Training Global Linear Models for Chinese Word Segmentation*

Dong Song and Anoop Sarkar

School of Computing Science, Simon Fraser University
Burnaby, BC, Canada V5A1S6,
dsong@alumni.sfu.ca, anoop@cs.sfu.ca

Abstract. This paper examines how one can obtain state of the art Chinese word segmentation using global linear models. We provide experimental comparisons that give a detailed road-map for obtaining state of the art accuracy on various datasets. In particular, we compare the use of reranking with full beam search; we compare various methods for learning weights for features that are full sentence features, such as language model features; and, we compare an Averaged Perceptron global linear model with the Exponentiated Gradient max-margin algorithm.

1 Introduction

The written form of many languages, including Chinese, do not have marks identifying words. Given the Chinese text "北京大学生比赛", a plausible segmentation would be "北京(Beijing)/大学生(university students)/比赛(competition)" (Competition among university students in Beijing). However, if "北京大学" is taken to mean Beijing University, the segmentation for the above character sequence might become "北京大学(Beijing University)/生(give birth to)/比赛(competition)" (Beijing University gives birth to competition), which is less plausible. Chinese word segmentation has a large community of researchers, and has resulted in three shared tasks: the SIGHAN bakeoffs [1–3]. Word segmentation can be treated as a supervised sequence learning (or tagging) task. As in other tagging tasks, the most accurate models are discriminative models such as conditional random fields (CRFs) [4], perceptron [5], or various max-margin sequence models, such as [6, 7]. [5] provides a common framework collectively called *global linear models* to describe these approaches.

In this paper we show that using features that have been commonly used for Chinese word segmentation, plus adding a few additional global features, such as language model features, we can provide state of the art accuracy on several standard datasets using global linear models. In particular, the accuracy numbers obtained by our approach do not use any post-processing heuristics. Several types of ad-hoc post-processing heuristics are commonly used by other systems to obtain high accuracy on certain data sets but not others. The main contribution of this paper is to motivate the various choices that need to be made while

* This research was partially supported by NSERC, Canada (RGPIN: 264905) and by an IBM Faculty Award. Thanks to Michael Collins and Terry Koo for help with the EG implementation (any errors are our own), to the anonymous reviewers, and to the SIGHAN bakeoff organizers and participants.

training global linear models. We provide experimental evidence for choices made that provide state of the art accuracy for Chinese word segmentation.

2 Global Linear Models

Michael Collins [5] provides a common framework called *global linear models* for the *sequence learning* task (also called *tagging*): Let $\mathbf{x}$ be a set of inputs, and $\mathbf{y}$ be a set of possible outputs. In our experiments, $\mathbf{x}$ are unsegmented Chinese sentences, and $\mathbf{y}$ are the possible word segmentations for $\mathbf{x}$.

- Each $x \in \mathbf{x}$ and $y \in \mathbf{y}$ is mapped to a d-dimensional feature vector $\Phi(x,y)$, with each dimension being a real number, summarizing partial information contained in (x,y).
- A weight parameter vector $\mathbf{w} \in \Re^d$ assigns a weight to each feature in $\Phi(x,y)$, representing the importance of that feature. The value of $\Phi(x,y) \cdot \mathbf{w}$ is the score of (x,y). The higher the score, the more plausible it is that y is the output for x.
- The function $GEN(x)$ generates the set of possible outputs y for a given x.

Having $\Phi(x,y)$, $\mathbf{w}$, and $GEN(x)$ specified, we would like to choose the highest scoring candidate y^* from $GEN(x)$ as the most plausible output. That is,

$$F(x) = \underset{y \in GEN(x)}{\operatorname{argmax}}\ p(y \mid x, \mathbf{w})$$

where $F(x)$ returns the highest scoring output y^* from $GEN(x)$. A *conditional random field* (CRF) [4] defines the conditional probability as a linear score for each candidate y and a *global* normalization term:

$$\log p(y \mid x, \mathbf{w}) = \Phi(x,y) \cdot \mathbf{w} - \log \sum_{y' \in GEN(x)} \exp(\Phi(x,y') \cdot \mathbf{w})$$

In our experiments we find that a simpler global linear model that ignores the normalization term is faster to train and provides comparable accuracy.

$$F(x) = \underset{y \in GEN(x)}{\operatorname{argmax}}\ \Phi(x,y) \cdot \mathbf{w}$$

For this model, we learn the weight vector from labeled data using the perceptron algorithm [5]. A global linear model is global is two ways: it uses features that are defined over the entire sequence, *and* the parameter estimation methods are explicitly related to errors over the entire sequence.

