Similarity Measure Optimization for Target Detection: A Case Study for Detection of Keywords in Telephone Conversations

Chapter · July 2018

DOI: 10.4018/978-1-5225-5513-1.ch015

2 authors:

Batuhan Gundogdu
University of Chicago
32 PUBLICATIONS **167** CITATIONS
SEE PROFILE

Murat Saraclar
Bogazici University
189 PUBLICATIONS **3,979** CITATIONS
SEE PROFILE

Similarity Measure Optimization for Target Detection: A Case Study for Detection of Keywords in Telephone Conversations

Batuhan Gundogdu[1] and Murat Saraclar[2]

[1]National Defence University, Turkey
[2]Bogazici University, Turkey

INTRODUCTION

The digitalization of data and the ever-increasing storage capacity has brought people closer to a life like the fictional world in Jorge Luis Borges' *Library of Babel* (Borges, 1998), —an infinite library *whose books contain every possible string of letters and, therefore, somewhere an explanation of why the library exists and how to use it.* Though, it was highly doubted that the librarians would ever find that book amid the miles of nonsense. The situation the society faces today is somewhat similar: When it comes to retrieval of data, and especially the spoken data, the conundrum appears to be the *"Paradox of Abundance"* defined in Haidt (2006): *Quantity undermines the quality of our engagement*. Hence, it is vital now, in the Internet era; to efficiently and correctly retrieve the data of interest to make our lives easier as evidenced by the well-developed text retrieval technology, which is undoubtedly one of the biggest life-changing innovations of the past century. On the other hand, speech retrieval technology is still nascent even though speech is definitely one of the most appealing forms of information content. The retrieval of speech is therefore not only attractive but also necessary for users to efficiently browse across the vast quantities of multimedia content.

Detection of certain words in telephone conversations is also a very important task for security reasons and fight against terrorism. Such a task may require a dynamic list of 'suspicious' keywords, so that there no longer exists a necessity for developing the retrieval system from the beginning once new keywords become of interest.

This chapter focuses on a speech retrieval task called spoken term detection, also known as keyword search (KWS). In KWS, the user provides a query in text form consisting of one or more words. This query is searched in an un-transcribed audio archive and the locations in the audio where the query exits are returned to the user in an ordered list, with respect to a similarity score the KWS system attributes to them. The text counterpart of the task that was just defined is a very common one in everyday life. Nowadays, people rarely spend a day without searching something in Google©. Text browsing also comes in handy and used in text editors with very well-known short-cut button configurations. Despite this wide usage of text retrieval acknowledged by almost every computer user, searching for speech still remains a young research. Searching for audio-visual content in multimedia archives, such as YouTube, only provides videos whose title or meta-data has high similarity scores to the text query. The actual locations where the terms of interest are uttered are not available upon such a search. In KWS, on the other hand, the audio documents are returned to the user along with the time stamps where the query occurs. Such a system maybe desired in situations where the user is interested in finding locations of their queries in lecture videos, broadcast news reports, radio communications, meeting recordings. Furthermore, detecting and locating the utterance of certain keywords in telephone conversations may be very important for security applications for prevention and/or investigation of criminal or terrorist activities. System diagram demonstrating the KWS task is given in Figure 1.

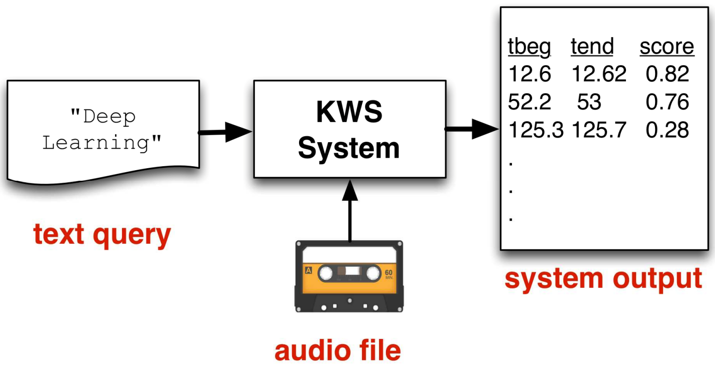


Figure 1 A brief demonstration of the keyword search task

Since the text retrieval task is now a mature product and has been providing reliable results, the first practiced approach to KWS was using an automatic speech recognition (ASR) system on the audio archive and converting the task into a text retrieval task. Once the spoken document is converted into a string of words by the ASR system, the task is now to finding text in text, which is not so hard. However, only depending on the ASR system output limits the retrieval system's performance to the accuracy and the vocabulary size of the ASR systems being used. Furthermore, when circumstances are harder as in the situations when the recording conditions are very noisy and there is little or no available data for the language of interest, which is referred to as a "*low resource language*" in the literature.

