# A Maximum Entropy Chinese Character-Based Parser

**Xiaoqiang Luo**
1101 Kitchawan Road, 23-121
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
xiaoluo@us.ibm.com

## Abstract

The paper presents a maximum entropy Chinese character-based parser trained on the Chinese Treebank ("CTB" henceforth). Word-based parse trees in CTB are first converted into character-based trees, where word-level part-of-speech (POS) tags become constituent labels and character-level tags are derived from word-level POS tags. A maximum entropy parser is then trained on the character-based corpus. The parser does word-segmentation, POS-tagging and parsing in a unified framework. An average label F-measure 81.4% and word-segmentation F-measure 96.0% are achieved by the parser. Our results show that word-level POS tags can improve significantly word-segmentation, but higher-level syntactic strutures are of little use to word segmentation in the maximum entropy parser. A word-dictionary helps to improve both word-segmentation and parsing accuracy.

## 1 Introduction: Why Parsing Characters?

After Linguistic Data Consortium (LDC) released the Chinese Treebank (CTB) developed at UPenn (Xia et al., 2000), various statistical Chinese parsers (Bikel and Chiang, 2000; Xu et al., 2002) have been built. Techniques used in parsing English have been shown working fairly well when applied to parsing Chinese text. As there is no word boundary in written Chinese text, CTB is manually segmented into words and then labeled. Parsers described in (Bikel and Chiang, 2000) and (Xu et al., 2002) operate at word-level with the assumption that input sentences are pre-segmented.

The paper studies the problem of parsing Chinese unsegmented sentences. The first motivation is that a character-based parser can be used directly in natural language applications that operate at character level, whereas a word-based parser requires a separate word-segmenter. The second and more important reason is that the availability of CTB, a large corpus with high quality syntactic annotations, provides us with an opportunity to create a highly-accurate word-segmenter. It is widely known that Chinese word-segmentation is a hard problem. There are multiple studies (Wu and Fung, 1994; Sproat et al., 1996; Luo and Roukos, 1996) showing that the agreement between two (untrained) native speakers is about upper 70% to lower 80%. The agreement between multiple human subjects is even lower (Wu and Fung, 1994). The reason is that human subjects may differ in segmenting things like personal names (whether family and given names should be one or two words), number and measure units and compound words, although these ambiguities do not change a human being's understanding of a sentence. Low agreement between humans affects directly evaluation of machines' performance (Wu and Fung, 1994) as it is hard to define a gold standard. It does not necessarily imply that machines cannot do better than humans. Indeed, if we train a model with consistently segmented data, a machine may do a better job in "remembering" word segmentations. As will be shown shortly, it is straightforward to encode word-segmentation information in a character-based parse tree. Parsing Chinese character streams therefore does effectively word-segmentation, part-of-speech (POS) tagging and constituent labeling at the same time. Since syntactical information influences directly word-segmentation in the proposed character-based parser, CTB allows us to test whether or not syntactic information is useful for word-segmentation. A third advantage of parsing Chinese character streams is that Chinese words are more or less an open concept and the out-of-vocabulary (OOV) word rate is high. As morphology of the Chinese language is limited, extra care is needed to model unknown words when building a word-based model. Xu et al. (2002), for example, uses an independent corpus to derive word classes so that unknown words can be parsed reliably. Chinese characters, on the other hand, are almost closed. To demonstrate the OOV problem, we collect a word and character vocabulary from the first $90\%$ sentences of CTB, and compute their coverages on the corresponding word and character tokenization of the last $10\%$ of the corpus. The word-based OOV rate is $5.61\%$ while the character-based OOV rate is only $0.18\%$.

The first step of training a character-based parser is to convert word-based parse trees into character-based trees. We derive character-level tags from word-level POS tags and encode word-boundary information with a positional tag. Word-level POSs become a constituent label in character-based trees. A maximum entropy parser (Ratnaparkhi, 1997) parser is then built and tested. Many language-independent feature templates in the English parser can be reused. Lexical features, which are language-dependent, are used to further improve the baseline models trained with language-independent features only. Word-segmentation results will be presented and it will be shown that POSs are very helpful while higher-level syntactic structures are of little use to word-segmentation – at least in the way they are used in the parser.