3 Feature Templates and Experimental Setup

In this section, we look at the choices to be made in defining the feature vector $\Phi(x,y)$. In our experiments the feature vector is defined using local feature templates and global feature templates. For local features, the 14 feature types from [8] are used, shown in Table 1a. The local features for the entire sequence are summed up to provide global features.

1	word w
2	word bigram $w_1 w_2$
3	single character word w
4	space-separated characters c_1 and c_2
5	character bigram $c_1 c_2$ in any word
6	word starting with character c with length l
7	word ending with character c with length l
8	first and last characters c_1 and c_2 of any word
9	word w immediately before character c
10	character c immediately before word w
11	starting chars c_1, c_2 for 2 consecutive words
12	ending chars c_1, c_2 for 2 consecutive words
13	a word of length l and the previous word w
14	a word of length l and the next word w

(a)

15	sentence confidence score
16	sentence language model score

(b)

Table 1: Feature templates for (a) local features and (b) global features

In our experiments we also use global features previously used by [9, 10] that are not simply a sum of local features over the sequence. These features cannot be decomposed into a sequence of local features, which we will henceforth refer to as *global features* (the italics are important!), are listed in Table 1b.

Sentence confidence scores are calculated by a model that is also used as the *GEN* function for the global linear model (for instance, in our experiments the sentence confidence score is provided by a baseline character-based CRF tagger and *GEN* is the n-best list it produces). Sentence language model scores are produced using the *SRILM* [11] toolkit[1]. They indicate how likely a sentence can be generated given the training data, and they help capture the usefulness of features extracted from the training data. We use a trigram language model trained on the entire training corpus. We normalize the sentence LM probability: $P^{1/L}$, where P is the probability-based language model score and L is the length of the sentence in words (not in characters). Using logs the value is $\mid \log(P)/L \mid$. We explore different methods for learning the weights for these *global features*[2].

We build a baseline system which is a character based tagger using only the character features from Table 1a. We use the 'IOB' tagset where each character is tagged as 'B' (first character of multi-character word), or 'I' (character inside a multi-character word), or 'O' (indicating a single character word). The baseline system is built using the *CRF++* toolkit by Taku Kudo[3]. It is also used in our reranking experiments as the source of possible segmentations for the *GEN* function in our global linear models.

[1] http://www.speech.sri.com/projects/srilm/

[2] Our global features are different from commonly used "global" features in the literature, which either enforce consistency in a sequence (e.g. ensuring that the same word type is labeled consistently in the token sequence) or examine the use of a feature in the entire training or testing corpus.

[3] http://crfpp.sourceforge.net/

Inputs: Training Data $\langle (x_1, y_1), \ldots, (x_m, y_m) \rangle$; number of iterations T
Initialization: Set $\mathbf{w} = \mathbf{0}$, $\gamma = \mathbf{0}$, $\sigma = \mathbf{0}$
Algorithm:

```
for t = 1, ..., T do
  for i = 1, ..., m do
    y_i' = argmax_{y ∈ n-best list} Φ(x_i, y) · w
    y^b is closest to y_i in terms of f-score and y^b ∈ n-best list
    if y_i' ≠ y^b then
      w = w + Φ(x_i, y^b) − Φ(x_i, y_i')
    end if
    σ = σ + w
  end for
end for
```

Output: Avg. weight parameter vector $\gamma = \sigma/(\mathrm{mT})$

Fig. 1: Averaged perceptron learning algorithm using an n-best list

The experimental results are reported on datasets from the first and third SIGHAN bakeoff shared task datasets [1, 3]. From the 1st SIGHAN bakeoff we use the Peking University (PU) dataset. From the 3rd SIGHAN bakeoff we use the CityU (City University of Hong Kong), the MSRA (Microsoft Research Asia) and UPUC (University of Pennsylvania and University of Colorado) datasets. We strictly follow the closed track rules, where no external knowledge is used[4].

