ABSTRACT

Today, most systems use large vocabulary continuous speech recognition tools to produce word transcripts which have indexed transcripts and query terms retrieved from the index. However, query terms that are not part of the recognizer’s vocabulary cannot be retrieved, thereby affecting the recall of the search.

Such terms can be retrieved using phonetic search methods. Phonetic transcripts can be generated by expanding the word transcripts into phones using the baseforms in the dictionary. In addition, advanced systems can provide phonetic transcripts using sub-word based language models. However, these phonetic transcripts suffer from inaccuracy and do not provide a good alternative to word transcripts.

We demonstrate how to retrieve information from speech data by presenting a novel approach for vocabulary independent retrieval combining search on transcripts that are produced according to different word and sub-word decoding methods. We present two different algorithms: the first is based on the Threshold Algorithm (TA); the second uses a Boolean retrieval model on inverted indices.

The value of this combination is demonstrated on data from NIST 2006 Spoken Term Detection evaluation.

Categories and Subject Descriptors
H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval

General Terms
Algorithms

Keywords
Spoken Information Retrieval, Out-Of-Vocabulary, Phonetic Search, Threshold Algorithm.

1. INTRODUCTION

The rapidly increasing volumes of spoken data have brought about a need for solutions to index and search this data.

The classical approach consists of converting speech to word transcripts using large vocabulary continuous speech recognition (LVCSR) tools and extending classical Information Retrieval (IR) techniques to word transcripts.

A significant drawback of this approach is that search on queries containing Out-Of-Vocabulary (OOV) terms do not return any results. OOV terms are words missing in the automatic speech recognition (ASR) system vocabulary. These words are replaced in the output transcript by alternatives that are probable, given the acoustic and language models of the ASR. In previous experiments, Logan et al. [9] that over 10% of user queries can contain OOV terms, as queries often relate to named entities that typically have poor coverage in the ASR vocabulary. The effects of OOV query terms in spoken data retrieval are discussed by Woodland et al. [25]. In many applications, the OOV rate may worsen over time unless the recognizer’s vocabulary is periodically updated.

One approach for solving the OOV issue consists of converting the speech to phonetic transcripts and representing the query as a sequence of phones. Such transcripts can be generated by expanding the word transcripts into phones using the pronunciation dictionary of the ASR system. Another approach is to use subword-based language models that include phones, syllables, or word-fragments. The retrieval is based on searching the sequence of subwords representing the query in the subword transcripts. Some of these works were done in the framework of the NIST TREC Spoken Document Retrieval tracks in the 1990s and are described by Garofolo et al. [6].

The contribution of the paper is twofold. First, we present a novel method for vocabulary independent retrieval by merging search results obtained from both word and phonetic transcripts using the idea of the Threshold Algorithm presented by Fagin et al. [5]. Though this algorithm is widely used for merging multiple search results, it has never been used for merging word and phonetic search in the context of speech retrieval. Second, we propose a novel efficient combination of search results in the context of speech retrieval.
using Boolean constraints on inverted indices. We analyze the retrieval effectiveness of this approach on the NIST 2006 Spoken Term Detection (STD) evaluation data.\footnote{http://www.nist.gov/speech/tests/std/2006/index.html}

The paper is organized as follows. We describe the audio processing in Section 2. The indexing and retrieval models are presented in Section 3. Experimental results are given in Section 4. In Section 5, we give an overview of related work, and Section 6 contains our conclusions.

2. AUTOMATIC SPEECH RECOGNITION SYSTEM

This section presents different techniques for generating transcripts of broadcast news data.

2.1 Word Decoding

We used an ASR system in speaker-independent mode to transcribe speech data. For best recognition results, an acoustic model and a language model were trained in advance on data with similar characteristics.

The ASR system is a speaker-adapted, English Broadcast News recognition system that uses discriminative training techniques. The speaker-independent and speaker-adapted models share a common alphabet of 6000 quipphone context-dependent states, and both models use 250K Gaussian mixture components. The speaker-independent system uses 40-dimensional recognition features computed via an LDA-MLLT projection of nine spaced frames of 19-d PLP features normalized with utterance-based cepstral mean subtraction, as presented by Saon et al. [18]. The speaker independent models were trained using the MMI criterion. The speaker-adapted system uses 40-dimensional recognition features computed via an LDA-MLLT projection of nine spaced frames of 19-d PLP features normalized with VTLN and speaker-based cepstral mean subtraction and cepstral variance normalization. Our implementation of VTLN uses a family of piecewise-linear warpings of the frequency axis prior to the Mel binning in the front-end feature extraction. The recognition features were further normalized with a speaker-dependent FMLLR transform as presented by Saon et al. [19]. The speaker-adapted models were trained using FMPE and MPE, which is a feature-space transform that is trained to maximize the MPE objective function. The FMPE transform operates by projecting from a very high-dimensional, sparse feature space derived from Gaussian posteriors to the normal recognition feature space and adding the projected posteriors to the standard LDA-MLLT feature vector as described by Povey et al. [16, 15]. Both supervised and lightly supervised training were used, depending on the available transcripts.

