Linear Reranking Model for Chinese Pinyin-to-Character Conversion

1Xinxin Li, 1Xuan Wang, 1Lin Yao and 1, 2Muhammad Waqas Anwar

1Computer Application Research Center, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China
2Department of Computer Science, COMSATS Institute of Information Technology, Abbottabad, Pakistan

Abstract: Pinyin-to-character conversion is an important task for Chinese natural language processing tasks. Previous work mainly focused on n-gram language models and machine learning approaches, or with additional hand-crafted or automatic rule-based post-processing. There are two problems unable to solve for word n-gram language model: out-of-vocabulary word recognition and long-distance grammatical constraints. In this study, we proposed a linear reranking model trying to solve these problems. Our model uses minimum error learning method to combine different sub models, which includes word and character n-gram LMs, part-of-speech tagging model and dependency model. Impact of different sub models on the conversion are fully experimented and analyzed. Results on the Lancaster Corpus of Mandarin Chinese show that our new model outperforms word n-gram language model.

Keywords: Dependency model, minimum error learning method, part-of-speech tagging, word n-gram model

INTRODUCTION

Research on pinyin-to-character conversion is beneficial for many Chinese language processing tasks, such as speech recognition, intelligent inputting methods. However, this task is still difficult due to the existence of homophone. For Chinese characters, there are about 418 different syllables without tones (more or less in different standards), 3500 common characters covering 99.48% of all characters and 7000 general characters. Therefore, there might be multiple possible characters for each syllable. For example, totally 60 characters in 3500 common characters have the same syllable “yi”. The task of Chinese pinyin-to-character conversion is to disambiguate the corresponding character for a syllable with other characters.

The common approach for Chinese pinyin-to-character conversion is n-gram Language Model (LM), which selects the word/character candidate sequence with greatest probability. The determination of word/character for current syllables depends on the probability on previous words/characters (detailed description in next section). There are two major problems for word n-gram language model. One problem is the mistaken recognition of Out-Of-Vocabulary (OOV) words. Since the OOV words are not in segmented training data, they might be given low probabilities in word n-gram LM. The other problem is that n-gram language model only embodies the constraints on nearby words, which lacks long distance constraints in grammatic or syntactic structure.

Many approaches are proposed to deal with these problems. Discriminative methods, such as maximum entropy model (Xiao et al., 2007), support vector machines (Jiang et al., 2007), can employ pinyin features after current word/character rather than only using information before current word/character in word n-gram LM. Rule-based post-processing approach brings grammatical rule constraints to candidates, where these rules can be obtained by hand-crafted or statistical methods. Wang employed rough set theory to extract rules automatically from training dataset (Wang et al., 2004).

In this study, we proposed a linear reranking model for Chinese pinyin-to-character conversion problem. Our model can utilize the information from word/character n-gram LMs, character-based discriminative model, pinyin-word occurrence model, Part-Of-Speech (POS) tagging model, word-POS co-occurrence model and dependency model. We can combine these sub models by using their probability outputs for candidates. Minimum error training method is used to obtain the weights for sub models. We perform experiments on Lancaster Corpus of Mandarin to analyze the effect of different sub models on pinyin-to-character conversion and the results show that the information from sub models helps the conversion.

PINYIN-TO-CHARACTER CONVERSION

Given a syllable sequence $S = s_1, s_2, ..., s_n$, the aim of Chinese pinyin-to-character conversion is to find
a sequence of characters \( C = c_1, c_2, \ldots, c_n \), or a sequence of words \( W = w_1, w_2, \ldots, w_n \), where each word \( w_k \) is composed of one or more characters \( c_i, \ldots, c_j \). An example of Chinese pinyin-to-character conversion is shown in Fig. 1.