This chapter aims to address the problem of KWS especially for low resource languages by presenting the methodologies adopted in literature and introduces a novel approach that conducts a similarity measure optimization and a dynamic search. In the next section the definitions are presented and the conventional techniques are introduced. The problem brought by scarcity of resources is addressed in a thorough manner and the solution methodology is described. The validity of the approach is presented with the experiments conducted on three low resource languages of the IARPA Babel Program (Harper, 2014).

BACKGROUND

This section summarizes the definition of the task in hand and the presents the background information explaining the methodologies adopted in the literature. KWS is defined as the task of obtaining the locations of a user provided text query term in an audio speech archive. The speech archive is referred to as the *"document"* in this book and in literature. The earliest mature works date back to mid-1990s when the ASR and information retrieval (IR) technologies began to achieve satisfactory performances. The Text Retrieval Conference (TREC) evaluations conducted in 1999 and 2000 are considered a milestone in the KWS applications. The application in (Van Thong et al, 2002) was the first KWS system used in web. Since these applications mainly focused on using broadcast news speech, the ASR results had fewer errors due to better recording conditions when compared to conversational speech. The actual and deep endeavor on KWS was motivated with the 2006 NIST Spoken Term Detection (STD) evaluations (Fiscus et al, 2007). Many successful applications of KWS were implemented on broadcast news (Wang, D. et al, 2008), video lectures (Garcia & Gish, 2006) and web-videos (Lee, S. W. et al, 2005) since then. As an extension to the 2006 NIST STD evaluations, which worked on KWS systems in English, Mandarin, and Arabic, the IARPA Babel Program (Harper, 2014), was initiated in 2011 to specifically address low resourced languages.

Large Vocabulary Continuous Speech Recognition (LVCSR) Based KWS Systems

The most convenient approach to approach the KWS problem would be thought of using an ASR system, more specifically a large vocabulary continuous speech recognition (LVCSR) system to obtain a text transcription from the audio and applying text retrieval techniques on the LVCSR output. The next subsection briefly discusses how an LVCSR system operates on a speech document.

A Brief Description of LVCSR Systems

Before proceeding onto the mathematical definitions and operations taking place under the hood, the explanation the task *large vocabulary continuous speech recognition* is needed. “Speech recognition” is a subtask of machine learning and basically defined as the process of a machine to identify words and phrases in spoken language and convert them to into understandable sequences of texts of actions. “Continuous speech recognition” is, in contrast to “isolated speech recognition” or “isolated word recognition” is a speech recognition task where the word boundaries are unknown by the machine. A machine conducting continuous speech recognition will convert the speech into combinations of the words or phrases it was trained to “know”. The set of words that the system knows is defined as the “*vocabulary*” of the speech recognition system. The vocabulary size of the speech recognition system depends on the application it is used. For applications such as “command recognition” or “digit recognition” will have certainly a limited vocabulary and the recognition accuracy is expected to be high in these applications. An example of such a system can be the automated systems used by the call centers of banks asking you to explain what you wish to do in a brief sentence. Large vocabulary continuous speech recognition, on the other hand, has a more comprehensive vocabulary and a pronunciation lexicon, which primarily aims to operate on conversational speech. The problem is put in a statistical perspective. Since the goal of LVCSR is stated as obtaining the sequence of words $\mathbf{W} = \{w_1, w_2, ..., w_N\}$ from the acoustics $\mathbf{X} = \{x_1, x_2, ..., x_T\}$. The task is nothing but a maximum likelihood optimization problem on the conditional probability distribution

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}\in\mathbf{W}^*} p(\mathbf{W} \mid \mathbf{X}) \tag{1}$$

where, the maximization is taken over all possible word sequences $\mathbf{W}^*$ that can be produced from the words in the vocabulary. There’s hardly a way to model this probability distribution for an LVCSR task, yet the feasible means is to obtain the maximum a posteriori optimization equation from Bayes formula i.e.

