A Joint Graph Model for Pinyin-to-Chinese Conversion with Typo Correction*

Zhongye Jia 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@gmail.com, zhaohai@cs.sjtu.edu.cn

Abstract

It is very import for Chinese language processing with the aid of an efficient input method engine (IME), of which pinyin-to-Chinese (PTC) conversion is the core part. Meanwhile, though typos are inevitable during user pinyin inputting, existing IMEs paid little attention to such big inconvenience. In this paper, motivated by a key equivalence of two decoding algorithms, we propose a joint graph model to globally optimize PTC and typo correction for IME. The evaluation results show that the proposed method outperforms both existing academic and commercial IMEs.

1 Introduction

1.1 Chinese Input Method

The daily life of Chinese people heavily depends on Chinese input method engine (IME), no matter whether one is composing an E-mail, writing an article, or sending a text message. However, every Chinese word inputted into computer or cellphone cannot be typed through one-to-one mapping of key-to-letter inputting directly, but has to go through an IME as there are thousands of Chinese characters for inputting while only 26 letter keys are available in the keyboard. An IME is an essential software interface that maps Chinese characters into English letter combinations. An efficient IME will largely improve the user experience of Chinese information processing.

Nowadays most of Chinese IMEs are pinyin based. Pinyin is originally designed as the phonetic symbol of a Chinese character (based on the standard modern Chinese, mandarin) , using Latin letters as its syllable notation. For example, the pinyin of the Chinese character "爱"(love) is "ài". Most characters usually have unique pinyin representations, while a few Chinese characters may be pronounced in several different ways, so they may have multiple pinyin representations. The advantage of pinyin IME is that it only adopts the pronunciation perspective of Chinese characters so that it is simple and easy to learn. But there are only less than 500 pinyin syllables in standard modern Chinese, compared with over 6,000 commonly used Chinese characters, which leads to serious ambiguities for pinyin-to-character mapping. Modern pinyin IMEs mostly use a "*sentence-based*" decoding technique (Chen and Lee, 2000) to alleviate the ambiguities. "*Sentence based*" means that IME generates a sequence of Chinese characters upon a sequence of pinyin inputs with respect to certain statistical criteria.

1.2 Typos and Chinese Spell Checking

Written in Chinese characters but not alphabets, spell checking for Chinese language is quite different from the same task for other languages. Since Chinese characters are entered via IME, those user-made typos do not immediately lead to spelling errors. When a user types a wrong letter, IME will be very likely to fail to generate the expected Chinese character sequence. Normally, the user may immediately notice the inputting error and then make corrections, which usually means doing a bunch of extra operations like cursor

*This work was partially supported by the National Natural Science Foundation of China (Grant No.60903119, Grant No.61170114, and Grant No.61272248), the National Basic Research Program of China (Grant No.2013CB329401), the Science and Technology Commission of Shanghai Municipality (Grant No.13511500200), and the European Union Seventh Framework Program (Grant No.247619).

†Corresponding author

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1512–1523,
Baltimore, Maryland, USA, June 23-25 2014.

movement, deletion and re-typing. Thus there are two separated sub-tasks for Chinese spell checking: 1. *typo checking* for user typed pinyin sequences which should be a built-in module in IME, and 2. *spell checking* for Chinese texts in its narrow sense, which is typically a module of word processing applications (Yang et al., 2012b). These two terms are often confused especially in IME related works such as (Chen and Lee, 2000) and (Wu et al., 2009).

Pinyin typos have always been a serious problem for Chinese pinyin IMEs. The user may fail to input the completely right pinyin simply because he/she is a dialect speaker and does not know the exact pronunciation for the expected character. This may be a very common situation since there are about seven quite different dialects in Chinese, among which being spoken languages, six are far different from the standard modern Chinese, mandarin. With the boom of smart-phones, pinyin typos worsen due to the limited size of soft keyboard, and the lack of physical feedback on the touch screen. However, existing practical IMEs only provide small patches to deal with typos such as *Fuzzy Pinyin* (Wu and Chen, 2004) and other language specific errors (Zheng et al., 2011b).

Typo checking and correction has an important impact on IME performance. When IME fails to correct a typo and generate the expected sentence, the user will have to take much extra effort to move the cursor back to the mistyped letter and correct it, which leads to very poor user experience (Jia and Zhao, 2013).

2 Related Works

The very first approach for Chinese input with typo correction was made by (Chen and Lee, 2000), which was also the initial attempt of "sentence-based" IME. The idea of "statistical input method" was proposed by modeling PTC conversion as a hidden Markov model (HMM), and using Viterbi (Viterbi, 1967) algorithm to decode the sequence. They solved the typo correction problem by decomposing the conditional probability $P(H|P)$ of Chinese character sequence H given pinyin sequence P into a language model $P(w_i|w_{i-1})$ and a typing model $P(p_i|w_i)$. The typing model that was estimated on real user input data was for typo correction. However, real user input data can be very noisy and not very convenient to obtain. As we will propose a joint model in this paper, such an individual typing model is not necessarily built in our approach.

(Zheng et al., 2011a) developed an IME system with typo correction called CHIME using noisy channel error model and language-specific features. However their model depended on a very strong assumption that input pinyin sequence should have been segmented into pinyin words by the user. This assumption does not really hold in modern "sentence-based" IMEs. We release this assumption since our model solves segmentation, typo correction and PTC conversion jointly.

Besides the common HMM approach for PTC conversion, there are also various methods such as: support vector machine (Jiang et al., 2007), maximum entropy (ME) model (Wang et al., 2006), conditional random field (CRF) (Li et al., 2009) and statistical machine translation (SMT) (Yang et al., 2012a; Wang et al., 2013c; Zhang and Zhao, 2013), etc.

Spell checking or typo checking was first proposed for English (Peterson, 1980). (Mays et al., 1991) addressed that spell checking should be done within a context, i.e., a sentence or a long phrase with a certain meaning, instead of only in one word. A recent spell correction work is (Li et al., 2006), where a distributional similarity was introduced for spell correction of web queries.