4 Reranking v.s. beam search

There are two choices for the definition of *GEN* in a global linear model:

- *GEN*(x) enumerates all possible segmentations of the input x. In this case, search is organized either using dynamic programming or using beam search.
- *GEN*(x) is the n-best output of another auxiliary model and the global linear model is used as a *reranking* model.

In this section we compare beam search with reranking across many different corpora to test the strength and weakness of both methods.

Reranking To produce a reranking system, we produce a 10-fold split of the training data: in each fold, 90% of the corpus is used for training and 10% is used to produce an n-best list of candidates. The n-best list is produced using the character-based CRF tagger described earlier. The true segmentation can now be compared with the n-best list in order to train using an averaged perceptron algorithm [5] shown in Figure 1. This system is then used to predict the best word segmentation from an n-best list for each sentence in the test data.

We used the development set of the UPUC corpus to find a suitable value for the parameter n, the maximum number of n-best candidates. This oracle procedure proceeds as follows: 80% of the training corpus is used to train the CRF model, which is used to produce the n-best outputs for each sentence on the remaining 20% of the corpus. Then, these n candidates are compared with

[4] We do not even use the encoding of the dataset (dates and non-Chinese characters are used in encoding-specific heuristics to improve performance, we do not do this).

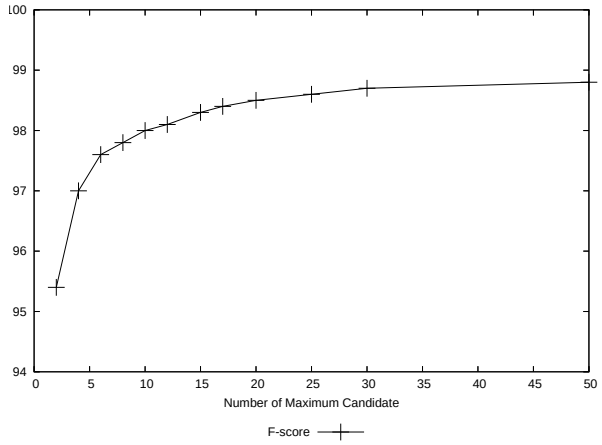

Fig. 2: F-score on the UPUC development set with different n

the true segmentation, and for each training sentence, the candidate closest to the truth is chosen as the final output. As we increase the value of n, for some sentences, its n-best candidate list is more likely to contain a segmentation that will improve the overall F-score (Figure 2). To balance accuracy and speed, we choose $n = 20$ in all our reranking experiments.

We do not use the algorithm in Figure 1 to train the weights for the *global features* defined in Table 1b. The sentence confidence score feature weight and the language model feature weight is chosen to be 15 for the CityU corpus, to be 15 for the MSRA corpus, and to be 20 for the UPUC corpus. The reason for this choice is provided in Section 5.

Beam Search Decoding In [8], instead of applying the n-best reranking method, their word segmentation system uses beam search decoding [12], where the global features are only those that are the sum of the local features.

In beam search, the decoder generates segmentation candidates incrementally. It reads one character at a time from the input sentence, and combines it with each existing candidate in two ways, either appending this new character to the last word, or considering it as the beginning of a new word. This combination process generates segmentations exhaustively; that is, for a sentence with k characters, all 2^{k-1} possible segmentations are generated. In global linear models which contain a normalization term it is common to use dynamic programming. However, for mistake-driven training such as the perceptron, beam search is more effective. We implemented the decoding algorithm following the pseudo-code described in [8] which is based on the algorithm in [12]. The beam size B is used to limit the number of candidates preserved after processing each character. The performance of the beam search system is compared with that of the n-best reranking system on the PU corpus from the first SIGHAN bakeoff, and on the CityU, MSRA, UPUC corpora from the third SIGHAN bakeoff (closed track). In the n-best reranking system, 20 is chosen to be the maximum number of n-best candidates. Using the approach described in Section 5 the weight for sentence confidence score and that for language model score are determined