$$\hat{\mathbf{W}} = \operatorname*{argmax}_{\mathbf{W}\in\mathbf{W}^*} p(\mathbf{X} \mid \mathbf{W})\, p(\mathbf{W}) \tag{2}$$

where the term $p(\mathbf{X} \mid \mathbf{W})$ is called the "*acoustic model*" and $p(\mathbf{W})$ is referred to as the "*language model*". The acoustic model is estimated from a dataset of transcribed speech data and presents a probability distribution in the acoustic feature space for each possible word sequence $\mathbf{W}$. Since it is practically impossible to obtain enough samples to estimate the distribution on word level, the general approach is to use sub-word units like *phonemes* and model their duration with Hidden Markov Models (HMM). On the other hand, the language model is obtained as n-gram conditional probabilities in word level, from transcriptions in the target language.

Acoustic Modeling with Gaussian Mixture Models

As mentioned above, the acoustic model is a probability distribution over the set of phonetic units in the target language, estimated from the training data. Each sub-word unit is generally approximated by a multivariate Gaussian Mixture Model (GMM).

$$p(\mathbf{X}) = \sum_{i=1}^{K} c_i \mathcal{N}(\mathbf{X} \mid \mu_i, \Sigma_i) \qquad (3)$$

where μ_i and Σ_i are the mean and variance parameters, respectively.

Each sub-word unit is modeled as a mixture of K Gaussians and the set of parameters are estimated using the training set via a expectation-maximization procedure called "Baum-Welch Algorithm". Further discussion on LVCSR would be out of the scope of this book, hence interested readers are referred to the very useful sources such as (Rabiner, 1989) and (Huang et al., 2001).

Acoustic Modeling with Deep Neural Networks

With the invent of high computing power brought by Graphical Processing Units (GPU) and the increasing amounts of available digital data, neural network computation yielded satisfactory results in many fields. Deep neural networks (DNN) are also used in acoustic modeling in speech recognition field (Hinton et al, 2012). DNNs are used in LVCSRs as stochastic structures whose inputs are acoustic feature vectors and outputs are posterior probabilities of the input frame belonging to one of the many classes. In the LVCSR task, these classes are phonetic sub-word units. A simple DNN structure is given in Figure 2. In the training, the weights $\mathbf{w}_{i,j}$ that connect the nodes of the network are learned.

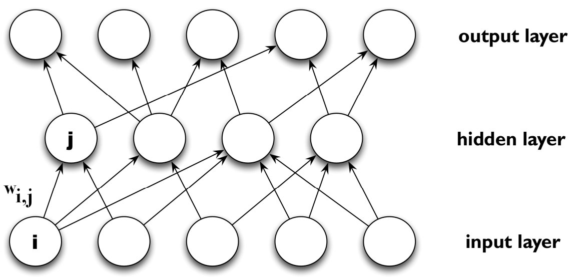


Figure 2 A sample DNN model with two layers

The output layer size of a DNN is equal to the number of phonetic sub-word units in the target language. Hence, the output of the network for a given time instant can be considered as a vector depicting a posterior probability mass function along the classes. The acoustic model from this posterior vector can easily be obtained by dividing them by the priors, again estimated from the training data. The most widely used acoustic feature vectors inputted to the neural network are linear prediction cepstral coefficients (LPCC) or mel frequency cepstral coefficients (MFCC). On the other hand, it should be noted that, with deep enough networks, the gaps of efficiencies of different acoustic feature vectors are closed, due to the representation learning power of DNNs.

KWS on LVCSR Output

Since LVCSR systems may fail to return perfect transcriptions when the available data to estimate the model parameters and the recording conditions are bad, the performance of the KWS systems that use them will be far from efficient as well. In order to address this problem, practical LVCSR based KWS systems use stochastic structures like *lattices* or *confusion networks* obtained from the LVCSR systems instead of the single-best transcriptions. Lattices are graph structures that store weighted hypotheses in a compact form. Indexing ASR lattices, instead of the best ASR hypothesis, is a widely used KWS method that partly compensates for the negative effects of recognition errors (Can & Saraçlar, 2011). In this method, spoken documents are divided into short segments called utterances. For each utterance, an ASR lattice, representing ASR hypotheses, is generated. A sample lattice structure is given in Figure 3.

