USING PROXIES FOR OOV KEYWORDS IN THE KEYWORD SEARCH TASK

Guoguo Chen, Oguz Yilmaz, Jan Trmal, Daniel Povey, Sanjeev Khudanpur

Center for Language and Speech Processing & Human Language Technology Center of Excellence
The Johns Hopkins University, Baltimore, MD 21218, USA
radical@clsp.jhu.edu

ABSTRACT

We propose a simple but effective weighted finite state transducer (WFST) based framework for handling out-of-vocabulary (OOV) keywords in a speech search task. State-of-the-art large vocabulary continuous speech recognition (LVCSR) and keyword search (KWS) systems are developed for conversational telephone speech in *Tagalog*. Word-based and phone-based indexes are created from word lattices, the latter by using the LVCSR system's pronunciation lexicon. Pronunciations of OOV keywords are hypothesized via a standard grapheme-to-phoneme method. In-vocabulary *proxies* (word or phone sequences) are generated for each OOV keyword using WFST techniques that permit incorporation of a phone confusion matrix. Empirical results when searching for the Babel/NIST evaluation keywords in the Babel 10 hour development-test speech collection show that (i) searching for word proxies in the word index significantly outperforms searching for phonetic representations of OOV words in a phone index, and (ii) while phone confusion information yields minor improvement when searching a phone index, it yields up to 40% improvement in actual term weighted value when searching a word index with word proxies.

Index Terms— Speech Recognition, Keyword Search, OOV Keywords, Proxy Keywords, Low Resource LVCSR.

1. SEARCHING FOR OOV WORDS IN SPEECH

Keyword search (KWS) for spoken documents has become more and more important nowadays as large speech repositories, such as oral history archives [1, 2] and online lectures [3, 4] are easily accessible. However, searching for keywords in spoken documents remains a changeling problem. Manual transcription of speech is usually prohibitively expensive, and given the amount of the spoken material available online, it is impractical to manually transcribe any nontrivial portion for search. Therefore, automatic KWS is highly desirable. State-of-the-art KWS systems usually rely on the large vocabulary continuous speech recognition (LVCSR) systems [5, 6]. In such systems, lattices of speech segments in the search collection are generated. An inverted index (postings list) is then created from the lattices. KWS may then be performed by searching for a given keyword via the inverted index. The KWS task is ideally "open vocabulary." However, LVCSR systems typically have a fixed vocabulary [7], making it impossible to search for out-of-vocabulary (OOV) words. One therefore uses as large an LVCSR vocabulary as feasible.

We are interested in KWS in low resource settings, where only limited resources are available to develop the LVCSR system, e.g. 10 hours of transcribed speech corresponding to about 100K words of transcribed text, and a pronunciation lexicon that covers only the words in the training text. Due to the low coverage of the pronunciation lexicon, the rate of encountering OOV keyword could be as large as 50%, leaving a lot of room for improvement over word-based KWS methods.

One way to minimize the OOV problem is to *preemptively* expand the the LVCSR lexicon. In other words, one adds automatically generated pronunciations of a large number of words in the LVCSR lexicon before lattice generation and indexation. In [6] it is shown that if one can anticipate the OOV keywords ahead of time, such a method leads to remarkable improvement in KWS performance. However, advance knowledge of all possible keywords is rarely the typical operating condition for KWS systems.

Another way to handle OOV keywords is via sub-word units such as phones, syllables or word-fragments. A sub-word index is created by either generating a sub-word lattice [8, 9, 10], or by converting a word lattice into a sub-word lattice with or without the use of an appropriate phone confusion matrix [11]. OOV keywords are represented as sequences of sub-word units, and matched against the sub-word index.

The idea of *query expansion* in text retrieval has also been adopted to tackle the OOV problem in speech search. A confusion matrix is used in [12, 13, 7, 14] to generate alternative words or syllables for OOV keywords, and a word or syllable index is searched for *them* instead of the original keywords. Unlike the idea of query expansion in text retrieval, wherein

The authors were supported by DARPA BOLT contract No HR0011-12-C-0015, and IARPA BABEL contract No W911NF-12-C-0015. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, IARPA, DoD/ARL or the U.S. Government.

one augments a possibly unseen keyword with other keywords that are semantically similar (e.g. synonyms), speech search entails other keywords that are acoustically similar. We will therefore call them *proxy keywords* in this paper.

