COMBINATION OF SEARCH TECHNIQUES FOR IMPROVED SPOTTING OF OOV KEYWORDS

Damianos Karakos and Richard M. Schwartz

Raytheon BBN Technologies, Cambridge, MA, USA

{dkarakos,schwartz}@bbn.com

ABSTRACT

The most common pipelines in keyword spotting involve some kind of speech recognition, which leads to the generation of sets of plausible hypotheses (e.g., word lattices), which are subsequently explored. The case of out-of-vocabulary (OOV) keywords is of special interest, because it requires representing keywords and/or lattices in an alternative format, so that the two can match. A number of techniques for dealing with OOV keywords have appeared in the literature; here, we focus on (i) fuzzy-phonetic search using phonetic confusion networks [1], and (ii) proxy-keyword search [2]. As we demonstrate in this paper, the combination of these two diverse techniques improves the ATWV of OOV keywords by at least 3% on average over the five development languages used in the second year of the IARPA Babel program.

Index Terms— keyword search, speech recognition, lattices, confusion networks, time quantization

1. INTRODUCTION

In commercial applications, as well as in most research studies related to speech recognition, decoding with whole-word units seems to be the most preferred method. In keyword spotting, where the goal is to detect the presence of given word phrases, this can be useful; when the keywords of interest are composed entirely of known (in-vocabulary) words, it has been shown [3, 4] that the best single-system performance is that of whole-word decodes.

When dealing with OOV keywords, things become more complicated; as demonstrated in the literature, [3, 5], state-of-the-art performance is obtained only when using multiple decodes involving sub-word units. Furthermore, combining together decodes that involve diverse sub-word units can be very beneficial [3]. However, if one is constrained to use whole-word decodes (when, e.g., the speech is pre-indexed using a whole-word vocabulary, or, when it is too expensive to run decoding with multiple recognition units), we would like to obtain comparable performance on the OOV keywords. For instance, [1] generates phonetic confusion networks and then performs fuzzy search in them. The work of [2] generates "proxy" keywords based on phonetic confusability and then matches them against a word lattice. This latter technique is almost identical to the one mentioned in [6].

In this paper, we combine the techniques of [1] and [2] in the case of whole-word decodes and show that one can obtain substantial ATWV gains of at least 3% over the five development languages used in the second year of the IARPA Babel program[1] Additionally, we introduce a new lattice construction, the time-quantized lattice (TQL), which is an order of magnitude smaller than the regular lattice, and can be used as an additional search output in the combination.

The paper is organized as follows: Section 2 briefly describes the pipelines used in this paper. Section 3 presents the procedure for generating TQLs. Section 4 describes the data used to generate the results of this paper. Section 5 presents keyword spotting results, and finally, Section 6 presents concluding remarks.

2. KEYWORD SPOTTING PIPELINES FOR DEALING WITH OOV KEYWORDS

For detecting OOV keywords, one must either resort to (i) hybrid fuzzy phonetic search strategies, in which recognition is done in terms of words, but the search is done fuzzily based on phonetic strings, allowing for inexact matches [1, 8, 9, 2], or, (ii) recognition in terms of shorter units [10, 3, 11, 12, 13, 5, 14] that have a higher chance of allowing a new word. In this paper, we focus mainly on the first category.

We next describe the keyword spotting techniques that we

We thank all members of the BBN Speech and Language group for useful discussions, especially those on the Babel project. We also acknowledge the contribution of the BUT partners of the BABELON team who provided MLP acoustic features. We also thank Guoguo Chen from JHU for sharing some implementation details of the proxy method. This work was supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense US Army Research Laboratory contract number W911NF-12-C-0013. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: 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 IARPA, DoD/ARL, or the U.S. Government.

[1] Our study is related, but quite different, from the work of [7], which focused only on IV keyword search.

experimented with. They can both be applied to decodes with whole-words (although other subword units are possible) and they specifically target the detection of OOV keywords. Both pipelines are applied on word posterior lattices; these are lattices generated after (i) computing the total score of each arc (a linear combination of the acoustic and language model log scores of that arc); (ii) running forward-backward in order to compute the posterior of the arc. (This is the total probability of all paths which pass through that arc.)

The first pipeline [1] performs a series of lattice transformations which aim at generating the right amount of connections, at the highest possible granularity. The steps are as follows.