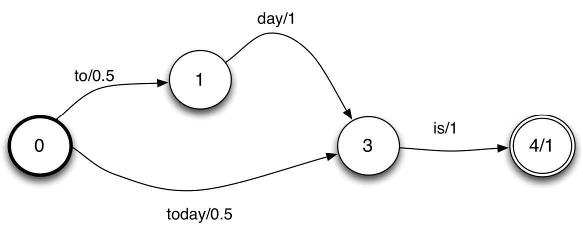


Figure 3 A sample lattice structure

Once the lattices are created from the LVCSR output, the search is then conducted by means of weighted finite state transducer (WFST) operations (Mohri et al, 2002), (Saraçlar & Sproat, 2004).

Retrieval of Out-of-Vocabulary Terms

As mentioned above, the LVCSR systems will produce lattices based on how similar the acoustics to the sequences of words within its vocabulary. However, if a query term includes words that are not in the vocabulary of the LVCSR system, they will never appear on the lattice. Hence it will be impossible to retrieve them. Such terms are referred to as out-of-vocabulary (OOV) terms, in the literature. Furthermore, users generally tend to search for distinctive words like names and technical terms, not common words like “sun” and “sea”. This problem presents a bigger conundrum even when working on low resource languages, since there is already not enough data to estimate parameters for the words in vocabulary. Also, the vocabulary size is going to be consequently small for such languages. In order to cope with this problem, one solution is to create lattices for sub-word units instead of whole words (Karakos & Schwartz, 2014). In this case, the usage of language model is surpassed; making the KWS systems create many false alarms. While sub-word indexing provides a method to retrieve OOV query words given a sub-word level representation for the query, it does not answer how to obtain these representations. Most KWS systems tackle this problem at the retrieval stage by employing pronunciation models. However, it is not an easy task to generate proper pronunciations for words that haven’t been seen before. For that reason, most KWS systems utilize inexact matching methods to handle the discrepancies between the actual and hypothesized pronunciations. This solution is implemented by searching the web for expansion of the language model to obtain automatically generated pronunciations (Gandhe et al., 2013). A very efficient approach to OOV handling is obtaining a confusion model from the training set and using acoustically similar in-vocabulary (IV) terms, instead of the OOV terms. These terms are called “proxy keywords” and this approach is preferred by many LVCSR based KWS systems due to its efficiency (Chen et al, 2013). The methodology adopted in this chapter follows a very different scheme and conducts the search in the feature space without using the LVCSR based systems. The addressing of the OOV terms in this works methodology is the main strength of the work. Next subsection introduces a pertinent task called query by example (QbE) spoken term detection, for what this chapter does is nothing but extending the success of QbE techniques onto KWS to provide a means of OOV retrieval.

Query by Example Spoken Term Detection

Query by example (QbE) spoken term detection, which is a very pertinent task, aims to search audio in audio. In this task, the user provides an audio snippet, instead of text, as a query and this term is searched in an audio archive (Shen, et al., 2009). Examples of this can be found in applications like Shazaam and SoundHound. Two main approaches proved useful for QbE STD.

1. Symbol-based approaches.
2. Frame-based approaches.

The symbol-based approaches use a decoder system and express the query and document as sequences of symbols from a finite alphabet. An LVCSR system can be thought of such a decoder for these systems. Since the query is also an audio, it will also be modeled as a lattice, in other words a weighted finite state transducer conveying the uncertainty of the decoding. A similarity score depicting the probability that the query exists in the utterance is then obtained by weighted finite state transducer operations.

Frame-based approaches, on the other hand, aims to obtain feature representations for each time window, called *frames*, from the document and the query, and then conduct the similarity search by means of versions dynamic time warping algorithms. In the MediaEval campaign (Rodriguez-Fuentes & Penagarikano, 2013), which focused on QbE spoken term detection on low resource languages, frame-based approaches yielded better performances compared to symbol-based approaches due to their freedom on search. The following subsection describes the dynamic time warping algorithm and their versions to conduct keyword search.