Using word \( n \)-gram LM, word sequence \( W \) for inputting syllable sequence \( S \) satisfying:

\[
\begin{align*}
  w^* &= \arg \max \ P(W \mid S) \\
  &= \arg \max \ P(S \mid W) P(W) / P(S)
\end{align*}
\]

where,

\[
\begin{align*}
  P(S \mid W) &= \prod_{i=1}^{m} p(s_i \mid w_i) \\
  P(W) &= \prod_{i=1}^{m} p(w_i \mid w_{i-2}, w_{i-1})
\end{align*}
\]

For trigram language model,

\[
P(W) = p(w_1)p(w_2 \mid w_1)p(w_3 \mid w_2, w_1)\prod_{i=3}^{m} p(w_i \mid w_{i-2}, w_{i-1})
\]

Using character \( n \)-gram LM, the character sequence \( C \) can be determined by:

\[
\begin{align*}
  C^* &= \arg \max \ P(C \mid S) \\
  &= \arg \max \ P(S \mid C) P(C) / P(S)
\end{align*}
\]

where, \( P(S \mid C) \) and \( P(C) \) are calculated similar as \( P(S \mid C) \) and \( P(W) \). When applying word \( n \)-gram LM, a beam search decoding method is used for Chinese pinyin-to-character conversion. Normally, \( P(S \mid W) \) and \( P(S \mid C) \) are omitted in the decoding phase. For each position \( j \) in syllable sentence, the decoding algorithm will generate all words in vocabulary according to current syllable sequence \( s_1, \ldots, s_j \) ending on position \( j \) and calculate the probability \( p(w_i \mid w_{i-2}, w_{i-1}) \) of current word \( w_i \) over previous words.

For every position \( j \), it keeps the \( n \)-best word sequences. The best word sequence will be the one with greatest probability on sentence end.

**Preliminary experiments:** We perform experiments on The Lancaster Corpus of Mandarin Chinese (LCMC) to analysis the performance of word \( n \)-gram LM and then compare it with single character \( n \)-gram LM, oracle of mixed result with character \( n \)-gram LM and oracle of k-best word sequences generated by word \( n \)-gram LM.

LCMC contains 15 different text categories, with totally 45735 segmented, POS tagged, pinyin annotated sentences. The original pinyin text is annotated using pinyin4j tool which only considers the most probable syllable for each character separately. We re-annotated the pinyin of sentences using minimum word number matching algorithm on Sogou dictionary. The re-annotated pinyin sequence is more accurate than the original one.

To evaluate word \( n \)-gram LM, we selected 200 sentences from each category as development and test dataset separately and the rest 39735 sentences as training data. All the training, development and test sentences are first segmented by Chinese end punctuations and then filtered by eliminating those containing English words. The statistics of our training, development and test data are listed in Table 1.

The training data for our word and character \( n \)-gram LMs are taken from People's Diary of year 1998, 2006, 2007 and 2009-2012. To obtain the full list of Chinese words is impossible because the specification of word segmentation is not determined and the number of words is still grown. Even all words are given, it's impractical because an accurate word \( n \)-gram LM needs a huge training data and the size of the trained language model is too huge to use. Our word dictionary for word \( n \)-gram LM contains 130750 words selected from four sources: the 7000 general characters, the 56064 common words, words from XinHua dictionary and 94412 highest frequency words extracted from Google Chinese web 5g version 1. For character \( n \)-gram LM, the word dictionary contains only the 7000 general characters. We use a minimum word number method to segment the raw training text and SRILM tool to train our language model (Stolcke, 2002). The Out-Of-Vocabulary (OOV) words in Table 1 are defined as words that are not contained in word dictionary.

The performance of word and character \( n \)-gram LMs are evaluated on development dataset. The results are shown on Table 2. Both models are evaluated on Character Error Rate (CER) measure:

**Table 1:** Statistics of training, development and test data

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Train</th>
<th>Dev</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Sentences</td>
<td>1135296</td>
<td>8251</td>
<td>8637</td>
</tr>
<tr>
<td>#All words</td>
<td>875397</td>
<td>62961</td>
<td>63608</td>
</tr>
<tr>
<td>#Chn words</td>
<td>723424</td>
<td>52039</td>
<td>52263</td>
</tr>
<tr>
<td>#Chn characters</td>
<td>1144559</td>
<td>82527</td>
<td>82788</td>
</tr>
<tr>
<td>#OOV word</td>
<td>13692</td>
<td>1043</td>
<td>1245</td>
</tr>
<tr>
<td>OOV rate</td>
<td>1.56%</td>
<td>1.66%</td>
<td>1.96%</td>
</tr>
</tbody>
</table>