Early attempts for Chinese spelling checking could date back to (Chang, 1994) where character tables for similar shape, pronunciation, meaning, and input-method-code characters were proposed. More recently, the 7th SIGHAN Workshop on Chinese Language Processing (Yu et al., 2013) held a shared task on Chinese spell checking. Various approaches were made for the task including language model (LM) based methods (Chen et al., 2013), ME model (Han and Chang, 2013), CRF (Wang et al., 2013d; Wang et al., 2013a), SMT (Chiu et al., 2013; Liu et al., 2013), and graph model (Jia et al., 2013), etc.

3 Pinyin Input Method Model

3.1 From English Letter to Chinese Sentence

It is a rather long journey from the first English letter typed on the keyboard to finally a completed Chinese sentence generated by IME. We will first take an overview of the entire process.

The average length of pinyin syllables is about 3 letters. There are about 410 pinyin syllables used in the current pinyin system. Each pinyin sylla-

ble has a bunch of corresponding Chinese characters which share the same pronunciation represented by the syllable. The number of those homophones ranges from 1 to over 300. Chinese characters then form words. But word in Chinese is a rather vague concept. Without word delimiters, linguists have argued on what a Chinese word really is for a long time and that is why there is always a primary word segmentation treatment in most Chinese language processing tasks (Zhao et al., 2006; Huang and Zhao, 2007; Zhao and Kit, 2008; Zhao et al., 2010; Zhao and Kit, 2011; Zhao et al., 2013). A Chinese word may contain from 1 to over 10 characters due to different word segmentation conventions. Figure 1 demonstrates the relationship of pinyin and word, from pinyin letters "*nihao*" to the word "你好 *(hello)*". Typically, an IME takes the pinyin input, segments it into syllables, looks up corresponding words in a dictionary and generates a sentence with the candidate words.

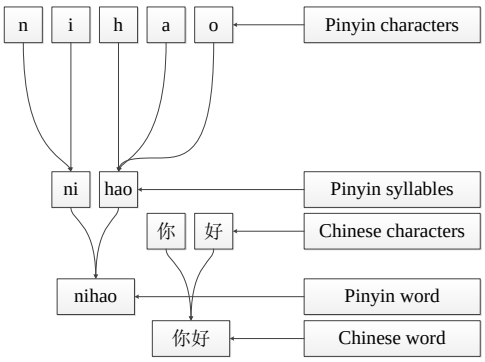


Figure 1: Relationship of pinyin and words

3.2 Pinyin Segmentation and Typo Correction

Non-Chinese users may feel confused or even surprised if they know that when typing pinyin through an IME, Chinese IME users will never enter delimiters such as "Space" key to segment either pinyin syllables or pinyin words, but just input the entire un-segmented pinyin sequence. For example, if one wants to input "你好世界 *(Hello world)*", he will just type "*nihaoshijie*" instead of segmented pinyin sequence "*ni hao shi jie*". Nevertheless, pinyin syllable segmentation is a much easier problem compared to Chinese word segmentation. Since pinyin syllables have a very limited vocabulary and follow a set of regularities strictly, it is convenient to perform pinyin syllable segmentation by using rules. But as the pinyin input is not segmented, it is nearly impossible to adopt previous spell checking methods for English to pinyin typo checking, although techniques for English spell checking have been well developed. A bit confusing but interesting, pinyin typo correction and segmentation come as two sides of one problem: when a pinyin sequence is mistyped, it is unlikely to be correctly segmented; when it is segmented in an awkward way, it is likely to be mistyped.

Inspired by (Yang et al., 2012b) and (Jia et al., 2013), we adopt the graph model for Chinese spell checking for pinyin segmentation and typo correction, which is based on the shortest path word segmentation algorithm (Casey and Lecolinet, 1996). The model has two major steps: segmentation and correction.

3.2.1 Pinyin Segmentation

The shortest path segmentation algorithm is based on the idea that a reasonable segmentation should minimize the number of segmented units. For a pinyin sequence $p_1 p_2 \dots p_L$, where p_i is a letter, first a directed acyclic graph (DAG) $G_S = (\mathbb{V}, \mathbb{E})$ is built for pinyin segmentation step. The vertex set $\mathbb{V}$ consists of the following parts:

- Virtual start vertex S_0 and end vertex S_E;
- Possible legal syllables fetched from dictionary $\mathbb{D}_p$ according to the input pinyin sequence:

$$\{S_{i,j} | S_{i,j} = p_i \dots p_j \in \mathbb{D}_p\};$$

- The letter itself as a fallback no matter if it is a legal pinyin syllable or not:

$$\{S_i | S_i = p_i\}.$$

The vertex weights w_S are all set to 0. The edges are from a syllable to all syllables next to it:

$$\mathbb{E} = \{E(S_{i,j} \to S_{j+1,k}) | S_{i,j}, S_{j+1,k} \in \mathbb{V}\}.$$

The edge weight the negative logarithm of conditional probability $P(S_{j+1,k}|S_{i,j})$ that a syllable $S_{i,j}$ is followed by $S_{j+1,k}$, which is give by a bigram language model of pinyin syllables:

$$W_{E(S_{i,j} \to S_{j+1,k})} = -\log P(S_{j+1,k}|S_{i,j})$$

The shortest path P^* on the graph is the path P with the least sum of weights:

$$P^* = \underset{(v,E) \in G \wedge (v,E) \in P}{\arg\min} \sum_v w_v + \sum_E W_E.$$

Computing the shortest path from S_0 to S_E on G_S yields the best segmentation. This is the single source shortest path (SSSP) problem on DAG which has an efficient algorithm by preprocessing the DAG with topology sort, then traversing vertices and edges in topological order. It has the time complexity of $O(|\mathbb{V}| + |\mathbb{E}|)$. For example, one intends to input "你好世界 *(Hello world)*" by typing "*nihaoshijie*", but mistyped as "*mihaoshijiw*". The graph for this input is shown in Figure 2. The shortest path, i.e., the best segmentation is "*mi hao shi ji w*". We will continue to use this example in the rest of this paper.

