# A Chinese Word Segmentation Experiment Using Conditional Random Field

Kuan-Wen Lo, Yiran Luo, Dingkang Wang, Zhaoyu Duan

April 28, 2015

**Abstract**

It's well-known that the standard Chinese text has no space or other delimiter between words. Hence, word segmentation becomes the most primary task in Chinese Word Processing, and the accuracy of Chinese word segmentation is essential to the performance of following procedures, such as Parsing, Part-of-speech Tagging and Machine Translation.

## 1 Introduction

This paper is to demonstrate our study of building a Chinese word segmenter. Our system relies on a Conditional Random Filed (CRF) model, which makes binary decision labels on each character whether or not it should be segmented. Then we utilize Stochastic Gradient Descent as our training method to find the optimal parameters ($\alpha$ and $\beta$), which we train using the backward-forward algorithm to construct the probability table. Given the probability table, we eventually decode the character sequences by Viterbi's Algorithm for the most probable label sequence. Most of our algorithms are applications following the tutorial by Charles Sutton and Andrew McCallum published in 2010 [1].

## 2 Approaches

In this section, we lay down the methodology of this experiment, introducing our statistic models, algorithms, and test datasets.

### 2.1 Labeling the Segmentation

Since Chinese is by nature a language with few white-spaces, we use 0 and 1 on each character to mark the non-starts and the start of each properly segmented Chinese word. For example:

Original: 服务品质
Segmented: 服务 品质
Labeled sequence: (服, 1) (务, 0) (品, 1) (质, 0)

## 2.2 Conditional Random Field (CRF)

Introduced by Lafferty et al. (2001) [2], Conditional Random Field is a statistical framework that helps build a discriminative model to segment and label sequential data. Later CRF was adapted for Chinese segmentation by Peng et al. (2004) [3] and Chang et al. (2008) [4] as a highly anticipated tool for such tasks.

In the big picture, we acquire the conditional log likelihood $L$ of a training set which contains N instances, for the entire source sequence set $\mathbf{s}$, and observation sequence set $\mathbf{o}$:

$$L = \sum_{n=1}^{N} \log(P(\mathbf{s}^{(n)}|\mathbf{o}^{(n)})) \tag{2.2.1}$$

For each sequence pair s and o, we acquire the likelihood for all K feature functions $f_k$, along with the weights $\lambda_k$. At position $t$, feature function $f_k$ takes the arguments of the current state $s_t$, the previous state $s_{t-1}$ and the current observation $o_t$. $\lambda_k$ serves as the parameter of the CRF model.

$$P(\mathbf{s}|\mathbf{o}) = \frac{1}{Z_{\mathbf{o}}} \exp\left(\sum_{t=1}^{T}\sum_{k=1}^{K} \lambda_k f_k(s_{t-1}, s_t, o_t)\right) \tag{2.2.2}$$

In the last setup, in addition to substituting 2.2.2 into 2.2.1, we have to consider overfitting, due to the fact that often the number of $\lambda$s is very large. To avoid overfitting , we need to hence place in the penalty determined by an arbitrary regularization factor $\sigma$. Eventually, our log likelihood function with penalty becomes:

$$L = \sum_{n=1}^{N} \frac{1}{Z_{\mathbf{o}^{(n)}}} \exp\left(\sum_{t=1}^{T}\sum_{k=1}^{K} \lambda_k f_k(s_{t-1}, s_t, o_t^{(n)})\right) - \sum_{k=1}^{K} \frac{\lambda_k^2}{\sigma} \tag{2.2.3}$$

## 2.3 Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent is a simple yet very efficient approach to discriminative learning of linear classifiers under loss functions. The objective loss function often has the form of a sum:

$$L(w) = \sum_{n} L_i(w) \tag{2.3.1}$$

When used to minimize the above function, a standard gradient descent method would perform the following iterations, where $\theta$ is a step size.

$$w = w - \theta \nabla L(w) \tag{2.3.2}$$

### 2.4 Test Data

We use Traditional Chinese corpora from Academia Sinica, Taiwan, which contains 708,953 sentences with 5,449,698 words and 8,368,050 characters [5] . It is the largest open-source corpora we can find and includes the most comprehensive Chinese text data. The corpora provides us segmented sentences as our training data and unsegmented sentence as our test data.

## 3 Experimental Design

This part of the paper will present how each segment of our program works based on the approaches as we have demonstrated in the previous section.

