IMPROVING MULTI-LATTICE ALIGNMENT BASED SPOKEN KEYWORD SPOTTING

Hui Lin, Alex Stupakov and Jeff Bilmes

Department of Electrical Engineering, University of Washington, Seattle, Washington, USA

ABSTRACT

In previous work, we showed that using a lattice instead of the 1-best path to represent both the query and the utterance being searched is beneficial for spoken keyword spotting. In this paper, we introduce several techniques that further improve our multi-lattice alignment approach, including edit operation modeling and supervised training of the conditional probability table, something which cannot be directly trained by traditional maximum likelihood estimation. Experiments on TIMIT show that the proposed methods significantly improve the performance of spoken keyword spotting.

***Index Terms*—** Spoken keyword spotting, lattice alignment, edit operation modeling, negative training, auxiliary training

1. INTRODUCTION

In certain cases, speech-specified keyword spotting is more appropriate than text-based keyword detection, such as whenever it is inconvenient, unsafe, or impossible for the user to enter a search query using a standard keyboard. For example, modern police officers or soldiers are sometimes equipped with a multi-sensor platform that has been augmented with a close-talking microphone, a camera, and a wrist-mounted display. During many on-the-job scenarios (such as while driving, walking, or whenever the hands are unavailable), spoken queries may be more appropriate to search through recordings of conversations in order to locate audio, photos, and video that have been recorded on the device during an investigation.

In [1], we proposed a new approach to spoken keyword spotting that uses a joint alignment between multiple phone lattices. The first phone lattice comes from the database itself and can be created offline. We refer to this as the *utterance lattice*. A second phone lattice is generated once the user has spoken a query phrase. This *query lattice* is then modified by removing its time marks, and then the two lattices are jointly aligned. Every region of time where the query lattice is properly aligned then becomes a candidate spoken keyword detection.

In this paper, we propose several methods to improve the performance of our multi-lattice alignment approach. The first method is related to edit operation modeling, which focuses on providing robustness against mistakes in the phone lattices. The learning of the consistency conditional probability table (CPT) is also investigated in this paper. As seen in [1] (a prerequisite reference for understanding the current paper), the consistency CPT plays an important role in gluing the query lattice and utterance lattice together to produce accurate alignments. Maximum likelihood training alone, however, is inappropriate for the consistency CPT — we address this issue in this paper using negative training data [2]. Experiments on TIMIT show that these methods significantly improve the performance of our spoken keyword spotting system relative to [1].

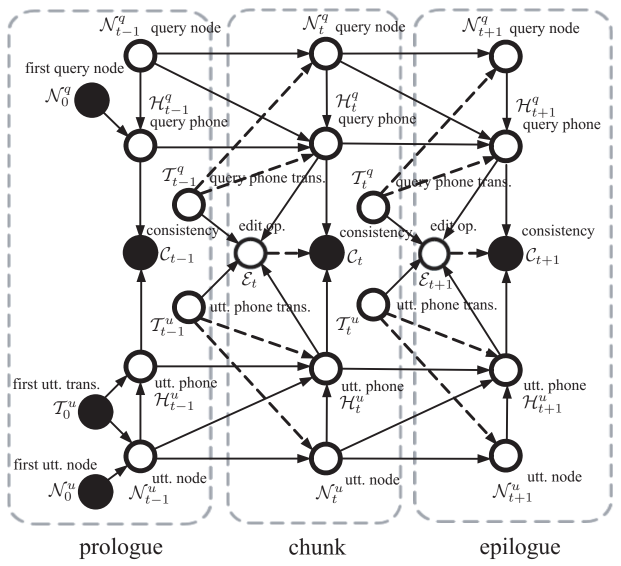
This work was supported by DARPA's ASSIST Program (No. NBCH-C-05-0137) and an ONR MURI grant (No. N000140510388).

2. MULTI-LATTICE ALIGNMENT

We first briefly review the multi-lattice alignment approach to spoken keyword spotting that we proposed in [1]. We refer the reader to [1] for full details of our method.