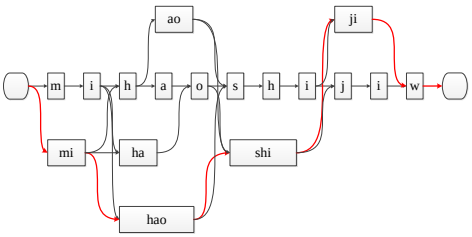


Figure 2: Graph model for pinyin segmentation

3.2.2 Pinyin Typo Correction

Next in the correction step, for the segmented pinyin sequence $S_1, S_2, \ldots, S_M$, a graph G_c is constructed to perform typo correction. The vertex set $\mathbb{V}$ consists of the following parts:

- Virtual start vertex S'_0 and end vertex S'_E with vertex weights of 0;
- All possible syllables similar to original syllable in G_s. If the adjacent syllables can be merged into a legal syllable, the merged syllable is also added into $\mathbb{V}$:

$$\{S'_{i,j} | S'_{i,j} = S'_i \ldots S'_j \in \mathbb{D}_p, \\ S'_k \sim S_k, k = i \le j\},$$

where the similarity $\sim$ is measured in Levenshtein distance (Levenshtein, 1966). Syllables with Levenshtein distance under a certain threshold are considered as similar:

$$\mathcal{L}(S_i, S_j) < T \leftrightarrow S_i \sim S_j.$$

The vertex weight is the Levenshtein distance multiply by a normalization parameter:

$$w_{S'_{i,j}} = \beta \sum_{k-i}^{j} \mathcal{L}(S'_k, S_k).$$

Similar to G_s, the edges are from one syllable to all syllables next to it and edge weights are the conditional probabilities between them. Computing the shortest path from S'_0 to S'_E on G_c yields the best typo correction result. In addition, the result has been segmented so far. Considering our running example, the graph G_c is shown in Figure 3, and the typo correction result is "*mi hao shi jie*".

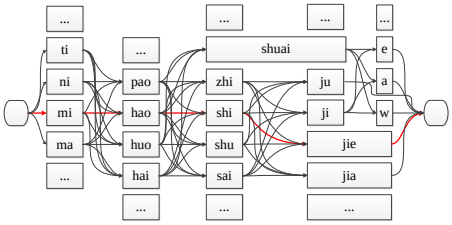


Figure 3: Graph model for pinyin typo correction

Merely using the above model, the typo correction result is not satisfying yet, no matter how much effort is paid. The major reason is that the basic semantic unit of Chinese language is actually word (tough vaguely defined) which is usually composed of several characters. Thus the conditional probability between characters does not make much sense. In addition, a pinyin syllable usually maps to dozens or even hundreds of corresponding homophonic characters, which makes the conditional probability between syllables much more noisy. However, using pinyin words instead of syllables is not a wise choice because pinyin word segmentation is not so easy a task as syllable segmentation. To make typo correction better, we consider to integrate it with PTC conversion using a joint model.

3.3 Hidden Markov Model for Pinyin-to-Chinese Conversion

PTC conversion has long been viewed as a decoding problem using HMM. We continue to follow this formalization. The best Chinese character sequence W^* for a given pinyin syllable sequence S is the one with the highest conditional probability $P(W|S)$ that

$$\begin{aligned} W^* &= \arg\max_W P(W|S) \\ &= \arg\max_W \frac{P(W)P(S|W)}{P(S)} \\ &= \arg\max_W P(W)P(S|W) \\ &= \arg\max_{w_1, w_w, \ldots, w_M} \prod_{w_i} P(w_i|w_{i-1}) \prod_{w_i} P(s_i|w_i) \end{aligned}$$

In the HMM for pinyin IME, observation states are pinyin syllables, hidden states are Chinese words, emission probability is $P(s_i|w_i)$, and transition probability is $P(w_i|w_{i-1})$. Note the transition probability is the conditional probability between words instead of characters. PTC conversion is to decode the Chinese word sequence from the pinyin sequence. The Viterbi algorithm (Viterbi, 1967) is used for the decoding.

The shortest path algorithm for typo correction and Viterbi algorithm for PTC conversion are very closely related. It has been strictly proven by (Forney, 1973) that the sequence decoding problem on HMM is formally identical to finding a shortest path on a certain graph, which can be constructed in the following manner.

Consider a first order HMM with all possible observations $\mathbb{O} = \{o_1, o_2, \ldots, o_M\}$, hidden states $\mathbb{S} = \{s_1, s_2, \ldots, s_N\}$, a special start state s_0, emission probabilities $(\mathcal{E}_{s_i,o_k}) = P(o_k|s_i)$, transition probabilities $(\mathcal{T}_{s_i,s_j}) = P(s_j|s_i)$, and start probabilities $(\mathcal{S}_{s_i}) = P(s_i|s_0)$. For an observation sequence of T time periods $Y = \{y_1, y_2, \ldots, y_T | y_t \in \mathbb{O}, t = 1, \ldots, T\}$, the decoding problem is to find the best corresponding hidden state sequence X^* with the highest probability, i.e.,

$$X^* = \arg\max_{x_1, x_t \in \mathbb{S}} \mathcal{S}_{x_1} \mathcal{E}_{x_1,y_1} \prod_{t=2}^{T} \mathcal{E}_{x_t,y_t} \mathcal{T}_{x_{t-1},x_t}. \quad (1)$$

Then we will construct a DAG $G = (\mathbb{V}, \mathbb{E})$ upon the HMM. The vertex set $\mathbb{V}$ includes:

- Virtual start vertex v_0 and end vertex v_E with vertex weight of 0;
- Normal vertices v_{x_t}, where $t = 1, \ldots, T$, and $\forall x_t \in \mathbb{S}$. The vertex weight is the negative logarithm of emission probability:

$$w_{v_{x_t}} = -\log \mathcal{E}_{x_t,y_t}.$$

The edge set $\mathbb{E}$ includes:

- Edges from the start vertex $E(v_0 \to v_{x_1})$ with edge weight

$$W_{E(v_0 \to v_{x_1})} = -\log \mathcal{S}_{x_1},$$

where $\forall x_1 \in \mathbb{S}$;

- Edges to the end vertex $E(v_{x_T} \to v_E)$ with vertex weights of 0;
- Edges between adjacent time periods $E(v_{x_{t-1}} \to v_{x_t})$ with edge weight

$$W_{E(v_{x_{t-1}} \to v_{x_t})} = -\log \mathcal{T}_{x_{t-1},x_t},$$

where $t = 2, \ldots, T$, and $\forall x_t, x_{t-1} \in \mathbb{S}$.