We used the ASR system described above and the static decoding graphs to produce lattices and subsequently word-mixture networks (WCNs) as described by Mangu et al. [12] and Hakkani-Tur and Riccardi [7]. Each edge \((u, v)\) was labeled with a word hypothesis and its posterior probability, i.e., the probability of the word given the signal. One of the main advantages of WCN is that it also provides an alignment for all of the words in the lattice. As stated by Mangu et al. [12], although WCNs are more compact than word lattices, in general, the 1-best path obtained from WCN has a better word accuracy than the 1-best path obtained from the corresponding word lattice.

We converted the 1-best path of the generated word transcript to its phonetic representation using the pronunciation dictionary of the ASR system.

2.2 Word-Fragment Decoding

We also generated a 1-best phonetic output using a word-fragment decoder, where word-fragments were defined as variable-length sequences of phonemes as described by Siohan and Bacchiani [22]. This method of generating word-fragment was explored by Mamou et al. [11] and used in the NIST STD 2006 Evaluation. The word-fragment decoder generates 1-best word-fragments, that are then converted into the corresponding phonetic strings. We believe that the use of word-fragments can lead to better phone accuracy compared to a phone-based decoder because of the strong constraints that it implies on the phonetic string.

Our word-fragment dictionary was created by building a phone-based language model and pruning it to keep non-redundant higher order N-grams. The list of the resulting phone N-grams was then used as word-fragments. For example, assuming that a 5-gram phone language model is built, the entire list of phone 1-gram, 2-gram, . . . , 5-gram was used as the word-fragment vocabulary. The language model pruning threshold was used to control the total number of phone N-grams, hence the size of the word-fragment vocabulary. Given the word-fragment inventory, we could easily derive a word fragment lexicon. The language model training data was then tokenized in terms of word-fragments, and an N-gram word-fragment language model could be built.

2.3 ASR Training data

We used a small language model, pruned to 3M n-grams, to build a static decoding graph for decoding. We used a larger language model, pruned to 30M n-grams, in a final lattice rescoring pass as described by Stolcke [23, 24]. Both language models use up to 4-gram context, and were smoothed using modified Kneser-Ney smoothing. The recognition lexicon contained 67K words with a total of 72K variants.

The language model was trained on a 198M word corpus comprising the 1996 English Broadcast News Transcripts (LDC97T22), the 1997 English Broadcast News Transcripts (LDC98T28), the 1996 CSR Hub4 Language Model data (LDC98T31), the EARS BN03 English closed captions, the English portion of the TDT4 Multilingual Text and Annota-
tions (LDC2005T16), and the English closed captions from the GALE Y1Q1 (LDC2006E82) and Y1Q2 (LDC2006E33) data releases.

For word-fragment decoding, we used the same training data to generate the word-fragment language model. We used a total of 20K word-fragments, where each fragment was up to 5-phones long and used a 3-gram word-fragment language model.

The acoustic models were trained on a 430-hour corpus comprising the 1996 English Broadcast News Speech collection (LDC97S44), the 1997 English Broadcast News Speech collection (LDC98S71), and English data from the TDT4 Multilingual Broadcast News Speech Corpus (LDC2005S11). Lightly supervised training was used for the TDT4 audio because only closed captions were available.

3. INDEXING AND RETRIEVAL

The main difficulty with retrieving information from spoken data is the low accuracy of the transcription, particu-
larly on terms of interest such as named entities and content words. Generally, the accuracy of a transcript is measured by its word error rate (WER), which is characterized by the number of substitutions, deletions, and insertions with respect to the correct audio transcript. Substitutions and deletions reflect that an occurrence of a term in the speech signal has not been recognized. These misses reduce the recall of the search. Substitutions and insertions reflect that a term that is not part of the speech signal has appeared in the transcript. These misses reduce the precision of the search.