In our approach, the lattice of the query keyword and the lattice of the utterance are both represented by graphical models, as shown in Fig 1. The upper half of the graph corresponds to the query, and the lower half to the utterance. The graphical model representations of the two lattices have similar topology — for each lattice, two distinct random variables represent the lattice node and lattice link. For instance, for the query lattice, we have the `query node` variable $\mathcal{N}_t^q$ for the lattice node, and the `query phone` variable $\mathcal{H}_t^q$ for the lattice links (which represent phones) in the phone lattice. The major difference between the two lattices is that for the graph that represents the query lattice, the time information associated with each node in the original lattice is discarded, while the graphical model for the utterance lattice uses a time inhomogeneous conditional probability table (CPT) to encode the starting/ending time points of links in the lattice. Specifically, the `utterance phone transition` variable $\mathcal{T}_t^u$ can take the value 1 (meaning there will be a transition for the `utterance phone` variable $\mathcal{H}_{t+1}^u$) only when there is an actual transition in the original lattice.

The `consistency` variable $\mathcal{C}_t$, which is always observed with value 1, couples the query and utterance lattices together. In particular, the CPT $p(\mathcal{C}_t = 1|\mathcal{H}_t^q, \mathcal{H}_t^u) = f(\mathcal{H}_t^q, \mathcal{H}_t^u)$ is simply a function of $\mathcal{H}_t^q$ and $\mathcal{H}_t^u$. If $\mathcal{H}_t^q$ is identical or similar to $\mathcal{H}_t^u$, $f(\mathcal{H}_t^q, \mathcal{H}_t^u)$ should take larger values,



$\mathcal{N}^q_{t-1}$ query node
$\mathcal{N}^q_t$ query node
$\mathcal{N}^q_{t+1}$ query node
first query node
$\mathcal{N}^q_0$
$\mathcal{H}^q_{t-1}$ query phone
$\mathcal{H}^q_t$ query phone
$\mathcal{H}^q_{t+1}$ query phone
$\mathcal{T}^q_{t-1}$ query phone trans.
$\mathcal{T}^q_t$ query phone trans.
consistency $\mathcal{C}_{t-1}$
edit op. $\mathcal{E}_t$
consistency $\mathcal{C}_t$
edit op. $\mathcal{E}_{t+1}$
consistency $\mathcal{C}_{t+1}$
$\mathcal{T}^u_{t-1}$ utt. phone trans.
$\mathcal{T}^u_t$ utt. phone trans.
first utt. trans.
$\mathcal{T}^u_0$
utt. phone $\mathcal{H}^u_{t-1}$
utt. phone $\mathcal{H}^u_t$
utt. phone $\mathcal{H}^u_{t+1}$
first utt. node
$\mathcal{N}^u_0$
utt. node $\mathcal{N}^u_{t-1}$
utt. node $\mathcal{N}^u_t$
utt. node $\mathcal{N}^u_{t+1}$
prologue
chunk
epilogue

Fig. 1. Graph for keyword spotting with spoken query. Dark circles represent observed variables and light circles represent hidden variables.

and $f(\cdot)$ should take smaller values otherwise. This ensures that better matched phone hypothesis string pairs will likely survive any pruning stage in decoding, potentially indicating a discovered keyword.

3. EDIT OPERATION MODELING

When transforming a source string into a target string, edit operations (*deletion*, *insertion*, or *substitution*) may be performed. The weighted sum of the minimum cost set of edit operations needed to transform one string into another, namely the Levenshtein or edit distance, is commonly used to measure the distance between two strings. In [3], where the task is keyword spotting based on phone lattices, the minimum edit distance is successfully used during a lattice search to compensate for phone recognizer insertion, deletion and substitution errors. In our case, lattices comprising multiple hypothesized strings (rather than single hypotheses) are aligned. One naive way of finding the minimum edit distance between lattices is to consider all possible strings contained in both the query and utterance lattice, and to compute the edit distance for all the string pairs. This would be computationally intractable as a linear length lattice can represent an exponential number of hypotheses. Actually, the tree edit distance and alignment distance problems are NP-hard [4]. Here we propose an edit operation modeling method for lattice alignment under the Dynamic Bayesian Network (DBN) framework, which allows us to quickly evaluate a variety of different alignment algorithms all using the same underlying (exact or approximate) inference code. We note that DBNs have been used as a generalized string alignment scheme in the past [5].

