A Comparison of Query-by-Example Methods for Spoken Term Detection

Wade Shen, Christopher M. White and Timothy J. Hazen

MIT/Lincoln Laboratory
244 Wood St.
Lexington, MA 02420
{swade,hazen}@ll.mit.edu, cmileswhite@jhu.edu

Abstract

In this paper we examine an alternative interface for phonetic search, namely query-by-example, that avoids OOV issues associated with both standard word-based and phonetic search methods. We develop three methods that compare query lattices derived from example audio against a standard ngram-based phonetic index and we analyze factors affecting the performance of these systems. We show that the best systems under this paradigm are able to achieve 77% precision when retrieving utterances from conversational telephone speech and returning 10 results from a single query (performance that is better than a similar dictionary-based approach) suggesting significant utility for applications requiring high precision. We also show that these systems can be further improved using relevance feedback: By incorporating four additional queries the precision of the best system can be improved by 13.7% relative. Our systems perform well despite high phone recognition error rates ($> 40\%$) and make use of no pronunciation or letter-to-sound resources.

Index Terms: spoken term detection, string-edit distance, keyword spotting, spoken document retrieval

1. Introduction

As the volume of recorded speech data has increased through sources such as podcasts, internet audio, etc., the need for technologies that allow fast access to this data based on content has also grown. Research in content-based retrieval methods from spoken data has a long tradition in both the information retrieval and speech recognition communities. A number of content-based retrieval methods have been explored including topic detection/tracking, spoken term detection, spoken document retrieval, etc. This research has been supported by multiple evaluations including the recent 2006 NIST Spoken Term Detection (STD) evaluation.

In this paper we focus on the retrieval of spoken utterances containing search terms specifically in context of low-resource spoken term detection. Much of the work done to date has focused on spoken term detection in languages and domains where transcribed speech and phonetic lexicon resources are widely available. As such, the best current methods make heavy use of word-based speech recognition during the indexing process. These systems assume that 1) well-trained recognizers are available for the STD domain/language of interest and that 2) the search vocabulary is likely to be well covered by the language models used during indexing (i.e. low OOV for query terms).

Many state of the art systems also make use of phonetic search, especially when language resources are scarce. As phonetic recognition is typically more error prone during indexing, soft-matching procedures are needed to account for these errors during search. These systems typically assume that at query time, an orthographic representation of the search term can be converted into a phonetic representation for comparison against an index using either pronunciation lexica or grapheme-to-phoneme conversion algorithms. Such systems can be subject to OOV issues as grapheme-to-phoneme conversion is not always available and query terms may not be found in pronunciation lexica.

In this work we examine an alternative interface for phonetic search, namely query-by-example, that avoids OOV issues associated with both standard word-based and phonetic search methods. We assume that a user provides query examples either from excised speech cuts corresponding to a search term or via a speech recording interface. The resulting audio is used to then create a query lattice for comparison against a standard phonetic index. We compare three methods for the retrieval of utterances from ngram-based phonetic indices given a query lattice and we analyze factors affecting the performance of these systems. We show that the best systems under this paradigm are able to achieve 77% precision on conversational telephone speech when returning 10 results (performance that is better than a similar dictionary-based approach), suggesting significant utility for applications requiring high precision. We also show that the system can be further improved using relevance feedback: By incorporating four additional queries the precision of the best system can be improved by 13.7% relative. Our systems perform well despite high phone recognition error rates ($> 40\%$) and make use of no pronunciation or letter-to-sound resources.

We treat the problem as a 'string-distance comparison' to leverage and improve existing stochastic string-distance methods [1, 2, 3]. We develop two types of algorithms to calculate confusion network to confusion network (lattice to lattice) distance based on 1-1 alignments and string-edit distance. The strength of the former is the ease of implementation and speed, while the latter can cope with misalignments by explicitly modeling insertions and deletions.

1.1. Related Work

Significant work has been done on the use of phonetic units for content-based retrieval from speech (e.g. [4, 5, 6]). Though our work is closely related to these efforts, we do not use pronunciations derived from pronunciation lexica or letter-to-sound sys-

This work was sponsored by the Department of Defense under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

tems. Furthermore, we make use of a lattice of possible alternatives from both the query term and search indices. This aspect of our matching procedure is related to Spoken Document Retrieval (SDR) work done by [7] and [8], suggesting that lattice representations of search indices and queries can improve SDR performance.

Less work has been done involving methods for speech search by example. In [9], the authors describe a method for example-based query generation for general search. In related music retrieval work, [10] attempts to retrieve drum loops from onomatopoeic queries.