Dynamic Time Warping

Dynamic time warping (DTW) is a dynamic programming algorithm that finds an alignment between two temporal sequences possibly of different length (Albrecht & Muller, 2009). The algorithm also outputs a similarity score for them. The traditional DTW assumes that the two sequences start and end at the same time and they are monotonic. The DTW algorithm has been widely used in speech recognition or verification tasks since speech is a monotonic signal (Sakoe & Chiba, 1978). An illustration of DTW can be seen in Figure 4. Of the two sequences, the red line is faster than the blue line. DTW algorithm fines the alignment of the two lines based on their similarities at any given location.

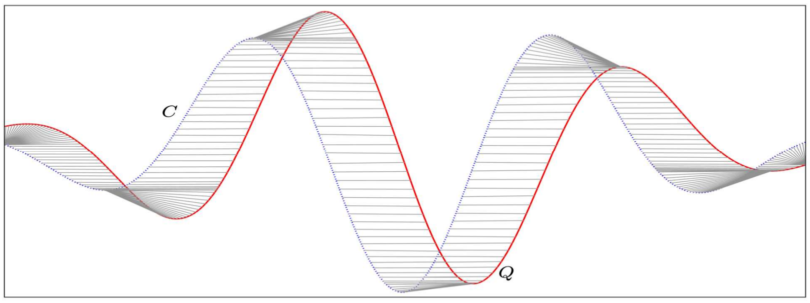


Figure 4 An illustration of the DTW alignment

Source : (Hundt et al, 2014).

The dynamic programming algorithm is as follows: Given two sequences (henceforth called utterances) $\mathbf{Q}$ and $\mathbf{X}$ to be aligned, each of the sequences are represented as a time series of feature vectors, $\mathbf{Q}=\{\mathbf{q}_1,...,\mathbf{q}_M\}$ and $X=\{\mathbf{x}_1,...,\mathbf{x}_N\}$, respectively. The optimal alignment path between $\mathbf{Q}$ and $\mathbf{X}$ is denoted as $\hat{\Phi}$. An alignment Φ between $\mathbf{Q}$ and $\mathbf{X}$ is a sequence of pairs $\Phi=(i_k, j_k); k=1,...,T$ that represents the mapping $(q_{i_1}, x_{j_1}), (q_{i_2}, x_{j_2}),...,(q_{i_T}, x_{j_T})$, where T is the length of the alignment path.

In the traditional DTW, the two utterances are aligned in such a way that their starting and ending frames coincide, i.e. $i_1 = j_1 = 1$, $i_T = M$ and $j_T = N$. The optimal alignment path, $\hat{\Phi}$, is the one that has the minimum total distance, henceforth called as accumulated distance, defined as:

$$D_\Phi = \sum_{k=1}^{T} d(\mathbf{q}_{i_k}, \mathbf{x}_{j_k}) \qquad (4)$$

where $d(\mathbf{q}_{i_k}, \mathbf{x}_{j_k})$ is a distance measure between the two frames on the alignment path. The accumulated distance on the optimal path between the two sequences is found using the following algorithm:

DTW Algorithm

```
for i=1 to M
      DTW(i,0) = infinity
for j=1 to N
      DTW(0,j) = infinity
DTW(0,0) = 0
for i=1 to M
      for j=1 to N
            cost = d(q(i), x(j))
            DTW(i,j)= cost + min(DTW(i-1,j),
                                 DTW(i,j-1),
                                 DTW(i-1,j-1))
return DTW(M,N)
```

However, in QbE spoken term detection, the two sequences do not start and end at the same time as required by DTW, rather one of the sequences (the document) is significantly longer than the other (the query) and the query can be aligned with any subsequence of the document. For this reason, two versions of DTW are used for the search: segmental DTW and subsequence DTW.

Segmental Dynamic Time Warping

In segmental dynamic time warping, short and overlapping segments from the document are obtained and the DTW distance is calculated between the query and each of the segments. The down part of this method is that the selection of the segment size is another optimization parameter that needs to be chosen. The segmental DTW employs the regional search by incorporating global constraints and restricting the allowable shapes that the alignment path can take. Likewise, it is also possible to allow generating multiple alignment paths, in many sub-sequences of the two utterances. It is intuitive to expect that for the two utterances to be aligned, one should not get too far ahead of frames from the other. An illustration of the segmental DTW algorithm can be seen in Figure 5.