Search recall can be enhanced by expanding the transcript with extra words. These words can be taken from the other alternatives provided by the WCN; these alternatives may have been spoken but were not the top choice of the ASR. Such an expansion tends to correct the substitutions and deletions and consequently might improve recall but probably reduces precision. Using an appropriate ranking model, we can avoid the decrease in precision. Mamou et al. [10] presented the enhancement in recall and precision by searching on WCN instead of considering only the 1-best path word transcript. We used this model for In-Vocabulary (IV) search. In word transcripts, OOV terms are deleted or substituted; therefore, the usage of phonetic transcripts is desirable. A possible combination of word and phonetic search was presented by Mamou et al. [11] in the context of spoken term detection. We further improve this combination in the context of speech document retrieval with fuzzy phonetic search. Moreover, due to the lower accuracy of phonetic search, we merge results obtained from different kinds of phonetic transcripts, e.g., fragment decoding and phonetic representation of the word decoding. Our results show that the retrieval on phonetic transcripts tends to improve the recall without substantially affecting the precision, using an appropriate ranking.

### 3.1 Indexing Model

We describe now the indexing process for both IV and OOV.

For phonetic transcripts, we extract N-grams of phones from the transcripts and index them. We extend the document at its beginning and its end with wildcard phones, so that each phone appears exactly in N N-grams. We ignore space characters. To compress the phonetic index, we represent each phone by a single character.

Both word and phonetic transcripts are indexed in an inverted index. Each occurrence of a unit of indexing (word or N-gram of phones) u in a transcript D is indexed with its position. In addition, for WCN indexing, we store the confidence level of the occurrence of u at the time t that is evaluated by its posterior probability Pr(u|t, D).

### 3.2 Retrieval Approach

This section presents our approach to accomplishing the document retrieval task using the indices described above.

Our goal is to answer written queries. Since the vocabulary of the ASR system used to generate the word transcripts is given, we can easily identify IV and OOV parts of the query. In this vocabulary, we include IV terms; the other terms are OOV.

Our approach uses word search for retrieving from indices based on word transcripts and phonetic search for retrieving from indices based on phonetic transcripts. We use a combination of the Boolean Model and the Vector Space Model (Salton and McGill [17]) with modifications of the term frequency and document frequency to determine the relevance of a document to a query. Afterward, we assign an aggregate score to the result based on the scores of the query terms from the search of the different indices, as explained in Section 3.5.

### 3.3 Word Search: Weighted Term Frequency

The word search approach can only be used for IV terms. The posting lists are extracted from the word inverted index. We extend the classical TFIDF method using the confidence level provided by the WCN, as explained by Mamou et al. [10].

Let us denote by occ(u, D) = (t_1, t_2, ..., t_n) the sequence of all the occurrences of an IV term u in the document D. The term frequency of u in D, tf(u, D), is given by the following formula:

\[
    tf(u, D) = \sum_{i=1}^{\|occ(u, D)\|} Pr(u|t_i, D)
\]

Note that we have not modified the computation of the document frequency.

### 3.4 Fuzzy Phonetic Search

This section presents our approach to fuzzy phonetic search. Although this approach is more appropriate for OOV query terms, it can be also used for IV query terms. However, the retrieval will probably be less accurate, because we ignore the space character during the indexing process of the phonetic transcripts.

If the query term is OOV, it is converted to its N-best phonetic pronunciation using the joint maximum entropy N-gram model presented by Chen [3]. For ease of representation, we first describe our fuzzy phonetic search using only the 1-best presentation, and in the next section we extend the search to N-best. If the query term is IV, it is converted to its phonetic representation.

The search is composed of two steps: query processing and then pruning and scoring.

#### 3.4.1 Query Processing

Each pronunciation is represented as a phrase of N-grams of phones. For indexing, we extend the query at its beginning and its end with wildcard phones such that each phone appears in N N-grams.

During the query processing, we retrieve several fuzzy matches from the phonetic inverted index for the phrase representation of the query.

To control the level of fuzziness, we define the following two parameters: δ_i, the maximal number of inserted N-grams, and δ_d, the maximal number of deleted N-grams. Those parameters are used in conjunction with the inverted indices of the phonetic transcript to efficiently find a list of indexed phrases that are different from the query phrase by at most δ_i insertions and δ_d deletions of N-grams. Note that a substitution is also allowed by an insertion and a deletion.

At the end of this stage, we get a list of fuzzy matches and the list of documents in which each match appears.

#### 3.4.2 Pruning and Scoring

The next step consists of pruning some of the matches using a cost function and then scoring each document ac-
According to its remaining matches.