2. Algorithms for Query-to-Index Matching

In this section we describe three different approaches to the query-by-example search problem. All systems assume that a compact phone ngram index is used. The form of this index is essentially a pruned confusion network as shown in Figure 1. In our particular implementation, we keep epsilon arc information for use during term matching. This representation allows very compact indexing of large data sets.

Times:	0.0 - 0.05		0.05 - 0.08		0.08 - 0.16	
Phones	`ax`	0.8	`l`	0.7	`ao`	0.8
	`EPS`	0.2	`r`	0.2	`ow`	0.1
			`EPS`	0.1	`EPS`	0.1

Figure 1: *Example 1-gram phonetic index for the term* `allow`

All three models presented below can be viewed as stochastic measures of string-to-string distance/similarity. In [1] and [3] cost parameters are learned from supervised data for either HMM or CRF models, while in our case, these costs are fixed as our systems, by design, make no use of supervised data other than the query. As in [2], we compare the performance of different alignment procedures for scoring a query lattice against a large indexed database of speech. Our alignment models vary from 1) fixed 1-1 alignment, to 2) Viterbi string edit distance, to 3) full Forward probability. As in [2] and [4] our systems make use of term weighting (either IDF or a background model) to improve scoring.

2.1. Direct Index Matching

Our first system assumes queries and indices do not require alignment during the matching procedure. At query time, we convert the query lattice into an ngram index (order 2) as described above. The resulting query structure is then compared against indexed speech of the same form column-by-column in sliding windows over all utterances. Scores are computed for each sliding window. This system is inspired by [4] though it differs from this approach in that we enforce a 1-1 query-to-index alignment scheme in place of the bag-of-ngrams model. Also, empirically, we found it better to use a likelihood ratio in place IDF weighting. Note: this procedure does not allow for insertions or deletions. Equation 1 shows the scoring function for used for each column.

$$p(\mathbf{I}_j|\mathbf{Q}_i) \quad = \quad \sum_{p\in \mathbf{I}_j} E(p|\mathbf{I}_j) * p(p|\mathbf{Q}_i) \qquad (1)$$

where $\mathbf{Q}_i$ and $\mathbf{I}_j$ represent the set of ngram posteriors associated with columns i and j of the query and index respectively and p is a phone ngram in both $\mathbf{Q}_i$ and $\mathbf{I}_j$. $E(p|\mathbf{I}_j)$ represents the expected value of a phone ngram p in index $\mathbf{I}_j$.

For each window with index offset w, we compute the joint sequence probability:

$$p(\mathbf{I}|\mathbf{Q}, w) = \prod_{i=1}^{|\mathbf{Q}|} p(\mathbf{I_{i+w}}|\mathbf{Q_i}) \qquad (2)$$

For detection purposes, we compute the likelihood ratio of joint sequence probability against a background model:

$$\begin{aligned} LR(\mathbf{Q}|w,\mathbf{I}) &= \frac{p(\mathbf{I}|\mathbf{Q},w)}{p(\mathbf{I}|\lambda_{bkg},w)} && (3)\\ p(\mathbf{I}|w,\lambda_{bkg}) &= \prod_{i=1}^{|\mathbf{Q}|} p(\mathbf{I}_{i+w}|\lambda_{bkg}) && (4)\\ &= \prod_{i=1}^{|\mathbf{Q}|} \sum_{p\in\mathbf{I}_{i+w}} E(p|\mathbf{I}_{i+w}) * p(p|\lambda_{bkg}) && (5) \end{aligned}$$

We estimate the background model parameters $p(p|\lambda_{bkg})$ from recognizer transcripts on a 100+ hour unlabeled set of Fisher data.

2.2. Edit Distance Alignment System

Consider two ngram indices $\mathbf{Q}$ and $\mathbf{I}$, a query and an indexed utterance respectively. A column i in such an index defines a distribution $p(p|\mathbf{I}_i)$ over symbols p (phones) drawn from an alphabet A. We would like to determine whether $\mathbf{I}$ contains an instance of $\mathbf{Q}$. To do this we will compute a string-edit distance in which we explicitly model insertions and deletions of index columns.

We consider the string-edit distance between strings of symbols in A to be defined by the tuple $\phi =< A, p(E) >$, where $p(E)$ is a probability distribution over edits $p : E \rightarrow [0,1]$, with $E = E_s \cup E_d \cup E_i$ ($E_s = A \times A$ set of substitutions, $E_d = A \times \{\epsilon\}$ set of deletions, $E_i = \{\epsilon\} \times A$ set of insertions).

