EFFICIENT SPEECH INDEXING AND SEARCH FOR EMBEDDED DEVICES USING UNITERMS

Changxue Ma and Woojay Jeon

Applied Research and Technology Center
Motorola, Inc.
Schaumburg, IL, U.S.A.

ABSTRACT

In this paper, we present an efficient method of speech indexing and search using phoneme sequences called uniterms. In the indexing stage, a collection of uniterms and uniterm sequences is extracted from the target speech database by applying statistical scoring to each data item's phoneme lattice. In the search stage, each speech query's phoneme lattice is used to select candidate uniterms from the collection. These uniterms are applied in a speech recognition engine to convert the speech query into a uniterm lattice, from which we obtain a set of candidate uniterm sequences, each of which can be mapped to a search result item. Not only is this method a significant improvement over previous phoneme-based methods, it is shown that explicit sequential comparison of uniterms in query and target data can be avoided using the proposed method without loss of search performance. Avoiding sequential comparison allows better handling of transposition of words, and for the case where queries have word orders different from their intended targets, the proposed method can potentially bring about significant improvement.

Index Terms— speech indexing, speech search

1. INTRODUCTION

Embedded devices are ever increasing in functionality and stored content, creating demand for more advanced and innovative interaction methods. Speech-based search is a natural way for people to retrieve content such as personal contacts, photos, videos, and voice mail, as well as weather forecasts, news, and device-specific commands and bookmarks.

The purpose of speech search is to identify speech segments in a database that putatively match a given speech query. While one could achieve this by using automatic speech recognition (ASR) to convert speech to text and then applying text search, there is much interest in directly searching over phonemic lattice transcriptions, rather than words or sentences, in a vocabulary-independent, content-independent manner [1]. This "sememeless" method can be more appropriate for efficiently searching arbitrary, spontaneous speech containing unknown vocabulary without requiring accurate ASR. It is ever more feasible in resource-limited embedded devices that must practically handle all sorts of names, places, and foreign terms that cannot be easily covered by a phonemic dictionary.

For robust phoneme-based indexing of conversational speech, past studies have used multiple phoneme hypotheses rather than best paths to compare query with target [2, 1]. More precise comparison algorithms have also been proposed [3], but phonemic lattices still suffer from high phoneme recognition error rates as well as the loss of context information. A general method for indexing weighted automata has also been shown to give results comparative to subword methods [4]. One limitation of phoneme-based indexing and sequential matching is that it requires word sequences in queries to be more or less identical to those of their intended targets. However, a query such as "I went to San Francisco yesterday" should be regarded the same as "Yesterday, I went to San Francisco" in a practical speech search system.

Recently, we showed that speech indexing and search can be made more robust by lumping phoneme sequences into more discriminative units called "uniterms," [5]. In this paper, we exploit uniterms further by noting that they allow more flexible matching techniques which can better handle word order problems. We employ a statistical scoring method that uses local conditional probabilities between consecutive uniterms, instead of matching whole sequences via dynamic programming, thereby alleviating errors with "out-of-order" queries (word order different from intended target). With "in-order" queries (word order coincides with intended target), we show that there is no performance loss when using the proposed method. At the same time, we continue to maintain vocabulary-independence and content-independence because the uniterms are data driven. Our method is also computationally efficient and therefore suitable for application in embedded devices.

2. UNITERM-BASED INDEXING

The first step in the proposed method is to extract a set of uniterms and uniterm sequences from the target database.

