunk` (action 5 to 8), `extend` (action $9, 11, 13$ and 15) and `reduce` (action $10, 12, 14$ and 16). Each type has its own vocabulary of actions and probabilistic model. Thus the model $P(a_i|S, a_1^{(i-1)})$ is one of the four models: tag model $P_t(\cdot|\cdot)$, chunk model $P_c(\cdot|\cdot)$, extend model $P_e(\cdot|\cdot)$ and reduce model $P_r(\cdot|\cdot)$. A more precise

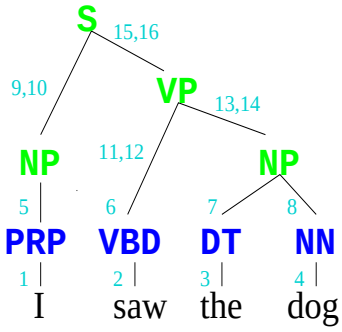


Figure 1: Parse actions (cf. Table 1): 1-4 – tag actions; 5-8 – chunk actions; 11-16 – extend and reduce actions. An extend action is always followed by a reduce action.

equation of (2) is:

$$P(T|S) = P(a_1, \cdots, a_{N_T}|S) \quad (3)$$

$$= \prod_{x \in \mathcal{T}_t} P_t(x|S, h_x) \prod_{y \in \mathcal{T}_c} P_c(y|S, h_y)$$

$$\prod_{z \in \mathcal{T}_e} P_e(z|S, h_z) \prod_{u \in \mathcal{T}_r} P_r(u|S, h_u), \quad (4)$$

where $\mathcal{T}_t, \mathcal{T}_c, \mathcal{T}_e$ and $\mathcal{T}_r$ are the set of tagging, chunking, extending and reduce actions, respectively, and h_x, h_y, h_z and h_u are short-hands for actions proceeding to x, y, z and u, respectively. The order of these actions is: tagging is done first from left to right; then chunking from left to right too; extending action starts from left too and it alternates with a reduce action, that is, immediately after an extending action, a reduce action is taken to see if a constituent resulted from the extending action should be closed or not. Table 1 summarizes parse actions corresponding to the parse tree in Figure 1, where action 1-4, 5-8, 9,11,13,15 and 10,12,14,16 are $\mathcal{T}_t, \mathcal{T}_c, \mathcal{T}_e$ and $\mathcal{T}_r$, respectively.

It is worth pointing out that the history of a model component can include a parse action of the other component. For example, while chunking the first word `I` (the fifth action in Table 1), the first tag action – `I` tagged as `PRP` – is part of the history h_y in (4). When a sentence is long, this type of dependency between two parse actions is common. Therefore, it is inappropriate to make Markovian assumption while building conditional models in (4). We will see shortly that this has implication for the EM algorithm implementation.

Since a parse tree can be decomposed as a sequential parse actions, unspecified parse tree structure corresponds to missing parse actions. For example, in subfigure (a) of 2, only POS tags are specified while constituent labels and high-level structures are not available – the dashed-lines and boxes indicate a possible completion of the POS-tagged subtrees. When represented by

Order	action type	value
1	TAG	PRP
2	TAG	VBD
3	TAG	DET
4	TAG	NN
5	CHUNK	startNP
6	CHUNK	other
7	CHUNK	startNP
8	CHUNK	join
9	EXTEND	extS
10	REDUCE	NO
11	EXTEND	extVP
12	REDUCE	NO
13	EXTEND	join
14	REDUCE	YES
15	EXTEND	join
16	REDUCE	YES

Table 1: Parse actions of the parser tree in Figure 1. The first column contains action indices in Figure 1.

parse actions, missing part of the parse tree are parse actions from 5 to 16 in Table 1.

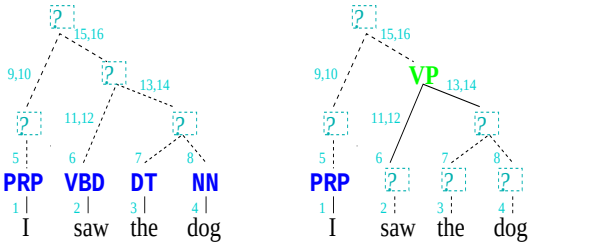
Partially-labeled data imposes constraints, and constraints can come in many forms. Sometimes only certain constituent boundaries are given while their internal structures may be left unspecified. Sometimes constituent labels are given together with boundaries. Sometimes there can be mixed tag and constituent label constraints. Subfigure (b) of 2 is such an example, where `saw the dog` is marked as `VP`, and the first tag `PRP` is specified, but other POS tags and the `VP`'s internal structure are missing.Subfigure (b) shows one of possible completions that are consistent with given constraints. In general, a constraint reduces parse actions. For instance, the word `saw` can not end a phrase which includes the word `I` because this would cause crossing brackets in the subfigure (b) of Figure2.