Corpus	Setting	F	P	R	R_{IV}	R_{OOV}
PU	Avg. perc. with beam search	**94.1**	94.5	93.6	69.3	95.1
	Avg. perc. with reranking, global & local features	93.1	93.9	92.3	94.2	61.8
	Avg. perc. with reranking, local features	92.2	92.8	91.7	93.4	62.3
	Baseline character based CRF	93.1	94.0	92.3	94.1	61.5
	CRF with subword tagging	91.9	91.7	92.2	94.5	53.5
CityU	Avg. perc. with beam search	96.8	96.8	96.8	97.6	77.8
	Avg. perc. with reranking, global & local features	**97.1**	97.1	97.1	97.9	78.3
	Avg. perc. with reranking, local features	96.7	96.7	96.6	97.5	77.4
	Baseline character based CRF	95.7	95.7	95.7	96.5	78.3
	CRF with subword tagging	95.9	95.8	96.0	96.9	75.2
MSRA	Avg. perc. with beam search	95.8	96.0	95.6	96.6	66.2
	Avg. perc. with reranking, global & local features	95.8	95.9	95.7	96.9	62.0
	Avg. perc. with reranking, local features	95.5	95.6	95.3	96.3	65.4
	Baseline character based CRF	94.7	95.2	94.3	95.3	66.9
	CRF with subword tagging	94.8	94.9	94.6	95.7	64.9
UPUC	Avg. perc. with beam search	92.6	92.0	93.3	95.8	67.3
	Avg. perc. with reranking, global & local features	**93.1**	92.5	93.8	96.1	69.4
	Avg. perc. with reranking, local features	92.5	91.8	93.1	95.5	68.8
	Baseline character based CRF	92.7	92.2	93.1	95.2	71.4
	CRF with subword tagging	91.8	91.0	92.7	95.2	66.6

Table 2: Performance (in percentage) comparing averaged perceptron with beam search with reranking. F is F-score, P is precision, R is recall, R_{IV} is in-vocabulary (words in training) recall, and R_{OOV} is out of vocabulary recall. CRF with subword tagging is our implementation of [13]. Boldface is statistically significant improvement over all other methods (see [14] for detailed results).

to be 15 for the CityU and MSRA corpora, 20 for the UPUC corpus, and 40 for the PU corpus. Similarly, using the dev set, the training iterations were set to 7 for the CityU and MSRA corpora, 9 for the UPUC corpus, and 6 for the PU corpus. In the beam search method, the beam size was set to be 16 for all corpora, and the number of iterations was set to be 7, 7 and 9 for the CityU, MSRA and UPUC corpora, respectively, corresponding to the iteration values we applied on each corpus in the reranking system. Table 2 shows the comparison between the averaged perceptron training using the beam search method v.s. the reranking method. For each corpus, the bold number represents the highest F-score. From the result, we see that on the CityU, MSRA and UPUC corpora, the beam search decoding based system outperforms the reranking using only local features. However, reranking based with *global features* is more accurate than the beam search decoding based system, except on the PU corpus.

For the PU corpus from the first SIGHAN bakeoff, the reranking does worse than beam search (and no better than the baseline). To see why we examine how many sentences in the gold standard also appear within the 20-best candidate list. For each corpus test set, the results are: CityU (88.2%), MSRA (88.3%), UPUC (68.4%), and PU (54.8%). For the PU test set, almost half of the true segmentations are not seen in the 20-best list, which seriously affects the rerank-

ing approach. While for the CityU and MSRA corpora, nearly 90% of the gold standard segmentations appear in the 20-best candidate list. Beam search has the advantage of not requiring a separate model to produce n-best candidates, but training and testing are much slower than reranking[5] and further research is required to make it competitive with reranking for Chinese word segmentation[6].

5 Learning Global Feature Weights

In this section we explore how to learn the weights for those features that are not simply the sum of local features. These so-called global features have an important property: they are real numbers that correspond to the quality of the entire segmentation, and cannot be identified with any portion of it. Algorithm 1 updates features for a segmentation but is restricted to local features collected over the entire segmentation (see line 6 of Algorithm 1 where the weight vector **w** is updated). For this reason, alternative strategies to obtain weights for global features need to be explored.