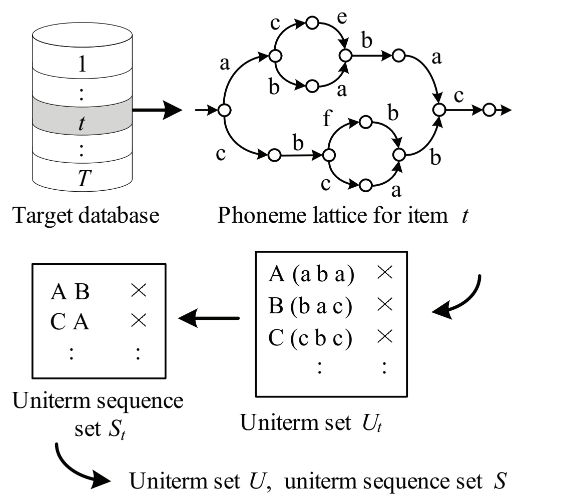

Fig. 1. Schematic of data indexing process. For the t-th target data item, a phoneme recognizer generates a phoneme (lowercase letters) lattice, from which a set U_t of uniterms (uppercase letters) is extracted. Uniterms with high scores are used to create a set S_t of uniterm sequences. $U_1, \cdots, U_T$ and $S_1, \cdots, S_T$ are consolidated into uniterm set U and uniterm sequence set S, respectively. "×" represents arbitrary scores.

Fig.1 shows the overall stages. From each speech recording item in the database, a phoneme lattice is created using an automatic speech recognizer with a phoneme loop grammar. From all paths in the lattice, all possible phoneme sequences of a fixed length (3 in the example) are extracted to create a list of uniterms. While a large number of phoneme strings can be extracted from a typical phoneme lattice, many can be erroneous or have little discriminative information. To extract those that are reliable, we apply a simple method of statistical scoring to find the strings that occur most frequently. To extract those that are discriminative, we make each uniterm long enough to contain a sufficient number of phonemes.

For a uniterm $\mathbf{u} = (x_1, x_2, \cdots, x_M)$ consisting of M phonemes $x_1, x_2, \cdots, x_M$, we define a score representing how likely the uniterm lies in a given phoneme lattice $\mathcal{L}$, as

$$S(\mathbf{u}) = \frac{1}{M} \log p\left(x_1, \cdots, x_M \middle| \mathcal{L}\right) + f\left(M\right) \qquad (1)$$

The conditional probability represents how likely the phoneme string $x_1, \cdots, x_M$ occurs in phoneme lattice $\mathcal{L}$, and the log probability is divided by M to normalize for length. A heuristic function $f(M)$ is also added to penalize short strings because longer strings are seen to be more discriminative. For example, $f\left(M\right) = b \log\left(M\right)$ where $b = 0.02$.

Since it is hard to estimate the conditional probability in (1), we approximate it using N-gram probabilities:

$$\begin{aligned} p\left(x_1, \cdots, x_M \middle| \mathcal{L}\right) &= \prod_{i=1}^{M} p\left(x_i \middle| x_1, \cdots, x_{i-1}, \mathcal{L}\right) \\ &\approx \prod_{i=1}^{M} p\left(x_i \middle| x_{i-N+1}, \cdots, x_{i-1}, \mathcal{L}\right) \end{aligned} \qquad (2)$$

The N-gram conditional probability $p(x_i | x_{i-N+1}, \cdots, x_{i-1}, \mathcal{L})$ represents the probability of a phoneme occurring given $N-1$ previously seen phonemes in $\mathcal{L}$. Setting $N = M$ would yield the most accurate computation of (2), but it is hard to reliably estimate the probabilities for large values of N, so we use some lower value. The simplest case of $N = 1$ uses only unigrams $p\left(x_i \middle| \mathcal{L}\right)$, while higher values can use more context via bigrams ($N = 2$), $p\left(x_i \middle| x_{i-1}, \mathcal{L}\right)$, or trigrams ($N = 3$), $p\left(x_i \middle| x_{i-1}, x_{i-2}, \mathcal{L}\right)$.