Specifically, we incorporate insertion and deletion operations explicitly into the aforementioned joint alignment graph. There are two benefits to representing edit operations using a DBN. First, by representing a lattice with random variables, all combinations of all paths from both query and utterance lattices can be taken into consideration during alignment — this is done, thanks to the underlying DBN dynamic programming inference algorithms, without needing to consider a combinatorial explosion of hypotheses. Second, since "cost" is equivalent to negative log probability, the cost functions for insertion and deletion can be expressed as probability tables associated with phone random variables, which makes the learning of such cost functions possible [5], as we show in Section 4.

Our approach starts by employing an `edit operation` variable $\mathcal{E}_t$ that indicates the type of edit operation occurring at time t. It is a discrete variable with cardinality 4, taking values from set $\{\text{I,D,S,M}\}$, where I indicates insertion, D represents deletion, S represents substitution, and M represents a good match. As shown in Fig. 1, $\mathcal{E}_t$ has four parents, and their logical relationship is illustrated in Algorithm 1.

Algorithm 1
```
if $\mathcal{H}^q_t = \mathcal{H}^u_t$ then
    match operation: $\mathcal{E}_t = \text{M}$
else
    if $\mathcal{T}^q_{t-1} = 1$ and $\mathcal{T}^u_{t-1} = 0$ then
        insertion operation: $\mathcal{E}_t = \text{I}$
    else if $\mathcal{T}^q_{t-1} = 0$ and $\mathcal{T}^u_{t-1} = 1$ then
        deletion operation: $\mathcal{E}_t = \text{D}$
    else if $\mathcal{T}^q_{t-1} = 1$ and $\mathcal{T}^u_{t-1} = 1$ then
        substitution operation: $\mathcal{E}_t = \text{S}$
    else if $\mathcal{T}^q_{t-1} = 0$ and $\mathcal{T}^u_{t-1} = 0$ then
        no operation: $\mathcal{E}_t = \text{M}$
    end if
end if
```

The `edit operation` variable $\mathcal{E}_t$ serves as a switching parent of the `consistency` variable $\mathcal{C}_t$. Specifically, the per-frame consistency variable score used during decoding is given by

$$\begin{cases} (p_s(\mathcal{C}_t = 1|\mathcal{H}^q_t, \mathcal{H}^u_t))^{w_s} & : \text{ if } \mathcal{E}_t = \text{S} \\ (p_i(\mathcal{C}_t = 1|\mathcal{H}^q_t, \mathcal{H}^u_t))^{w_i} & : \text{ if } \mathcal{E}_t = \text{I} \\ (p_d(\mathcal{C}_t = 1|\mathcal{H}^q_t, \mathcal{H}^u_t))^{w_d} & : \text{ if } \mathcal{E}_t = \text{D} \\ 1 & : \text{ if } \mathcal{E}_t = \text{M}. \end{cases} \tag{1}$$

Subscripts s, i, and d to $p(\cdot)$ are used to indicate that although the graph topology is the same, probability distributions (which are determined by the switching parent $\mathcal{E}_t$) can be different. Moreover, the exponential weights are also potentially different depending on the switching parent — i.e., we use both different scaling factors (w_s, w_i and w_d) different consistency CPTs for different edit operations. When there is a match operation, $\mathcal{E}_t$ takes value M, resulting in a unity score being used, i.e., $p_m(\mathcal{C}_t = 1|\mathcal{H}_t^q, \mathcal{H}_t^u) = 1$.

While the consistency CPTs can be manually assigned by phonetic and/or prior knowledge, a potentially better way is to automatically learn them from data, as introduced in the following section.

