Graph Model for Chinese Spell Checking*

Zhongye Jia, Peilu Wang and Hai Zhao†
MOE-Microsoft Key Laboratory for Intelligent Computing and Intelligent Systems,
Center for Brain-Like Computing and Machine Intelligence
Department of Computer Science and Engineering, Shanghai Jiao Tong University
800 Dongchuan Road, Shanghai 200240, China
jia.zhongye,plwang1990@gmail.com,zhaohai@cs.sjtu.edu.cn

Abstract

This paper describes our system in the Bake-Off 2013 task of SIGHAN 7. We illustrate that Chinese spell checking and correction can be efficiently tackled with by utilizing word segmenter. A graph model is used to represent the sentence and a single source shortest path (SSSP) algorithm is performed on the graph to correct spell errors. Our system achieves 4 first ranks out of 10 metrics on the standard test set.

1 Introduction and Task Description

Spell checking is a common task in every written language, which is an automatic mechanism to detect and correct human errors. However, spell checking in Chinese is very different from that in English or other alphabetical languages. In Bake-Off 2013, the evaluation includes two sub-tasks: detection and correction for Chinese spell errors. The errors are collected from students' written essays.

The object of spell checking is word, but "word" is not a natural concept for Chinese since there are no word delimiters between words. Most Chinese natural language processing tasks require an additional word segmentation phase beforehand. When a word is misspelled, the word segmentation could not be processed properly. Another problem with Chinese is that the difference between "characters" and "words" is not very clear. Most Chinese characters itself can also be words which are called "single-character words" in Chinese. Thus Chinese is a language that may never encounter "out-of-vocabulary (OOV)" problem. Spell errors in alphabetical languages, such as English, are always typically divided into two categories:

- The misspelled word is a non-word, for example "come" is misspelled into "cmoe";
- The misspelled word is still a legal word, for example "come" is misspelled into "cone".

Spell errors in Chinese are quite different. In Chinese, if the misspelled word is a non-word, the word segmenter will not recognize it as a word, but split it into two or more words with fewer characters. For example, if "你好世界 (hello world)" is misspelled into "你好世节", the word segmenter will segment it into "你好/世/节" instead of "你好/世节". For non-word spell error, the misspelled word will be mis-segmented.

Thus spell checking for Chinese cannot directly use those edit distance based methods which are commonly used for alphabetical languages. Spell checking for Chinese have to deal with word segmentation problem first, since misspelled sentence cannot be segmented properly by a normal word segmenter. And it is necessary to use information beyond word level to detect and correct those mis-segmented words.

In this paper, we describe the system submitted from the team of Shanghai Jiao Tong University (SJTU). We are inspired by the idea of shortest path word segmentation algorithm. A directed acyclic graph (DAG) is built from the input sentence similar to the shortest path word segmentation algorithm. The spell error detection and correction problem is transformed to the SSSP problem on the DAG. We also tried filters based on sen-

*This work was partially supported by the National Natural Science Foundation of China (Grant No.60903119, Grant No.61170114, and Grant No.61272248), and the National Basic Research Program of China (Grant No.2009CB320901 and Grant No.2013CB329401).

†Corresponding author

Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing (SIGHAN-7), pages 88–92,
Nagoya, Japan, 14 October 2013.

tence perplexity (PPL) and character mutual information (MI).

2 System Architecture

We utilize a modified shortest path word segmenter as the core part of spell checker. The original shortest path word segmentation algorithm is revised for spell checking. Instead of the segmented sentence, the output sentence of the modified word segmenter is both segmented and spell-checked.

2.1 The Shortest Path Word Segmentation Algorithm

Shortest path word segmentation algorithm (Casey and Lecolinet, 1996) is based on the following assumption: a reasonable segmentation should maximize the lengths of all segments or minimize the total number of segments. For a sentence S of m characters $\{c_1, c_2, \ldots, c_m\}$, the best segmented sentence S^* of n^* words $\{w_1^*, w_2^*, \ldots, w_{n^*}^*\}$ should be:

$$S^* = \underset{\{w_1, w_2, \ldots, w_n\}}{\arg\min} \; n. \qquad (1)$$

This optimization problem can be easily transformed to a SSSP problem on a DAG.