Our work is most similar to that of [7], where proxy keywords are created using a phone confusion matrix. However, instead of searching for the proxy keywords in an n-best list generated by the LVCSR system, we introduce a weighted finite state transducer (WFST) based framework for directly matching multiple proxies against entire LVCSR lattices. We further demonstrate that using word proxies with the word index usually outperforms searching for OOV words in a converted phone index.

The rest of the paper is organized as follows. We describe our WFST based framework for proxy keywords generation in Section 2, and our KWS pipeline in Section 3. The experimental setup, results and some discussion follow in Section 4. Section 5 closes with some conclusions.

2. PROXY KEYWORD GENERATION

Let K represent a finite-state acceptor for an OOV keyword, and L_2 a finite state transducer for the pronunciation of the OOV keyword; e.g. pronunciations hypothesized via the joint-sequence model implemented in *S*equitur software [15]. Let E be an edit-distance transducer that maps a phone sequence to any other phone sequence with costs estimated from a phone confusion matrix. Let L_1 denote the pronunciation lexicon of the LVCSR system. Our WFST procedure for generating a proxy keyword K' may be described as,

$$K' = \mathrm{Project}\big(\mathrm{ShortestPath}\big(K \circ L_2 \circ E \circ (L_1^*)^{-1}\big)\big)\,. \quad (1)$$

This framework is similar to the method proposed in [6], where Levenshtein distance was used as the cost in E. Other phone confusion costs can easily be encoded in E, as we will do in this paper. The WFST based framework also makes it possible to convey the confusion cost to the final stage in which retrieved keywords are sorted for presentation to a user. Finally, the WFST framework supports both phone and word proxies, as will be explain below.

2.1. Sequitur Pronunciation Generation (L_2)

Pronunciations for OOV keywords are automatically obtained with the use of the Sequitur G2P software [15]. This method is based on automatically aligning graphemes and phonemes in a set of training examples of words+pronunciations to create "graphones," and building a joint multigram model of graphone sequences. Our model is trained on the baseline Babel low resource Tagalog lexicon containing 5.5K word pronunciation pairs. Pronunciations of an OOV keyword are "read off" the most likely graphone sequences corresponding to the grapheme sequence of the keyword.

2.2. Phone Confusion Matrix Estimation (E)

The phone confusion probabilities needed for E are obtained through standard maximum likelihood estimation. Training data for these conditional probabilities are collected by aligning the reference phone string for some held-out speech to the phone string corresponding its ASR hypothesis. The alignment maximizes the phone matching rate. A small subset of the development-test speech is utilized for this. Deletions and insertions are treated separately from phone substitutions, so that high rates of deletions and insertions will not adversely affect the estimation of E.

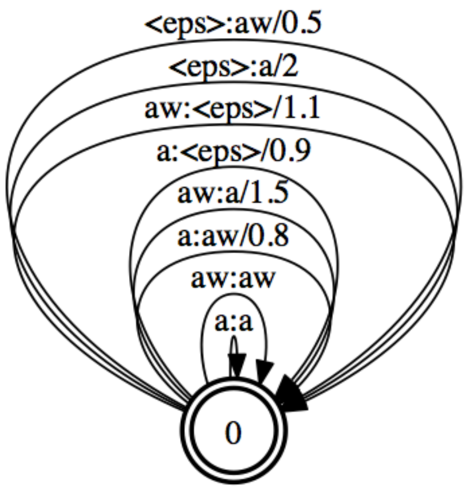

Fig. 1. Example of a phone confusion encoding transducer E.

These phone confusion statistics are encoded in the edit-distance transducer, as illustrated in Figure 1.

2.3. Using Proxies of OOV Keywords with a Phone Index

Since languages usually have a closed phone set, a KWS system based on a phone index may be considered open vocabulary. Furthermore, as claimed in [14], adding phone confusion to either the index or the phoneme representation $K \circ L_2$ of the (OOV) keyword can help improve KWS performance. To generate such phone proxies in our framework requires only a minor modification of (1) as,

$$K'' = \mathrm{Project}\left(\mathrm{ShortestPath}\left(K \circ L_2 \circ E\right)\right). \quad (2)$$

A phone index, however, represents a much larger search space by removing lexical constraints. Therefore phone-based KWS systems often suffer higher false alarm rates.

2.4. Improving the Word Proxy Generation of [6]

Instead of searching for phone proxies K'' of an OOV keyword K in a phone index, we can generate word proxies K' and search directly against a word index. The obvious advantage is that we do not have to keep a separate index for

OOV keywords — they share the index with the in-vocabulary (IV) keywords. Another potential advantage is that by imposing the lexical constraints on the permissible phone sequence via L_1^*, the search space is greatly limited, which may reduce false alarms.