4. LEARNING OF THE CONSISTENCY CPT

The consistency CPT $p(\mathcal{C}_t = 1|\mathcal{H}_t^q, \mathcal{H}_t^u)$ plays an crucial role in the alignment process, as we can see from the scoring functions in Eq.1. From the perspective of decoding, $p(\mathcal{C}_t = 1|\mathcal{H}_t^q, \mathcal{H}_t^u)$ is proportional to a non-negative function of $\mathcal{H}_t^q$ and $\mathcal{H}_t^u$ (any proportionality constant does not change the results). We may thus write, for example, $p_s(\mathcal{C}_t = 1|\mathcal{H}_t^q, \mathcal{H}_t^u) = f_s(\mathcal{H}_t^q, \mathcal{H}_t^u)$ for non-negative function $f_s(\cdot, \cdot)$. If $\mathcal{H}_t^q$ is "similar" to $\mathcal{H}_t^u$, $f_s(\mathcal{H}_t^q, \mathcal{H}_t^u)$ should take larger values, and otherwise $f_s(\mathcal{H}_t^q, \mathcal{H}_t^u)$ should take smaller values. Such a function can be formed in a number of ways. For example, it can be derived from linguistic/phonetic knowledge [6, 1], or it can be approximated by estimating the Kullback-Leibler distance of the acoustic models for different phones [1]. Another natural way of obtaining the consistency CPT is to train it in place, within the model, using a supervised training procedure. The maximum likelihood estimate of this CPT, however, is simply the ratio of counts:

$$\begin{aligned} p(\mathcal{C}_t = 1|\mathcal{H}_t^q, \mathcal{H}_t^u) &= \frac{\mathbf{N}(\mathcal{C}_t = 1, \mathcal{H}_t^q, \mathcal{H}_t^u)}{\mathbf{N}(\mathcal{H}_t^q, \mathcal{H}_t^u)} \\ &= \frac{\mathbf{N}(\mathcal{C}_t = 1, \mathcal{H}_t^q, \mathcal{H}_t^u)}{\mathbf{N}(\mathcal{C}_t = 0, \mathcal{H}_t^q, \mathcal{H}_t^u) + \mathbf{N}(\mathcal{C}_t = 1, \mathcal{H}_t^q, \mathcal{H}_t^u)} \end{aligned} \tag{2}$$

where $\mathbf{N}(\cdot)$ is the count function (or the sum of expected sufficient statistics in the EM case). Note that without any data labeled $\mathcal{C}_t = 0$, known as "negative training data" [2], the estimated ratio is always 1 (unity), and no statistical relationship between $\mathcal{H}_t^q$ and $\mathcal{H}_t^u$ is learnt. In this paper, we propose two methods to overcome this problem.

4.1. Negative Training

Our first method is based on the negative training approach proposed in [2]. We express the count function as in [2]:

$$\begin{aligned} \mathbf{N}(\mathcal{C}_t = 1, \mathcal{H}_t^q, \mathcal{H}_t^u) &= M_1 \cdot p(\mathcal{H}_t^q, \mathcal{H}_t^u) \\ \mathbf{N}(\mathcal{C}_t = 0, \mathcal{H}_t^q, \mathcal{H}_t^u) &= n \cdot M_1 \cdot p(\mathcal{H}_t^q) p(\mathcal{H}_t^u) \end{aligned}$$

where M_1 is the number of samples in the positive training data, nM_1 is the total number of samples in the induced negative training data, and n is the ratio of the amount of positive to negative training data, which can be assigned based the prior beliefs about consistency. The consistency CPT can be formed without generating any actual negative training data using the following sigmoidal form:

$$p(\mathcal{C}_t = 1|\mathcal{H}_t^q, \mathcal{H}_t^u) = \frac{1}{1 + n\left(\frac{p(\mathcal{H}_t^q, \mathcal{H}_t^u)}{p(\mathcal{H}_t^q)p(\mathcal{H}_t^u)}\right)^{-1}} \tag{3}$$

Based on the above, we utilize the following EM procedure to train the consistency CPT:

1. Select a set of keywords for the *training* set, then segment out all instances of the selected keywords from the audio files, and generate the query lattices.
2. Initialize $p(\mathcal{C}_t = 1|\mathcal{H}_t^q, \mathcal{H}_t^u)$ either by phonetic knowledge, or by estimating the Kullback-Leibler distance of the acoustic models for different phones.
3. With the keyword location observed, get the alignments for each query in all the utterances where the query occurs, and obtain the frame by frame Viterbi outputs in the aligned keyword regions.
4. Re-estimate $p(\mathcal{C}_t = 1|\mathcal{H}_t^q, \mathcal{H}_t^u)$ using Eq. 3, where the $p(\mathcal{H}_t^q, \mathcal{H}_t^u)$, $p(\mathcal{H}_t^q)$ and $p(\mathcal{H}_t^u)$ are calculated based on the Viterbi outputs obtained in step 3.
5. Repeat steps 3 and 4 until apparent convergence, i.e., the numerical changes in the consistency CPT fall below a preset threshold.

4.2. Auxiliary training

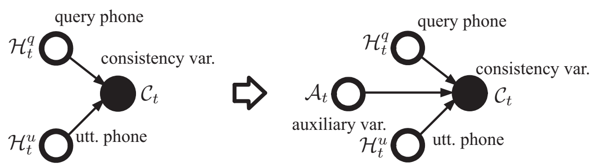


Fig. 2. Training with an auxiliary variable

The second method we propose here is to train the consistency CPT with an auxiliary variable, similar to an approach used in [7]. Originally, we have one observed child (the consistency variable) with two parents (the query and utterance phone variables). A discrete auxiliary variable $\mathcal{A}_t$ is added as another parent of the consistency variable, forming a graph with one observed child and three parents, as shown in Fig. 2 on the right. $\mathcal{A}_t$ has cardinality d^2, where d is the cardinality for $\mathcal{H}_t^q$ and $\mathcal{H}_t^u$. The logical relationship among these variables is:

$$\begin{aligned} p(\mathcal{C}_t = 1|\mathcal{H}_t^q = i, \mathcal{H}_t^u = j, \mathcal{A}_t = k) = 1 \quad &: \text{ if } k = d \cdot i + j \\ p(\mathcal{C}_t = 1|\mathcal{H}_t^q = i, \mathcal{H}_t^u = j, \mathcal{A}_t = k) = 0 \quad &: \text{ otherwise} \end{aligned}$$

In other words, there is a one-to-one mapping from $k \in \{0, ..., d^2 - 1\}$ to the pair (i, j) so that the only way to explain the observed child is that values i and j match k. Thus, after EM training, $p(\mathcal{A}_t = k)$ is the learnt score for $f(\mathcal{H}_t^q = i, \mathcal{H}_t^u = j)$, and can be used to form the consistency

CPT. Auxiliary training does not make any assumptions about quantities of negative training data, but it is computationally more expensive since the added auxiliary variable $\mathcal{A}_t$ has large cardinality d^2.

5. EXPERIMENTS

We performed experiments on the TIMIT speech database for evaluating the effect of our proposed methods on spoken keyword spotting performance. The experimental setup was the same as in [1]. Keyword query phrases were selected from the TIMIT test set by choosing single-word and multi-word phrases with lengths of 6-12 phones that occurred at least twice in the test set. 67 total such keywords were chosen, distributed evenly over lengths. The TIMIT test set was used as the evaluation data set, which has about 1.5 hours of audio. In total, there were about 400 instances of the keywords in the test set. One instance of each keyword was chosen as the query.

Both the query and utterance phone lattices were generated with the CMU Sphinx 3.7 decoder in allphone mode, using a phone bigram language model and a speaker-independent monophone acoustic model. The resulting lattices yielded a 33.4% PER (phone error rate) and 11.1% oracle PER on the NIST core test set.

The TIMIT training set was used for training the consistency CPT. Keyword query phrases were selected in the same way as the test set. For each selected keyword, audio that corresponds to all instances from the training set was excised and processed to create phone lattices to be used as the queries in the training. In total, over 1500 query lattices were generated. Each query lattice was forced aligned to the utterances which contain the query occurrences. The initial consistency CPT was derived from acoustic models using KL distances, as was done in [1]. The frame-by-frame Viterbi outputs were then used as the supervised training data for both negative training and auxiliary training.