First a graph $G = (V, E)$ must be built to represent the sentence to be segmented. The vertices of G are possible candidate words of adjacent characters. The words are fetched from a dictionary $\mathbb{D}$. Two special vertices $w_{-,0}$ = "<S>" and $w_{n+1,-}$ = "</S>" are added to represent the start and end of the sentence:

$$V = \{w_{i,j} | w_{i,j} = c_i \ldots c_j \in \mathbb{D}\} \cup \{w_{-,0}, w_{n+1,-}\}.$$

The edges are from a word to the next word:

$$E = \{< w_{i,j} \rightarrow w_{j+1,k}, \omega > | w_{i,j}, w_{j+1,k} \in V\},$$

where ω is the weight of edge which is set to 1, $\omega = \omega_0 \equiv 1$.

For example, the Chinese sentence "床前明月光" can be represented by the graph shown in Figure 1. It can be easily proved that the graph G is a DAG, and finding the best segmentation according to Equation 1 is finding the shortest path from "<S>" to "</S>", which is an SSSP problem on DAG.

The SSSP problem on DAG have an simple algorithm (Eppstein, 1998) with time complexity of $O(|V| + |E|)$, The algorithm is shown in Algorithm 1.

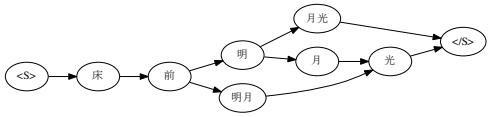


Figure 1: A sample of graph for segmentation

Algorithm 1 SSSP algorithm for word segmentation

Require: sentence of characters S
Require: dictionary $\mathbb{D}$
Ensure: segmented sentence s^*
Build DAG $G = (V, E)$ from S with $\mathbb{D}$
Topologically sort G into L
Init $D[v] \leftarrow -\infty, \forall v \in V$
Init $B[v] \leftarrow \Phi, \forall v \in V$
D[<S>] $\leftarrow 0$
for $u \in L$ **do**
  for v, ω s.t. $< u \rightarrow v, \omega > \in E$ **do**
    if $D[v] > D[u] + \omega$ **then**
      $D[v] \leftarrow D[u] + \omega$
      $B[v] \leftarrow u$
    end if
  end for
end for
$S^* = \Phi$
$v \leftarrow$ </S>
while $v \neq \Phi$ **do**
  Insert v into the front of S^*
  $v \leftarrow B[V]$
end while

The segmented sentence of the above example "床前明月光" is "床/前/明月/光" or "床/前/明/月光" by using the SSSP algorithm.

2.2 Using SSSP Algorithm fo Spell Checking

The basic idea of using SSSP algorithm for spell checking comes from the observation that a misspelled word is often splitted into two or more words by the shortest path word segmenter. If we can substitute the misspelled character with the correct one and provide it as a candidate word, then the shortest path word segmenter will choose the correct one, since it has less words.

Then there is no need to change the SSSP algorithm. Only the graph of sentence is built in a different way. The vertices consists not only candidate words composed of original adjacent charac-

ters, but also characters substituted by those similar to the original ones. An additional map of similar characters $\mathbb{C}$ is needed. The revised vertices V are:

$$\begin{aligned} V =& \{w_{i,j} | w_{i,j} = c_i \dots c_j \in \mathbb{D}\} \\ & \cup \{w_{i,j}^k | w_{i,j}^k = c_i \dots c'_k \dots c_j \in \mathbb{D}, \\ & \qquad \tau \le j - i \le T, \\ & \qquad c'_k \in \mathbb{C}[c_k], k = i, i+1, \dots, j\} \\ & \cup \{w_{-,0}, w_{n+1,-}\}. \end{aligned}$$

The substitution is only performed on those words with lenth between some thresholds τ and T. The weight of edges are respectively changed:

$$\omega = f(\omega_0, \omega_s),$$

where ω_s measures the similarity between the two characters and $f(\cdot, \cdot)$ is a function to be selected.