Learning Weights from Development Data We use development data to determine the weight for the sentence confidence score S_{crf} and for the language model score S_{lm}.[7] In this step, each training corpus is separated into a training set, which contains 80% of the training corpus, and a development set containing the remaining 20% of the training corpus. Then, the perceptron algorithm is applied on the training set with different S_{crf} and S_{lm} weight values, and for various number of iterations. The weight values we test include 2, 4, 6, 8, 10, 15, 20, 30, 40, 50, 100 and 200, across a wide range of scales. The reason for these discrete values is because we are simply looking for a confidence threshold over which the sum of local features can override global features such as the confidence score (cf. Figure 3). As we can see, there will be a significant number of testing scenarios (i.e. $12 \times 12 = 144$ testing scenarios) in order to pick the most suitable weight values for each corpus. To simplify the process, we assume that the weights for both S_{crf} and S_{lm} are equal – this assumption is based on the fact that weights for these global features simply provide an importance factor so only a threshold value is needed rather than a finely tuned value that interacts with other feature weights for features based on local feature templates. Figure 3 shows the F-scores on each of the three corpora using different S_{crf} and S_{lm} weight values with different number of iterations t. From the tables, we observe that when the weight for S_{crf} and S_{lm} increases, F-score improves; however, if the weight for S_{crf} and S_{lm} becomes too large to overrule the effect of weight learning on local features, F-score suffers. For our experiments, the weight for S_{crf} and S_{lm} is chosen to be 15 for the CityU corpus, 15 for the MSRA corpus, and 20 for the UPUC corpus. Iterations over the development set also allows us

[5] We added the language model (LM) *global feature* as part of beam search, but could not use it in our experiments as training was prohibitively slow. Rescoring the final output using the LM probability led to lower accuracy.

[6] In general, in this paper we are not making general claims about algorithms, but rather what works and does not work for typical Chinese word segmentation datasets.

[7] This process is the same for all datasets. Heuristics are tuned per dataset.

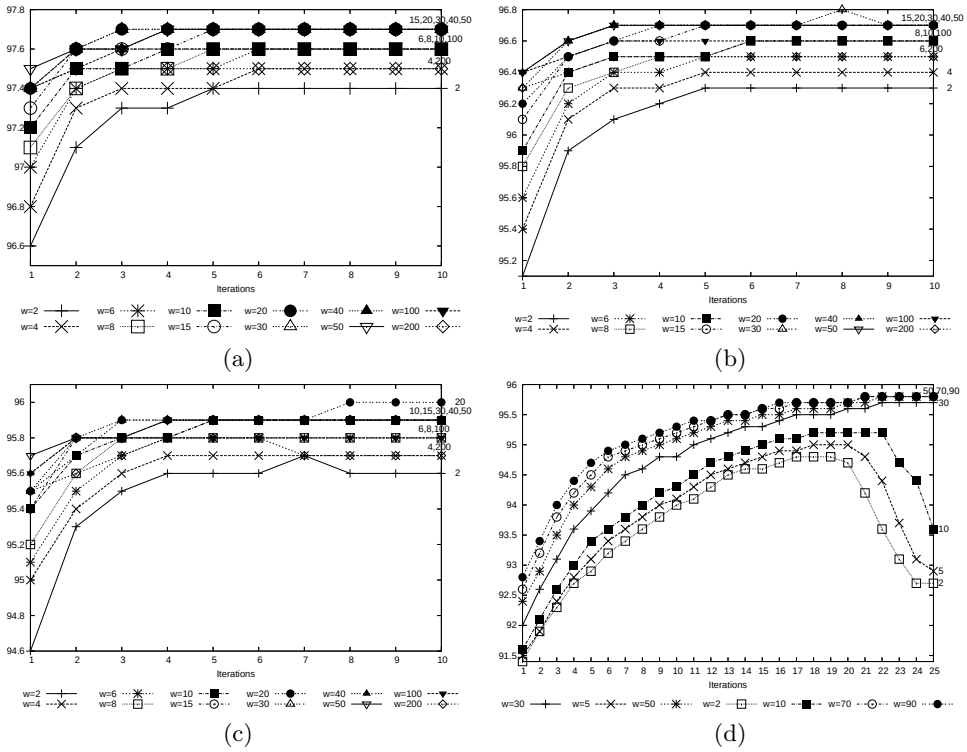

Fig. 3: F-scores on the (a) CityU, (b) MSRA, (c) UPUC development set for the avg. perceptron algorithm, and (d) UPUC development set for the EG algorithm.

to find the optimal number of iterations of training for the perceptron algorithm which is then used on the test set.