However, there is a disadvantage of using (1) to generate word proxies, as illustrated by the following made-up English example. Suppose `balloon` $\equiv$ `/B AH L UW N/` is an OOV word, but `some` $\equiv$ `/S AX M/`, `Samba` $\equiv$ `/S AA M B AH/` and `loon` $\equiv$ `/L UW N/` are IV words. If the decoder encounters the sequence `some balloon` in the speech, it may hypothesize the word sequence `Samba loon` in that location. Now, generating $K' =$ `Samba loon` from $K =$ `baloon` by (1) requires E to insert 3 phones `/S AA M/`, while generating $K' =$ `loon` requires E to delete 2 phones `/B AH/`. So both appear to be poor proxies, even though a perfect phone match for K exists in the lattice!

To address the problem described in the previous example, we modify the edit-distance transducer of Figure 1 to the transducer E' shown in Figure 2.

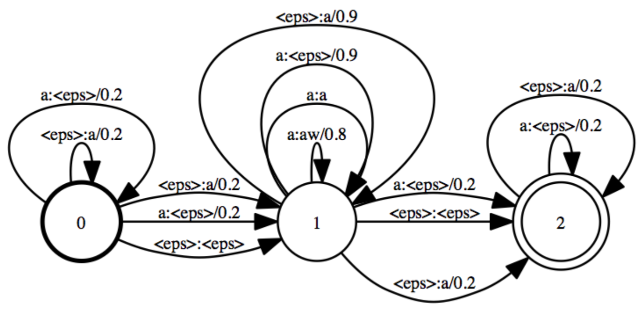

Fig. 2. Modified phone confusion encoding transducer E', with freer edits permitted at the keyword-boundary.

In this modified transducer, insertions and deletions at the boundaries of the keyword K are allowed at a lower cost, making search-by-word-proxy closer to phonetic search, while still retaining lexical constraint on the phone sequence of K'. Word proxies are thus generated as,

$$K' = \text{Project}\big(\text{ShortestPath}\big(K \circ L_2 \circ E' \circ (L_1^*)^{-1}\big)\big). \quad (3)$$

Note that the *ShortestPath* algorithm can be computationally expensive on large WFSTs. We therefore resort to pruning in the *ShortestPath* algorithm as needed.

2.5. On the Language Model Score of the Proxies

The proxy generation process (3) takes acoustic confusion into account, so that occurrences of K' in the index are good candidates for actual occurrences of K. But the word lattices from which the index was created contain both acoustic and language model scores. The language model score for K' is arguably not appropriate for comparing/sorting these occurrences. We will evaluate the impact of retaining/discarding this score in Section 4.

3. KEYWORD SEARCH PIPELINE

Our KWS pipeline is comprised of two major parts: lattice generation and lattice indexation. We build our pipeline using the open source toolkit Kaldi [16], which includes all the necessary tools for building and using an LVCSR system, as well as OpenFST [17] based indexation tools created for this work. All the code and scripts needed to reproduce the experiments in this paper have been checked into the Kaldi repository on SourceForge.

3.1. The Kaldi-based LVCSR System for Tagalog

We use the IARPA Babel Tagalog language collection release `babel106b-v0.2g-sub-train`, which contains 10 hours of transcribed conversational telephone speech and a 5.5K pronunciation lexicon that covers the training transcripts, to build our LVCSR system. This setting simulates typical low resource conditions. In addition to the 10 hours of training data, the Babel Tagalog collection also provides a 10 hour dataset of conversational telephone speech for development-testing. We use these 10 hours of speech data as our search collection. A small (1.5 hour) subset of it is also used for tuning a handful of parameters, such as the language model scale factor.

Standard PLP analysis is employed to extract 13 dimensional acoustic feature, and a maximum likelihood acoustic training recipe is followed to train speaker adaptive models. This is followed by the training of a universal background model from speaker-transformed data, which is then used to train a subspace Gaussian mixture model (SGMM). Finally, all the training speech is decoded using the SGMM system, and boosted maximum mutual information (BMMI) training of the SGMM parameters is performed.

The language model is trained using the SRILM tools [18]. A trigram language model with Good Turing smoothing is used. The model order, smoothing method and count-cutoffs are selected to minimize the perplexity of the 1.5 hour subset mentioned above.

Word lattices are generated with the SGMM+BMMI model using the Kaldi decoder. To support the creation of a phone index, the word latices are converted to phone lattices; no separate phone decoding is performed.