We intend to develop a learning algorithm that can accommodate arbitrarily partial labeling in the context of history-based parsing. To that end, we treat given structures or corresponding parse actions as "observation", and missing structures, or corresponding parse action as "hidden", and employ the EM algorithm to tackle this problem.

2.2 EM Algorithm for Partially-labeled Data

Formally, let $T = (T_a, T_m)$ be a parse tree for a sentence S, where T_a is the annotated part, and T_m the missing part. The set of parse trees for the sentence S is denoted as

$$\mathcal{T}(S) = \{(T_a, T_m) : T_m \texttt{ compatible with } T_a\}. \quad (5)$$



(a) tag constraints (b) tag, bracket and label constraint

Figure 2: Solid line and tag/label are constraints. The dotted-line shows a possible completion of given constraints. Missing tag/labels are indicated by enclosed '?'s. (a) POS tags are given. (b) The 1st tag and VP are constraints. The internal structure of VP is missing.

We can also define the set of missing structures to be

$$\mathcal{T}_m(S) = \{T_m : (T_a, T_m) \texttt{is a tree of } S\}. \quad (6)$$

T_m being compatible with T_a in (5) means that T_a and T_m together form a valid parse tree for S. In subfigure (a) of Figure 2, for example, T_a is the four tags and T_m represents any structure on top of the four tags that form a tree.

Let θ be the set of model parameters. We will write the model $P(T|S)$ as $P_\theta(T|S)$ from now on to highlight parameter values. Our goal is to maximize the probability of "observation", or given annotation T_a:

$$\max_\theta P_\theta(T_a|S) = \max_\theta \sum_{T_m \in \mathcal{T}_m(S)} P_\theta(T_a, T_m|S). \quad (7)$$

Maximizing directly (7) is difficult. However, if we regard T_a as observation and T_m as missing data, the EM (Dempster et al., 1977) algorithm can be utilized to derive an iterative procedure as follows. Recall that the EM auxiliary function is

$$Q(\theta'|\theta) = E[\log P_{\theta'}(T|S)|T_a, S, \theta], \quad (8)$$

which is the conditional expectation of the complete data given the "observation" T_a and S. A simple expansion leads to

$$\begin{aligned} Q(\theta'|\theta) &= E[\log P_{\theta'}(T|S)|T_a, S, \theta] \\ &= \sum_{T \in \mathcal{T}(S)} P_\theta(T|T_a, S) \log P_{\theta'}(T|S) \quad (9) \\ &= \sum_{T \in \mathcal{T}(S)} \frac{P_\theta(T|S)}{P_\theta(T_a|S)} \log P_{\theta'}(T|S) \quad (10) \\ &= \frac{1}{P_\theta(T_a|S)} \sum_{T \in \mathcal{T}(S)} P_\theta(T|S) \log P_{\theta'}(T|S) \quad (11) \end{aligned}$$

(9) to (10) is due to the fact that $P(T, T_a|S) = P(T|S)$ and $P(T|T_a, S) = \frac{P(T|S)}{P(T_a|S)}$. The denominator $P_\theta(T_a|S)$ is just $\sum_{T_m} P\big((T_m, T_a)|S\big)$.

It is straightforward to show that $Q(\theta'|\theta) \geq Q(\theta|\theta)$ implies $P_{\theta'}(T_a|S) \geq P_\theta(T_a|S)$. Therefore, instead of maximizing (7), we can maximize iteratively (11).

Two special cases of (11) are intuitively easy to understand: when all training data is fully labeled, $\mathcal{T}$ consists of only $T = T_a$ and (11) degenerates into the familiar training likelihood: $\log P_{\theta'}(T|S)$; On the other hand, if no supervision information is available, iteratively maximizing (11) is nothing but minimizing the training entropy.