The N-gram probabilities are estimated by counting the occurrence of phoneme sequences in all possible paths in $\mathcal{L}$. To compensate for data sparsity, smoothing techniques are used to improve the measurements. For example,

$$\begin{aligned} p\left(x_i \left| x_{i-1}, x_{i-2}\right.\right) = \alpha \hat{p}\left(x_i \left| x_{i-1}, x_{i-2}\right.\right) + \\ \beta \hat{p}\left(x_i \left| x_{i-1}\right.\right) + \gamma \hat{p}\left(x_i\right) + \varepsilon \end{aligned} \qquad (3)$$

where α, β, γ, and ε are empirical constants under the constraint $\alpha + \beta + \gamma + \varepsilon = 1$.

After keeping only those uniterms that exceed some score threshold, a set of uniterm sequences is obtained by substituting uniterms for the paths in the phoneme lattice. The uniterms and uniterm sequences extracted from all data items are consolidated into a universal uniterm set U and uniterm sequence set S. An inverted index ensures that each item in S can be mapped to one or more target data items that contain the uniterm sequence.

3. UNITERM-BASED SEARCH

3.1. Selection of uniterms from a phoneme lattice

We now show how a search query is processed and matched against the target database. Fig.2 shows the overall search process. As we did for the target data items, a phoneme recognizer is used to convert a speech query into a phoneme lattice. Using the lattice, an initial set of uniterms L_1 is selected from the universal uniterm set U by scoring each uniterm according to the N-gram method described in Sec.2 and retaining only those uniterms whose scores exceed some threshold.

In the next stage, the set of uniterms extracted from the previous step in Sec.3.1 is further refined. Phoneme-level dynamic programming is used to measure the similarity between each uniterm and the best path of the phoneme lattice from which it was derived. Since each uniterm constitutes only a subset of the phoneme lattice, there will be insertion errors be-

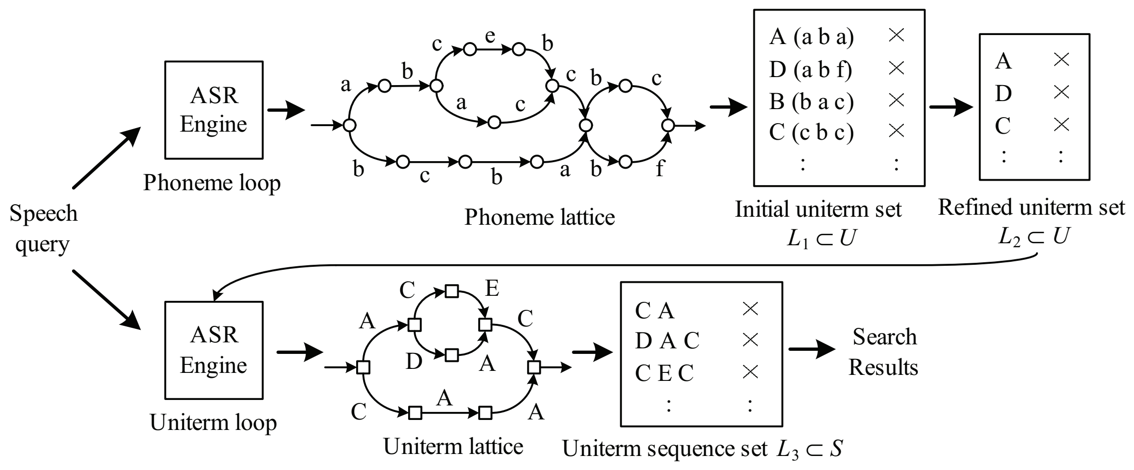

Fig. 2. Schematic of data search process. Using the query phoneme lattice, a set of uniterms is selected from the universal set U via statistical scoring and then refined via dynamic programming. The set of uniterms L_2 is used with the ASR engine to obtain a uniterm lattice, which is then used to select a set L_3 of uniterm sequences from the universal set S. Each sequence can be mapped to one or more target database items to obtain the final search results. The lattices in the figure are arbitrary examples.

fore or after the uniterm. If all uniterms are the same length, however, this becomes a constant bias that can be ignored.