1. Whole-word posterior lattices are converted to phonetic lattices. This is done by decomposing each word arc into a sequence of phone arcs, and copying the word posterior over to each one of the new arcs. The timing of each of the new arcs is approximated by the simple heuristic of dividing the word duration uniformly (although it is certainly possible to use an annotated form of lattice that contains the time at which each state is visited in the decoding graph).
2. Phonetic lattices are converted to phonetic confusion networks. This is done by using the same algorithm [15] as for whole-words.
3. Phonetic confusion networks are searched using fuzzy matching and dynamic programming. Each keyword is represented as one or more sequences of phones. We use dynamic programming with a beam search to find all paths through the c-net for each pronunciation. The score of a match is the product of the posteriors of the c-net arcs and a predefined mismatch probability for each phone insertion, deletion and substitution [1].

The second pipeline follows [2], but has been re-implemented in-house to allow more flexibility. The main steps are as follows.

1. Each keyword is represented as a finite-state transducer (FST) [16] which encodes each keyword as a sequence of phones (again, allowing multiple pronunciations).
2. The keyword FST, which is the union of the individual keyword-specific FSTs, is composed with a confusion FST which represents the cost of confusing one phone with another, as well as the cost for inserting/deleting a phone. (We set the identity cost to zero.)
3. The resulting FST is pruned with a threshold based on cost, so that implausible paths are removed. For multi-word keywords, each constituent word is treated separately so that longer keywords are not more severely penalized.
4. A second (optional) threshold prevents too many insertions by imposing a limit on the number of phones generated.
5. The resulting FST is subsequently composed with the "Kleene +" of a phone-to-word (P2W) FST, which is the FST representation of the inverse of the word lexicon.
6. The resulting "proxy-keyword FST" encodes all the "proxy"' forms of each keyword. It can be optionally pruned so that the length of each proxy does not differ too much from the length of the original keyword.
7. The "proxy-keyword FST" is further composed with the indexed version of each posterior lattice (see, e.g., Figure 1 of [6] for an example of an indexed set of c-nets—the same idea can be used for general lattices). Extraction of hits from the resulting FST is done by keeping a priority queue of partial paths at each node, which is further constrained using a beam.
8. Before generation of the indexed version, the posterior lattice is processed so that intra-word and inter-word silences are limited to 0.1 and 0.5 seconds, respectively.

Each one of the resulting sets of detections (hits) is normalized using the linear fit method of [17], followed by the KST normalization technique of [18]. The normalized lists are then combined together (two at a time and all together) using the algorithm mentioned in [18].

3. CONSTRUCTION OF TIME-QUANTIZED LATTICES

In this section we describe how a new lattice structure, the time-quantized lattice (TQL) is constructed. The main parameter to consider in the construction is the duration d between the "anchor" nodes. Smaller (resp. larger) duration corresponds to finer (resp. coarser) time quantization and more (resp. less) faithfulness to the original lattice.

To build the TQLs, the following steps are followed for each utterance:

1. The start and end times of the utterance correspond to the start and end times of the TQL. When the utterance duration is less than d, the original lattice is used.
2. The nodes of the TQL are "anchor" nodes, placed at regular time intervals. That is, starting with the start time of the utterance, each TQL node corresponds uniquely to a multiple of d seconds after that. Note that d may be slightly adjusted to make sure that the time of the last node is exactly equal to the end time of the utterance.
3. Each lattice arc is "inserted" into the TQL by attaching its start/end nodes to the closest anchor nodes. To prevent deleting arcs (or introducing self-loops) a lattice