**Table 2:** Performance of different LMs on development data

<table>
<thead>
<tr>
<th>Models</th>
<th>CER</th>
<th>IVWER</th>
<th>OOWER</th>
</tr>
</thead>
<tbody>
<tr>
<td>character</td>
<td>12.92</td>
<td>11.25</td>
<td>15.04</td>
</tr>
<tr>
<td>word</td>
<td>11.27</td>
<td>9.03</td>
<td>14.23</td>
</tr>
<tr>
<td>mixed oracle</td>
<td>7.38</td>
<td>6.89</td>
<td>11.58</td>
</tr>
<tr>
<td>500-best oracle</td>
<td>2.01</td>
<td>4.01</td>
<td>5.94</td>
</tr>
</tbody>
</table>
Both models are also evaluated by IV word error rate, OOV word error rate. The OOV words are defined on table 1 are Chinese words not in word dictionary. The length of both IV words and OOV words is larger than 1. Table 2 shows that word n-gram LM achieves better performance than character n-gram LM in all evaluation measures, providing evidence that word n-gram LM provides more constraints than character n-gram LM. And word n-gram LM achieves better performance on IV words than OOV words.

Oracle of mixed result of word and character n-gram LMs and oracle of k-best word lists are calculated to compare with word n-gram LM. The k-best word lists are generated using beam search algorithm on word-based model. Table 2 shows that the word and character n-gram LMs cause different errors and oracle of their mixed result obtains better performance. The oracle of k-best word sequences achieves the best CER. The fact that the performance of word n-gram LM is much lower than both oracles leaves us a space to improve it by using other information.

LINER RERANKING MODEL

Experimental results show single word n-gram LM is insufficient for Chinese pinyin-to-character conversion. We proposed a linear reranking model to combine different information. In this study, several sub models are introduced into our model, including part-of-speech tagging model, pinyin-word co-occurrence model, word-POS occurrence model and dependency model. To utilize these different sub models, we can combine their probability outputs of each candidate and use minimum error training method to obtain the weight for these models. The k-best candidates for each pinyin sentence are first generated by word n-gram LM using beam search algorithm and then POS tagged and dependency parsed. Our linear reranking model is used to select the best one from these candidates.

Minimum Error Training (MERT) algorithm is proposed to combine different features for machine translation (Och, 2003). It has been successfully used in joint word segmentation and POS tagging (Jiang et al., 2008). The probability of a possible corresponding word sequence W for a syllable sequence S is calculated as:

\[
P_{\text{score}}(W | S) = P_{\text{score}}(W, C, T, D | S) \\
= \sum_{ij} w_i * P_{\text{sub}}(W, C, T, D | S) \\
= w_1 * P(W) + w_2 * P(C) + w_3 * P(S | C) \\
+ w_4 * P_{\text{score}}(W | S) + w_5 * P_{\text{score}}(S | W) \\
+ w_6 * P(T | W) + w_7 * P(T) \\
+ w_8 * P_{\text{score}}(W | T) \\
+ w_9 * P(D | W, T)
\]

where, \(\sum_{i=1}^{8} w_i = 1\) and \(P_{\text{sub}}(W, C, T, D | S)\) represents the probability output of sub models, including word n-gram LM \(P(W)\), character n-gram LM \(P(C)\), character-based discriminative model \(P(S | C)\), pinyin-word co-occurrence model \(P_{\text{score}}(W | S)\) and \(P_{\text{score}}(S | W)\), POS tagging model \(P(T | W)\), POS N-gram LM \(P(T)\), POS-word occurrence model \(P_{\text{score}}(W | T)\) and a dependency model \(P(D | W, T)\). The first line in the equation is valid for syllables s, there will be multiple possible corresponding words w. For a syllable sequence S and its corresponding word sequence W, we define pinyin-word co-occurrence as:

**Feature templates for character-based model**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Template</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character-based discriminative model</td>
<td>(\Phi(S, C)\times \theta)</td>
</tr>
<tr>
<td>Pinyin-word co-occurrence</td>
<td>(F(C) = \text{arg max}_{C \in \mathbb{D}(P)} P(S</td>
</tr>
</tbody>
</table>
Table 4: Feature template for POS model

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>w-2 t0 End (w-1) w0 start (w1) t0</td>
</tr>
<tr>
<td>2</td>
<td>w-1 t0 When len (w0) = 1</td>
</tr>
<tr>
<td>3</td>
<td>w0 t0 Start (w0) t0</td>
</tr>
<tr>
<td>4</td>
<td>w1 t0 End (w1) t0</td>
</tr>
<tr>
<td>5</td>
<td>w2 t0 cnt0, (n = 1, len (w0-2)</td>
</tr>
<tr>
<td>6</td>
<td>t-1 t0 Start (w0) cnt0, (n = above)</td>
</tr>
<tr>
<td>7</td>
<td>t-2 t-1 t0 End (w0) cnt0, (n = above)</td>
</tr>
<tr>
<td>8</td>
<td>t-1 w0 cnt0+1 t0, (cn = cn+1)</td>
</tr>
<tr>
<td>9</td>
<td>w0 t0 end (w1) Class (start (w0)) t0</td>
</tr>
<tr>
<td>10</td>
<td>w0 t0 start (w1) Class (end (w1)) t0</td>
</tr>
</tbody>
</table>

Fig. 2: An example of a dependency tree

\[
P_{\text{occ}}(W \mid S) = \prod_{i=1}^{m} p(w_i \mid s_i),
\]

\[
P_{\text{occ}}(S \mid W) = \prod_{i=1}^{m} p(s_i \mid w_i),
\]

\[
p(w_i \mid s_i) \text{ and } p(s_i \mid w_i) \text{ can be calculated using minimum likelihood estimation (MLE) method, where,
}\]

\[
p(w_i \mid s_i) = \frac{N(w_i, s_i)}{N(s_i)}, \quad p(s_i \mid w_i) = \frac{N(w_i, s_i)}{N(w_i)}
\]

We can statistic the number of syllables \(N(s_i)\) and syllable-word pairs \(N(w_i, s_i)\) from annotated dataset. The training dataset is the same as the dataset for word n-gram LM.

**POS tagging model:** Given the word sequence \(W\), we can determine its Part-Of-Speech (POS) tags. Previous work reveals that POS information improves word segmentation (Ng and Low, 2004; Zhang and Nivre, 2011). In this study, we introduce POS tagging model trying to help selecting the best word sequence. Averaged perceptron algorithm is used for parameter estimation.

The POS tag sequence \(T\) for word sequence \(W\) is chosen as:

\[
P(T \mid W) = \arg \max_{T \in \text{GEN}(W)} P(T \mid W)
\]

\[
= \arg \max_{T \in \text{GEN}(W)} \Phi(T, W) \times \alpha
\]

The features \(\Phi\) (T, W) for POS tagging model are defined similar as Zhang and Clark (2008), shown in Table 4.

The training data for POS tagging is taken from Chinese Treebank (CTB5) and the distribution of training, development and test dataset is same as Zhang and Clark (2008). The F-1 measure of our POS model on golden segmented development dataset is 95.26%.

**POS N-gram language model:** The POS n-gram LM is trained using the data described before and POS tagged. We define the probability of POS sequence \(T\) as:

\[
P(T) = \prod_{i=1}^{m} p(t_i \mid t_{i-1}, \ldots, t_{i-n})
\]

**POS-word co-occurrence:** The POS-word Co-occurrence is defined similar as syllable-word co-occurrence. For word sequence \(w\) and its corresponding POS sequence \(T\), the co-occurrence is defined as:

\[
P_{\text{occ}}(W \mid T) = \prod_{i=1}^{m} p(w_i \mid t_i),
\]

\[
P_{\text{occ}}(T \mid W) = \prod_{i=1}^{m} p(t_i \mid w_i)
\]

where, \(p(w_i \mid t_i)\) and \(p(t_i \mid w_i)\) are also calculated using MLE method.