Phoneme duration and score (log of the phoneme's conditional probability) are incorporated in the calculation of the cost function for the dynamic programming. The cost V of a path from the start node at time t_0 to end node at time t_1 is

$$V = \sum_k l\left(k, t_0, t_1\right) \tag{4}$$

where $l\left(k, t_0, t_1\right)$ denotes the cost function of the k-th edge (phoneme) in the path, defined as

$$l\left(k, t_0, t_1\right) = \max \begin{cases} s_k \cdot const\_a & \text{(equal)} \\ \left(t_1 - t_0\right) \cdot const\_b & \text{(substitution)} \\ const\_c & \text{(insertion)} \\ const\_c & \text{(deletion)} \end{cases} \tag{5}$$

Here, s_k is the score of the kth phoneme and $const\_a$, $const\_b$, and $const\_c$ are empirical scaling factors. In our experiments, we choose 1/5, -100, and -3000, respectively. Higher cost indicates greater similarity.

3.2. Fine search using uniterm lattices

In the final stage, a variety of search strategies are possible using the refined list of candidate uniterms and scores. The list itself may already be considered a "coarse search" result, and an inverted index can be used to retrieve the database items containing the uniterms with high scores. A more effective method is to apply speech recognition once again to the utterance, but this time using the selected uniterms as components of a loop grammar to obtain a *uniterm lattice* $\mathcal{L}'$. Using $\mathcal{L}'$, we can select a set of uniterm sequences from the universal set S by computing smoothed trigram conditional probabilities $p\left(\mathbf{u}_i \middle| \mathbf{u}_{i-1}, \mathbf{u}_{i-2}, \mathcal{L}'\right)$ in a manner similar to Sec.2, where $\mathbf{u}_i$ is a uniterm. Since each uniterm sequence can be immediately mapped to one or more target data items that contain the sequence, the sequences with high scores give us our final "fine search" results.

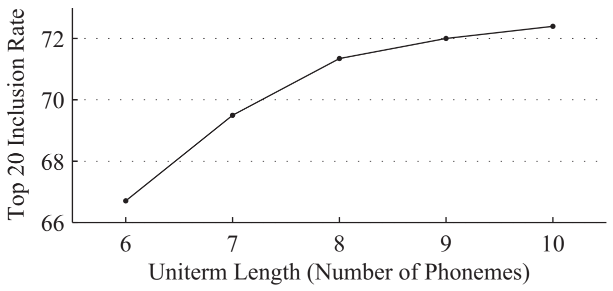

Fig. 3. Inclusion rate (recall) of coarse search for $n = 20$ in equation (6) as a function of phoneme string length

4. EXPERIMENT AND DISCUSSION

We use the ETSI advanced frontend standard for distributed speech recognition, which generates feature vectors of 39 dimensions per frame (12 MFCC plus energy, delta, and acceleration coefficients). The speech recognizer is MLite++, a Motorola proprietary HMM-based ASR engine for embedded platforms. The engine uses both context-independent (CI) and context-dependent (CD) subword HMMs trained on a large speaker-independent American English database. From each utterance, around 50 uniterms, each with a fixed length

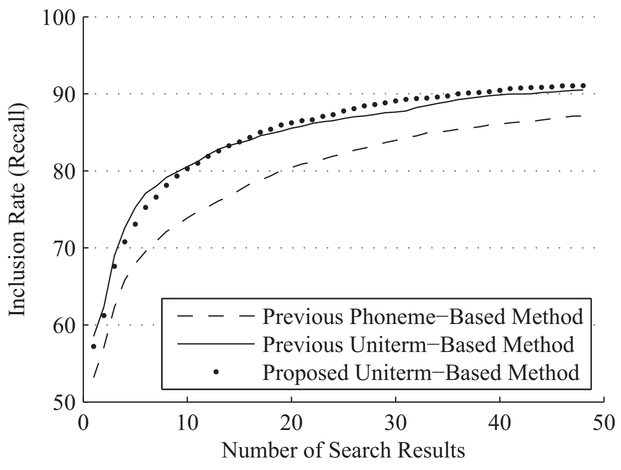

Fig. 4. Results of fine search (with "in-order" queries) for varying number of search results n in equation (6). The proposed unitem-based method has similar performance compared to the previous uniterm-based method and significantly better performance than the phoneme-based method.

of 8 phonemes, are extracted.