### 3.1 Preprocessing

The training text is encoded in UTF-8 and it cannot be recognized by Python, so decoding the text first is necessary and all the spaces in the text are full-width in that we convert them to half-width space and delete newline characters ('`\n`'). We tag each character with 1 or 0 which means whether the character is the beginning of the word. Then, we convert each sentence into a list of tuples which contain a character and a tag as our training data.

### 3.2 Training

According to our CRF Model, the conditional log-likelihood of a set of training instances as shown in equation 2.2.3, and now, as shown in equation 3.2.1, we can take the partial derivative of the log-likelihood in order to maximize it. $F_k$ here is the frequency or count that the particular global feature $f_k$ appears in the training data.

$$\frac{\partial L}{\partial \lambda_k} = \sum_{n=1}^{N} \left( F_k(\mathbf{s}^{(n)}, \mathbf{o}^{(n)}) - \sum_{\mathbf{s}} P(\mathbf{s}|\mathbf{o}^{(n)}) F_k(\mathbf{s}, \mathbf{o}^{(n)}) \right) - \frac{\lambda_k}{\sigma} \tag{3.2.1}$$

After we got the expression of the partial expression, we found that the frequency term and penalty term are not hard to calculate, so we focused on how to simplify the second term $\sum_{\mathbf{s}} P(\mathbf{s}|\mathbf{o}^{(n)}) F_k(\mathbf{s}, \mathbf{o}^{(n)})$. If we want to compute this term directly, we need to go through every possible labeling for a given Chinese sentence, suppose that the length of the sentence is $l$, then the complexity of computing this term will be $O(l2^l)$ which is not acceptable. Therefore, we have to make it easier to computer. Equation 3.2.2 shows the transferred partial derivative.

$$\frac{\partial L}{\partial \lambda_k} = \sum_{n=1}^{N} \left( F_k(\mathbf{s}^{(n)}, \mathbf{o}^{(n)}) - \sum_{t} \sum_{s,s'} P(s_{t-1} = s, s_t = s'|\mathbf{o}^{(n)}) f_k(s, s', o_t^{(n)}) \right) - \frac{\lambda_k}{\sigma} \tag{3.2.2}$$

Then, we need to define $\alpha$ table and $\beta$ table which borrow the ideas from backward-forward training. $\alpha(t, s)$ represents the probability of a partial labeling ending at position $t$ with label $s$, similarly, $\beta(t, s)$ represents the probability of a partial labeling starting at position $t + 1$ with label $s$ at position $t$. Equation 3.2.3 and equation 3.2.4 show how to compute the elements in these two table.

$$\alpha(t, s) = \sum_{s'} \alpha(t-1, s') \exp\left(\sum_k \lambda_k f_k(s', s, o_t)\right) \tag{3.2.3}$$

$$\beta(t, s) = \sum_{s'} \beta(t+1, s') \exp\left(\sum_k \lambda_k f_k(s, s', o_{t+1})\right) \tag{3.2.4}$$

At last, as shown in equation 3.2.5 we can make use of $\alpha$ table, $\beta$ table and current $\lambda$ values to calculate the derivatives. $Z_{\mathbf{o}}$ here represents $P(\mathbf{o})$, and we use it here fit target probability into (0, 1), and $Z_{\mathbf{o}}$ can be calculated easily by equation 3.2.6.

$$P(s_{t-1} = s, s_t = s'|\mathbf{o}) = \frac{1}{Z_{\mathbf{o}}} \alpha(t-1, s) \exp\left(\sum_k \lambda_k f_k(s, s', o_t)\right) \beta(t, s') \tag{3.2.5}$$

$$Z_{\mathbf{o}} = \beta(1, 0) + \beta(1, 1) \tag{3.2.6}$$

Now, we are ready to implement Stochastic Gradient Descent (SGD) algorithm using the derivatives. For each parameter $\lambda_k$, after computing the derivative of the log-likelihood function, modification method of $\lambda_k$ is shown in following equation. Notice since our goal is to find the global maximum instead of minimum, we use addition here.

$$\lambda_k \leftarrow \lambda_k + \theta \frac{\partial L}{\partial \lambda_k} \tag{3.2.7}$$

For the SGD, the learning rate $\theta$ can exert important influence on the final result. If learning rate $\theta$ is very small it would take so long time to change $\lambda$ to the proper values. If learning rate $\theta$ is too large, it will make the model unable to get converged. Actually, it will make the likelihood function even get a worse result.