Learning Weights from Training Data The word segmentation system, designed by Liang in [15], incorporated and learned the weights for real-valued mutual information (MI) features by transforming them into alternative forms:

- Scale the values from $[0, \infty)$ into some fixed range [a, b], where the smallest value observed maps to a, and the largest value observed maps to b.
- Apply z-scores instead of the original values. The *z-score* of value x from $[0, \infty)$ is defined as $\frac{x-\mu}{\sigma}$ where μ and σ represent the mean and standard deviation of the distribution of x values.
- Map any value x to a if $x < \mu$, the mean value from the distribution of x values, or to b if $x \geq \mu$.

We use Liang's method to learn weights for our two global features during perceptron training, instead of manually fixing their weight using the development set. We experiment with the transformations on the two global features defined previously with the UPUC and CityU corpora[8]. Table 3 provides the perfor-

[8] Due to the large number of experimental settings, we do not test on the CityU and PU corpora due to their size.

Method	F-score (UPUC corpus)		F-score (CityU corpus)	
	held-out set	test set	held-out set	test set
Without global features	95.5	92.5	97.3	96.7
Fixed global feature weights	**96.0**	**93.1**	**97.7**	**97.1**
Threshold at mean to 0,1	95.0	92.0	96.7	96.0
Threshold at mean to -1,1	95.0	92.0	96.6	95.9
Normalize to [0,1]	95.2	92.1	96.8	96.0
Normalize to [-1,1]	95.1	92.0	96.8	95.9
Normalize to [-3,3]	95.1	92.1	96.8	96.0
Z-score	95.4	92.5	97.1	96.3

Table 3: F-scores (in percentage) obtained by using various ways to transform global feature weights and by updating their weights in averaged perceptron learning. The experiments are done on the UPUC and CityU corpora.

mance on their development and test sets. Z-scores perform well but do not out-perform fixing global feature weights using the development set. The likely reason is that the two global features have different properties than the MI feature. They do not have shared components across different training sentences and they describe the entire sentence unlike the MI features.

6 Exponentiated Gradient

In this section, we explore the use of max-margin methods for global linear models. In many tasks, the use of large margin or max-margin methods provides better generalization error over unseen data. We would like to know if Chinese word segmentation can benefit from a max-margin approach. We implement the batch exponentiated gradient (EG) algorithm [6, 7] with the same feature set as the perceptron experiments, including the two global features defined previously, and compare the performance on the UPUC corpus in the reranking setting.[9]. In EG, a set of dual variables $\alpha_{i,y}$ is assigned to data points $\mathbf{x}$. Specifically, to every point $x_i \in \mathbf{x}$, there corresponds a distribution $\alpha_{i,y}$ such that $\alpha_{i,y} \geq 0$ and $\sum_y \alpha_{i,y} = 1$. The algorithm attempts to optimize these dual variables $\alpha_{i,y}$ for each i separately. In the word segmentation case, x_i is a training example, and $\alpha_{i,y}$ is the dual variable corresponding to each possible segmented output y for x_i. EG is also expressed as a global linear model:

$$F(x) = \underset{y \in GEN(x)}{\operatorname{argmax}} \Phi(x, y) \cdot \mathbf{w}$$

The weight parameter vector $\mathbf{w}$ is expressed in terms of the dual variables $\alpha_{i,y}$:

$$\mathbf{w} = \sum_{i,y} \alpha_{i,y} \left[\Phi(x_i, y_i) - \Phi(x_i, y)\right]$$

[9] Because EG is computationally expensive we test only on UPUC. We obtain F-score of 93.1% on UPUC (lower than other corpora) so there is room for improvement using max-margin methods, however the baseline CRF model performs quite well on UPUC at F-score of 92.7%. This section is about comparing perceptron and EG.