The shortest path P^* from v_0 to v_E is the one with the least sum of vertex and edge weights, i.e.,

$$\begin{aligned} P^* &= \arg\min_{v_{x_t} \in \mathbb{V}} \sum_{t=1}^{T} (w_{v_{x_t}} + W_{E(v_{x_{t-1}} \to v_{x_t})}) \\ &= \arg\min_{v_{x_1}, v_{x_t} \in \mathbb{V}} \{-\log \mathcal{S}_{x_1} - \log \mathcal{E}_{x_1,y_1} \\ &\quad + \sum_{t=2}^{T} (-\log \mathcal{E}_{x_t,y_t} - \log \mathcal{T}_{x_{t-1},x_t})\} \\ &= \arg\max_{v_{x_1}, v_{x_t} \in \mathbb{V}} \mathcal{S}_{x_1} \mathcal{E}_{x_1,y_1} \prod_{t=2}^{T} \mathcal{E}_{x_t,y_t} \mathcal{T}_{x_{t-1},x_t}. \quad (2) \end{aligned}$$

The optimization goal of P^* in Equation (2) is identical to that of X^* in Equation (1).

3.4 Joint Graph Model For Pinyin IME

Given HMM decoding problem is identical to SSSP problem on DAG, we propose a joint graph model for PTC conversion with typo correction. The joint graph model aims to find the global optimal for both PTC conversion and typo correction on the entire input pinyin sequence. The graph $G = (\mathbb{V}, \mathbb{E})$ is constructed based on graph G_c for typo correction in Section 3.2. The vertex set $\mathbb{V}$ consists of the following parts:

- Virtual start vertex V_0 and end vertex V_E with vertex weight of 0;
- Adjacent pinyin syllables in G_c are merged into pinyin words. Corresponding Chinese words are fetched from a PTC dictionary $\mathbb{D}_c$, which is a dictionary maps pinyin words to Chinese words, and added as vertices:

$$\{V_{i,j} | \forall V_{i,j} \in \mathbb{D}_c[S'_i \ldots S'_j], i \le j\};$$

The vertex weight consists of two parts: 1. the vertex weights of syllables in G_c, and 2. the emission probability:

$$\begin{aligned} w_{V_{i,j}} =& \beta \sum_{k=i}^{j} \mathcal{L}(S'_k, S_k) \\ & - \gamma \log P(S'_i \ldots S'_j | V_{i,j}); \end{aligned}$$

If the corresponding pinyin syllables in G_c have an edge between them, the vertices in G also have an edge:

$$\mathbb{E} = \{E(V_{i,j} \rightarrow V_{j+1,k}) | E(S_{i,j} \rightarrow S_{j+1,k}) \in \mathbb{G}_c\}.$$

The edge weights are the negative logarithm of the transition probabilities:

$$W_{E(V_{i,j} \rightarrow V_{j+1,k})} = -\log P(V_{j+1,k} | V_{i,j})$$

Although the model is formulated on first order HMM, i.e., the LM used for transition probability is a bigram one, it is easy to extend the model to take advantage of higher order n-gram LM, by tracking longer history while traversing the graph.

Computing the shortest path from V_0 to V_E on G yields the best pinyin-to-Chinese conversion with typo correction result. Considering our running example, the graph G is shown in Figure 4.

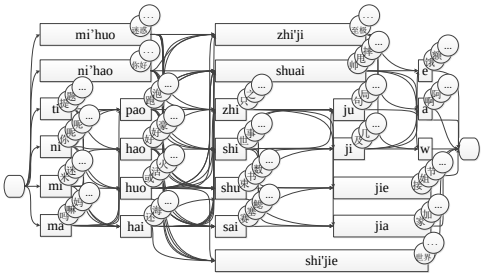


Figure 4: Joint graph model

The joint graph is rather huge and density. According to our empirical statistics, when setting threshold $T = 2$, for a sentence of M characters, the joint graph will have $|\mathbb{V}| = M \times 1,000$, and $|\mathbb{E}| = M \times 1,000,000$.

3.5 K-Shortest Paths

To reduce the scale of graph G, we filter graph G_c by searching its K-shortest paths first to get G'_c and construct G on top of G'_c. Figure 5 shows the 3-shortest paths filtered graph G'_c and Figure 6 shows the corresponding G for our running example. The scale of graph may be thus drastically reduced.

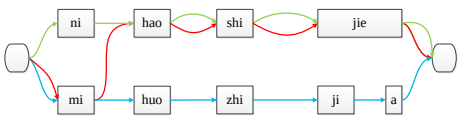


Figure 5: K-shortest paths in typo correction

An efficient heap data structure is required in K-shortest paths algorithm (Eppstein, 1998) for backtracking the best paths to current vertex while traversing. The heap is implemented as a priority queue of size K sorted according to path length that should support efficient push and pop operations. Fibonacci heap (Fredman and Tarjan, 1987) is adopted for the heap implementation since it has a push complexity of $O(1)$ which is better than the $O(K)$ for other heap structures.