## 2 Word-Tree to Character-Tree

CTB is manually segmented and is tokenized at word level. To build a Chinese character parser, we first need to convert word-based parse trees into character trees. A few simple rules are employed in this conversion to encode word boundary information:

1. Word-level POS tags become labels in character trees.
2. Character-level tags are inherited from word-level POS tags after appending a positional tag;
3. For single-character words, the positional tag is "s"; for multiple-character words, the first character is appended with a positional tag "b", last character with a positional tag "e", and all middle characters with a positional tag "m".

An example will clarify any ambiguity of the rules. For example, a word-parse tree
"(IP (NP (NP 天津港/NR ) (NP 扩建/NN 工程/NN ) ) (VP 开工/VV ) 。/PU )"
would become
"(IP (NP (NP (NR 天/nrb 津/nrm 港/nre ) ) (NP (NN 扩/nnb 建/nne ) (NN 工/nnb 程/nne ) ) ) (VP (VV 开/vvb 工/vve ) ) (PU 。/pus ) )." (1)

Note that the word-level POS "NR" becomes a label of the constituent spanning the three characters "天津港". The character-level tags of the constituent "天津港" are the lower-cased word-level POS tag plus a positional letter. Thus, the first character "天" is assigned the tag "nrb" where "nr" is from the word-level POS tag and "b" denotes the beginning character; the second (middle) character "津" gets the positional letter "m", signifying that it is in the middle, and the last character "港" gets the positional letter "e", denoting the end of the word. Other words in the sentence are mapped similarly. After the mapping, the number of terminal tokens of the character tree is larger than that of the word tree.

It is clear that character-level tags encode word boundary information, and chunk-level[1] labels are word-level POS tags. Therefore, parsing a Chinese character sentence is effectively doing word-segmentation, POS-tagging and constructing syntactic structure at the same time.

## 3 Model and Features

The maximum entropy parser (Ratnaparkhi, 1997) is used in this study, for it offers the flexibility of integrating multiple sources of knowledge into a model. The maximum entropy model decomposes $P(T|S)$, the probability of a parse tree $T$ given a sentence $S$, into the product of probabilities of individual parse

[1]A chunk is here defined as a constituent whose children are all preterminals.

actions, i.e., $\prod_{i=1}^{N_T} P(a_i|S, a_1^{(i-1)})$. The parse actions $a_1^{N_T}$ are an ordered sequence, where $N_T$ is the number of actions associated with the parse $T$. The mapping from a parse tree to its unique sequence of actions is 1-to-1. Each parse action is either `tagging` a word, `chunking` tagged words, `extending` an existing constituent to another constituent, or `checking` whether an open constituent should be closed. Each component model takes the exponential form:

$$P(a_i|S, a_1^{(i-1)}) = \frac{exp\Big[\sum_k \lambda_k g_k(S, a_1^{(i-1)}, a_i)\Big]}{Z(S, a_1^{(i-1)})}, \tag{2}$$

where $Z(S, a_1^{(i-1)})$ is a normalization term to ensure that $P(a_i|S, a_1^{(i-1)})$ is a probability, $g_k(S, a_1^{(i-1)}, a_i)$ is a feature function (often binary) and $\lambda_k$ is the weight of $g_k$.

Given a set of features and a corpus of training data, there exist efficient training algorithms (Darroch and Ratcliff, 1972; Berger et al., 1996) to find the optimal parameters $\{\lambda_k\}$. The art of building a maximum entropy parser then reduces to choosing "good" features. We break features used in this study into two categories. The first set of features are derived from predefined templates. When these templates are applied to training data, features are generated automatically. Since these templates can be used in any language, features generated this way are referred to *language-independent* features. The second category of features incorporate lexical information into the model and are primarily designed to improve word-segmentation. This set of features are *language-dependent* since a Chinese word dictionary is required.

### 3.1 Language-Independent Feature Templates

The maximum entropy parser (Ratnaparkhi, 1997) parses a sentence in three phases: (1) it first tags the input sentence. Multiple tag sequences are kept in the search heap for processing in later stages; (2) Tagged tokens are grouped into chunks. It is possible that a tagged token is not in any chunk; (3) A chunked sentence, consisting of a forest of many subtrees, is then used to extend a subtree to a new constituent or join an existing constituent. Each `extending` action is followed by a `checking` action which decides whether or not to close the extended constituent. In general, when a parse action $a_i$ is carried out, the context information, i.e., the input sentence $S$ and preceding parse actions $a_1^{(i-1)}$, is represented by a forest of subtrees. Feature functions operate on the forest context and the next parse action. They are all of the form:

$$g_k\Big((S, a_1^{(i-1)}), a_i)\Big) = h_k(S, a_1^{(i-1)})\delta(a_i = a_k), \tag{3}$$

where $h_k(S, a_1^{(i-1)})$ is a binary function on the context.