(11) is obtained for a single training sentence S. Extension to multiple sentences is trivial. We need only to sum (11) over S, assuming that each sentence S gets equal weight.

(11) provides us with the familiar EM procedure to train the model: we start with a seed model with the parameter θ, then for each sentence S, find all parse trees $\mathcal{T}_m(S)$ compatible with T_a, normalize the probability $P_\theta(T|S)$ by their sum, $P_\theta(T_a|S)$ (E-step), and use them as "counts" to get the next optimal θ' (M-step). The procedure is repeated until the change of training likelihood falls below a threshold. However, naive implementation of the procedure is very costly: the set $\mathcal{T}_m$ can be prohibitive large if T_a does not contain many constraints and S is reasonably long. Thus, a more efficient algorithm is needed.

In the framework of probabilistic context-free grammar (PCFG), the inside-outside (Baker, 1979) algorithm has been developed to estimate parameters efficiently, inspired by the forward-backward algorithm in training hidden Markov models (HMM) (Baker, 1975). History-based statistical parsers (e.g., (Magerman, 1995) and (Ratnaparkhi, 1997)) do not use explicit rules, and, the Markov assumption between parse actions, as discussed in Section 2.1, is violated. Violation of Markov assumption makes it impossible to construct a trellis as used in the forward-backward algorithm.

Nevertheless, nested parser states can be organized as a tree to represent compactly the hypothesized parses.

3 Efficient Implementation Via Nested Parser State

The key quantity needed for optimizing (11) is $P_\theta(T_a, T_m|S)$. Once we have $P_\theta(T_a, T_m|S)$, the denominator $P_\theta(T_a|S)$ is just a sum of $P_\theta(T_a, T_m|S)$ over T_m. After the posterior probability $\frac{P_\theta(T_a,T_m|S)}{P_\theta(T_a|S)}$ is computed, θ' can be optimized by the same algorithm used in training the parser with fully-annotated data: in the case of maximum entropy model, the empirical expectation is the accumulated score $\frac{P_\theta(T_a,T_m|S)}{P_\theta(T_a|S)}$ for each feature, and the *same iterative scaling algorithm* (Berger et al., 1996) can be applied with these fractional "empirical" counts.

It has been pointed out that the set of missing structure $\mathcal{T}_m$ can be large, and it is essential to compute $\Sigma\frac{P_\theta(T_a,T_m|S)}{P_\theta(T_a|S)}$ efficiently. Nested parser state is used to serve this purpose.

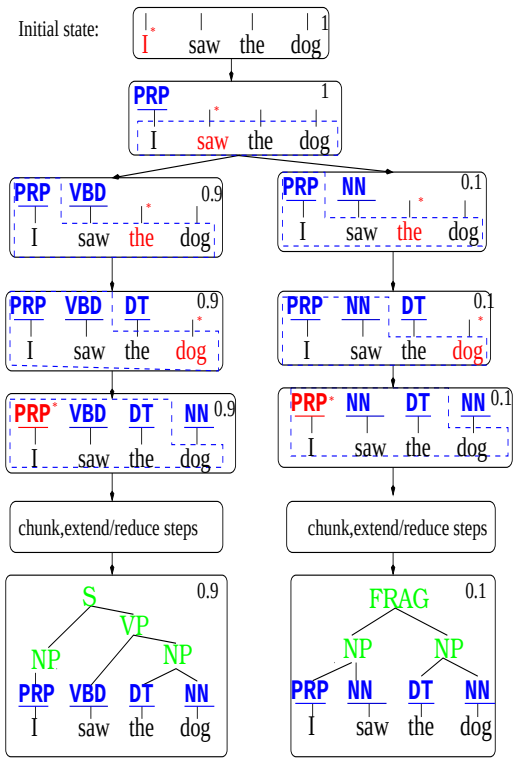


Figure 3: Each solid box represents a state. Arrows show evolution of parser states corresponding to two possible parses in the two bottom boxes. Active subtree is indicated by red *. Note that after all words are tagged, the first tagged subtree becomes active for the next chunking action. Subtrees enclosed in blue dashed lines are shared from parent states. Chunking, extending/reducing actions are omitted due to space limit. The number on the up right corner is the cumulative score of a parser state.