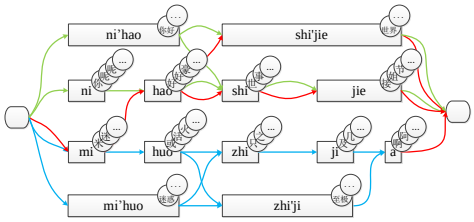


Figure 6: Filtered graph model

Another benefit provided by K-shortest paths is that it can be used for generating N-best candidates of PTC conversion, which may be helpful for further performance improvement.

4 Experiments

4.1 Corpora, Tools and Experiment Settings

The corpus for evaluation is the one provided in (Yang et al., 2012a), which is originally extracted from the *People's Daily* corpus and labeled with pinyin. The corpus has already been split into training **Train**, development **Dev** and test **Test** sets as shown in Table 1.

	Train	**Dev**	**Test**
#Sentence	1M	2K	100K
#character	43,679,593	83,765	4,123,184

Table 1: Data set size

SRILM (Stolcke, 2002) is adopted for language model training and KenLM (Heafield, 2011; Heafield et al., 2013) for language model query. The Chinese part of the corpus is segmented into words before LM training. Maximum matching word segmentation is used with a large word vocabulary $\mathcal{V}$ extracted from web data provided by (Wang et al., 2013b). The pinyin part is segmented according to the Chinese part. This vocabulary $\mathcal{V}$ also serves as the PTC dictionary. The original vocabulary is not labeled with pinyin, thus we use the PTC dictionary of *sunpinyin*[1] which is an open source Chinese pinyin IME, to label the

[1] http://code.google.com/p/sunpinyin/

vocabulary $\mathcal{V}$ with pinyin. The emission probabilities are estimated using the lexical translation module of MOSES (Koehn et al., 2007) as "translation probability" from pinyin to Chinese.

4.2 Evaluation Metrics

We will use conventional sequence labeling evaluation metrics such as sequence accuracy and character accuracy[2].

Chinese characters in a sentence may be separated by digits, punctuation and alphabets which are directly inputted without the IME. We follow the so-called term *Max Input Unit* (MIU), the longest consecutive Chinese character sequence proposed by (Jia and Zhao, 2013). We will mainly consider MIU accuracy (MIU-Acc) which is the ratio of the number of completely corrected generated MIUs over the number of all MIUs, and character accuracy (Ch-Acc), but the sentence accuracy (S-Acc) will also be reported in evaluation results.

We will also report the conversion error rate (ConvER) proposed by (Zheng et al., 2011a), which is the ratio of the number of mistyped pinyin word that is not converted to the right Chinese word over the total number of mistyped pinyin words[3].

4.3 Baseline System without Typo Correction

Firstly we build a baseline system without typo correction which is a pipeline of pinyin syllable segmentation and PTC conversion. The baseline system takes a pinyin input sequence, segments it into syllables, and then converts it to Chinese character sequence.

The pinyin syllable segmentation already has very high (over 98%) accuracy with a trigram LM using improved Kneser-Ney smoothing. According to our empirical observation, emission probabilities are mostly 1 since most Chinese words have unique pronunciation. So in this step we set $\gamma = 0$. We consider different LM smoothing methods including Kneser-Ney (KN), improved Kneser-Ney (IKN), and Witten-Bell (WB). All of the three smoothing methods for bigram and trigram LMs are examined both using back-off models and interpolated models. The number of N-best candidates for PTC conversion is set to 10. The results on **Dev** are shown in Figure 7 in which the "-i" suffix indicates using interpolated model. According to the results, we then choose the trigram LM using Kneser-Ney smoothing with interpolation.

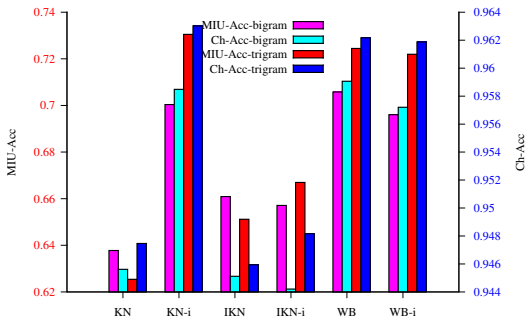


Figure 7: MIU-Acc and Ch-Acc with different LM smoothing

The choice of the number of N-best candidates for PTC conversion also has a strong impact on the results. Figure 8 shows the results on **Dev** with different Ns, of which the N axis is drawn in logarithmic scale. We can observe that MIU-Acc slightly decreases while N goes up, but Ch-Acc largely increases. We therefore choose $N = 10$ as trade-off.

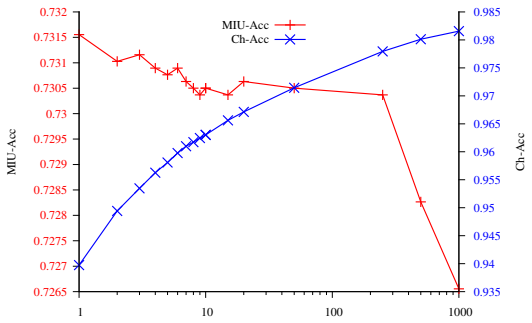


Figure 8: MIU-Acc and Ch-Acc with different Ns

The parameter γ determines emission probability. Results with different γ on **Dev** is shown in Figure 9, of which the γ axis is drawn in logarithmic scale. $\gamma = 0.03$ is chosen at last.

We compare our baseline system with several practical pinyin IMEs including *sunpinyin* and *Google Input Tools* (Online version)[4]. The results on **Dev** are shown in Table 2.

[2]We only work on the PTC conversion part of IME, thus we are unable to use existing evaluation systems (Jia and Zhao, 2013) for full Chinese IME functions.

