Addressing the Out-Of-Vocabulary Problem for Large-Scale Chinese Spoken Term Detection

Sha Meng\textsuperscript{1,2}, Jian Shao\textsuperscript{3,2}, Roger Peng Yu\textsuperscript{2}, Jia Liu\textsuperscript{1}, and Frank Seide\textsuperscript{2}

\textsuperscript{1}Department of Electronic Engineering, Tsinghua University  
\textsuperscript{2}Microsoft Research Asia  
\textsuperscript{3}ThinkIT Speech Lab, Institute of Acoustics, Chinese Academy of Sciences

\begin{abstract}
While the Out-Of-Vocabulary (OOV) problem remains a challenge for English spoken term detection tasks, it is underestimated for Chinese. This is because an English OOV query can still be matched as a sequence of Chinese characters, with each character itself being a word in the vocabulary. However, our experiments show that search accuracy levels differ significantly when a query is or is not in the vocabulary. In-Vocabulary (INV) queries outperform OOV queries for more than 20%. We examine this problem with a word-lattice-based spoken term detection task. We propose a two-stage method by first locating candidates by partial phonetic matching and then refining the matching score with word lattice rescoring. Experiments show that the proposed method achieves a 24.1% relative improvement for OOV queries on a large-scale Chinese spoken term detection task.

\textbf{Index Terms}: Spoken Term Detection, OOV, Word Lattice, Large-Vocabulary Continuous Speech Recognition
\end{abstract}

1. Introduction

Improving accessibility for the overwhelming amount of speech data available today necessitates the development of robust Spoken Term Detection (STD, also known as keyword spotting) and Spoken Document Retrieval (SDR) techniques. A straightforward approach is to convert spoken content to text by Large-Vocabulary Continuous Speech Recognition (LVCSR) systems and apply text search technologies. However, with most STD scenarios, speech recognition accuracy levels are still not satisfying. Previous research [1, 2, 3, 4] has shown that indexing word lattices instead of only Speech-To-Text transcripts can effectively deal with the high Word Error Rate (WER) in speech recognition. [4] has shown that a word-lattice-based system can be effectively used to search a 600-hour corpus.

A disadvantage of LVCSR-based systems is vocabulary dependency, which is widely known as the Out-Of-Vocabulary (OOV) problem. For automatic speech transcription tasks, the OOV problem is usually acceptable as the OOV rates are typically only around 1-2%, while it becomes more serious in an STD task, since those OOV words are more likely to be used as user queries. A study in [5] shows that more than 12% of the queries can be OOV words, while the transcripts OOV rate is less than 1.5%.

To solve this problem, some research [6, 7, 8, 9] took sub-word based approaches. On small-scale tasks (e.g., speech databases below 10 hours), where recall is a higher priority than precision, these approaches have achieved promising performance. However, with large-scale speech database (e.g., 100 hours or more), where precision is more important for users, the huge number of false alarms generated by the sub-word based systems becomes unacceptable. The OOV problem remains a challenge for large-scale English STD tasks.

For Chinese, the OOV problem is seriously underestimated or even ignored even for word-based approaches. This is because for Chinese, even a query is not directly in the recognition vocabulary, it can still be matched as a sequence of Chinese characters, with each character itself being a word in the vocabulary\textsuperscript{1}. However, our experiments will show that, even for the same set of queries, the search accuracy differs significantly (75.8% versus 59.1%) when the queries are in the vocabulary versus out of vocabulary. We identify this performance gap as the OOV problem for Chinese STD tasks. The problem is especially important when dealing with proper names for people, organization and other proper nouns. While the proper names are very likely to be used as queries, it is difficult to include all of them in the vocabulary when creating an index.

In this paper, we examine the OOV problem for a Chinese STD task within a 100-hour database. The word-lattice-based baseline system achieves an accuracy of 66.5% with In-Vocabulary (INV) queries, while only 43.3% with OOV queries. We propose a two-stage approach for searching OOV queries. We first find a candidate result list by partial phonetic matching, then refine the matching score with word-lattice rescoring. Experiments show that with the proposed method, OOV performance is improved to 53.7%.