Before we discuss the nested parser state, we need to talk about briefly how search is done in a history-based parser. The initial state consists of a forest of $|S|$ degenerated trees, each of which is just a single node containing a word. The score of the initial state is 1. The first subtree (i.e., the first word) is tagged and for each possible tag, a new parser state is created, and the hypothesized tag along with its score is stored. In constrained decoding, only tags compatible with constraints are retained. For each newly-created parser state, the parser moves on and tags the second subtree (the second word). Again, a new parser state is created for each hypothesized tag. The process is repeated until all words are tagged. Then control is passed to chunking, which operates in the same manner. The parsing process ends when all subtrees are

joined under a single root node.

Figure 3 shows the evolution of parser states for the input sentence `I saw the dog`. For the convenience of drawing, the picture is made under the assumption that the existing model only allows an alternative POS tag for the word `saw` and all other parse action has probability 1. From the picture, we know that a parser state consists of a forest of trees, the current score and what the next parse action is. A new parser state is evolved from an old one, or its parent state $(S, a_1^{(i-1)})$, together with a future action a_i and its associated probability $P(a_i|S, a_1^{(i-1)})$. Note that a parser state shares a significant amount of information with its parent state. The common subtrees are shared across parser states. This is highlighted by dashed blue lines.

Observe that each parser state in Figure 3 corresponds to a parse event that will appear in one of the possible parse outcomes. Therefore, we can first construct a tree of parser states as shown in Figure 3. A leaf node of this tree corresponds to a complete parse. We then back track, starting from each leaf, and dump a parse event at each state as we traverse back to the root. Note that each parse state needs to be visited once.

Compared with the straightforward implementation of E-step of (11), an evolution tree of parser states like Figure 3 keeps track of all parse actions[2], and is effectively a compact representation of top-N parse trees. The benefit of using this nested parser state tree is obvious: we do not need to count an event twice, as one would have to if first top-N parses are dumped, and the model is retrained.

4 Experimental Results

The EM algorithm developed in Section 2 can be applied to a class of problems that training data is only partially annotated. Since partial annotation can often be more efficiently obtained, statistical parsers can thus achieve better performance with less demanding of labeling efforts. We consider this scenario as *in-domain* learning. A related but more interesting scenario is *cross-domain* learning, that is, to take advantage of corpora from other domains, e.g. POS tagging. Partial annotations are derived through some automated processes and annotations in other domains are recycled.

4.1 In-Domain EM Learning on WSJ and CTB

The very first question for semi-supervised learning is: how much supervision information is needed? Are POS tags and constituent labels equally important? To answer these questions, we conduct two groups of in-domain experiments: 1) strip all POS tags and vary the amount of labels in the training data; 2) strip all labels and keep POS tags only in the training data. Partially-labeled data created this way are used to simulate the EM learning process.

Experiments are carried out on both the WSJ Penn treebank and the Chinese Penn treebank. CTB is used to build a Chinese character parser. To train the Chinese character parser, we convert the word-based Chinese treebank into character-based. For WSJ, we start with an initial model trained on Section 2 and 3. The rest of the training data, namely Section 4 to 21 are reserved for the EM learning experiments. Section 23 is used as the test set. For CTB, we start with the official LDC treebank with about $100K$ words, and use another $120K$ words of unofficial data as the EM set. The last 400 sentences of the official LDC release is used as the test set. Table 2 shows the performance of the initial model, as well as the full models' performance. The average sentence length in Table 2 is in characters under the column "Chinese".

	WSJ		Chinese	
	Train	Full Set	Train	Full Set
Size	83.8k	950k	100k	220k
F-measure	0.788	0.850	0.748	0.827
Avg. Length	24		40	

Table 2: Label F-measures for WSJ and CTB

4.1.1 Results with Partial Labels

In this set of experiments, we remove all POS tags and vary the amount of labels in the EM data. One EM iteration is carried out. Fig 4 shows the percent changes (relative to the full model) of F-measure versus the percentage of labels used in the EM learning. As can be seen, the changes of F-measure are roughly linear to the amount of labels we keep in the EM learning. While raw data can not benefit the EM process, it takes only a small fraction of the total labels (6% for WSJ, and 16% for CTB) for the EM algorithm to learn helpful information, even when POS tags are absent.