**Dependency model:** For a long sentence \(W\), the determination of a word for syllables might not be determined only using the information from nearby words and syllables. It might also need grammatic and syntactic information. For example, in a syllable sequence "yi zhi mei li ke ai de xiao hua mao". The determination of word for syllables "yi zhi" is dependent on its relation with syllables "xiao hua mao". Dependency model brings long-distance word relation for Chinese pinyin-to-character conversion. For a segmented and POS tagged sentence \((W, T)\), we use a deterministic transition-based algorithm for dependency parsing (Zhang and Nivre, 2011). Figure 2 gives an example of a dependency tree structure.

The probability of dependency tree \(D\) is:

\[
P(D \mid W, T) = \prod_{i=1}^{m} w_i \times f_i(D \mid W, T)
\]

**EXPERIMENTS AND ANALYSIS**

The experimental dataset for linear reranking model is the same as preliminary experiments on word n-gram LM in section 2. Same as calculating oracle of k best word sequences, we take 500 candidate dependency trees for each syllable sequence. Then linear reranking model is used to select the best one from these candidate ones. Character Error Rate (CER) is used as our evaluation measure.

**Experiments on sub models:** All our sub models can generate a probability for a give dependency tree \(D\), as we listed in section (Linear Reranking Model). We train these sub models on training dataset and evaluate the linear reranking model on development dataset.
Table 5 shows that all the sub models have positive impact for the task, in which word n-gram LM improves the performance most, character N-gram LM ranks second and pinyin-word occurrence ranks third. The dependency model also brings an improvement on development data, better than POS-word occurrence model.

We then evaluate the impact of each sub model on the task. The experimental results are shown in Fig. 3. The numbers 0, ..., 8 in X axis denote sub models listed in Table 5. The above curve in Fig. 3 describes the results of only one sub model used for evaluation. The results reveal similar phenomenon as Table 5 that word n-gram LM achieves the lowest CER and character n-gram LM ranked the second. The dependency model gives the lowest accuracy.

The below curve shows the results of linear reranking models of word n-gram LM combining with one other sub models. The linear reranking models outperform all single sub models. Given word n-gram LM, adding pinyin-word occurrence model achieves the best performance where it's omitted in the original decoding algorithm described in (Pinyin-to-Character Conversion).

We then evaluate different groups of sub models to validate which information has the most influence on the conversion. All sub models are split into four groups: character group, pinyin group, POS group, dependency group. The results in Table 6 show that except word n-gram model, the pinyin group provides the most useful information and the dependency model gives the least benefit.

Figure 4 shows experimental results of the linear reranking models beginning from word n-gram LM and adding one sub model group each time. The entire linear reranking model benefits from every sub model group. The final model achieves 1.51 points decrease than word N-gram LM, about 14.49% increase on performance.

**Experiments on test data:** We then evaluate our models on test data. The experimental results exhibit similar performance as development data. With more sub models added, the performance of linear reranking model increases. The final model achieves 10.94 CER, about 8.89% decrease than 12.03 CER of word n-gram LM.

**Complexity analysis:** The space and time complexity of the linear reranking model is vital for Chinese pinyin-to-character conversion. All sub models can be generated along with character appending in sentence with linear time complexity of sentence length \( O(n) \). Then our linear reranking model is also linear with sentence length since it's a linear combination of these sub models.

**Experiments on test data:** We then evaluate our models on test data. The experimental results exhibit similar performance as development data. With more sub models added, the performance of linear reranking model increases. The final model achieves 10.94 CER, about 8.89% decrease than 12.03 CER of word n-gram LM.
CONCLUSION

In this study, we employ a linear reranking model to solve Chinese pinyin-character conversion problem. Our reranking model can improve the performance of word n-gram LM by utilizing information from different sub models. There are two directions we can continue to work on. One is extending our model to speech recognition. The second direction is to reduce the size of sub models while keeping its accuracy.

REFERENCES