Let us consider a query term ph represented by the following sequence of phones \((p_1, p_2, \ldots, p_n)\) and ph' a sequence of phones \((p'_1, p'_2, \ldots, p'_m)\) that appears in the indexed corpus and that was matched as relevant to ph.

We define the confusion cost of ph with respect to ph' to be the smallest sum of insertions, deletions, and substitutions penalties required to change ph into ph'. We assign a penalty \(\alpha_t\) to each insertion and a penalty \(\alpha_d\) to each deletion. For substitutions, we give a different penalty to those substitutions that are more likely to happen than others. As described by Amir et al. [1], we identified seven groups of substitutions that are more likely to happen than others. As tion. For substitutions, we give a different penalty to those substitutions that are more likely to happen than others. As described by Amir et al. [1], we identified seven groups of substitutions that are more likely to happen than others.

The penalty factors are determined such that \(0 \leq \alpha_{sm} \leq \alpha_t, \alpha_d, \alpha_s \leq 1\). Note that this is different from the classical Levenshtein distance since it is non-symmetric and we assign different penalties to each kind of error. We derive the similarity of ph with ph' from the confusion cost of ph with ph'.

We use a dynamic programming algorithm to compute the confusion cost extending the commonly used algorithm that computes the Levenshtein distance. Our implementation is fail-fast since the procedure is aborted if it is discovered that the minimal cost between the sequences is greater than a certain threshold, \(\theta(n)\), given by the following formula:

\[
\theta(n) = \theta \cdot n \cdot \max(\alpha_t, \alpha_d, \alpha_s)
\]

where \(\theta\) is a given parameter, \(0 \leq \theta < 1\). Note that the case of \(\theta = 0\) corresponds to exact match.

The cost matrix, \(C\), is an \((n + 1) \times (m + 1)\) matrix. The element \(C(i, j)\) gives the confusion cost between the subsequences \((p_1, p_2, \ldots, p_i)\) and \((p'_1, p'_2, \ldots, p'_j)\). C is filled using a dynamic programming algorithm. During the initialization of \(C\), the first row and the first column are filled, corresponding to the case where one of the subsequences is empty.

\[
C(0, 0) = 0, \ C(i, 0) = i \cdot \alpha_d \text{ and } C(0, j) = j \cdot \alpha_s.
\]

After the initialization step, we traverse each row \(i\) to compute the values of \(C(i, j)\) for each value of \(j\). The following recursion is used to fill in row \(i\):

\[
C(i, j) = \min_{0 \leq l \leq j} \{C(i - 1, j - 1) + \alpha_{sd}, C(i, j - 1) + \alpha_d, C(i - 1, j - 1) + cc(p_i, p'_j)\}.
\]

\(cc(p_i, p'_j)\) represents the cost of the confusion of \(p_i\) and \(p'_j\), and is computed in the following way:

\- if \(p_i = p'_j\), \(cc(p_i, p'_j) = 0\),
\- if \(p_i\) and \(p'_j\) are in the same metaphone group, \(cc(p_i, p'_j) = \alpha_{sm}\),
\- if \(p_i\) and \(p'_j\) are not in the same metaphone group, \(cc(p_i, p'_j) = \alpha_s\).

After the filling of row \(i\), we abort the computation if:

\[
\min_{0 \leq i \leq m} \{|C(i, j)| > \theta(n)\}.
\]

We define \(sim(ph, ph')\), the similarity of ph with respect to ph', as follows: if the computation is aborted, the similarity \(sim(ph, ph')\) is 0; else

\[
sim(ph, ph') = 1 - \frac{C(n, m)}{n - \max(\alpha_t, \alpha_d, \alpha_s)}.
\]

Note that \(0 \leq sim(ph, ph') \leq 1\). Finally, we compute the score of ph in a document D, \(score(ph, D)\), using TFIDF. We define the term frequency of ph in D, \(tf(ph, D)\) by the following formula:

\[
\text{tf}(ph, D) = \sum_{ph' \in D} sim(ph, ph')
\]

and the document frequency of ph in the corpus, \(df(ph)\), by

\[
df(ph) = \lvert \{D \mid \exists ph' \in D \text{ s.t. } sim(ph, ph') > 0\} \rvert.
\]

3.5 Combination of the Results Lists

We propose two different approaches for combining the lists of results. The first approach is based on the Threshold Algorithm [5] for combining the results from the different indices. The second approach indexes the different transcripts in a single inverted index and retrieves information using the Boolean Model.