[3]Other evaluation metrics are also proposed by (Zheng et al., 2011a) which is only suitable for their system since our system uses a joint model

[4]http://www.google.com/inputtools/try/

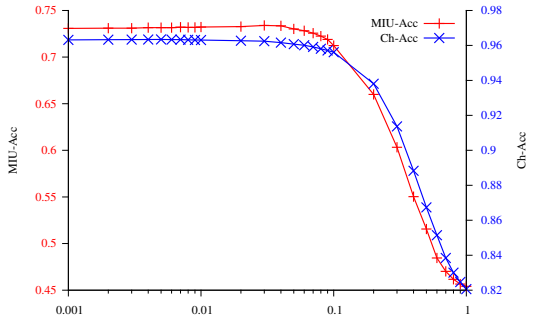


Figure 9: MIU-Acc and Ch-Acc with different γ

	MIU-Acc	Ch-Acc	S-Acc
Baseline	73.39	96.24	38.00
sunpinyin	52.37	87.51	13.95
Google	74.74	94.81	40.2
Yang-ME	-	93.3	30.2
Yang-MT	-	95.5	45.4

Table 2: Baseline system compared to other IMEs (%)

4.4 PTC Conversion with Typo Correction

Based upon the baseline system, we build the joint system of PTC conversion with typo correction.

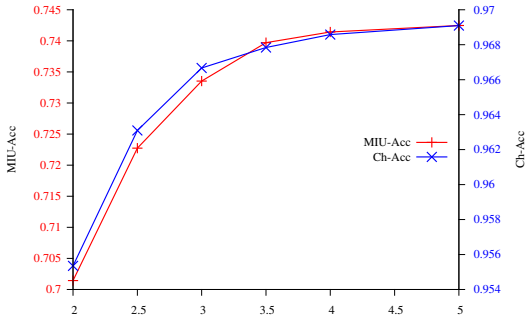
We simulate user typos by randomly generating errors automatically on the corpus. The typo rate is set according to previous Human-Computer Interaction (HCI) studies. Due to few works have been done on modeling Chinese text entry, we have to refer to those corresponding results on English (Wobbrock and Myers, 2006; MacKenzie and Soukoreff, 2002; Clarkson et al., 2005), which show that the average typo rate is about 2%. (Zheng et al., 2011a) performed an experiment that 2,000 sentences of 11,968 Chinese words were entered by 5 native speakers. The collected data consists of 775 mistyped pinyin words caused by one edit operation, and 85 caused by two edit operations. As we observe on **TRAIN** that the average pinyin word length is 5.24, then typo rate in the experiment of (Zheng et al., 2011a) can be roughly estimated as:

$$\frac{775 + 85 \times 2}{11968 \times 5.24} = 1.51\%,$$

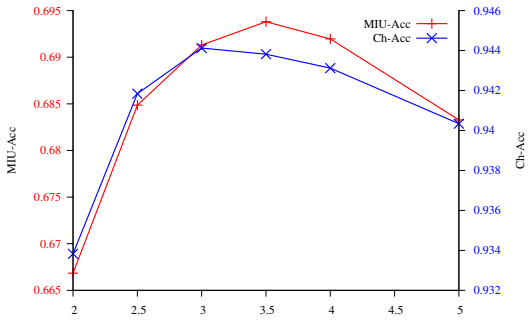
which is similar to the conclusion on English. Thus we generate corpora from **DEV** with typo rate of 0% (**0-P**), 2% (**2-P**), and 5% (**5-P**) to evaluate the system.

According to (Zheng et al., 2011a) most mistyped pinyin words are caused by one edit operation. Since pinyin syllable is much shorter than pinyin word, this ratio can be higher for pinyin syllables. From our statistics on **TRAIN**, with 2% randomly generated typos, $Pr(\mathcal{L}(S', S) < 2) = 99.86\%$. Thus we set the threshold T for $\mathcal{L}$ to 2.

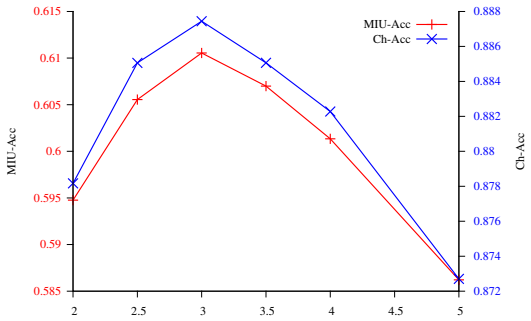
We first set K-shortest paths filter to $K = 10$ and tune β. Results with different β are shown in Figure 10. With $\beta = 3.5$, we select K. Results with different K are shown in Figure 11. We choose $K = 20$ since there is no significant improvement when $K > 20$.



(a) **0-P**



(b) **2-P**

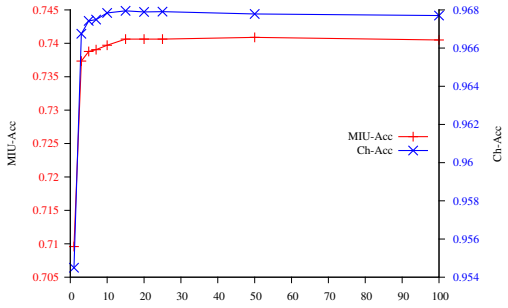


(c) **5-P**

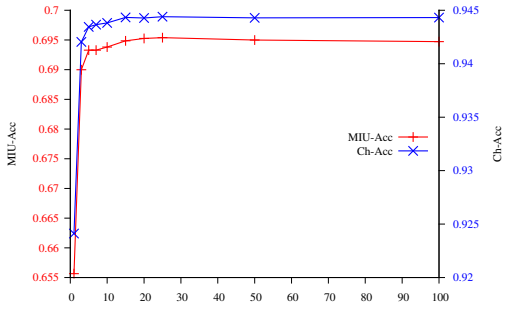
Figure 10: MIU-Acc and Ch-Acc with different β

The selection of K also directly guarantees the running time of the joint model. With $K = 20$, on a normal PC with Intel Pentium Dual-Core E6700 CPU, the PTC conversion rate is over 2000 characters-per-minute (*cpm*), which is much faster than the normal typing rate of 200 *cpm*.