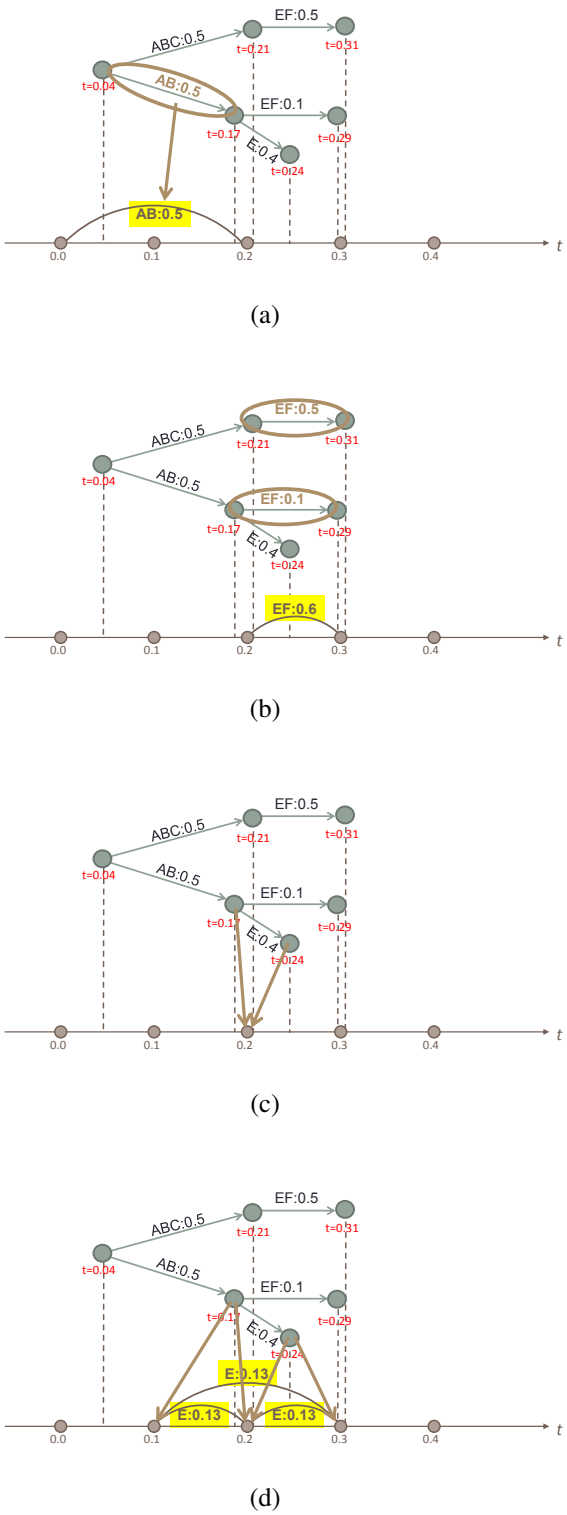

Fig. 1: (Caption appears in next column.)

Fig. 1: Steps in the generation of a TQL. The "x-axis" represents time and it illustrates the connection between TQL nodes and times. (a) The default of inserting a lattice arc to the TQL is to attach to the closest TQL nodes. (b) When two or more lattice arcs with the same label (e.g., word) get aligned at the same anchor nodes their posteriors are added together. (c) The default operation of (a) can lead to missing arcs, or self-loops (depending on how one would like to handle cases of arcs whose endpoints are both aligned to the same anchor node). (d) To prevent this from happening, we instead align each of the endpoints of a lattice arc to at most *two* anchor nodes. This can lead to more than one arc in the TQL; the original posterior is divided equally among the resulting TQL arcs.

arc can give rise to multiple TQL arcs, each of which attach to adjacent anchor nodes. The original posterior is distributed accordingly (we have found that dividing uniformly works well). See Figure 3 for an illustration of the generation process.

We have also experimented with alternatives, such as: (i) attaching to more than two adjacent anchor nodes, (ii) dividing the original posterior non-uniformly, but according to the proximity to the anchor node. Since we did not observe any significant gain from these alternatives, we decided to follow the much simpler approach of dividing the posterior uniformly. We also varied the time duration between anchor nodes, and we found that the best setting is 50ms. This is the setting chosen for the results in this paper.

As expected, TQLs lie between lattices and c-nets in terms of density. They are about 5–6 times larger than c-nets and 20 times smaller than lattices. So, they provide an interesting alternative as far as storage and computation are concerned.

4. DATA

Here, we briefly describe the data used in the experiments of this paper.

First, the languages considered were Assamese, Bengali, Haitian Creole, Lao and Zulu[2]. These were the development languages delivered during the second year of the IARPA Babel project.

The primary condition that the program participants were evaluated on designated only 10 hours of transcribed audio for acoustic and language model training. This is the condition considered in this paper as well.