Given a training set $\{(x_i, y_i)\}_{i=1}^{m}$ and the weight parameter vector $\mathbf{w}$, the margin on the segmentation candidate y for the i^{th} training example is defined as the difference in score between the true segmentation and the candidate y. That is,

$$M_{i,y} = \Phi(x_i, y_i) \cdot \mathbf{w} - \Phi(x_i, y) \cdot \mathbf{w}$$

For each dual variable $\alpha_{i,y}$, a new $\alpha'_{i,y}$ is obtained as

$$\alpha'_{i,y} \leftarrow \frac{\alpha_{i,y} e^{\eta \nabla_{i,y}}}{\sum_y \alpha_{i,y} e^{\eta \nabla_{i,y}}} \text{ where } \nabla_{i,y} = \begin{cases} 0 & \text{for } y = y_i \\ 1 - M_{i,y} & \text{for } y \neq y_i \end{cases}$$

and η is the learning rate which is positive and controls the magnitude of the update. In implementing the batch EG algorithm, during the initialization phase, the initial values of $\alpha_{i,y}$ are set to be 1/(number of n-best candidates for x_i). In order to get $\alpha'_{i,y}$, we need to calculate $e^{\eta \nabla_{i,y}}$. When each ∇ in the n-best list is positively or negatively too large, numerical underflow occurs. To avoid this, ∇ is normalized:

$$\alpha'_{i,y} \leftarrow \frac{\alpha_{i,y} e^{\eta \nabla_{i,y} / \sum_y |\nabla_{i,y}|}}{\sum_y \alpha_{i,y} e^{\eta \nabla_{i,y} / \sum_y |\nabla_{i,y}|}}$$

As before, the weight for global features is pre-determined using the development set and is fixed during the learning process. Considering the difference in training time between online update for perceptron learning and batch update for EG method, the maximum number of iterations is set to be larger (T = 25) in the latter case during parameter pruning. The weight for the global features are tested with 2, 5, 10, 30, 50, 70, and 90. Figure 3(d) shows the performance on the UPUC held-out set with various parameters. We select the number of iterations to be 22 and the weight for global features to be 90, and apply these parameters on the UPUC test set. Table 4 lists the resulting performance. Performance of the EG method with 22 iterations and with the same number of iterations (9 iterations) as the averaged perceptron method is provided, along with the use of different feature sets. The bold number represents the highest F-score.

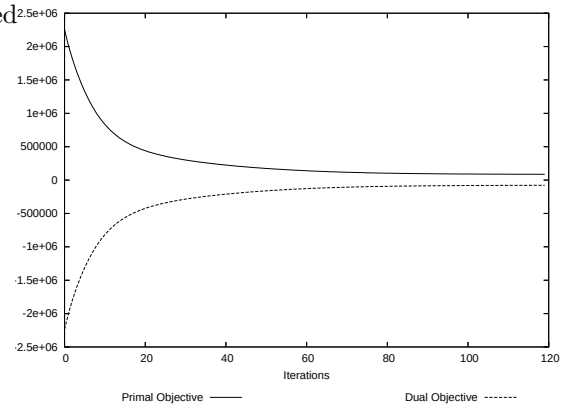

Fig. 4: EG algorithm convergence on UPUC

From Table 4, we see that the averaged perceptron with global features provides the highest F-score. Continuing to run the EG algorithm for more iterations (T = 120) with the weight of global features being fixed at 90, Figure 4 shows the convergence in terms of the primal and dual objective functions. From the

Setting	F	P	R	R_{IV}	R_{OOV}
EG algorithm, global & local features	93.0	92.3	93.7	96.1	68.2
EG algorithm, local features	90.4	90.6	90.2	92.2	69.7
EG algorithm, global & local features, 9 iterations	92.4	91.7	93.1	95.5	67.6
Avg. perc. global & local features	**93.1**	92.5	93.8	96.1	69.4

Table 4: Performance (percentage) of the EG algorithms, compared to the perceptron learning methods. All are in the reranking setting. Cf. UPUC results in Table 2.

figure, we can see that the algorithm does in fact converge to the maximum margin solution on this data set. However, at iteration 120, the F-score remains 0.930, which is the same as the F-score produced in the 22nd iteration.