Let $\mathbf{z}_N$ be an N-length sequence of edit operations $z_i \in E$ required to transduce any path through a query index $\mathbf{Q}$ into any path though confusion network $\mathbf{I}$, with $z_i =< \mathbf{Q}_j, \mathbf{I}_k >$ the edit event of transducing column $\mathbf{Q}_j$ into $\mathbf{I}_k$, and $p(\mathbf{z}_N|\phi)$ the product of the probabilities $p(z_i)$ of the specific edit operations.

This formulation naturally leads to two distances: $d_\phi^{Vit}(\mathbf{Q},\mathbf{I})$ calculates the distance between two confusion networks as the mostly likely transduction between any path in $\mathbf{Q}$ with any path in $\mathbf{I}$, and $d_\phi^{Fwd}(\mathbf{Q},\mathbf{I})$ sums over all possible transductions between all members of $\mathbf{Q}$ and $\mathbf{I}$.

The first distance $d_\phi^{Vit}(\mathbf{Q},\mathbf{I})$, a generalization of the Viterbi path, is the negative logarithm of the probability of the most likely edit sequence between any path through $\mathbf{Q}$ into any path through $\mathbf{I}$:

$$d_\phi^{Vit}(\mathbf{Q},\mathbf{I}) = -log(max_{\mathbf{z}_N} p(\mathbf{z}_N|\phi)) \qquad (6)$$

$$p(\mathbf{z}_N|\phi) = \Pi_{i=0}^{N} p(z_i) \qquad (7)$$

$$p(z_i) = \sum_{\forall q\in\mathbf{Q}_j} \sum_{\forall p\in\mathbf{I}_k} \frac{1}{2}p(p|q)p(p|\mathbf{I}_k) + \frac{1}{2}p(q|p)p(q|\mathbf{Q}_j) \quad (8)$$

where p and q are phone ngrams within columns $\mathbf{I}_k$ and $\mathbf{Q}_j$ respectively and $p(p|\mathbf{I}_k)$-like terms taken from the posterior distribution of the confusion network at column k. While the $p(q|p)$-like terms could be learned, we opt for the unsupervised standard edit-distance of $p(\cdot|\cdot)$ as 1 for a match, and 0 for substitutions, insertions, and deletions. Also, note that for implementation the networks were converted into bigram networks, while using the above equations for each unigram in a column.

2.2.1. Hybrid distance

In [2] the variants of edit-distance performed best when combined with TF-IDF. Here we combine the two by using the alignment generated from $d_\phi^{Vit}(\mathbf{Q}, \mathbf{I})$, while weighting each of the bigram matches with the IDF_p factor, and sum the score in addition to a unigram match score.

2.3. HMM Alignment System

In addition to an edit distance-based alignment, we employed a discrete HMM model for matching. In this framework, we treat the query as a discrete HMM in which each column of the ngram index is interpreted as an HMM state and unigram probabilities are used as observation probabilities. Epsilon arcs in the index structure are interpreted as skip arc probabilities and a fixed stay probability parameter allows for insertions to be observed in indexed utterances given the query as a model.

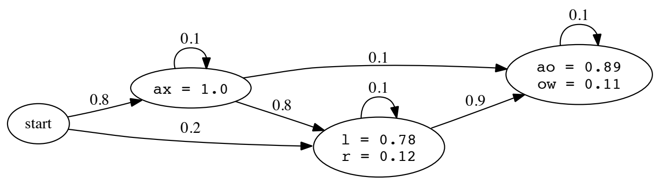

Figure 2: *Example HMM with fixed stay probability of 0.1*

Figure 2 shows an example of the resulting query/model corresponding to the index shown in Figure 1. During search, the discrete HMM model of the query is aligned to the index to compute:

$$p(\mathbf{I}_1^k|\lambda_Q) = \sum_{\forall \mathbf{s}} \prod_{i=1}^{k} p(s_{i-1}|s_i) p(\mathbf{I}_i|s_i) \quad (9)$$

$$= \sum_{\forall \mathbf{s}} \prod_{i=1}^{k} p(s_{i-1}|s_i) \sum_{p \in \mathbf{I}_i} p(p|s_i) * E(p|\mathbf{I}_i) \quad (10)$$

The $E(p|\mathbf{I}_i)$ term in the above equation uses the posterior of each phone p in each index column i. This formulation allows for full alignment of all index alternatives. Scoring is done by computing full Forward probability of the index I given the query model λ_Q. Note: with fixed transition, skip, and stay probabilities, this model computes d_ϕ^{fwd} as described in the previous section. As we make use of per state skip information derived from the query index, these models are not fully equivalent.