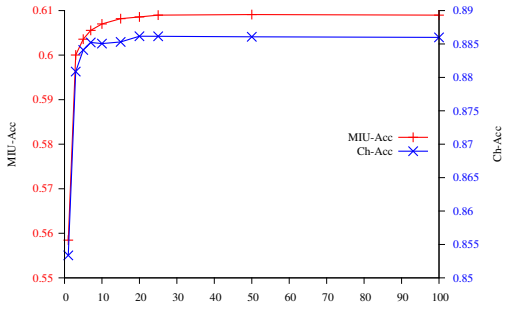
With all parameters optimized, results on **TEST**



(a) **0-P**



(b) **2-P**



(c) **5-P**

Figure 11: MIU-Acc and Ch-Acc with different K

using the proposed joint model are shown in Table 3 and Table 4. Our results are compared to the baseline system without typo correction and *Google Input Tool*. Since *sunpinyin* does not have typo correction module and performs much poorer than our baseline system, we do not include it in the comparison. Though no direct proofs can be found to indicate if Google Input Tool has an independent typo correction component, its outputs show that such a component is unlikely available.

Since *Google Input Tool* has to be accessed through a web interface and the network connection cannot be guaranteed. we only take a subset of 10K sentences of **TEST** to perform the experiments, and the results are shown in Table 3.

The scores reported in (Zheng et al., 2011a) are not listed in Table 4 since the data set is different. They reported a ConvER of 53.56%, which is given here for reference.

Additionally, to further inspect the robustness of our model, performance with typo rate ranges from 0% to 5% is shown in Figure 12. Although the performance decreases while typo rate goes up, it is still quite satisfying around typo rate of 2% which is assumed to be the real world situation.

	MIU-Acc	Ch-Acc	S-Acc	ConvER
Baseline **0-P**	79.90	97.47	48.87	-
Baseline **2-P**	50.47	90.53	11.12	99.95
Baseline **5-P**	30.26	82.83	3.32	99.99
Google **0-P**	79.08	95.26	46.83	-
Google **2-P**	49.47	61.50	11.08	91.70
Google **5-P**	29.18	36.20	3.29	94.64
Joint **0-P**	79.90	97.52	49.27	-
Joint **2-P**	75.55	95.40	40.69	18.45
Joint **5-P**	67.76	90.17	27.86	24.68

Table 3: Test results on 10K sentences from **TEST** (%)

	MIU-Acc	Ch-Acc	S-Acc	ConvER
Baseline **0-P**	74.46	96.42	40.50	-
Baseline **2-P**	47.25	89.50	9.62	99.95
Baseline **5-P**	28.28	81.74	2.63	99.98
Joint **2-P**	74.22	96.39	40.34	-
Joint **2-P**	69.91	94.14	33.11	21.35
Joint **5-P**	62.14	88.49	22.62	27.79

Table 4: Test results on **TEST** (%)

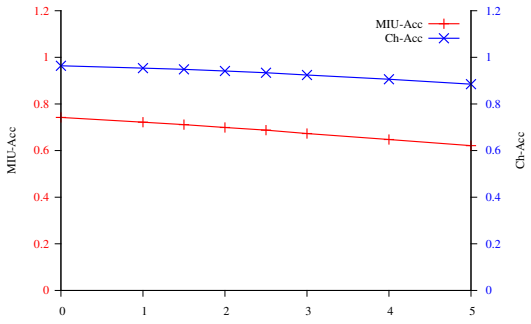


Figure 12: MIU-Acc and Ch-Acc with different typo rate (%)

5 Conclusion

In this paper, we have developed a joint graph model for pinyin-to-Chinese conversion with typo correction. This model finds a joint global optimal for typo correction and PTC conversion on the entire input pinyin sequence. The evaluation results show that our model outperforms both previous academic systems and existing commercial products. In addition, the joint model is efficient enough for practical use.

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.

Chao-Huang Chang. 1994. A Pilot Study on Automatic Chinese Spelling Error Correction. *Journal of Chinese Language and Computing*, 4:143–149.

Zheng Chen and Kai-Fu Lee. 2000. A New Statistical Approach To Chinese Pinyin Input. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 241–247, Hong Kong, October.

Kuan-Yu Chen, Hung-Shin Lee, Chung-Han Lee, Hsin-Min Wang, and Hsin-Hsi Chen. 2013. A Study of Language Modeling for Chinese Spelling Check. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 79–83, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Hsun-wen Chiu, Jian-cheng Wu, and Jason S. Chang. 2013. Chinese Spelling Checker Based on Statistical Machine Translation. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 49–53, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Edward Clarkson, James Clawson, Kent Lyons, and Thad Starner. 2005. An Empirical Study of Typing Rates on mini-QWERTY Keyboards. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '05, pages 1288–1291, New York, NY, USA. ACM.

David Eppstein. 1998. Finding the K Shortest Paths. *SIAM Journal on computing*, 28(2):652–673.

Jr G. David Forney. 1973. The Viterbi Algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Michael L. Fredman and Robert Endre Tarjan. 1987. Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms. *Journal of the ACM (JACM)*, 34(3):596–615, July.

Dongxu Han and Baobao Chang. 2013. A Maximum Entropy Approach to Chinese Spelling Check. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 74–78, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable Modified Kneser-Ney Language Model Estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696, Sofia, Bulgaria, August.

Kenneth Heafield. 2011. KenLM: Faster and Smaller Language Model Queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom, July.

Changning Huang and Hai Zhao. 2007. Chinese Word Segmentation: A Decade Review. *Journal of Chinese Information Processing*, 21(3):8–20.

Zhongye Jia and Hai Zhao. 2013. KySS 1.0: a Framework for Automatic Evaluation of Chinese Input Method Engines. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1195–1201, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Zhongye Jia, Peilu Wang, and Hai Zhao. 2013. Graph Model for Chinese Spell Checking. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 88–92, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Wei Jiang, Yi Guan, Xiaolong Wang, and BingQuan Liu. 2007. PinYin-to-Character Conversion Model based on Support Vector Machines. *Journal of Chinese information processing*, 21(2):100–105.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June. Association for Computational Linguistics.

Vladimir I. Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. In *Soviet physics doklady*, volume 10, page 707.