3.5.1 Using the Threshold Algorithm

As described above, a query is composed of IV and OOV terms. Let us denote the query \(Q = (iv_1, \ldots, iv_n, oov_1, \ldots, oov_m)\) where each \(iv\) is an IV term and each \(oov\) is an OOV term. In a general setup, we may have several indices based on word transcripts and several indices based on phonetic transcripts.

For each term, we need to decide the forms with which it should be queried (i.e., word or phone), the indices to which it should be sent, and how to merge the search results obtained on the different indices. We start with a simple setup where we have one index based on word transcripts for IV terms and one index based on phonetic transcripts for OOV terms. We then describe the general case.

In the simple case, the query is decomposed into subqueries such that each \(iv_i\) is sent to the word index, and each \(oov_i\) is converted to its phonetic representation and sent to the phone index. Each index runs its subquery using its scoring as described above and returns a result list of documents sorted by decreasing order of relevance to the given query, where scores are normalized in the range \([0, 1]\). A relevant document can now appear in one or both of the lists with a score in each. The next step is to merge the two result lists into a single list of documents that are most relevant to the query, using some aggregation function to combine the scores of the same document from the different lists. An example of this aggregation could be to assign different weights to each list and use a weighted sum for the aggregation.

A simple algorithm to merge the lists would fully scan the lists and find the \(k\) documents (for a given \(k\)) with the highest aggregated score. This could work for short lists, but when the lists grow it could be an expensive task. Therefore, we use the Threshold Algorithm (TA) as described by Fa-gin et al. [5], the state-of-the-art for such problems. The algorithm simultaneously scans the different lists row by row. For each scanned document in the current row, it finds its score in the other lists and then uses the aggregate function to find its total score. It then calculates the total score of the current row using the same aggregate function. The algorithm stops when there are \(k\) documents with a total score larger than the last row’s total score.

We now turn to the general setup in which we may have
several word indices and several phonetic indices. Each iv term can be sent to one or more word indices and to one or more phone indices using its phonetic representation; each oov term can be sent to one or more phone indices using its phonetic representation. In this general case, we run the TA as follows: for each query term, we first decide to which indices to send it and then send it as a subquery to each of the selected indices. The decision where to send each query term is based on the availability of each index, in the case of a distributed system, or on the quality of the index using some collected statistics. The description of such statistics, however, is outside the scope of this paper. Each index returns a list of results sorted by decreasing score order. We then apply a TA on those lists and get a single list of documents that contain the term in at least one of the indices. We refer to this step as local TA. After we finish with the local TA for all query terms, we then apply a TA on all those lists and get the final set of top-k documents that best matches the query. We refer to this step as global TA.

It is worth mentioning that the TA can be run using OR or AND semantics. In OR semantics, a document is returned if it appears only in some of the result lists. In AND semantics, a document is returned only if it appears in all the result lists. Moreover, we can use a different aggregation on the local TA (for example, taking the max score), and a different aggregation on the global TA (for example, using a weighted sum). We usually prefer to run the local TA under OR semantics; it is sufficient that the query term appears in the document according to at least one of the indices selected to run this query term. We then run the global TA, combining the different query terms using the semantics defined in the query between the query terms.

3.5.2 Using Inverted Index with Boolean Constraints

The TA solution described in the previous section is a general solution for merging lists. It can work for any setup of indices, such as a distributed environment where each index resides on a different machine, or for combining results from different implementations of indices. Its drawback is that it may require deep scanning of the result lists coming from each index until it can reach the stopping condition.

When all indices are on the same machine and we can control their implementation, we can achieve better performance by implementing all indices in a single inverted index and exploiting the strength of Boolean constraints as described below. Furthermore, we show that with such an implementation, we can perform both local and global TA merges using a single Boolean expression.

We index word and phonetic corpora into a single inverted index. Each corpus is indexed under a predefined field, enabling the easy access of data from a specific corpus. For query processing, the query is reformulated into Boolean clauses (field: term) with Boolean constraints such that each clause is matched, according its field, against the appropriate transcript corpus.

For example, let us suppose that we indexed one word transcript corpus under the field W and two phonetic transcript corpora under the fields P1 and P2. We can reformulate each IV term iv, as the Boolean clauses ((W: iv) ∨ (P1: iv)) ∨ (P2: iv)); each IV term is searched against word and phonetic corpora under OR semantics. In a similar manner, we can reformulate each OOV term oov, as the Boolean clauses ((P1 : oov) ∨ (P2 : oov)); each OOV term is searched against phonetic corpora under OR semantics.