3.2. OpenFST-based System for Indexation and KWS

We implement the lattice indexation algorithm of [19] within the Kaldi suite. Specifically, the lattice of each utterance is converted into a finite-state acceptor with the posterior score, start-time and end-time for each word encoded as a

3805 keywords in 106b-v0.2g_conv-eval.kwlist2.xml			
1736 in-vocab keywords		2069 OOV keywords	
1067 Found in Dev	669 Not in Dev	670 Found in Dev	1399 Not in Dev

Table 1. Keyword statistics relative to the LVCSR lexicon and the Dev speech transcripts (search collection).

3-dimensional weight. An inverted index is then created from these individual acceptors, with paths to accept every possible word sequence in the original lattices. By applying standard WFST operations, one can work out the posterior score, start- and end-time of each occurrence of a word sequence.

To carry out KWS, keywords (which include multiword key-*phrases*) are typically compiled into linear acceptors K. By composing the acceptor K with the inverted index, one obtains the posterior score, start- and end-time of each occurrence of that keyword. The keyword acceptors do not have to be linear acceptors. They can be any acceptor, as long as each path in the acceptor represents a meaningful keyword or keyword phrase, such as the acceptor K' of (3) that represents multiple proxies for K. Furthermore, weights can be encoded in the keyword acceptors (such as the edit-distance supplied by E') if a single keyword has more than one representation.

Finally, a YES/NO decision is made according to the posterior scores from the search. We apply a keyword specific threshold proposed in [20], which uses the expected count of the keyword to estimate the number of the "true hits" in the formula for the actual term weighted value (ATWV). ATWV is computed using the NIST scoring tool F4DE.

4. EMPIRICAL EVALUATION OF KWS BY PROXY

4.1. Experimental Setup

We use `babel106b-v0.2g_conv-eval.kwlist2.xml`, the keyword list provided by NIST for the IARPA Babel Base Phase evaluation in Spring 2013, and the Babel Tagalog lexicon release `babel106b-v0.2g-sub-train` to designate a subset of the keywords as OOV. The search collection is the 10 hours of development-test data mentioned above.

Keyword occurrence statistics are shown in Table 1. Note that 2069 of 3805 keywords (54%) are OOVs w.r.t. the lexicon, which is not atypical in a low resource setting. This list was created for a separate (evaluation) search collection, and many keywords do not appear in the 10 hour development-test speech we use as our search collection. Indeed, only 1067 of 1736 in-vocabulary (IV) keywords, and 670 of 2069 OOV keywords occur in our search collection. Since the ATWV metric ignores keywords with zero true-positives, our KWS evaluation is *effectively* based on 1737 keywords. Yet, 670 (39%) of them are OOV. Ignoring OOV keywords therefore still significantly degrades the average ATWV.

In our experiments, we search for the IV keywords directly in the word index. We search for the OOV keywords by seeking either their word proxies (3) in the word index, or their phone proxies (2) in the phone index. The phone index is created, using the same OpenFST tools as the word index, but from phone lattices that are, in turn, converted from Kaldi word lattices. We generate and use phone proxies only for OOV keywords with *at least 5 phones*; shorter keywords generate too many false alarms, so it is better to ignore them.

4.2. Keyword Search Results

We conduct KWS experiments using keyword proxies for the 670 OOV keywords in 5 different conditions shown in Table 2. The richness of the proxy set is controlled by pruning in the *ShortestPath* step, and may be measured by either the average number of proxies per OOV keyword, or by the average number of hits retrieved (correct or false alarms) per OOV keyword. KWS performance in terms of the average ATWV over the 670 OOV keywords is plotted in Figures 3 and 4.

Several conclusions may be drawn from these two figures. Comparing Condition 4 (magenta) with Condition 2 (blue) in Figure 3 suggests that given a predetermined number of proxies per OOV keyword, searching a word index using word proxies is more effective than searching a phone index using phone proxies. Comparing Conditions 4 and 2 in Figure 4 suggests that for any given number of hits returned per OOV, word proxies are again the better choice. The quick drop in ATWV for Condition 2 in both Figures 3 and 4 suggest that even some highly ranked phone proxies cause significant false alarms. This affirms the value of relying on phone sequences that satisfy lexical constraints implicit in the word proxies. And since we use an expected count based threshold for YES/NO decisions, reducing the number of false alarms may indirectly increase the number of true hits marked YES.