## 3.3 Decoding

We implement Viterbi Algorithm to for decoding, which makes use of the states and corresponding $\lambda$ values for a given triples using equation 3.3.1. In the equation, $V(t, s)$ represent the probability of a partial labeling ending at position t with a label s, and we set $V(1, 1) = 1$ and $V(1, 0) = 0$, since the first character must be a beginning of a word.

$$V(t, s) = \begin{cases} \max_{s'}(V(t-1, s') \exp\left(\sum_k \lambda_k f_k(s', s, o_t)\right)) & t > 1 \\ s == 1 & t = 1 \end{cases} \tag{3.3.1}$$

In addition, in order to prevent overflow, we use the log trick to normalize the updating equation. Equation 3.3.2 shows the new updating rules after normalization.

$$V(t,s) = \begin{cases} \max_{s'}(V(t-1,s') + \sum_k \lambda_k f_k(s', s, o_t)) & t > 1 \\ V(1,0) = -\infty,\ V(1,1) = 0 & t = 1 \end{cases} \tag{3.3.2}$$

# 4 Experimental Results

We have so far tested our program with three extractions out of the AS training set, each with a random selection of 20, 200, and 1000 sentences. The best performances, shown below, are scored using the 'score' script provided along the package in bakeoff2005 [5]:

20-sentence-corpus:
TOTAL TRUE WORDS RECALL: 0.960
TOTAL TEST WORDS PRECISION: 0.954

200-sentence-corpus:
TOTAL TRUE WORDS RECALL: 0.774
TOTAL TEST WORDS PRECISION: 0.800

1000-sentence-corpus:
TOTAL TRUE WORDS RECALL: 0.465
TOTAL TEST WORDS PRECISION: 0.378

Meanwhile, we have implemented the Stanford Chinese segmenter ported into the NLTK as a comparison model of segmentation capability, and its score based upon the 200k-sentence training data and 2000-sentence test data is:

Stanford segmenter:
TOTAL TRUE WORDS RECALL: 0.946
TOTAL TEST WORDS PRECISION: 0.951

# 5 Analysis and Discussion

As is presented in the verdicts, our program is able to handle when the test case is relatively small. However, as the size of the test case increases, the inaccuracy floats along, leaving space for improvement in following aspects of the experiment.

## 5.1 Convergence detection

So far, we are only being able to detect manually when the parameters returned by SGD. We have only to manually control the number of loops running the regression process.

Therefore, it's necessary to introduce some automatic threshold, which depends on the size of the test input, to determine when the parameter $\lambda_k$ converges.

### 5.2 Unigram before bigram

In our experiment, we only focus on bigrams, which include the previous and current states and the current observation as a whole clique. We may introduce unigram into the parameter $\lambda_k$ for additional features, as well as a back-off factor for unseen words in testing data.

### 5.3 More precise container

We use Python's default float type for the precision calculation, which in some cases can result in underflow as probabilities can be extremely low. We may replace float with numpy's double type for a better precision.

## 6 Conclusion

In this report, we have traced our attempt on substantiating current methods on the practice of Chinese word segmentation from scratch. First we initialize the experiment with Conditional Random Field to assign binary labels on each input sentence to decide the cuts for segmentation. Treating each sentence as a feature, we then train the model with Stochastic Gradient Descent and the backward-forward algorithm to optimize the parameters of the features. Eventually, we decode the most probable segmentation sequence using Viterbi's. Although the current result with the optimal configuration is far from perfect, it shows that our method still has potential into better performance in future.

## 7 Acknowledgment

We would like to thank Prof. Eric Fosler-Lussier for his teachings throughout the semester offering us the opportunity to furthermore explore the knowledge in NLP.

## References

[1] Charles Sutton and Andrew McCallum: *An Introduction to Conditional Random Fields* Foundations and Trends in Machine Learning (FnT ML). (2010)

[2] John D. Lafferty, Andrew McCallum , and Fernando C. N. Pereira: *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data* Proceedings of the Eighteenth International Conference on Machine Learning, p.282-289 (2001)

[3] Fuchun Peng, Fangfang Feng, and Andrew McCallum: *Chinese segmentation and new word detection using conditional random fields* In Proc. of COLING. (2004)

[4] Pi-Chuan Chang, Michel Galley, and Chris Manning: *Optimizing Chinese Word Segmentation for Machine Translation Performance*, In ACL Workshop on Statistical Machine Translation. (2008)

[5] Second International Chinese Word Segmentation Bakeoff Data. (n.d.). `http://sighan.cs.uchicago.edu/bakeoff2005/`