Note that the learning rate of the EM algorithm with the Chinese parser is much slower. This is because the amount of EM data in the Chinese experiment is only about one-seventh of the WSJ EM data. Another reason is that the WSJ baseline model is better.

4.1.2 Results with POS Tags

In this set of experiments, only POS tags of the EM data are kept and all labels are stripped. For the Chinese character parser, character POS tags are inherited from word-level POS tags, plus a letter indicating the position of a character.

It turns out that POS tags are very helpful to the parser: For CTB, Fig 5 shows that POS tags contributes to about

[2] In practice, pruning is done: locally when adding an action, and globally when the cumulative score falls below a threshold relative to the best hypothesis.

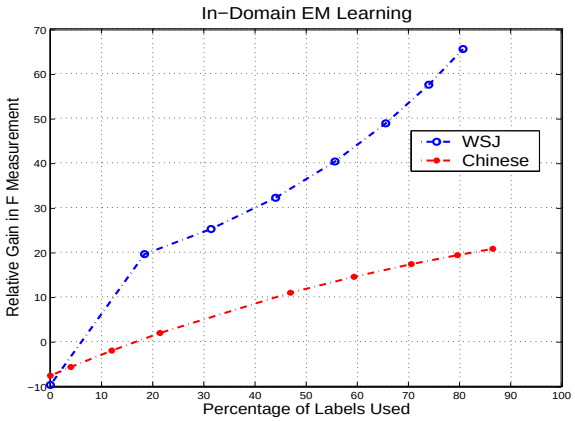


Figure 4: In-domain EM learning with labels. POS tags are removed in these experiments. It shows that percent changes of F-measures are roughly linear to the amount of labels. With the Chinese character parser, the learning rate is much slower. Raw data hurts performance.

one-half of the gain. For WSJ, a 14.5% relative gain is obtained. This is not very surprising, as Chinese character-level POS tags encode word segmentation information; and word segmentation and word sense disambiguation are extremely important for character-based parsing. Also the average length of a Chinese sentence is 40 characters, two-thirds longer than the average length of a WSJ sentence, as measured in modeling units (although similar in word count). Pruning errors pose a more severe problem for longer sentences. The presence of POS tags reduces more pruning errors, as POS tagging is done first in the parser.

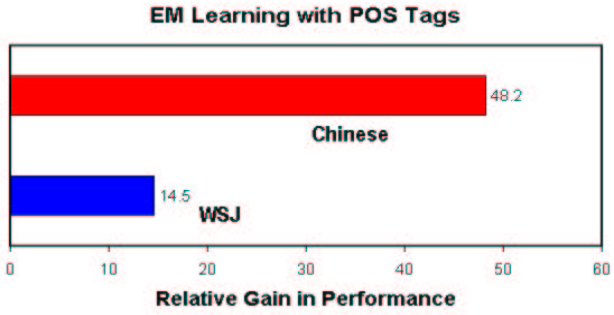


Figure 5: In-domain EM learning with POS tags.

4.2 Cross-Domain EM Learning

Our ultimate goal of this study is to make use of labeled data from a cross-domain to help the Chinese character parser. In-domain simulation results indicate that POS tags are more useful than constituent labels. Therefore, we consciously select the PKU POS-tagged corpus as the EM data.

PKU corpus uses a different and larger POS set. So the PKU-style tags are first mapped to the CTB-style. For the most part, the mapping is a table look-up, although not always 1-to-1 or m-to-1. For example, the particle "De (Bai2-Shao2)" in CTB can be tagged as either `DEC`, `DEG`, `AS`, or `SP`, depending on the context it is used; but in the PKU corpus, a single tag `u` is used for this character. When a POS cannot be mapped without context, the target (Penn-style) POS will be left unspecified. After doing this, we are able to map about 93% characters of PKU data into the Penn-style.

We train the seed models with various amounts of UPenn CTB data, and use $100K$ words of PKU data for EM learning. Results are tabulated in Table 3. The PKU data with POS tags is most useful when the seed model is trained with insufficient data. As high as 33.1% error reduction is observed when the seed model is trained with $40K$ words of UPenn CTB. The improvement decreases when the seed model is trained with more data. If we start with a seed model trained with $100K$ words, only 5.9% reduction is got.