Next, contrasting the pair of Conditions 1 & 2 against the pair of Conditions 3 & 4 in Figure 3 suggests that using a phone confusion transducer E or E' greatly improves KWS

Cond.	Index source	Proxy type	Uses E or E'
1	Phone lattice	Phone (K'')	No
2	Phone lattice	Phone (K'')	Yes (E)
3	Word lattice	Word (K')	No
4	Word lattice	Word (K')	Yes (E')
5	Word lattice w/ no LM scores	Word (K')	Yes (E')

Table 2. The numbered experimental conditions in Figures 3 and 4. The first two designate phone-based search without and with the use of phone proxies (2), the next two designate word-based search without and with the use of word proxies (3), and the last designates word-based search with word proxies after ignoring language model scores in the lattices.

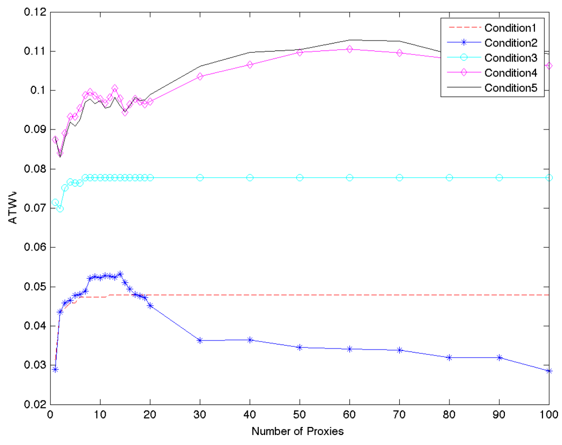

Fig. 3. ATWV versus the number of proxies per keyword.

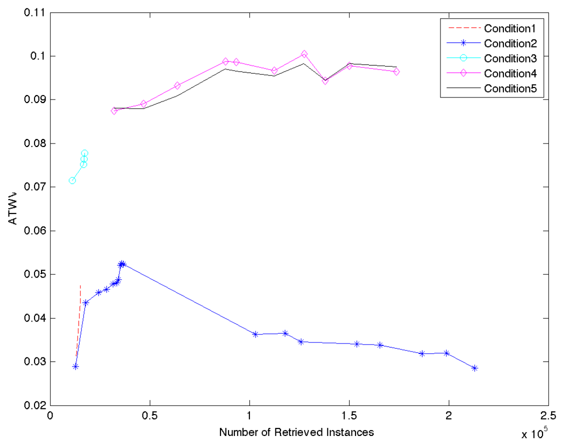

Fig. 4. ATWV versus number of retrieved hits per keyword.

performance with word proxies (comparison of Condition 3 with 4) but does not help much with phone proxies (comparison of Condition 1 with 2). In fact, using E hurts performance in Condition 2 when the number of phone proxies is large. This too may be explained by the inherent false alarm problem with phone proxies. Adding the phone confusion transducer simply aggravates the situation, unless the number of proxies is severely limited, e.g. to be around 10, as suggested by Figure 3. On the other hand, the word proxies appear to have better precision; admitting proxies permitted by phone confusion therefore further improves the performance. We thus conclude that phone confusion information is very helpful to word proxies (40% improvement in ATWV), but should be carefully constrained when applying to phone proxies.

Next, compare Condition 4 and Condition 5 in Figure 3. The index in Condition 5 is built without language model scores. As explained in Section 2.5, the mismatch between the proxies and the index may lead to some degradation in the KWS performance, because the proxies are created merely based on acoustic confusion while the index incorporates potentially incorrect language model scores. The result in Figure 3 shows that there is a small degradation by retaining the language model score in the index, but the degradation is usually negligible. Therefore, if only a single index must be retained/searched for both IV and OOV keywords, we suggest retaining the word index with language model scores.

ATWV for $\rightarrow$	IV Kwds	OOV Kwds	All Kwds
Without Proxies	0.351	0.000	0.216
With Proxies	0.351	0.110	0.258

Table 3. ATWV for searching a word index without and with the use of word proxies: 50 proxies per OOV keyword.

Some clarification may be in order for Condition 1 and Condition 3, where proxies are generated without a phone confusion transducer. If only one proxy were possible for each OOV keyword without the phone edits permitted by E or E', then curves for Condition 1 and Condition 3 in Figure 3 should be horizontal lines. However, multiple proxies are possible even without phone edits, because the lexicon L_1 contains multiple pronunciations for some words, and other legitimate ambiguities (e.g. homophones) in the phone-to-word transduction. Therefore, the curves first rise as multiple proxies are admitted by enlisting alternative shortest paths in (2) or (3), but quickly become horizontal lines once these limited alternatives are exhausted.