The paper is structured as follows. Section 2 introduces the lattice-based spoken term detection algorithm. Section 3 investigates the OOV problem with some preliminary experiments. Section 4 proposes a two-stage approach for improving OOV performance. Section 5 shows our experimental results and section 6 concludes.

2. Lattice-based Spoken Term Detection

We first recapitulate the method of lattice-based spoken term detection in [10]. A lattice $L = (\mathcal{N}, A, n_{\text{start}}, n_{\text{end}})$ is a directed acyclic graph (DAG) with $\mathcal{N}$ being the set of nodes, $A$ being the set of arcs, and $n_{\text{start}}, n_{\text{end}} \in \mathcal{N}$ being the unique

\begin{footnotesize}
\footnotesize\textsuperscript{1}Chinese words are graphemically made up of characters. The set of most common 6,000 Chinese characters has a sufficient coverage for most user scenarios.
\end{footnotesize}
initial and unique final node, respectively.

Each node $n \in \mathcal{N}$ has an associated time $t(n)$ and possibly an acoustic or Language-Model (LM) context condition. Arcs are 4-tuples $a = (S[a], E[a], I[a], w[a])$. $S[a], E[a], I[a] \in \mathcal{N}$ denote the start/end node of the arc. $I[a]$ is the word (or sub-word) identity. $w[a]$ shall be a weight assigned to the arc by the recognizer. Specifically, $w[a] = p_{la}(a) / |\lambda| \cdot P_{la}(a)$ with acoustic likelihood $p_{la}(a)$, LM probability $P_{la}$, and LM weight $\lambda$.

We define a path $\pi = (a_1, \ldots, a_K)$ as a sequence of connected arcs, and use the symbols $S, E, I$ and $w$ for paths as well to represent the respective properties for entire paths, i.e., the path start node $S[\pi] = S[a_1]$, end node $E[\pi] = E[a_K]$, label sequence $I[\pi] = (I[a_1], \ldots, I[a_K])$, and total path weight $w[\pi] = \prod_{k=1}^{K} w[a_k]$.

For a query $Q$, the posterior probability at time $(t_s, t_e)$ is calculated as

$$P(x, t_s, t_e, w, \pi | O) = \sum_{\pi : E[\pi] = t_e, \sum_{t=1}^{t_s} I[\pi] = Q} \alpha_{t_{\text{end}}} w[\pi] \cdot \beta_{t_{\text{end}}} \cdot \alpha_{t} (S[\pi]),$$

where $\alpha_n$, $\beta_n$ are the forward/backward probabilities [11]:

$$\alpha_n = \sum_{\pi : S[\pi] = n, E[\pi] = \pi} w[\pi],$$

$$\beta_n = \sum_{\pi : S[\pi] = n, E[\pi] = \pi} w[\pi].$$

It was proved that in a word-spotting task, ranking by the phrase posterior probability is theoretically optimal [12].

3. The OOV Problem for Chinese Spoken Term Detection

A fundamental difference between Chinese language and English language is that Chinese has no explicit word boundaries. For an STD task, a word breaker is required to segment a query into in-vocabulary words, and then matched against word lattices. For example, "语音文档检索" (spoken document retrieval) is segmented into three words "语音-文档检索" and is matched as a three-word phrase in lattices. To simplify the notation, we define query class $Q_k$ as queries with $x$ characters and segmented into $y$ in-vocabulary words. In the above example, "语音-文档检索" is a $Q_{0,3}$ query. With this definition, INV queries belong to $Q_{0,1}$, while typical OOV queries belong to $Q_{x,x}$. OOV queries can never be found in word lattices for English, while Chinese OOV queries can still be matched as sequences of characters (or shorter words). As a preliminary study, we want to investigate how different search accuracy will be if a query is INV versus OOV.