With the modified DAG G, the SSSP algorithm can perform both segmentation task and spell checking task. Suppose the sentence "床前明月光" is misspelled as "床前名月光", the modified graph is shown in Figure 2. The output of the

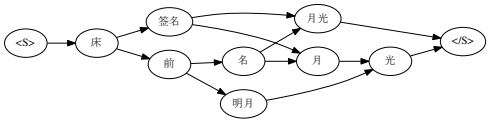


Figure 2: A sample of graph for spell checking

spell checker is "床/签名/月光", note this is not the expected result.

2.3 Using Language Model with SSSP Algorithm

The problem with simple SSSP spell chekcer is that it only tries to merge short words into longer ones without considering whether that is reasonable. To reduce the false-alarm rate (Wu et al., 2010), we add some statistical criteria to the SSSP spell checker.

A natural statistical criteria is the conditional probability between words, P, which can be given by a language model (LM). The conditional probability are combined into the weights of edges by some function $g(\cdot, \cdot, \cdot)$:

$$\omega = g(\omega_0, \omega_s, \frac{1}{P}),$$

Note that the higher conditional probability means the sentence is more reasonable, so for the SSSP algorithm, the inverse of conditional probability $\frac{1}{P}$ is used.

2.4 LM and MI Filter

In the sample set of Bake-Off 2013, we observed that there is at most one error in each sentence (Chen et al., 2011). But the spell checker may detect multiple errors. To choose the best correction, we run a filter and the one with lowest PPL or highest MI gain is selected.

For LM filter, sentence PPL is used as the metric. The correction with lowest PPL is considered as best.

MI indicates how possible two characters are collocated together. Character based MI is used, for two adjacent characters c_1 and c_2, the MI is:

$$\text{MI}(c_1, c_2) = \log \frac{P(c_1)P(c_2)}{P(c_1 c_2)}$$

The correction with highest MI gain Δ_{MI} is considered as best:

$$\begin{aligned} \Delta_{\text{MI}} = \max(&\text{MI}(c_{i-1}, c'_i) - \text{MI}(c_{i-1}, c_i), \\ &\text{MI}(c'_i, c_{i+1}) - \text{MI}(c_i, c_{i+1})). \end{aligned}$$

3 Experiments

3.1 Data Sets and Resources

The Bake-Off 2013 sample data, SAMPLE, consists 350 sentences without errors and 350 sentences with one error per sentence. The official test data, TEST, consists of 1,000 unlabeled sentences for subtask 1 and another 1,000 sentences for subtask 2. All the sentences are collected from students' written essays. All the data are in traditional Chinese.

The dictionary used in SSSP algorithm is *SogouW*[1] dictionary from *Sogou inc.*, which is in simplified Chinese. The *OpenCC*[2] converter is used to convert it into traditional Chinese. For similar character map the data set provided by (Liu et al., 2011) is used. The LM is built on the Academia Sinica corpus (Emerson, 2005) with IRSTLM toolkit (Federico et al., 2008). Prefix tree data structure is used to speed up the dictionary look-up. The implementation of Perl module `Tree::Trie`[3] is used with some modification.

3.2 Edge Weight Function selection

A series of experiments are performed to choose a good edge weight function $g(\cdot, \cdot, \cdot)$. A simplified metric is used to evaluate different functions:

[1] `http://www.sogou.com/labs/dl/w.html`
[2] `http://code.google.com/p/opencc/`
[3] `http://search.cpan.org/~avif/Tree-Trie-1.5/`

- Correction precision:

$$\mathcal{P} = \frac{\text{\# of correctly corrected characters}}{\text{\# of all corrected characters}};$$

- Correction recall:

$$\mathcal{R} = \frac{\text{\# of correctly corrected characters}}{\text{\# of wrong characters of gold data}};$$

- F1 macro:

$$\mathcal{F} = \frac{2\mathcal{P}\mathcal{R}}{\mathcal{P}+\mathcal{R}}.$$

The LM is set to 2-gram according to the observations of (Yang et al., 2012). Improved Kneser-Ney (Chen and Goodman, 1999) algorithm is used for LM smoothing.