Results are reported on the unsequestered part (Eval1) of the blind data used in the IARPA Babel official evaluation in the Spring of 2014. It consists of approximately 5 hours of audio per language. Tuning of the system combination weights

[2]The official codes for the final releases of the languages were IARPA-babel102b (Assamese), IARPA-babel103b (Bengali), IARPA-babel201b (Haitian Creole), IARPA-babel203b (Lao), IARPA-babel206b (Zulu).

		assamese		bengali		haitian		lao		zulu	
		IV	OOV	IV	OOV	IV	OOV	IV	OOV	IV	OOV
method of [1]	lat/c-net	38.9	19.2	40.0	24.1	52.8	26.8	46.1	5.9	34.4	17.7
	TQL/c-net	31.3	17.2	33.9	22.1	48.7	24.1	42.1	3.2	28.7	16.6
	$\text{comb}_{[1]}$	40.6	21.3	41.3	23.6	54.0	28.6	48.4	13.4	34.9	18.9
proxy	lat	40.4	20.7	40.8	23.6	53.2	23.4	47.6	9.0	33.3	14.1
	c-net	37.4	16.8	38.4	20.6	49.2	15.5	44.6	6.1	33.4	12.1
	TQL	**40.7**	22.7	40.5	23.7	53.4	22.5	47.9	12.4	34.0	13.3
	$\text{comb}_{\text{proxy}}$	40.6	22.7	41.3	23.3	53.2	20.0	48.1	16.2	34.2	14.9
$\text{comb}_{[1]}$ + $\text{comb}_{\text{proxy}}$		40.2	**26.1**	**42.1**	**29.3**	**54.9**	**31.8**	**48.6**	**17.6**	**35.8**	**21.3**
syllable (method of [1])		37.4	25.5	39.4	32.0	52.5	35.3	47.0	20.4	33.9	26.1
word + syllable		41.7	27.7	43.1	32.4	55.8	35.4	49.8	21.4	38.0	27.9

Table 1: ATWV results with various methods discussed in the paper. IV results are shown for reference. When using only whole-word decodes, the best ATWV on each language (shown in bold) is obtained by combining the proxy and non-proxy modes of search. Significantly better results (underlined) can be obtained when combining with syllabic decodes.

As	Be	Ha	La	Zu
24.2	26.2	21.8	11.5	45.3

Table 2: Percentage of OOV keywords per language.

and of the decision thresholds was done on a Dev set, consisting of approximately 10 hours of audio per language.

ATWV is the primary performance measure used in the IARPA Babel program. It is defined as one minus a linear combination of the rates of misses and false alarms [19], averaged over the keyword set.

The set of keywords on which we report performance consists of the union of the development and evaluation keywords used during the second year of the program. Their number varies between 4.7K and 5.1K across languages. The percentage of OOV keywords per language is shown in Table 2.

5. EXPERIMENTAL RESULTS

In this section we present ATWV results obtained on the Eval1 data set mentioned in Section 4. Table 1 shows ATWV results with the keyword spotting pipelines of Section 2, as well as their combination.

In the case of the method of [1], the lat/c-net rows correspond to using c-nets extracted from regular lattices (word c-nets for IV keywords with exact match, and phonetic c-nets for OOV keywords with fuzzy match). Similarly for the TQL/c-net rows.

For both methods ([1] and proxies), the "comb" rows correspond to the combination of hit lists obtained by using all available structures (lattices, cnets, TQLs).

For comparison, we also show the result when decoding with syllables ("syllable (method of [1])" row) and in combination with whole-word decodes ("word+syllable" row).

There are several conclusions from these results: (i) On average, when used with proxies, lattices and TQLs perform better than c-nets, while TQLs perform slightly better than lattices. (ii) On average, there is a significant 3% absolute ATWV gain on the OOV keywords from doing the combination of all search methods considered in the case of whole-word decoding. (iii) Syllables give better ATWVs for OOV keywords than whole-words. (iv) Whole-words and syllables combine well (a similar observation appeared in [3]).

6. CONCLUSIONS

We considered two techniques for dealing with OOV keywords: (i) fuzzy-phonetic search using phonetic confusion networks [1], and (ii) proxy-keyword search [2]. As we demonstrate in this paper for the case of whole-word decoding, the combination of these two diverse techniques improves the ATWV of OOV keywords by at least 3% on average over the five development languages used in the second year of the IARPA Babel program. We also introduced a new lattice structure, the time-quantized lattice, which has competitive performance, while being an order of magnitude smaller than a regular lattice. Having a variety of connections in the outputs of ASR systems is crucial for detecting different keyword types; system combination seems to be an important system component for exploiting this variability.