Compared with in-domain CTB experiments, the gain from cross-domain is much smaller. Two reasons contribute to this difference: first, EM data from in-domain data are in the same style, while mapping from cross-domain tags will introduce noises; Second, we find sometimes PKU and Penn treebank have different word-segmentations. No attempt is made to recover from this error. Furthermore, we lose about 7% tags after the mapping.

Train(K wds)	baseline	+PKU	Err. Reduction(%)
20	0.585	0.699	27.5
40	0.637	0.757	33.1
80	0.730	0.764	12.7
100	0.748	0.763	5.9

Table 3: Label F-measures on UPenn CTB: the 1st column is training set size (in K words), and the 2nd and 3rd columns are F-measure before and after using $100K$-word PKU data. The last column is the relative reduction of errors.

In summary, the in-domain analysis provides details of the type and amount of supervision information and their relative importance to a history-based parser. When training data is limited, which is often the case with a new domain or task, such information can point us to the right cross-domain corpus whose annotation can be re-used. Our experiments show that cross-domain data can be very helpful. Moreover, by utilizing cross-domain data, we can improve a parser without labeling extra data.

5 Related Work

As supervision information is costly to obtain, unsupervised and semi-supervised methods have been studied in many areas of natural language processing such POS tagging (Merialdo, 1994), word sense disambiguation (Yarowsky, 1995) and document classification (McCallum and Nigam, 1998). Mitchell (1999) has a general discussion of use of unlabeled data in supervised learning. Banko and Brill (2001) includes a result of using large amount of unlabeled data in a task of disambiguating English confusable words.

In the area of natural language parsing, Pereira and Schabes (1992) extended the inside-outside algorithm to incorporate partially-bracketed training data. That is, when carrying out the inside-outside algorithm, only constructs consistent with bracket constraints are used. The proposed algorithm is tested on the ATIS (Hemphill et al., 1990) task and significant better results are obtained, compared with a PCFG trained with raw text. The same technique was later applied to the WSJ Corpus (Schabes et al., 1993). Independently, Black et al. (1992) used the same technique in training a statistical grammar of computer manuals. Both (Pereira and Schabes, 1992) and our work are about taking advantage of partially-labeled data, however, CFG (and its probabilistic variation) is very different from the formalism of history-based statistical parsers (e.g., (Ratnaparkhi, 1997) and (Jelinek et al., 1994; Magerman, 1995)). In particular, neither (Jelinek et al., 1994; Magerman, 1995) nor (Ratnaparkhi, 1997) maintains an explicit list of CFG rules. This difference leads to different training algorithms: inside-outside re-estimates probabilities of CFG rules, while an improved iterative scaling (Berger et al., 1996) algorithm is used to estimate parameters in maximum entropy model. Therefore, while (Pereira and Schabes, 1992) and our work share the same aim, the actual problem is quite different. Lack of inside-outside-type of algorithm is a major hurdle in implementing the EM algorithm in history-based statistical parser, and, to the best of our knowledge, this work is a first attempt of solving the problem.

Another related work is (Charniak, 1997). In one of the experimental results in (Charniak, 1997), 30 million raw text was parsed with a seed model and then the best parses were added to training data. The same training scheme can be implemented in our work. That is, taking the best parsing result is a special case of the E-step in that the best parse is used to approximate the summation of (11) (or Viterbi training).

In the 2002 Johns Hopkins Summer Workshop, a group worked on semi-supervised training for natural language parsing (Steedman et al., 2003). Their focus is using co-training (Blum and Mitchell, 1998; Sarkar, 2001) to improve statistical parsing. The co-training paradigm uses *multiple* statistical models (Collins, 1997; Sarkar, 2001) built on a (small) amount of labeled data, and then one model "generates" labels for another model. A selection step, based on various utility measures, is used to pick up "useful" training sentences. The co-training is shown useful when the seed model is insufficiently trained, but the gain disappears if the amount of initially labeled data is reasonably large (Steedman et al., 2002). While developing the EM algorithm for history-based statistical parsers, our focus is semi-supervised learning, with the aim of re-using labeled data from another domain.

6 Conclusions and Future Work

We have proposed an instance of the EM algorithm to incorporate partially-labeled data in statistical natural language parsing. An efficient implementation of the E-step is proposed using nested parser states. The proposed algorithm is useful in taking advantage of multiple corpora with different labeling. We observe that significant improvement can be obtained if an under-trained seed model is enhanced with partially-labeled data.