These expanded query terms are then joined into a new Boolean query according to the semantics defined in the original query between the query terms.

After this reformulation of the query, the posting list of each clause is extracted from the index. The posting lists of all the clauses are then merged and combined according to the Boolean constraints to form a single result list. This step is efficiently handled by the runtime algorithm of any classical textual IR engine.

In addition, we can assign an appropriate weight to each Boolean clause according to its field. One possibility could be to assign a higher weight to word fields as opposed to phonetic fields. These weights are handled by the search engine when scoring a document matching the Boolean constraints: terms appearing in both the document and query contribute to the global document a score proportional to their field’s weight. In a same manner, the internal Boolean clauses can also be assigned a weight; in such configuration, we can assign, for example, a higher weight to IV terms over OOV terms. In a same way, the internal clauses contribute a score linearly boosted by these weights.

In the following section, we report experiments on the retrieval effectiveness based on the TA approach. Note that both the TA and Boolean constraint approaches return similar result lists; the only difference is in the implementation of the combination algorithm. The TA approach merges result lists of documents ordered according to their score, while the second approach merges posting lists ordered by the document identifiers according to Boolean constraints.

4. EXPERIMENTS

4.1 Experimental Setup

Our corpus consisted of the three hours of US English Broadcast News (BN) from the dry-run set provided by NIST for the Spoken Term Detection (STD) 2006 evaluation. Our experiments reports the retrieval performance on two different segmentations of this corpus:

- **BN-long** - segmentation into 23 audio documents of 470 sec on average, and
- **BN-short** - segmentation into 475 audio documents of 23 sec on average.

These two segmentations allow to analyze the retrieval performance according to the length of the speech document.

We used the IBM Research prototype ASR systems described in Section 2 for word decoding and word-fragment decoding. We used and extended Lucene\(^4\), an Apache open source search library written in Java, for indexing and search. In particular, we extended the capabilities of Lucene fuzzy search\(^5\) in the context of phonetic search on speech data and we have used the scoring function based on the Vector Space Model implemented in Lucene\(^6\).

\(^2\)See for example the Lucene field at http://lucene.apache.org/java/docs/api/org/apache/lucene/document/Field.html

\(^3\)http://lucene.apache.org/

\(^4\)http://lucene.apache.org/java/docs/api/org/apache/lucene/search/FuzzyQuery.html

\(^5\)http://lucene.apache.org/java/docs/api/org/apache/lucene/search/Similarity.html
We built three different indices:

- **Word Index** - a word index on the WCN,
- **WordPhone Index** - a phonetic N-gram index of the phonetic representation of the 1-best word decoding
- **Phone Index** - a phonetic N-gram index of the 1-best fragment decoding.

We compared two different search methods for phonetic retrieval:

- **exact** - exact search, i.e., $\delta_i = \delta_d = 0$, and
- **fuzzy** - fuzzy search, i.e., $\delta_i, \delta_d > 0$.

We also measured the Mean Average Precision (MAP) by comparing the results obtained over the automatic transcriptions to the results obtained over the reference manual transcripts. Our aim was to evaluate the ability of the suggested retrieval approach in handling transcribed speech data. Thus, the closer the automatic results to the manual results, the better the search effectiveness over the automatic transcriptions. The results returned from the manual transcription for a given query were considered relevant and were expected to be retrieved with highest scores.

The testing and determination of empirical values were achieved on the development set provided by NIST. The experiments were performed with the following parameters: $N = 2$ (bi-gram phonetic indices), the parameters $\delta_i = \frac{|ph| - 1}{N}$ and $\delta_d = \frac{|ph| - 1}{N}$ where $|ph|$ is the number of bi-grams in the phrase representing the query term, the penalty factors of the cost confusion $\alpha_s = 0.8$, $\alpha_d = 1$, $\alpha_s = 0.9$ and $\alpha_{sm} = 0.2$.

As mentioned in Section 3.5, we ran the local TA under OR semantics and the semantcs of the query affects only the global TA. The Weighted Sum was used as the aggregate function for both local and global TA; it is given by the following formula:

$$\sum_i w_i \cdot \text{score}(Q, D_i),$$

where $\text{score}(Q, D_i)$ is the score of a document $D$ in list $i$ for query $Q$ and $w_i$ is the weight assigned to this list. The weights in the local TA were respectively 5, 3, 2 for the Word, WordPhone and Phone indices. In the global TA, the weights of IV and OOV terms were respectively equal to 2 and 1.