Table 1: STD experiment when changing queries from INV to OOV words. Results in Figure Of Merit (FOM). Recall (REC) listed for reference. The same set of 336 queries occurring 971 times actually in references are used.

<table>
<thead>
<tr>
<th>setup</th>
<th>FOM</th>
<th>REC</th>
</tr>
</thead>
<tbody>
<tr>
<td>queries in vocabulary</td>
<td>75.8</td>
<td>77.8</td>
</tr>
<tr>
<td>removing queries from vocabulary</td>
<td>59.8</td>
<td>61.2</td>
</tr>
<tr>
<td>relative degradation</td>
<td>-21.2%</td>
<td>-21.3%</td>
</tr>
</tbody>
</table>

Table 1 shows the experiment of turning INV queries to OOV words. Experiment setup will be detailed in section 5. A set of 336 $Q_{4,4}$ INV queries is selected. The first line shows the baseline STD performance. In the second line, all queries and their sub-words are removed from the vocabulary so that they become $Q_{4,4}$ OOV words. The language model is retrained on the same training set (but with different word segmentation for those query words) and word lattices are re-generated. As a small set of words is manipulated in the language model, the impact on the overall recognition accuracy is negligible (WER difference less than 0.3%). It is fair to assume that the impact of manipulating other queries to a specific query can be ignored. As shown in the table, there is a 21.2% accuracy degradation by turning INV queries to OOV.

Table 2: STD experiment when changing queries from OOV to INV words. Results in Figure Of Merit (FOM). Recall (REC) listed for reference. The same set of 62 queries occurring 185 times actually in references are used.

<table>
<thead>
<tr>
<th>setup</th>
<th>FOM</th>
<th>REC</th>
</tr>
</thead>
<tbody>
<tr>
<td>queries not in vocabulary</td>
<td>44.8</td>
<td>50.3</td>
</tr>
<tr>
<td>adding queries to vocabulary</td>
<td>58.2</td>
<td>67.0</td>
</tr>
<tr>
<td>relative improvement</td>
<td>+29.8%</td>
<td>+33.3%</td>
</tr>
</tbody>
</table>

Table 2 gives results of another experiment of turning OOV queries to INV words. A set of 62 $Q_{0,1}$ OOV queries are selected. The first line shows the baseline performance. In the second line, the queries are added into the recognition vocabulary and the language model with a constant unigram score (calculated as the average unigram of all words in the vocabulary). Word lattices are then re-generated with the new vocabulary. A 29.8% improvement is observed by adding the queries into the vocabulary, even without a well-trained language model score.

Both experiments show that OOV queries are a performance bottleneck for an STD task. In the next section, we will present a method that address this problem.

4. A Two-stage Approach for OOV Term Detection

Table 1 and 2 show that poor performance of OOV queries comes from low recall. Based on this observation, we proposed...
4.1. Partial Phonetic Matching for High Recall

For word-lattice-based STD, a query is matched against lattices by word identities. To achieve a higher recall, we first loose the matching criterion by phonetic matching. This is achieved by converting word lattices to phonetic lattices, whose base units are toneless half syllables. The detailed conversion algorithm can be found in [10]. Fig. 2(b) shows an example of converted lattice with (a) being the raw lattice. To match a query against the converted lattices, the query is first converted to toneless half syllables.

It indicates that the phonetic matching does not give satisfying recall. We further loose the matching criterion by partial matching. For a query \( Q = (q_1, q_2, \ldots, q_K) \) and a path \( \pi = (a_1, a_2, \ldots, a_K) \) from lattices, where \( q_i \) and \( I[a_i] \) are toneless-half-syllables. The partial matching determines a match if

\[
\sum_{i=1}^{K} b(q_i, I[a_i]) \geq K \times \alpha,
\]

where \( \alpha = 0.7 \) in our experiments.