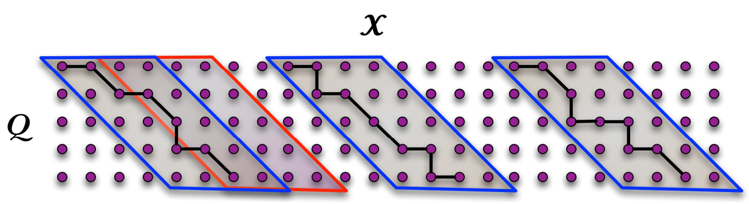


Figure 5 An illustration of the segmental DTW.

Subsequence Dynamic Time Warping

The subsequence dynamic time warping (sDTW) algorithm is another version of the DTW algorithm specifically suitable for the task of interest. While in the traditional DTW algorithm the best alignment between $\mathbf{Q}$ and $\mathbf{X}$ are found by fixing and aligning the start and end frames of the both utterances, in sDTW, it is a fact that $\mathbf{Q}$ is considerably shorter than $\mathbf{X}$ and the starting point of $\mathbf{Q}$ can be aligned with any point of $\mathbf{X}$. Furthermore the segmentation modification offered by segmental DTW fails to perform well when the durations of the two utterances differ by a larger margin. It has been reported in (Lee et al., 2015) that the audio recordings can be as long as twice the original term especially in broadcast news since the speaker tend to slow down the speaking pace drastically in order to make sure the pronunciations are good. Hence, sDTW is used for finding the best alignment between $\mathbf{Q}$ and all sub-sequences of $\mathbf{X}$ with a solution to this problem inherent in segmental DTW. The best alignment yields the most similar subsequence of $\mathbf{X}$ to $\mathbf{Q}$. The similarity search is done through the accumulated distance scores between $\mathbf{Q}$ and all sub-sequences of $\mathbf{X}$ averaged by the alignment path lengths T. An illustration of the recursive sDTW algorithm can be seen in Figure 6. Interested readers are referred to (Gundogdu et al, 2016) for more information about the algorithm and the usage in KWS.

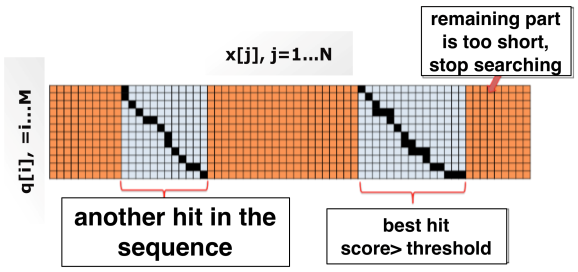

remaining part is too short, stop searching
x[j], j=1...N
q[i], =i...M
another hit in the sequence
best hit score> threshold

Figure 6 An illustration of the recurrent sDTW Algorithm

Source : (Sarı et al, 2015).

Evaluation Metrics

The evaluation methodology is a very important aspect for developing systems. For KWS applications, the success can be defined as the degree of relevance and perfection in detection. Two very pertinent evaluation metrics for this task are *Precision-Recall Rate* and the *F-Measure*, which is very similar to it. Precision (Pr) is the fraction of returned documents from the collection that are relevant to the query, and recall (Rc) is the fraction of relevant documents in the collection that are returned. These values are calculated as follows: Given Q queries, let R(q) be the total number of documents that are relevant to query q, A(q) be the total number of retrieved documents and C(q) be the number
of relevant documents correctly retrieved. Then:

$$P_r = \frac{1}{Q}\sum_{q=1}^{Q}\frac{C(q)}{A(q)} \tag{5}$$

$$R_c = \frac{1}{Q}\sum_{q=1}^{Q}\frac{C(q)}{R(q)} \tag{6}$$

$$F = \frac{2 \times P_r \times R_c}{P_r + R_c} \tag{7}$$

Similar metrics derived from precision and recall rates are *Mean Average Precision (MAP), R-Precision, Precision@N* metrics. For the computation of MAP, Pr values are calculated at each of the Rc values (it should be noted that Rc varies between 0 to 1) and the average of the areas under this Pr -Rc curve for all queries are calculated. Precision@N is the Pr value of the top N returned documents (where N = 10 is a common choice). R-precision is similar to Precision@N, only that N varies for each given query q, and is set to the number of relevant documents R(q). Other widely used evaluation metrics are *Receiver Operating Curve (ROC), Figure Of Merit (FOM), Detection Error Trade*-off (DET) Curve, and the *Term Weighted Value (TWV)* metrics. In order to describe these metrics, the methodology for detection of a given term should be introduced. The outputs of the system are ordered with respect to some confidence score. If the score of an output is above some given threshold, then the output is marked “YES” and it is marked “NO” if it is below the threshold. In other words, a binary decision-making is required per the task definition. The evaluation system looks for the overlaps between the reference occurrences and the outputs of the system and marks the outputs as “HIT” if the central point of the outputs labeled as “YES” overlaps with the 0.5 sec extension of the reference and “FA” (referring to False Alarm) otherwise. The missed detections and the detections labeled as “NO” which overlaps with the reference occurrence are marked “MISS”. This scoring scheme was introduced in NIST STD 2006 evaluations (Fiscus et al, 2007). The methodology of labeling of the system outputs is visualized in
Figure 7.

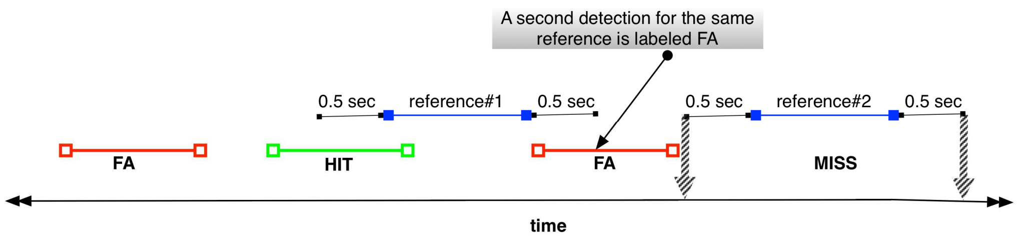


Figure 7 Detection and Hit Labeling Procedure

Given the following definitions:

- *term*: the term being searched,
- *th*: the decision threshold,
- *T:* the total duration of the document,
- $N_{target}(term)$: the number of all correct occurrences of the term in the dataset,
- $N_{HIT}(term, th)$: the number of correctly detected term occurrences for the given *th* value,
- $N_{FA}(term, th)$: the number of false alarms for the term for the given *th* value

With the definitions of the above counts, the following statistics are calculated:

$$p_{HIT}(term, th) = \frac{N_{HIT}(term, th)}{N_{target}(term)} \tag{8}$$

$$p_{MISS}(term, th) = 1 - P_{HIT}(term, th) = 1 - \frac{N_{HIT}(term, th)}{N_{target}(term)} \tag{9}$$

$$p_{FA}(term, th) = \frac{N_{FA}(term, th)}{T - N_{target}(term)} \tag{10}$$

Generally speaking, the performance of an STD system is defined by the trade-off between p_{HIT} and p_{FA}. In response to demands of the specific applications, different measures are considered using these statistics. An *ROC* for one *term* is the *term*'s p_{HIT} *(term; th)* as a function of N_{FA}*(term; th)* per hour. *FOM* is an upper-bound estimate on spoken term detection accuracy averaged over 1 to 10 false alarms per hour. It can be defined as the area under *ROC* from 0 to 10 false alarms per hour. The definitions of these metrics are illustrated in Figure 8.

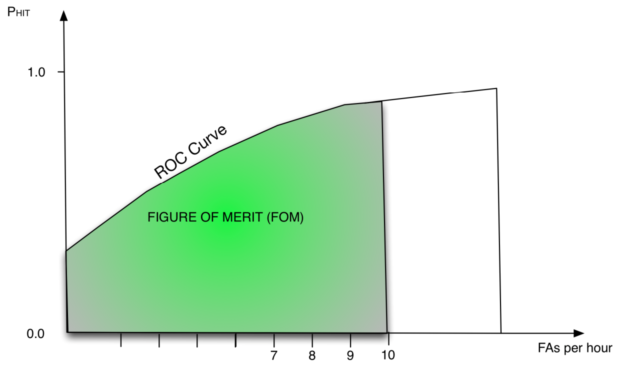


Figure 8 Receiver Operating Curve (ROC) and Figure Of Merit (FOM) Evaluation Metrics

However, the FOM metric is somewhat dependent on each terms' threshold and does not convey useful information about how the system would perform with regards to a single *th* for all queries. Hence the DET curve, which is a more explanatory metric considering the decision threshold, is introduced. For a given threshold th, the DET curve shows the dependency of term's false alarm probability p_F*A(term; th)*(x-axis) to the miss probability p_{MISS}*(term; th)* (y-axis). Contrary to ROC, the DET curve is a plot for all possible values of the threshold this is why it is widely used in the recent KWS applications. An example of a DET curve is illustrated in Figure 9.