Multiplication of similarity and log conditional probability is firstly used as weight function:

$$\omega^M = -\alpha(\omega_0 + \omega_s)\log P$$

where $\omega_0 \equiv 1$, and ω_s for different kinds of characters are shown in Table 1. The settings of ω_s is inspired by (Yang et al., 2012), in which pinyin[4] edit distance is used as weight. Word length threshold is set to $\tau = 2$ and $T = 5$. Experiments show that the choice of α does not have notable influence on the result which remains at $\mathcal{P} = 0.49$, $\mathcal{R} = 0.61$, $\mathcal{F} = 0.55$ on SAMPLE.

Type	ω_s
same pronunciation same tone	0
same pronunciation different tone	0
similar pronunciation same tone	1
similar pronunciation different tone	1
similar shape	1

Table 1: ω_s used in ω^M

Linear combination of similarity and log conditional probability is then tried:

$$\omega^L = \omega_s - \beta \log P$$

where $\omega_0 \equiv 0$ which is omitted in the equation, and ω_s for different kinds of characters are shown in Table 2.

We experimented with different β and observed that with lager β, the spell checker tends to get more reasonable corrections so $\mathcal{P}$ goes higher, but $\mathcal{R}$ goes lower. The $\mathcal{P}$, $\mathcal{R}$ and $\mathcal{F}$ on SAMPLE of different β are shown in Figure 3.

LM and MI filters slightly improves the result of the spell checker. The results of applying two filters are shown in Figure 4.

[4]Pinyin is the official phonetic system for transcribing the sound of Chinese characters into Latin script.

Type	ω_s
same pronunciation same tone	1
same pronunciation different tone	1
similar pronunciation same tone	2
similar pronunciation different tone	2
similar shape	2

Table 2: ω_s used in ω^L

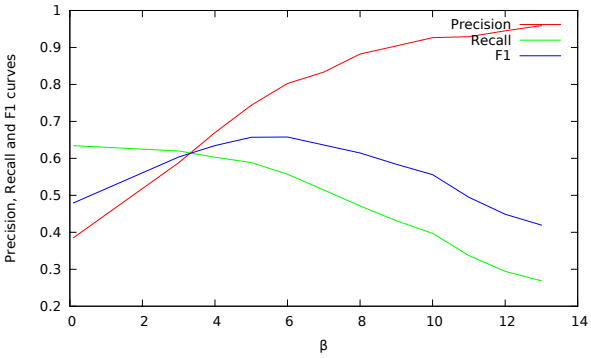


Figure 3: $\mathcal{P}$, $\mathcal{R}$ and $\mathcal{F}$ achieved by different β

3.3 Results

In the final test we submitted 3 runs, using edge weight function ω^L, of which β is set to 0, 6, and 10. Since there is no remarkable improvement by applying filters and the final test data has no claim that there's only one error per sentence, no filters are applied in the final test. The results on TEST are listed in Table 3 and Table 4, in which those metrics that we got first rank are marked as bold.

Metric	Run1	Run2	Run3
False-Alarm Rate	0.44	0.0957	**0.0229**
Detection Accuracy	0.662	0.856	0.844
Detection Precision	0.4671	0.769	**0.9091**
Detection Recall	0.9	0.7433	0.5333
Error Location Accuracy	0.522	0.805	0.809
Error Location Precision	0.2249	0.5931	**0.7102**
Error Location Recall	0.4333	0.5733	0.4167

Table 3: Final test results of subtask 1

Metric	Run1	Run2	Run3
Location Accuracy	0.372	0.475	0.37
Correction Accuracy	0.338	0.442	0.356
Correction Precision	0.3828	0.636	**0.705**

Table 4: Final test results of subtask 2

4 Conclusion and Future Work

In this paper we presented the system from team of Shanghai Jiao Tong University that participated in

the Bake-Off 2013 task. A graph model is utilized to represent the spell checking problem and SSSP algorithm is applied to solve it. By adjusting edge weight function, a trade-off can be made between precision and recall.