The path “L-AO-N-A-GER-ER-D-UO” in figure 2(b) is a partial match for the query “\( \text{罗纳尔多} \) (Ronaldo)”, which is pronounced as “L-UO-N-A-GER-ER-D-UO”.

4.2. Lattice Rescoring for High Precision

When partial phonetic matching increases the recall, they result in lower precision. Next step is to restore a reasonable precision.

Table 3: STD baseline for INV and OOV queries. Results reported in FOM, REC listed for reference.

<table>
<thead>
<tr>
<th>query set</th>
<th>#queries</th>
<th>#occurs</th>
<th>FOM</th>
<th>REC</th>
</tr>
</thead>
<tbody>
<tr>
<td>INV queries</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( Q_{3.3} )</td>
<td>768</td>
<td>3038</td>
<td>63.6</td>
<td>70.7</td>
</tr>
<tr>
<td>( Q_{4.1} )</td>
<td>314</td>
<td>913</td>
<td>75.9</td>
<td>77.9</td>
</tr>
<tr>
<td>( Q_{5.3} )</td>
<td>22</td>
<td>68</td>
<td>75.2</td>
<td>75.9</td>
</tr>
<tr>
<td>overall</td>
<td>1104</td>
<td>4409</td>
<td>66.5</td>
<td>75.1</td>
</tr>
<tr>
<td>OOV queries</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( Q_{3.3} )</td>
<td>233</td>
<td>738</td>
<td>36.9</td>
<td>42.7</td>
</tr>
<tr>
<td>( Q_{4.3} )</td>
<td>193</td>
<td>411</td>
<td>53.8</td>
<td>57.7</td>
</tr>
<tr>
<td>( Q_{4.4} )</td>
<td>54</td>
<td>155</td>
<td>44.7</td>
<td>50.3</td>
</tr>
<tr>
<td>( Q_{5.4} )</td>
<td>24</td>
<td>74</td>
<td>44.9</td>
<td>46.0</td>
</tr>
<tr>
<td>( Q_{5.5} )</td>
<td>8</td>
<td>30</td>
<td>45.3</td>
<td>50.0</td>
</tr>
<tr>
<td>overall</td>
<td>312</td>
<td>1408</td>
<td>43.3</td>
<td>48.2</td>
</tr>
</tbody>
</table>

Table 2 has shown that lattices generated by re-running speech recognition with OOV queries in the vocabulary provides satisfying precision even without a well-trained language model. It is certainly not realistic to re-run speech recognition for each OOV query. However, if we could “simulate” a lattice with OOV arcs as if it is generated by re-running the speech recognition with OOV word in vocabulary, we expect to achieve similar precision. To achieve that, we “insert” the candidates into the raw lattices as new arcs and recalculate the posterior probabilities of those candidates in the “simulated” lattices. The hope is that, if the arc is generated by re-running speech recognition, it will get reasonable posterior probability. If the arc will not be generated by re-running speech recognition which means it get pruned in the decoding process, its posterior probability should be low.

Let \( (t_s, t_e, Q) \) be a result candidate, with \( t_s \) and \( t_e \) being the start/end time, and \( Q \) being the OOV query. Let \( L \) be the raw word lattice. The lattice rescoring algorithm is as below.

- find the closest node \( n_i \in L \) to \( t_s \);
- if \( |t[n_i] - t_s| > 150 \text{ms} \), remove the candidate;
- define start node set \( N_s = \{ n \in L, t[n] \in (t[n_s] - 20 \text{ms}, t[n_s] + 20 \text{ms}) \} \);
- \( \forall n \in N_s \), remove \( n \) from \( N_s \), if \( \exists n' \in N_s, t[n] = t[n'] \) and \( \alpha_n < \alpha_{n'} \) (a heuristic to avoid over-counting the arc with different cross-word triphone contexts);
- get end node set \( N_e \) in a similar way as \( N_s \);
- \( \forall n \in N_s, m \in N_e \),
  - calculate the acoustic score \( p_{ac} \) of \( Q \) against time interval \( (t[n_i], I[a_i]) \) with the acoustic model;
  - calculate the language score \( P_{LM} \) of \( Q \), assuming \( Q \) has a constant unigram as used in table 2 (may need to incorporate the backoff of left context word);
  - insert a new arc \( a \) to \( L \) with \( S[a] = n, E[a] = m, I[a] = Q, u[a] = p_{ac}^{1/3} \cdot P_{LM} \);
- recalculate the candidate posterior probability as in Eq. (1).