Some notations are needed to present features. We use $w_n$ to denote an input terminal token, $t_n$ its tag (preterminal), $c_n$ a chunk, and $e_n$ a constituent label, where the index $n$ is relative to the current subtree: the subtree immediately left to the current is indexed as $-1$, the second left to the current subtree is indexed as $-2$, the subtree immediately to the right is indexed as 1, so on and so forth. $d_{n,l}$ represents the root label of the $l^{th}$-child of the $n^{th}$ subtree. If $l < 0$, the child is counted from right.

With these notations, we are ready to introduce language-independent features, which are broken down as follows:

**Tag Features**

In the tag model, the context consists of a window of five tokens – the token being tagged and two tokens to its left and right – and two tags on the left of the current word. The feature templates are tabulated in Table 1 (to save space, templates are grouped). At training time, feature templates are instantiated by the training data. For example, when the template "$w_{-1}, t_0$" is applied to the first character of the sample sentence,
"(IP (NP (NP (NR 天/nrb 津/nrm 港/nre ) ) (NP (NN 扩/nnb 建/nne ) (NN 工/nnb 程/nne ) ) ) (VP (VV 开/vvb 工/vve ) ) (PU 。/pus ) )",
a feature $g(w_{-1}$ =*BOUNDARY*$, t_0 = nrb)$ is generated. Note that $w_{-1}$ is the token on the left and in this case, the boundary of the sentence. The template "$w_0, t_0$" is instantiated similarly as $g(w_0 = 天, t_0 = nrb)$.

**Chunk Features**

As character-level tags have encoded the chunk label information and the uncertainly about a chunk action is low given character-level tags, we limit the chunk context to a window of three subtrees – the current one plus its left and right subtree. $c_n$ in Table 2 denotes the label of the $n^{th}$ subtree if it is not

| Index | Template (context,future) |
|---|---|
| 1 | $w_n, t_0$ $(n = -2, -1, 0, 1, 2)$ |
| 2 | $w_n w_{n+1}, t_0$ $(n = -1, 0)$ |
| 3 | $w_n w_{n+1} w_{n+2}, t_0$ $(n = -2, -1, 0)$ |
| 4 | $t_{-1}, t_0$ |
| 5 | $t_{-2} t_{-1}, t_0$ |

Table 1: Tag feature templates: $w_n(n = -2, 1, 0, 1, 2)$: current token (if $n = 0$) or $|n|^{th}$ token on the left (if $n < 0$) or right (if $n$ 0). $t_n(n = -2, -1, 0, 1, 2)$: tag.

a chunk, or the chunk label plus the tag of its right-most child if it is a chunk.

| Index | Template (context,future) |
|---|---|
| 1 | $c_n, a_0$ $(n = -1, 0, 1)$ |
| 2 | $c_n c_{n+1}, a_0$ $(n = -1, 0)$ |

Table 2: Chunk feature templates: $c_n(n = -1, 0, 1)$ is the chunk label plus the tag of its right most child if the $n^{th}$ tree is a chunk; Otherwise $c_n$ is the constituent label of the $n^{th}$ tree.

Again, we use the sentence (1) as an example. Assume that the current forest of subtrees is

(NR 天/nrb 津/nrm 港/nre ) 扩/nnb 建/nne 工/nnb 程/nne 开/vvb 工/vve 。/pus ,

and the current subtree is "扩/nnb", then instantiating the template $c_{-1}, a_0$ would result in a feature $g(c_{-1} = NR : nre, a_0 = chunkNN)$.

**Extend Features**

Extend features depend on previous subtree and the two following subtrees. Some features uses child labels of the previous subtree. For example, the interpretation of the template on line 4 of Table 3 is that $e_{-1}$ is the root label of the previous subtree, $d_{(-1,-1)}$ is the label of the right-most child of the previous tree, and $e_0$ is the root label of the current subtree.