ROC curves were generated by varying the dummy state (which is used to absorb the non-keyword region) score s_d defined in [1]. The x axis indicates the number of false alarms per hour per keyword, and the y axis indicates the recall rate of detection.

To see whether introducing the edit operation into the graphical model improves performance, we kept $p_s(\cdot)$ the same as the baseline (where the consistency CPT was derived from acoustic models using KL distances), while assigning $p_i(\cdot), p_d(\cdot)$ using prior knowledge (i.e., `silence` phone is allowed to be inserted/deleted). Suitable weights w_s, w_d, w_i were found by grid search. As shown in Fig. 3 (the edit op. modeling curve), this gives us significant improvements over the baseline. We also separately investigated the effects of methods for learning the consistency CPTs. In particular, the consistency CPTs in the baseline model were replaced by those learned from negative training and auxiliary training, and improvements were also achieved (the negative training and auxiliary training curves in Fig. 3). Finally, using learned CPTs for edit operation modeling (negative training + edit operation modeling) further improves the performance.

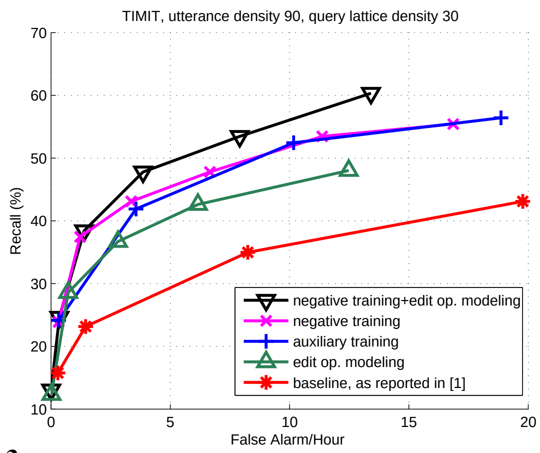


Fig. 3. ROC curves of different methods. Baseline is without edit operation modeling, and uses consistency CPT derived from the acoustic models by KL distances, as used in [1].

6. DISCUSSION

Our results show quite a significant improvements over past work [1], yet we believe there are additional steps that could improve results further. First, discriminative training should be evaluated against our negative training and auxiliary training approaches. Second, additional acoustic information could be incorporated directly into the model (Figure 1) rather than only indirectly via a phone variable. Third, more advanced context-dependent edit operations could be incorporated into the model and automatically learned, as in [5].

7. REFERENCES

[1] H. Lin, A. Stupakov, and J. Bilmes, "Spoken Keyword Spotting via Multi-lattice Alignment," *Interspeech*, 2008.

[2] S. Reynolds and J. Bilmes, "Part-of-speech tagging using virtual evidence and negative training," *Human Language Technology Conference/Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP-2005)*, Oct 2005.

[3] K. Thambiratnam and S. Sridharan, "Dynamic Match Phone-Lattice Searches For Very Fast And Accurate Unrestricted Vocabulary Keyword Spotting," in *Proc. ICASSP*, 2005.

[4] A. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, "Proof verification and hardness of approximation problems," *Proc. of the 33rd IEEE Symposium on the Foundations of Computer Science*, 1992.

[5] K. Filali and J. Bilmes, "A dynamic Bayesian framework to model context and memory in edit distance learning: An application to pronunciation classification," in *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2005.

[6] H. Lin, J. Bilmes, D. Vergyri, and K. Kirchhoff, "OOV detection by joint word/phone lattice alignment," *Automatic Speech Recognition & Understanding, 2007. ASRU. IEEE Workshop on*, pp. 478–483, 2007.

[7] K. Saenko and K. Livescu, "An Asynchronous DBN for Audio-Visual Speech Recognition," *IEEE Workshop on Spoken Language Technologies*, 2006.