Figure 2(c) shows a “simulated” lattice from the raw lattice in (a). A new arc with the OOV word “罗纳尔多” is added.

5. Experimental Results

5.1. Setup

We evaluate our method on a 104-hour long Chinese spontaneous speech corpus. The phone set contains 187 phones, with
28 “initials” (the consonant), 157 tonal “finals” (the vowel), and two silence phones. An acoustic model trained on 450-hour reading-style speech plus 50-hour spontaneous speech. 39-dimension MFCC is used. A dictionary with 68,933 words is used for both the LVCSR recognizer and for the word breaker. Word trigrams are trained from a text corpus that contains about 2.1 billion characters. Baseline character error rate is 42.2%.

The query set is generated by filtering a query log from Live Search Chinese [13] against our test database. Only queries occurring less than 20 times are selected. Examples queries are “鸟-罗曼多” (music), “北京烤鸭” (Beijing Duck), “埃菲尔铁塔” (Eiffel Tower) and “红楼梦” (The Story of the Stone - the name of a novel). The INV query set includes all $Q_{1,3}/Q_{4,3}/Q_{5,3}$ queries, while the OOV query set includes all $Q_{3,3}/Q_{4,3}/Q_{5,3}$ queries. We intentionally exclude 2-character queries $(Q_{2,1}/Q_{2,2})$ from the experiments since those queries have a low precision even with INV words, which is a different problem from the one we are addressing in this paper. The INV set contains 1,104 words while the OOV set contains 512 words.

STD results are evaluated in Figure of Merit (FOM), which is defined by National Institute of Science and Technology (NIST) as the detection/false-alarm curve averaged over $[0..10]$ false alarms per keyword per hour. Instead of the original $h=1$, we use $h = \text{data-set-duration}$.

5.2. STD Baseline for INV and OOV Queries

Table 3 shows the baseline performance for INV and OOV queries. Queries of different classes are reported separately. On average, the INV queries achieve an FOM of 66.5% and recall at 75.1%, while as expected, the OOV queries perform significantly worse, with an FOM of 43.3% and a recall of 48.2%.

5.3. Improved Method for OOV queries

Table 4 shows the experiments of the proposed method for OOV queries. The first line lists our baseline results. The second line lists the effect of our proposed approach step by step. By phonetic match, the overall recall is improved from 48.2% to 57.5%, with the FOM improved from 43.3% to 48.5%. The fact that FOM improves less than recall indicates that the precision goes down in this step. In the next line, by partial matching the recall is further improved to 72.4% while the FOM degrades to 34.2% due to a further precision decrease. In the last line, with lattice rescoring, FOM is improved to 53.7% as the precision goes up.

An overall 24.1% relative improvement is achieved with the proposed method. This is very close to the oracle setup of turning OOV to INV.

6. Conclusions

Although OOV words are still searchable in a word-lattice-based Chinese STD system, the performance was observed to be poor. On the other hand, those words are important as they are more likely to be used as queries. We proposed a method to improve the OOV query performance. We first generated a candidate result list by partial phonetic matching, and then refined the candidates by lattice rescoring. Experiments showed a 24.1% relative improvement on OOV queries with the proposed method.

The work is still an initial study. Further work needs to be done on a scalable index and on reducing time cost of the acoustic score calculation in lattice rescoring.

7. References