When making use of cross-domain data, it is often necessary to map one style of labeling into another. This requires domain knowledge. It would be nice if we can take cross-domain data as it is, and use it as an extra source of knowledge when training statistical models. Maximum entropy modeling provides us with a flexible framework to incorporate multiple sources of information. We will investigate this possibility in our future work.

7 Acknowledgments

[Omitted.]

References

J.K Baker. 1975. Stochastic modeling for automatic speech understanding. In D. R. Reddy, editor, *Speech Recognition*. Academic Press.

J.K. Baker. 1979. Trainable grammars for speech recognition. In Jared J. Wolf and Dennis H. Klatt, editors, *Speech Communication papers presented at the ninth Meeting of the Acoustic Society of America*.

Michele Banko and Eric Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*.

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, March.

E. Black, J. Lafferty, and S. Roukos. 1992. Development and evaluation of a broad-coverage probabilistic grammar of English-language computer mannuals. In *Proc. Annual Meeting of ACL*, pages 185–192.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*.

E. Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the 14th AAAI*, Menlo Park, CA.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL*, Seattle.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proc. Annual Meeting of ACL*, pages 16–23.

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, Series B*, 39(1):1–38.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS spoken language system pilot corpus. In *DARPA Speech and Natural Language Workshop*.

Rebecca Hwa. 2000. Sample selection for statistical grammar induction. In *Proc.* 5^{th} *EMNLP/VLC*, pages 45–52.

F. Jelinek, J. Lafferty, D. Magerman, R. Mercer, A. Ratnaparkhi, and S. Roukos. 1994. Decision tree parsing using a hidden derivation model. In *Proc. Human Language Technology Workshop*, pages 272–277.

D. Magerman. 1995. Statistical decision-tree models for parsing. In *Proc. Annual Meeting of ACL*, pages 276–283.

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

Andrew McCallum and Kamal Nigam. 1998. Employing EM and pool-based active learning for text classification. In *Machine Learning: Proceedings of the Fifteenth International Conference (ICML '98)*, pages 359–367.

B. Merialdo. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–172.

T. M. Mitchell. 1999. The role of unlabelled data in supervised learning. In *Proceedings of the Sixth International Colloquium on Cognitive Science*, San Sebastian, Spain.

F. Pereira and Y. Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proc. Annual Meeting of ACL*, pages 128–135.

Adwait Ratnaparkhi. 1997. A Linear Observed Time Statistical Parser Based on Maximum Entropy Models. In *Second Conference on Empirical Methods in Natural Language Processing*, pages 1 – 10.

Anoop Sarkar. 2001. Applying co-training methods to statistical parsing. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics*.

Y. Schabes, M. Roth, and R. Osborne. 1993. Parsing the wall street journal with Inside-Outside algorithm. In *Proceedings of the Sixth Conference of EACL*, pages 341–347.

M. Steedman, S. Baker, J. Crim, S. Clark, J. Hockenmaier, R. Hwa, M. Osborne, P. Ruhlen, and A. Sarkar. 2002. Semi-supervised training for statistical parsing. Technical report, JHU Summer Workshop. Also http://www.clsp.jhu.edu/ws2002/groups/wide/allfinal.pdf.

Mark Steedman, Miles Osborne, Anoop Sarkar, Stephen Clark, Rebecca Hwa, Julia Hockenmaier, Paul Ruhlen, Steve Baker, and Jeremiah Crim. 2003. Bootstrapping statistical parsers from small datasets. In *Proceedings of the 11th Conference of EACL*.

Min Tang, Xiaoqiang Luo, and Salim Roukos. 2002. Active learning for statistical natural language parsing. In *Proc. Annual Meeting of ACL*.

Cynthia A. Thompson, Mary Elaine Califf, and Raymond J. Mooney. 1999. Active learning for natural language parsing and information extraction. In *Proc.* 16^{th} *International Conf. on Machine Learning*, pages 406–414. Morgan Kaufmann, San Francisco, CA.

F. Xia, M. Palmer, N. Xue, M.E. Okurowski, J. Kovarik, F.D. Chiou, S. Huang, T. Kroch, and M. Marcus. 2000. Developing guidelines and ensuring consistency for Chinese text annotation. In *Proc of the 2nd Intl. Conf. on Language Resources and Evaluation (LREC 2000)*.

D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196.