A problem with the current result is that the test data set is a manually collected one with very high error rate. In subtask 1, nearly 50% sentences contains spell errors and in subtask 2, every sentence contains at least one spell error. This error rate is far higher than that in normal text. We may consider using data from normal text in future work.

References

Richard G Casey and Eric Lecolinet. 1996. A survey of methods and strategies in character segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(7):690–706.

Stanley F Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393.

Yong-Zhi Chen, Shih-Hung Wu, Ping-Che Yang, and Tsun Ku. 2011. Improve the detection of improperly used chinese characters in students' essays with error model. *International Journal of Continuing Engineering Education and Life Long Learning*, 21(1):103–116.

Thomas Emerson. 2005. The second international chinese word segmentation bakeoff. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 123–133.

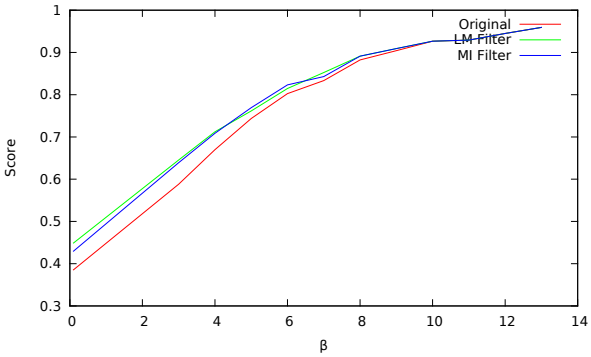
David Eppstein. 1998. Finding the k shortest paths. *SIAM Journal on computing*, 28(2):652–673.

Marcello Federico, Nicola Bertoldi, and Mauro Cettolo. 2008. Irstlm: an open source toolkit for handling large scale language models. In *Interspeech*, pages 1618–1621.

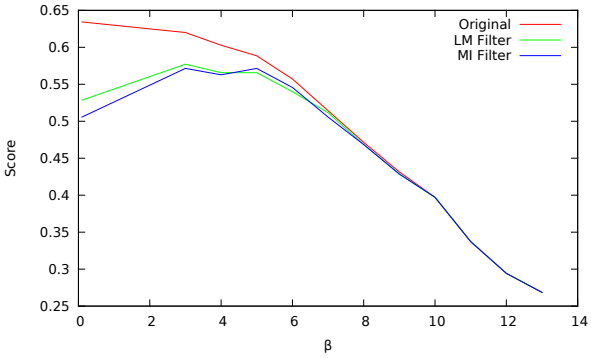
C.-L. Liu, M.-H. Lai, K.-W. Tien, Y.-H. Chuang, S.-H. Wu, and C.-Y. Lee. 2011. Visually and phonologically similar characters in incorrect chinese words: Analyses, identification, and applications. 10(2):10:1–10:39, June.

Shih-Hung Wu, Yong-Zhi Chen, Ping-Che Yang, Tsun Ku, and Chao-Lin Liu. 2010. Reducing the false alarm rate of chinese character error detection and correction. In *CIPS-SIGHAN Joint Conference on Chinese Language Processing*.

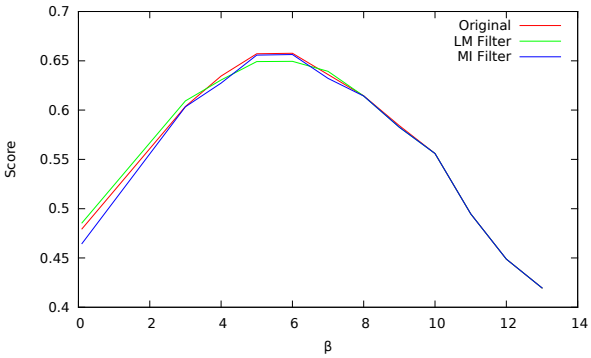
Shaohua Yang, Hai Zhao, Xiaolin Wang, and Baoliang Lu. 2012. Spell checking for chinese. In *International Conference on Language Resources and Evaluation*, pages 730–736, Istanbul, Turkey, May.



(a) Precision



(b) Recall



(c) F1

Figure 4: Precision, recall and F1 scores with filters