Experiments were carried out on an audio database consisting of 1,156 utterances from six speakers. The content text is chosen from a wide range of song titles. For each utterance in the database, two to three other utterances have identical content but are from different speakers. The system performance is measured by how well the system, given an utterance from the database as a query, can match this utterance to the other utterances with identical content. The experimental setup conducted for this paper was identical to that of the previous study [5], with the word orders of queries coinciding with the word orders of intended targets.

The inclusion rate (recall rate) is defined as

$$\text{Inclusion Rate} = \frac{TP_n}{TP_n + FN_n} \qquad (6)$$

where TP_n stands for the number of true positives, FN_n stands for the number of false negatives, and n denotes the number of search results returned.

Fig.3 shows the inclusion rate for the coarse search with varying uniterm length and $n = 20$. As previously mentioned[5], long uniterms (like long queries in text search) can provide more discriminative power in general, but can also cause relevant results to be missed: first, because they can make the system more vulnerable to phoneme recognition errors, and second, because they can become too narrow in scope and do not generalize well. In Fig.3, the performance improves linearly from length 6 to 8, but plateaus around 9 and 10 for our test data set. For even longer lengths, we expect the performance to drop. Also note that uniterms do not necessarily need to be shorter than the average word (5 or 6 phonemes) because partial matches are possible when performing uniterm recognition in the fine search stage.

Fig.4 shows the inclusion rate for varying values of n and fixed uniterm length. Consistent with previous findings [5], the uniterm-based methods perform significantly better than the phoneme-based method. We also see that there is no degradation of performance when we apply the proposed method in this experiment, where the word order of queries are consistent with those of the intended targets, compared to the previous uniterm-based method that relied on explicit sequential matching via dynamic programming. At the same time, preliminary experiments using queries with out-of-order words have indicated a significant increase in performance when using the proposed method.

5. CONCLUSION AND FUTURE WORK

In this paper we presented a speech indexing and search scheme for the fast retrieval of speech segments on embedded devices using phoneme sequences called uniterms. By using uniterms, we showed that speech search performance can be significantly improved over previous phoneme-based methods. Furthermore, by applying a combination of dynamic programming and statistical scoring, we showed that explicit sequential comparison of subword lattice paths via dynamic programming can be avoided without loss of search performance. This is encouraging because for the case where speech queries contain word orders different from their intended targets, this will make the new matching method significantly more effective. Extensive experimental results for this case scenario will be reported in the future.

6. REFERENCES

[1] D. A. James and S. J. Young, "A fast lattice-based approach to vocabulary independent wordspotting," in *IEEE Int. Conf. Acoust., Speech. Signal Processing*, 1994.

[2] K. Ng and V. W. Zue, "Subword-based approaches for spoken document retrieval," *Speech Communication*, vol. 32, no. 3, pp. 157–186, 2000.

[3] O. Siohan and M. Bacchiani, "Fast vocabulary-independent audio search using path-based graph index," in *Proc. INTERSPEECH*, 2005.

[4] C. Allauzen, M. Mohri, and M. Saraclar, "General indexation of weighted automata – application to spoken utterance retrieval," in *Proc. HLT/NAACL*, 2004, pp. 33–40.

[5] C. Ma, "Uniterm voice indexing and search for mobile devices," in *IEEE Int. Workshop on Multimedia Analysis and Proc.*, 2008.