7 Accuracy

In this section, we compare the accuracy we obtain with the best known accuracy (to our knowledge) on each dataset from the first and third SIGHAN bakeoff [1, 3]. On the PU corpus, we replicate the result in [8] and obtain $F = 0.941$ which is the best accuracy on that dataset. On the UPUC corpus the best result is obtained by Site 20. They use date and non-Chinese character information in their features (this depends on the encoding). Plus one local feature they use is the tone of the character which we do not use. They do not report results without the date and non-Chinese character features (they created these features for the training data using clustering). Their overall result using a log-linear tagger is $F = 0.933$ which is better than our result of $F = 0.931$. On the MSRA corpus the best result is obtained by Site 32.[10] They use three different kinds of post-processing: dictionary based heuristics, date specific heuristics, and knowledge about named entities that is added to the training data. Without any post-processing, using a log-linear tagger with n-gram features the result obtained is $F = 0.958$ which matches our score of $F = 0.958$. On the CityU corpus, the best result is obtained by Site 15. They build a specialized tagger for non-Chinese characters using clustering, plus template-based post-processing. Without these steps the accuracy is $F = 0.966$ compared to our score of $F = 0.971$. Our system (avg. perceptron with global & local features) is the only one that consistently obtains good performance across all these datasets except for the Peking University (PU) dataset[11].

8 Summary

In this paper, we explore several choices in building a Chinese word segmentation system. We explore the choice between using global features or not, and the choices involved in training their feature weights. In our experiments we find that using a development dataset to fix these global feature weights is better than learning them from data directly. We compare reranking versus the use of full

[10] Due to lack of space, we omit full references and ask that the reader refer to [1, 3]

[11] Testing on all available datasets would result in huge tables, so we report on the most recent dataset from the 5th SIGHAN, and we report on the 2nd SIGHAN data only for comparison with [8].

beam search decoding, and find that further research is required to make beam search competitive in all datasets. We explore the choice between max-margin methods and an averaged perceptron, and find that the averaged perceptron is typically faster and as accurate for our datasets. We show that our methods lead to state of the art accuracy and provide a transparent, easy to replicate design for highly accurate Chinese word segmentation.

References

1. Sproat, R., Emerson, T.: The 1st international chinese word segmentation bakeoff. In: Proceedings of the 2nd SIGHAN Workshop on Chinese Language Processing, Sapporo, Japan, ACL (July 2003) 123–133
2. Emerson, T.: The 2nd international chinese word segmentation bakeoff. In: Proceedings of the 4th SIGHAN Workshop on Chinese Language Processing, Jeju Island, Korea (October 2005) 123–133
3. Levow, G.A.: The 3rd international chinese language processing bakeoff. In: Proceedings of the 5th SIGHAN Workshop on Chinese Language Processing, Sydney, Australia, ACL (July 2006) 108–117
4. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the 18th International Conf. on Machine Learning (ICML). (2001) 282–289
5. Collins, M.: Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In: Proceedings of the Empirical Methods in Natural Language Processing (EMNLP), Philadelphia, PA, USA, ACL (July 2002) 1–8
6. Kivinen, J., Warmuth, M.: Exponentiated gradient versus gradient descent for linear predictors. Technical Report UCSC-CRL-94-16, UC Santa Cruz (1994)
7. Globerson, A., Koo, T., Carreras, X., Collins, M.: Exponentiated gradient algorithms for log-linear structured prediction. In: ICML. (2007) 305–312
8. Zhang, Y., Clark, S.: Chinese segmentation with a word-based perceptron algorithm. In: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, Prague, Czech Republic, ACL (June 2007) 840–847
9. Sproat, R., Gale, W., Shih, C., Chang, N.: A stochastic finite-state word-segmentation algorithm for chinese. Comput. Linguist. **22**(3) (1996) 377–404
10. Song, D., Sarkar, A.: Training a perceptron with global and local features for chinese word segmentation. In: Proceedings of the 6th SIGHAN Workshop on Chinese Language Processing. (2008) 143–146
11. Stolcke, A.: SRILM – an extensible language modeling toolkit. In: Proceedings of the ICSLP. Volume 2., Denver, Colorado (2002) 901–904
12. Collins, M., Roark, B.: Incremental parsing with the perceptron algorithm. In: Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume, Barcelona, Spain (July 2004) 111–118
13. Zhang, R., Kikui, G., Sumita, E.: Subword-based tagging by conditional random fields for chinese word segmentation. In: Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers, New York City, USA, ACL (June 2006) 193–196
14. Song, D.: Experimental comparison of discriminative learning approaches for chinese word segmentation. Master's thesis, Simon Fraser University (2008)
15. Liang, P.: Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology (2005)