### 4.2 WER Analysis

We used the WER in order to characterize the accuracy of the transcripts. The WER of the 1-best path transcripts extracted from WCN’s was 12.7%. The substitution, deletion and insertion rates were 49%, 42% and 9%, respectively. The WER of the 1-best fragment decodings was generally worse than the WER of 1-best word transcripts. In our ASR configuration, since the insertion rate was higher than the deletion rate, we have chosen the penalty parameters such that $\delta_i > \delta_d$.

### 4.3 In-Vocabulary Search

IV terms can be searched using both word and phonetic indices. We did not use fuzzy phonetic search since it generally decreases the MAP of the search. For each IV term, we combined the word search with exact phonetic search using the local TA. We processed the set of all the 1074 IV queries from the dry-run STD set. We tested both OR and AND semantics for multiple word queries. Table 1 summarizes the MAP according to each search approach. The MAP of the Word and WordPhone approaches are close but not equal since the space character between terms is ignored in the WordPhone index. As expected for IV terms, the MAP of the WordPhone is better than MAP of the Phone and using only the Phone approach drastically affects the MAP of the retrieval. The combination improved the MAP of the search by up to 4% with respect to the Word search.

### 4.4 Out-Of-Vocabulary Search

OOV terms can be retrieved only using phonetic indices. We have processed the set of all the OOV terms that appear in the manual transcripts and all the OOV terms from the dry-run STD set. There were ninety-seven queries, each query containing a single OOV term. Table 2 and Table 3 summarize the MAP according to each search approach on BN-long and BN-short corpora, respectively. As generally observed in phonetic OOV search (Ng and Zue [14]), we obtained higher performance with fuzzy search than with exact search. The combination of WordPhone and Phone leads to an improvement for both exact and fuzzy searches.

We have extended this set with ninety-seven other queries of two terms where each term has been chosen randomly from the OOV single word queries set described above. We ran the queries under OR and AND semantics. Each OOV term was searched on phonetic indices using fuzzy match, and the results were merged using local TA between the

---

### Table 1: MAP for IV search.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Semantics</th>
<th>Word</th>
<th>WordPhone</th>
<th>Phone</th>
<th>Combination</th>
</tr>
</thead>
<tbody>
<tr>
<td>BNEWS-long</td>
<td>OR</td>
<td>0.90</td>
<td>0.87</td>
<td>0.74</td>
<td>0.92</td>
</tr>
<tr>
<td></td>
<td>AND</td>
<td>0.90</td>
<td>0.85</td>
<td>0.66</td>
<td>0.93</td>
</tr>
<tr>
<td>BNEWS-short</td>
<td>OR</td>
<td>0.85</td>
<td>0.78</td>
<td>0.63</td>
<td>0.88</td>
</tr>
<tr>
<td></td>
<td>AND</td>
<td>0.88</td>
<td>0.82</td>
<td>0.60</td>
<td>0.90</td>
</tr>
</tbody>
</table>

### Table 2: MAP for OOV search on single word queries on BN-long corpus

<table>
<thead>
<tr>
<th>Method</th>
<th>WordPhone</th>
<th>Phone</th>
<th>Combination</th>
</tr>
</thead>
<tbody>
<tr>
<td>exact</td>
<td>0.32</td>
<td>0.30</td>
<td>0.39</td>
</tr>
<tr>
<td>fuzzy</td>
<td>0.52</td>
<td>0.52</td>
<td>0.58</td>
</tr>
</tbody>
</table>

### Table 3: MAP for OOV search on single word queries on BN-short corpus

<table>
<thead>
<tr>
<th>Method</th>
<th>WordPhone</th>
<th>Phone</th>
<th>Combination</th>
</tr>
</thead>
<tbody>
<tr>
<td>exact</td>
<td>0.31</td>
<td>0.27</td>
<td>0.37</td>
</tr>
<tr>
<td>fuzzy</td>
<td>0.40</td>
<td>0.39</td>
<td>0.47</td>
</tr>
</tbody>
</table>
4.5 Hybrid Search

Hybrid queries combine both IV and OOV terms. The IV terms were searched on both word and phonetic indices using exact match and the results were merged with the local TA between Word, WordPhone and Phone indices. The OOV terms were searched on phonetic indices using fuzzy match, and the results were merged with the local TA between the WordPhone and Phone indices. Afterwards, the results for each query term were merged into a single list of results using the global TA. We randomly generated a set of 1000 hybrid queries from the IV and OOV sets used for the above experiments. Each query contained two IV terms and two OOV terms. We tested both OR and AND semantics. Table 5 summarizes the experiments. Under AND semantics (i.e., all query terms must appear in a matched document), we could not obtain any results using the Word index since each query contains OOV terms. Moreover, for the BN-short corpus with AND semantics, there were no relevant results: the documents were short and no single document contained all four query terms.