Mu Li, Muhua Zhu, Yang Zhang, and Ming Zhou. 2006. Exploring Distributional Similarity Based Models for Query Spelling Correction. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 1025–1032, Sydney, Australia, July. Association for Computational Linguistics.

Lu Li, Xuan Wang, Xiao-Long Wang, and Yan-Bing Yu. 2009. A Conditional Random Fields Approach to Chinese Pinyin-to-Character Conversion. *Journal of Communication and Computer*, 6(4):25–31.

Xiaodong Liu, Kevin Cheng, Yanyan Luo, Kevin Duh, and Yuji Matsumoto. 2013. A Hybrid Chinese Spelling Correction Using Language Model and Statistical Machine Translation with Reranking. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 54–58, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

I. Scott MacKenzie and R. William Soukoreff. 2002. A Character-level Error Analysis Technique for Evaluating Text Entry Methods. In *Proceedings of the Second Nordic Conference on Human-computer Interaction*, NordiCHI '02, pages 243–246, New York, NY, USA. ACM.

Eric Mays, Fred J Damerau, and Robert L Mercer. 1991. Context Based Spelling Correction. *Information Processing & Management*, 27(5):517–522.

James L. Peterson. 1980. Computer Programs for Detecting and Correcting Spelling Errors. *Commun. ACM*, 23(12):676–687, December.

Andreas Stolcke. 2002. SRILM-An Extensible Language Modeling Toolkit. In *Proceedings of the international conference on spoken language processing*, volume 2, pages 901–904.

Andrew J. Viterbi. 1967. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *Information Theory, IEEE Transactions on*, 13(2):260–269.

Xuan Wang, Lu Li, Lin Yao, and Waqas Anwar. 2006. A Maximum Entropy Approach to Chinese Pin Yin-To-Character Conversion. In *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on*, volume 4, pages 2956–2959. IEEE.

Chun-Hung Wang, Jason S. Chang, and Jian-Cheng Wu. 2013a. Automatic Chinese Confusion Words Extraction Using Conditional Random Fields and the Web. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 64–68, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Peilu Wang, Ruihua Sun, Hai Zhao, and Kai Yu. 2013b. A New Word Language Model Evaluation Metric for Character Based Languages. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, pages 315–324. Springer.

Rui Wang, Masao Utiyama, Isao Goto, Eiichro Sumita, Hai Zhao, and Bao-Liang Lu. 2013c. Converting Continuous-Space Language Models into N-Gram Language Models for Statistical Machine Translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 845–850, Seattle, Washington, USA, October. Association for Computational Linguistics.

Yih-Ru Wang, Yuan-Fu Liao, Yeh-Kuang Wu, and Liang-Chun Chang. 2013d. Conditional Random Field-based Parser and Language Model for Traditional Chinese Spelling Checker. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 69–73, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Jacob O. Wobbrock and Brad A. Myers. 2006. Analyzing the Input Stream for Character- Level Errors in Unconstrained Text Entry Evaluations. *ACM Trans. Comput.-Hum. Interact.*, 13(4):458–489, December.

Jun Wu and Liren Chen. 2004. Fault-tolerant Romanized Input Method for Non-roman Characters, August 25. US Patent App. 10/928,131.

Jun Wu, Hulcan Zhu, and Hongjun Zhu. 2009. Systems and Methods for Translating Chinese Pinyin to Chinese Characters, January 13. US Patent 7,478,033.

Shaohua Yang, Hai Zhao, and Bao-liang Lu. 2012a. A Machine Translation Approach for Chinese Whole-Sentence Pinyin-to-Character Conversion. In *Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation*, pages 333–342, Bali,Indonesia, November. Faculty of Computer Science, Universitas Indonesia.

Shaohua Yang, Hai Zhao, Xiaolin Wang, and Bao-liang Lu. 2012b. Spell Checking for Chinese. In *International Conference on Language Resources and Evaluation*, pages 730–736, Istanbul, Turkey, May.

Liang-Chih Yu, Yuen-Hsien Tseng, Jingbo Zhu, and Fuji Ren, editors. 2013. *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*. Asian Federation of Natural Language Processing, Nagoya, Japan, October.

Jingyi Zhang and Hai Zhao. 2013. Improving Function Word Alignment with Frequency and Syntactic Information. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 2211–2217. AAAI Press.

Hai Zhao and Chunyu Kit. 2008. Exploiting Unlabeled Text with Different Unsupervised Segmentation Criteria for Chinese Word Segmentation. *Research in Computing Science*, 33:93–104.

Hai Zhao and Chunyu Kit. 2011. Integrating Unsupervised and Supervised Word Segmentation: The Role of Goodness Measures. *Information Sciences*, 181(1):163–183.

Hai Zhao, Chang-Ning Huang, and Mu Li. 2006. An Improved Chinese Word Segmentation System with Conditional Random Field. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 162–165, Sydney, Australia, July. Association for Computational Linguistics.

Hai Zhao, Chang-Ning Huang, Mu Li, and Bao-Liang Lu. 2010. A Unified Character-Based Tagging Framework for Chinese Word Segmentation. *ACM Transactions on Asian Language Information Processing (TALIP)*, 9(2):5.

Hai Zhao, Masao Utiyama, Eiichiro Sumita, and Bao-Liang Lu. 2013. An Empirical Study on Word Segmentation for Chinese Machine Translation. In *Computational Linguistics and Intelligent Text Processing*, pages 248–263. Springer.

Yabin Zheng, Chen Li, and Maosong Sun. 2011a. CHIME: An Efficient Error-tolerant Chinese Pinyin Input Method. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, pages 2551–2556. AAAI Press.

Yabin Zheng, Lixing Xie, Zhiyuan Liu, Maosong Sun, Yang Zhang, and Liyun Ru. 2011b. Why Press Backspace? Understanding User Input Behaviors in Chinese Pinyin Input Method. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Techologies*, pages 485–490, Portland, Oregon, USA, June. Association for Computational Linguistics.