3. Experimental Design

We evaluated the ability of the methods proposed above to retrieve utterances containing query terms on conversational telephone speech data from the Fisher corpus and we compared these methods to a retrieval system that makes use of a pronunciation lexicon as a baseline. From transcripts provided by LDC we extracted 248 single-word terms with an average of 19.2 target trials per term (minimum: 10, maximum: 500). An additional set of 967 utterances were also extracted as non-target trials.

For each term we constructed a held-out set of 1,843 query examples (minimum: 1, maximum: 10, average: 7.4 examples per term) extracted from continuous speech utterances. Words were selected from the STD 2006 Evaluation term list with a minimum of 5 orthographic characters. This was done in order to avoid sampling of unlikely search terms (e.g. short function words).

For the purpose of these experiments each trial is a single utterance (between 0.5-6 secs) and we measured the ability of each system to return utterances containing each target term. We evaluated performance in terms of precision at N, where N is the number of target trials for that word. This measures the operating point where precision is equal to recall. For example, the term `actor` has 22 target and 967 non-target trials. In this case, we measure the performance of a detector for this term by sorting its scores, and report precision for the top 22 scoring hits. We also report precision at 10, the typical Google metric, as well as the Equal-Error Rate (EER). In all cases, the performance per term is averaged.

We measure the performance of each system using single and multiple query examples. Performance using multiple query examples was examined in order to assess how these systems could perform with user feedback, i.e. using a preliminary query example and its search results, users might expand the set of examples corresponding to a given query.

Having many spoken queries for some words invites a method to combine them, or the scores resulting from their use. For systems making use of multiple queries we employed either a max or avg score criterion for combining results on the utterance level.

For all experiments phonetic indices were created by running a neural network-based TRAPs phonetic recognizer trained on 10 hours of Switchboard2, phase 4 data. This recognizer was also used to generate models from query examples using the aforementioned methods.

4. Results

Table 1 shows results for all methods at the three operating points described above on our Fisher test set using single and multiple query examples. The best method for combining multiple query returns varies for different systems. We observe that the use of multiple queries improves system precision by 13.7% @ 10. This suggests that users would be able to provide relevance feedback to these systems for improved performance.

Overall all systems perform comparably with the discrete HMM model being slightly better at most operating points. All systems benefit from the use of IDF term weighting and/or log likelihood ratio (llr) scoring against a background model.

This table also compares the result of the best system (dhmm) using queries constructed from examples and queries from dictionary pronunciations. Query-by-example outperforms the pronunciation-based model when a single query or pronunciation is used. This suggests that the underlying phonetic recognizer is making consistent “errors” between the index and the query. Few query terms have more than one pronunciation (avg. 1.1 prons. per term), as a result, there is little improvement when all pronunciations are combined.

We examined the performance of our best system as a function of the length of the input query. Figure 3 plots each precision metric as a function of query length (in characters). Although the variability in performance at specific queries lengths is high, generally, performance improves at all operating points as a function of the query length. Note that for very short queries, system performance is highly affected by indexing errors associated with missed columns during query extraction. This is due to the fact that we derive only approximate timing information from the ASR phone lattice during confusion

System	Configuration	P@10	P@N	EER
direct	*one query*	70.66	49.95	20.57
	one query (idf)	73.65	51.90	19.70
	one query (llr)	74.51	52.29	**19.53**
	all queries (avg+llr)	79.15	58.84	14.84
	all queries (max+llr)	75.44	55.23	16.06
d_{ϕ}^{vit}	*one query*	63.44	44.83	23.78
	one query (idf)	69.41	48.18	22.01
	all queries (avg+idf)	80.85	58.31	15.08
	all queries (max+idf)	77.74	54.82	16.57
dhmm	*one query*	74.83	53.04	20.80
	one query (llr)	**76.99**	**53.94**	20.48
	all queries (avg+llr)	81.17	54.92	18.04
	all queries (max+llr)	**82.26**	**61.34**	**13.98**
dhmm + pron lex.	*one dict entry (llr)*	73.01	47.66	21.11
	all dict entries (avg+llr)	73.99	48.16	20.92
	all dict entries (max+llr)	74.27	48.26	20.93

Table 1: *Performance of each QbE system (aggregation and background model indicated in parenthesis, best single and multi-query results are indicated in* **bold***)*

network construction. Table 2 shows the 1-best posterior decoding results for multiple query instances of the word `allow`. As shown in this example, extraneous and missing columns are common due to timing issues. From manual inspection, we found that timing issues dominate differences in the number of indexed columns extracted for different queries of the same term rather than deletion or insertion errors from the recognizer or pronunciation differences.