**Check Features**

Most of check feature templates again use constituent labels of the surrounding subtrees. The template on line 1 of Table 4 is unique to the check model. It essentially looks at children of the current constituent, which is intuitively a strong indication whether or not the current constituent should be closed.

| Index | Template (context,future) |
|---|---|
| 1 | $e_{-1} e_n, a_0$ $(n = 0, 1, 2)$ |
| 2 | $e_{-1} d_{(-1,-n)}, a_0$ $(n = 1, 2)$ |
| 3 | $e_{-1} e_0 e_1$ |
| 4 | $e_{-1} d_{(-1,-1)} e_0, a_0$ |
| 5 | $e_{-1} d_{(-1,-1)} e_0 e_1, a_0$ |
| 6 | $e_{-1} d_{(-1,-1)} d_{(-1,-2)}, a_0$ |

Table 3: Extend feature templates: $e_n(n = -1, 0, 1, 2)$ is the root constituent label of the $n^{th}$ subtree (relative to the current one); $d_{(-1,-n)}(n = 1, 2)$ is the label of the $n^{th}$ rightmost child of the previous subtree.

| Index | Template (context,future) |
|---|---|
| 1 | $e_0 \to d_{0,1} \cdots d_{0,n_d}, a_0$ |
| 2 | $e_{0,-1}, a_0$ |
| 3 | $e_0 d_{0,i}, a_0$ $(i = 1, 2, \cdots, n_d)$ |
| 4 | $e_{-1}, a_0$ |
| 5 | $e_1, a_0$ |
| 6 | $e_{-2} e_{-1}, a_0$ |
| 7 | $e_1 e_2, a_0$ |

Table 4: Check feature templates: $e_n(n = -1, 0, 1, 2)$ is the constituent label of the $n^{th}$ subtree (relative to the current one). $d_{(0,i)}$ is the $i^{th}$ child label of the current constituent.

### 3.2 Language-Dependent Features

The model described so far does not depend on any Chinese word dictionary. All features derived from templates in Section 3.1 are extracted from training data. A problem is that words not seen in training data may not have "good" features associated with them. Fortunately, the maximum entropy framework makes it relatively easy to incorporate other sources of knowledge into the model. We present a set of language-dependent features in this section, primarily for Chinese word segmentation.

The language-dependent features are computed from a word list and training data. Formerly, let $L$ be a list of Chinese words, where characters are separated by spaces. At the time of tagging characters (recall word-segmentation information is encoded in character-level tags), we test characters within a window of five (that is, two characters to the left and two to the right) and see if a character either starts, occurs in any position of, or ends *any* word on the list $L$. This feature templates are summarized in Table 5. $b(w_n, L)$ tests if the character $w_n$ starts any word on the list $L$. Similarly, $m(w_n, L)$ tests if the character $w_n$ occurs in any position of any word on the list $L$, and $e(w_n, L)$ tests if the character $w_n$ is the last position of any word on the list $L$.

| Index | Template (context,future) |
|---|---|
| 1 | $b(w_n, L), t_0$ $(n = -2, -1, 0, 1, 2)$ |
| 2 | $m(w_n, L), t_0$ $(n = -2, -1, 0, 1, 2)$ |
| 3 | $e(w_n, L), t_0$ $(n = -2, -1, 0, 1, 2)$ |

Table 5: Language-dependent lexical features.

A word list can be collected to encode different semantic or syntactic information. For example, a list of location names or personal names may help the model to identify unseen city or personal names; Or a closed list of functional words can be collected to represent a particular set of words sharing a POS. This type of features would improve the model robustness since unseen words will share features fired for seen words. We will show shortly that even a relatively small word-list improves significantly the word-segmentation accuracy.

## 4 Experiments

All experiments reported here are conducted on the latest LDC release of the Chinese Treebank, which consists of about $250K$ words. Word parse trees are converted to character trees using the procedure described in Section 2. All traces and functional tags are stripped in training and testing. Two results are reported for the character-based parsers: the F-measure of word segmentation and F-measure of constituent labels. Formally, let $W_r(i), W_p(i)$ be the number of words of the $i^{th}$ reference sentence and its parser output, respectively, and $C(i)$ be the number of common words in the $i^{th}$ sentence of test set, then the word segmentation F-measure is

$$F_{seg} = \frac{2\sum_i C(i)}{\sum_i \Big(W_r(i) + W_p(i)\Big)}. \qquad (4)$$

The F-measure of constituent labels is computed similarly:

$$F_{lab} = \frac{2\sum_i N(i)}{\sum_i \Big(L_r(i) + L_p(i)\Big)}, \qquad (5)$$

where $L_r(i)$ and $L_p(i)$ are the number of constituents in the $i^{th}$ reference parse tree and parser output, respectively, and $N(i)$ is the number of common constituents. Chunk-level labels converted from POS tags (e.g., “NR”, “NN” and “VV” etc in (1)) are included in computing label F-measures for character-based parsers.