In the combination approach, we saw an improvement of up to 25% with respect to the Word approach, under the OR semantics. Under AND semantics, we saw an improvement of 14% to 58% for the combination approach in respect to the Phonetic approaches.

5. RELATED WORK

Popular approaches for OOV search are based on search on subword decoding as described by Clements et al. [4], Seide et al. [21], and Siohan and Bacchiani [22].

Other approaches are based on search on the subword representation of the word decoding as described by Amir et al. [1]. They show improvement using phone confusion matrix and Levenshtein distance. Chaudhari and Picheny [2] present an approximate similarity measure for searching such transcripts using higher-order phone confusions.

Jones et al. [8] propose different ways (data fusion and data merging) to combine search on word transcript generated by an LVCSR system for IV terms with word spotting on phone-lattice for OOV terms, in the context of video mail applications. Saraclar and Sproat [20] show an improvement in word spotting accuracy for both IV and OOV queries, using phonemes and word lattices, where a confidence measure of a word or a phoneme can be derived. They propose three different retrieval strategies: search both the word and the phonetic indices and unify the two different sets of results; search the word index for IV queries and search the phonetic index for OOV queries; and search the word index and if no result is returned, search the phonetic index. However, the phonetic search is done by exact matches on the phonetic lattices. Mamou et al. [11] and Mathews et al. [13] propose to combine word and phonetic searches in the context of spoken term detection.

However, these approaches do not propose a method to combine multiple word indices with multiple phonetic indices.

6. CONCLUSIONS AND FUTURE WORK

In the past, word and phonetic approaches have been used for IR on speech data; the former suffers from low accuracy and the latter from limited vocabulary of the recognition system. In this paper, we presented a retrieval model that combines search on different speech transcripts. We compared this combination approach to the state-of-the-art word and phonetic approaches.

For future work, we will explore the approach more deeply based on Boolean constraints presented in Section 3.5.2 and compare it with the TA approach in terms of computation and time complexity.

Our TA approach can be particularly practical in a peer-to-peer (P2P) network because we can afford to run multiple decoding methods on the same audio data simultaneously on different peers. We intend to extend the search into a P2P environment in the context of the FP6 SAPIR project.

7. ACKNOWLEDGMENTS

This work was partially funded by the European Commission Sixth Framework Programme project SAPIR - Search in Audiovisual content using P2P IR.

The authors are grateful to Ron Hoory and Michal Shmueli-Scheuer.

8. REFERENCES


http://www.sapir.eu/
<table>
<thead>
<tr>
<th>Corpus</th>
<th>Semantics</th>
<th>Word</th>
<th>WordPhone</th>
<th>Phone</th>
<th>Combination</th>
</tr>
</thead>
<tbody>
<tr>
<td>BNEWS-long</td>
<td>OR</td>
<td>0.63</td>
<td>0.59</td>
<td>0.67</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AND</td>
<td>0.57</td>
<td>0.53</td>
<td>0.58</td>
<td></td>
</tr>
<tr>
<td>BNEWS-short</td>
<td>OR</td>
<td>0.35</td>
<td>0.36</td>
<td>0.41</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AND</td>
<td>0.51</td>
<td>0.50</td>
<td>0.52</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: MAP for OOV search on multiple word queries

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Semantics</th>
<th>Word</th>
<th>WordPhone</th>
<th>Phone</th>
<th>Combination</th>
</tr>
</thead>
<tbody>
<tr>
<td>BNEWS-long</td>
<td>OR</td>
<td>0.58</td>
<td>0.64</td>
<td>0.59</td>
<td>0.71</td>
</tr>
<tr>
<td></td>
<td>AND</td>
<td>0</td>
<td>0.5</td>
<td>0.36</td>
<td>0.57</td>
</tr>
<tr>
<td>BNEWS-short</td>
<td>OR</td>
<td>0.59</td>
<td>0.54</td>
<td>0.48</td>
<td>0.73</td>
</tr>
</tbody>
</table>

Table 5: MAP for hybrid search

on Research and development in information retrieval, pages 30–38, New York, NY, USA, 1996. ACM.