1-best indices for `allow`	`AX L AW`	`L UH`
	`AY L AE T`	`N AX L OW`
	`L`	

Table 2: *Examples of indexing errors for the term* `allow`

5. Discussion

All of the methods we present in this paper show promising performance on the utterance retrieval task. Our results suggest that making use of lattice-to-lattice alignments between queries and indices can improve STD performance. Similarly, term weighting either through IDF or through the use of a background model can provide significant performance improvements.

Performance of these systems is significantly influenced by the length of query. Generally, longer queries yield better results. Our experiments also show that use of relevance feedback can further improve performance by as much as 13% relative even using rather naive combination methods. It should be possible to allow the HMM-based system to create aggregate models via query alignment. Further work is needed to assess if this improves performance when multiple queries are available.

In terms of computational cost, the baseline direct matcher is very fast as ngrams can accessed in constant time. The other two systems make use of sequence alignment. With our implementation of the discrete hmm, alignment can be done in 270% xRT (faster) and could be made faster by first applying an direct match first to select regions for alignment-based matching.

These systems perform well despite poor phone recognition and indexing errors associated with approximate confusion network timing. We expect that better indexing methods could be employed to further improve these systems.

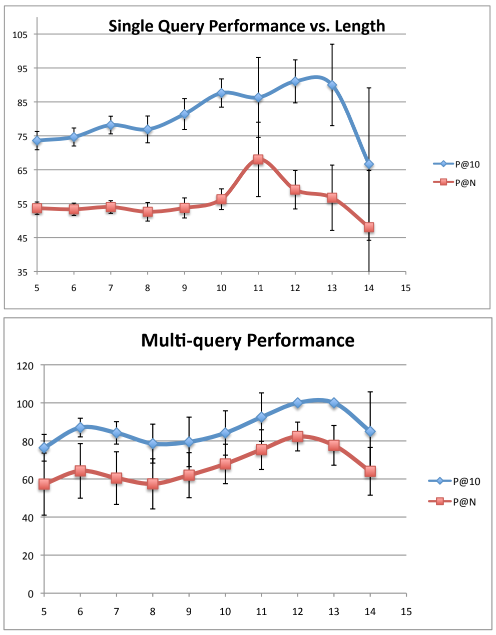

Figure 3: Precision vs. length for single and multiple queries

5.1. Acknowledgments

We would like to thank Ken Basye and Hugh Secker-Walker for their lattice-to-lattice distance and general purpose graph tools.

6. References

[1] E. Ristad and P. Yianilos, “Learning string-edit distance,” *IEEE Trans. on Pattern Analysis Machine Intelligence*, 1998.

[2] W.W. Cohen, P. Ravikumar, and S.E. Fienberg, “A comparison of string distance metrics for name-matching tasks,” in *IJCAI*, 2003, pp. 73–78.

[3] A. McCallum, K. Bellare, and F. Pereira, “A conditional random field for discriminatively-trained finite-state string edit distance,” in *Conference on Uncertainty in AI (UAI)*, 2005.

[4] K. Ng and V. Zue, “Subword-based approaches for spoken document retrieval,” in *PhD. Thesis*, 2000.

[5] L. Burget et al., “Indexing and search methods for spoken documents,” in *ICTSD*, 2006.

[6] T. Hori, I. Lee Hetherington, Timothy J. Hazen, and J. Glass, “Open-vocabulary spoken utterance retrieval using confusion networks,” in *ICASSP*, 2007.

[7] M. Saraclar and R. Sproat, “Lattice-based search for spoken utterance retrieval,” *HLT-NAACL*, 2004.

[8] T. Kiah Chia, K. Chai Sim, Haizhou L., and H. Tou Ng, “A lattice-based approach to query-by-example spoken document retrieval,” in *SIGIR*, 2008, pp. 363–370.

[9] H. Murao, N. Kawaguchi, S. Matsubara, and Y. Inagaki, “Example-based query generation for spontaneous speech,” *IEICE - Trans. Inf. Syst.*, 2005.

[10] O. Gillet and G. Richard, “Drum loops retrieval from spoken queries,” *Journal of Intelligent Information Systems*, 2005.