### 4.1 Impact of Training Data

The first question we have is whether CTB is large enough in the sense that the performance saturates. The first set of experiments are intended to answer this question. In these experiments, the first 90% CTB is used as the training set and the rest 10% as the test set. We start with 10% of the training set and increase the training set each time by 10%. Only language-independent features are used in these experiments.

Figure 1 shows the word segmentation F-measure and label F-measure versus the amount of training data. As can be seen, F-measures of both word segmentation and constituent label increase monotonically as the amount of training data increases. If all training data is used, the word segmentation F-measure is 95.3% and label F-measure 81.7%. These results show that language-independent features work fairly well – a major advantage of data-driven statistical approach. The learning curve also shows that the current training size has not reached a saturating point. This indicates that there is room to improve our model by getting more training data.

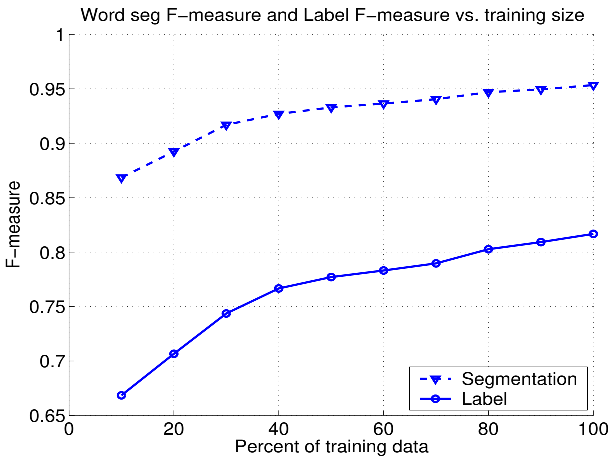


Figure 1: Learning curves: word-segmentation F-measure and parsing label F-measure vs. percentage of training data.

### 4.2 Effect of Lexical Features

In this section, we present the main parsing results. As it has not been long since the second release of CTB and there is no commonly-agreed training and test set, we divide the entire corpus into 10 equal partitions and hold each partition as a test set while the rest are used for training. For each training-test configuration, a baseline model is trained with only language independent features. Baseline word segmentation and label F-measures are plotted with dotted-line in Figure 2. We then add extra lexical features described in Section 3.1 to the model. Lexical questions are derived from a 58K-entry word list. The word list is broken into 4 sub-lists based on word length, ranging from 2 to 5 characters. Lexical features are computed by answering one of the three questions in Table 5. Intuitively, these questions would help the model to identify word boundaries, which in turn ought to improve the parser. This is confirmed by results shown in Figure 2. The solid two lines represent results with enhanced lexical questions. As can be seen, lexical questions improve significantly both word segmentation and parsing across all experiments. This is not surprising as lexical features derived from the word list are complementary to language-independent features computed from training sentences.

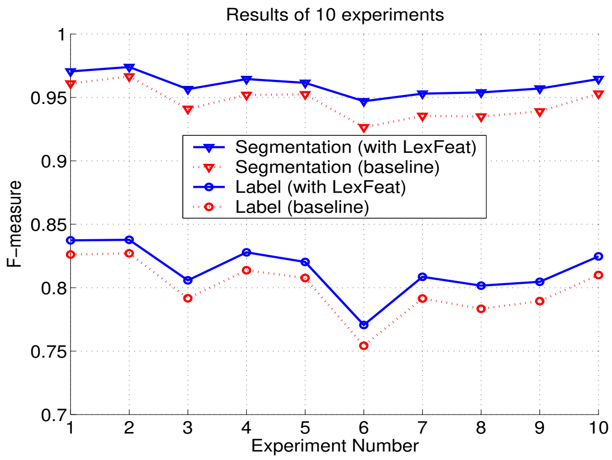


Figure 2: Parsing and word segmentation F-measures vs. the experiment numbers. Lines with triangles: segmentation; Lines with circles: label; Dotted-lines: language-independent features only; Solid lines: plus lexical features.