Finally, in Table 3, we compare the performance of word proxies on the 670 OOV keywords with search performance on the 1067 IV keywords, and note that while post-facto search by proxies is still much poorer than having them in-vocabulary, the average ATWV on the full NIST evaluation keyword set improves by 20% — from 0.216 to 0.258.

5. CONCLUSION

We have presented a simple, cheap and effective way to use *word proxies* to improve KWS performance for OOV keywords. Experiments were done with the Babel Tagalog data and the results suggest that such techniques are reasonably effective for handling OOV words.

6. ACKNOWLEDGEMENTS

We gratefully acknowledge many fruitful discussions with David Yarowsky, Aren Jansen and other members of the JHU Babel team. We also thank members of the CMU Babel team, particularly Florian Metze, collaboration with whom greatly facilitated this work.

7. REFERENCES

[1] The USC Shoah Foundation. Visited 07/02/2013. [Online]. Available: https://sfi.usc.edu/

[2] StoryCorps. Visited 07/02/2013. [Online]. Available: http://storycorps.org/

[3] TED: The Ideas Worth Spreading. Visited 07/02/2013. [Online]. Available: http://www.ted.com/

[4] Coursera. Visited 07/02/2013. [Online]. Available: https://www.coursera.org/

[5] I. Szoke, L. Burget, J. Cernocky, and M. Fapso, "Sub-word modeling of out of vocabulary words in spoken term detection," in *Spoken Language Technology Workshop, 2008. SLT 2008. IEEE.* IEEE, 2008, pp. 273–276.

[6] G. Chen, S. Khudanpur, D. Povey, J. Trmal, D. Yarowsky, and O. Yilmaz, "Quantifying the value of pronunciation lexicons for keyword search in low resource languages," in *Proceedings of ICASSP 2013*, 2013.

[7] B. Logan, J.-M. Van Thong, and P. J. Moreno, "Approaches to reduce the effects of oov queries on indexed spoken audio," *Multimedia, IEEE Transactions on*, vol. 7, no. 5, pp. 899–906, 2005.

[8] M. Saraclar and R. Sproat, "Lattice-based search for spoken utterance retrieval," in *Proceedings of HLT-NAACL 2004*, 2004.

[9] O. Siohan and M. Bacchiani, "Fast vocabulary-independent audio search using path-based graph indexing," in *Proceedings of Interspeech 2005*, 2005.

[10] J. Mamou, B. Ramabhadran, and O. Siohan, "Vocabulary independent spoken term detection," in *Proceedings of ACM SIGIR 2007*. ACM, 2007, pp. 615–622.

[11] U. V. Chaudhari and M. Picheny, "Improvements in phone based audio search via constrained match with high order confusion estimates," in *Proceedings of ASRU 2007*. IEEE, 2007, pp. 665–670.

[12] B. Logan and J.-M. Van Thong, "Confusion-based query expansion for oov words in spoken document retrieval," in *Procceedings of Interspeech 2002*, 2002.

[13] Y.-C. Li, W.-K. Lo, H. M. Meng, and P. Ching, "Query expansion using phonetic confusions for Chinese spoken document retrieval," in *Proceedings of IRAL 2000*, 2000.

[14] P. Karanasou, L. Burget, D. Vergyri, M. Akbacak, and A. Mandal, "Discriminatively trained phoneme confusion model for keyword spotting." in *Proceedings of Interspeech 2012*, 2012.

[15] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech Communication*, vol. 50, no. 5, pp. 434 – 451, 2008.

[16] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlíček, Y. Qian, P. Schwarz, J. Silovský, G. Stemmer, and K. Veselý, "The Kaldi speech recognition toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, Dec. 2011.

[17] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "OpenFst: a general and efficient weighted finite-state transducer library," in *Proceedings of the 12th international conference on Implementation and application of automata*, ser. CIAA'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 11–23.

[18] A. Stolcke, "SRILM - an extensible language modeling toolkit," in *7th International Conference on Spoken Language Processing, ICSLP2002*, 2002.

[19] D. Can and M. Saraclar, "Lattice indexing for spoken term detection," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, no. 8, pp. 2338 – 2347, nov. 2011.

[20] D. R. H. Miller, M. Kleber, C. Kao, O. Kimball, T. Colthurst, S. Lowe, R. M. Schwartz, and H. Gish, "Rapid and accurate spoken term detection," in *Proc. of Interspeech 2007*, vol. 7, 2007, pp. 314–317.