7. REFERENCES

[1] I. Bulyko, J. Herrero, C. Mihelich, and O. Kimball, "Subword speech recognition for detection of unseen words," in *Proc. of Interspeech*, Portland, Oregon, Sep 2012.

[2] G. Chen, O. Yilmaz, J. Trmal, D. Povey, and S. Khudanpur, "Using proxies for OOV keywords in the keyword search task," in *Proc. of ASRU*, 2013.

[3] D. Karakos and R. Schwartz, "Subword and phonetic search for detecting out-of-vocabulary keywords," in *Proc. of Interspeech*, Singapore, 2014.

[4] W. Hartmann, V.-B. Le, A. Messaoudi, L. Lamel, and J.-L. Gauvain, "Comparing decoding strategies for subword-based keyword spotting in low-resourced languages," in *Proc. of Interspeech*, Singapore, 2014.

[5] K. Narasimhan, D. Karakos, R. Schwartz, S. Tsakalidis, and R. Barzilay, "Morphological segmentation for keyword spotting," in *Proc. of EMNLP*, 2014.

[6] L. Mangu, B. Kingsbury, H. Soltau, H.-K. Kuo, and M. Picheny, "Efficient spoken term detection using confusion networks," in *Proc. of ICASSP*, 2014.

[7] J. Chiu, Y. Wang, J. Trmal, D. Povey, G. Chen, and A. Rudnicky, "Combination of FST and CN search in spoken term detection," in *Proc. of Interspeech*, Singapore, 2014.

[8] D. R.H. Miller, M. Kleber, C.-L. Kao, O. Kimball, T. Colthurst, S. A. Lowe, R. M. Schwartz, and H. Gish, "Rapid and accurate spoken term detection," in *Proc. of Interspeech*, 2007.

[9] J. Mamou, B. Ramabhadran, and O. Siohan, "Vocabulary independent spoken term detection," in *Proc. of SIGIR'07*, Amsterdam, The Netherlands, July 2007.

[10] I. Szoke, L. Burget, J. Cernocký, and M. Fapo, "Subword modeling of out of vocabulary words in spoken term detection," in *IEEE Workshop on Spoken Language Technology*, India, 2008.

[11] F. Seide, P. Yu, C. Ma, and E. Chang, "Vocabulary-independent search in spontaneous speech," in *Proc. of ICASSP*, 2004.

[12] A. Rastrow, A. Sethy, and B. Ramabhadran, "A new method for OOV detection using hybrid word/fragment system," in *Proc. of ICASSP*, 2009.

[13] A. Rastrow, A. Sethy, B. Ramabhadran, and F. Jelinek, "Towards using hybrid word and fragment units for vocabulary independent LVCSR systems," in *Proc. of Interspeech*, 2009.

[14] P. Baumann, H. Fang, R. He, B. Hutchinson, A. Jaech, E. Fosler-Lussier, M. Ostendorf, J. Pierrehumbert, A. Janin, and S. Wegmann, "Leveraging morphology for dealing with data sparsity," Presented at the IARPA Babel PI Meeting, January 2014.

[15] L. Mangu, E. Brill, and A. Stolcke, "Finding consensus in speech recognition: word error minimization and other applications of confusion networks," *Computer Speech and Language*, vol. 14, no. 4, pp. 373–400, 2000.

[16] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri, "Openfst: A general and efficient weighted finite-state transducer library," in *Proc. of CIAA, Springer*, 2007.

[17] D. Karakos, I. Bulyko, R. Schwartz, S. Tsakalidis, L. Nguyen, and J. Makhoul, "Normalization of phonetic keyword search scores," in *Proc. of ICASSP*, Florence, Italy, 2014.

[18] D. Karakos, R. Schwartz, S. Tsakalidis, L. Zhang, S. Ranjan, T. Ng, R. Hsiao, G. Saikumar, I. Bulyko, L. Nguyen, J. Makhoul, F. Grezl, M. Hannemann, M. Karafiat, I. Szoke, K. Vesely, L. Lamel, and V.-B. Le, "Score normalization and system combination for improved keyword spotting," in *Proc. of ASRU*, Olomouc, Czech Republic, 2013.

[19] NIST, "OpenKWS13 keyword search evaluation plan," http://www.nist.gov/itl/iad/mig/upload/OpenKWS13-EvalPlan.pdf, 2013.