Another observation is that results vary greatly across experiment configurations: for the model trained with lexical features, the second experiment has a label F-measure 83.8% and word-segmentation F-measure 97.4%, while the sixth experiment has a label F-measure 77.1% and word-segmentation F-measure 94.7%. The large variances justify multiple experiment runs. To reduce the variances, we report numbers averaged over the 10 experiments in Table 6. Numbers on the row starting with "WS" are word-segmentation results, while numbers on the last row are F-measures of constituent labels. The second column are average F-measures for the baseline model trained with only language-independent features. The third column contains F-measures for the model trained with extra lexical features. The last column are releative error reduction. The best average word-segmentation F-measure is 96.0% and label F-measure is 81.4%.

| | F-measure | | |
|---|---|---|---|
| | baseline | LexFeat | Relative(%) |
| WS(%) | 94.6 | **96.0** | 26 |
| Label(%) | 80.0 | **81.4** | 7 |

Table 6: WS: word-segmentation. Baseline: language-independent features. LexFeat: plus lexical features. Numbers are averaged over the 10 experiments in Figure 2.

### 4.3 Effect of Syntactic Information on Word-segmentation

Since CTB provides us with full parse trees, we want to know how syntactic information affects word-segmentation. To this end, we devise two sets of experiments:

1. We strip all POS tags and labels in the Chinese Treebank and retain only word boundary information. To use the same maximum entropy parser, we represent word boundary by dummy constituent label "W". For example, the sample sentence (1) in Section 2 is represented as:

   (W 天/wb 津/wm 港/we ) (W 扩/wb 建/we ) (W 工/wb 程/we ) (W 开/wb 工/we ) (W 。/ws ).

2. We remove all labels but retain word-level POS information. The sample sentence above is represented as:

   (NR 天/nrb 津/nrm 港/nre ) (NN 扩/nnb 建/nne ) (NN 工/nnb 程/nne ) (VV 开/vvb 工/vve ) (PU 。/pus ).

Note that positional tags are used in both setups.

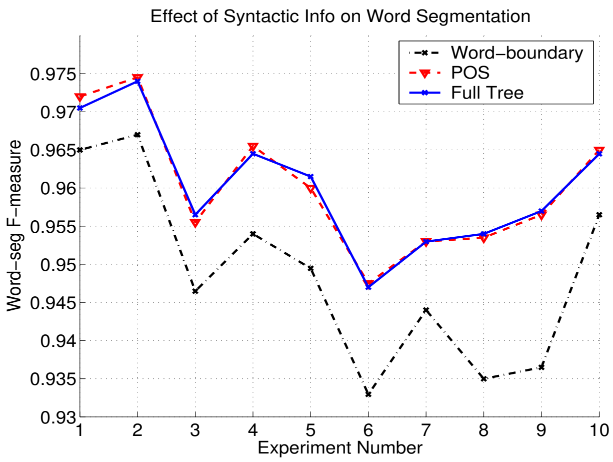


Figure 3: Usefulness of syntactic information: (black) dash-dotted line – word boundaries only, (red) dashed line – POS info, and (blue) solid line – full parse trees.

With these two representations of CTB, we repeat the 10 experiments of Section 4.2 using the same lexical features. Word-segmentation results are plotted in Figure 3. The model trained with word boundary information has the worst performance, which is not surprising as we would expect information such as POS tags to help disambiguate word boundaries. What is surprising is that syntactic information beyond POS tags has little effect on word-segmentation – there is practically no difference between the solid line (for the model trained with full parse trees) and the dashed-line (for the model trained with POS information) in Figure 3. This result suggests that most ambiguities of Chinese word boundaries can be resolved at lexical level, and high-level syntactic information does not help much to word segmentation in the current parser.

## 5 Related Work

Bikel and Chiang (2000) and Xu et al. (2002) construct word-based statistical parsers on the first release of Chinese Treebank, which has about 100K words, roughly half of the training data used in this study. Bikel and Chiang (2000) in fact contains two parsers: one is a lexicalized probabilistic context-free grammar (PCFG) similar to (Collins, 1997); the other is based on statistical TAG (Chiang, 2000). About 73% F-measure is reported in (Bikel and Chiang, 2000). Xu et al. (2002) is also based on PCFG, but enhanced with lexical features derived from the ASBC corpus[2]. Xu et al. (2002) reports an overall F-measure 75.2% when the same training and test set as (Bikel and Chiang, 2000) are used. Since our parser operates at character level, and more training data is used, the best results are not directly comparable. The middle point of the learning curve in Figure 1, which is trained with roughly 100K words, is at the same ballpark of (Xu et al., 2002). The contribution of this work is that the proposed character-based parser does word-segmentation, POS tagging and parsing in a unified framework. It is the first attempt to our knowledge that syntactic information is used in word-segmentation.

Chinese word segmentation is a well-known problem that has been studied extensively (Wu and Fung, 1994; Sproat et al., 1996; Luo and Roukos, 1996) and it is known that human agreement is relatively low. Without knowing and controlling testing conditions, it is nearly impossible to compare results in a meaningful way. Therefore, we will compare our approach with some related work only without commenting on segmentation accuracy. Wu and Tseng (1993) contains a good problem statement of Chinese word-segmentation and also outlines a few segmentation algorithms. Our method is supervised in that the training data is manually labeled. Palmer (1997) uses transform-based learning (TBL) to correct an initial segmentation. Sproat et al. (1996) employs stochastic finite state machines to find word boundaries. Luo and Roukos (1996) proposes to use a language model to select from ambiguous word-segmentations. All these work assume that a lexicon or some manually segmented data or both are available. There are numerous work exploring semi-supervised or unsupervised algorithms to segment Chinese text. Ando and Lee (2003) uses a heuristic method that does not require segmented training data. Peng and Schuurmans (2001) learns a lexicon and its unigram probability distribution. The automatically learned lexicon is pruned using a mutual information criterion. Peng and Schuurmans (2001) requires a validation set and is therefore semi-supervised.

[2] See `http://godel.iis.sinica.edu.tw/ROCLING`.

## 6 Conclusions

We present a maximum entropy Chinese character-based parser which does word-segmentation, POS tagging and parsing in a unified framework. The flexibility of maximum entropy model allows us to integrate into the model knowledge from other sources, together with features derived automatically from training corpus. We have shown that a relatively small word-list can reduce word-segmentation error by as much as 26%, and a word-segmentation F-measure 96.0% and label F-measure 81.4% are obtained by the character-based parser. Our results also show that POS information is very useful for Chinese word-segmentation, but higher-level syntactic information benefits little to word-segmentation.

## Acknowledgments

Special thanks go to Hongyan Jing and Judith Hochberg who proofread the paper and corrected many typos and ungrammatical errors. The author is also grateful to the anonymous reviewers for their insightful comments and suggestions. This work was partially supported by the Defense Advanced Research Projects Agency and monitored by SPAWAR under contract No. N66001-99-2-8916. The views and findings contained in this material are those of the authors and do not necessarily reflect the position of policy of the Government and no official endorsement should be inferred.

## References

Rie Kubota Ando and Lillian Lee. 2003. Mostly-unsupervised statistical segmentation of Japanese Kanji. *Natural Language Engineering*.

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.

Daniel M. Bikel and David Chiang. 2000. Two statistical parsing models applied to the chinese treebank. In *Proceedings of the Second Chinese Language Processing Workshop*, pages 1–6.

David Chiang. 2000. Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proc. Annual Meeting of ACL*, pages 1–6.

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

J. N. Darroch and D. Ratcliff. 1972. Generalized iterative scaling for log-linear model. *Ann. Math. Statist.*, 43:1470–1480.

Xiaoqiang Luo and Salim Roukos. 1996. An iterative algorithm to build chinese language models. In *Proc. of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 139–143.

David Palmer. 1997. A trainable rule-based algorithm for word segmentation. In *Proc. Annual Meeting of ACL*, Madrid.

Fuchun Peng and Dale Schuurmans. 2001. Self-supervised Chinese word segmentation. In *Advances in Intelligent Data Analysis*, pages 238–247.

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.

Richard Sproat, Chilin Shih, William Gale, and Nancy Chang. 1996. A stochastic finite-state word-segmentation algorithm for Chinese. *Computational Linguistics*, 22(3):377–404.

Dekai Wu and Pascale Fung. 1994. Improving chinese tokenization with linguistic filters on statistical lexical acquisition. In *Fourth Conference on Applied Natural Language Processing*, pages 180–181, Stuttgart.

Zimin Wu and Gwyneth Tseng. 1993. Chinese text segmentation for text retrieval: Achievements and problems. *Journal of The American Society for Information Science*, 44(9):532–542.

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)*.

Jinxi Xu, Scott Miller, and Ralph Weischedel. 2002. A statistical parser for Chinese. In *Proc. Human Language Technology Workshop*.