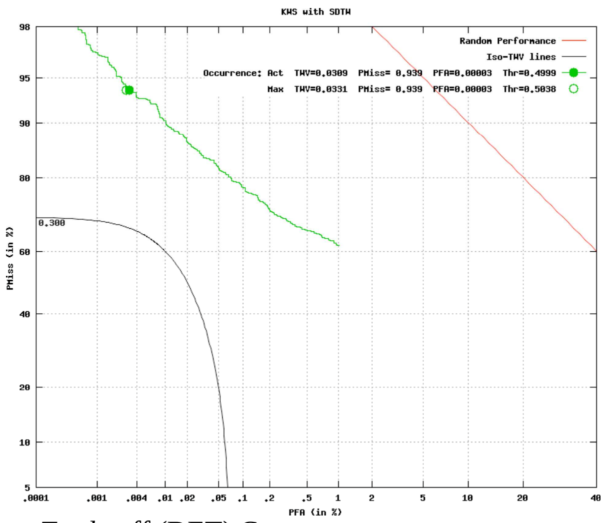


Figure 9 Detection Error Trade-off (DET) Curve

In order to be able to compare two systems based on one number, the metric *Term Weighted Value (TWV)* was introduced. Although helpful in monitoring the system performance, DET curves cannot give such a number. The definition of TWV is as follows:

$$TWV(th) = 1 - \frac{1}{Q}\sum_{term}\left(p_{MISS}(term, th) + \beta p_{FA}(term, th)\right) \qquad (11)$$

where, β is a constant that gives an intuition on how valuable a “HIT” is when compared to many number of false alarms. Hence, TWV gives a number between negative infinity and 1. A system that succeeds to detect all of the occurrences of all terms with no false alarms will have a TWV of 1, and a system that gives no outputs will give a TWV of 0. Hence, it is theoretically possible to do worse than doing nothing at all. Similar to TWV, Actual Term Weighted Value (ATWV) is the TWV using the actual decisions. ATWV represents the system's ability to predict the optimal operating point given the TWV scoring metric. Maximum Term Weighted Value (MTWV) is the maximum value the TWV can get over all *th* values. Optimum Term Weighted Value (OTWV) is calculated using the best *th* value for each query. Supremum Term Weighted Value (STWV) is calculated by only considering the “HIT”s and ignoring the cost of “FA”s. It is simply the ratio of the system's correctly detected outputs to the actual number of occurrences averaged over all terms. Clearly;

$$ATWV \leq MTWV \leq OTWV \leq STWV \qquad (12)$$

Score Normalization

As stated in the earlier sections, the KWS systems are expected to return the hit hypotheses based on confidence scores that are assigned to them. The decision of a hypothesis being relevant or irrelevant is made based on this similarity scores. Since the system is developed before the run time, the user can only use one threshold value to decide the relevancy. On the other hand, keywords and utterances vary in length and acoustic ingredient. Because of this reason, the similarity scores for each keyword tend to vary in range, too. A very unlikely hypothesis for a specific keyword may have higher scores than the best (and relevant) hypothesis for some other keyword, for this reason.

In order to address this problem, the hypotheses of the keywords are normalized so that a single threshold value can be used for decision in the run time. Many normalization techniques have been used in the literature. First and the most widely used normalization technique is called the *Sum to One (STO) Normalization.* In STO normalization all hypothesis for a certain keyword are normalized by their sum, so that they add up to one. It is a great means of taking the ranges of hypotheses belonging to different keywords to the same scale by protecting their order. Another very widely used normalization technique is called the *Histogram Equalization (HE),* which rescales the hypotheses' scores so that the best hypothesis has a score of 1 and the worst has a score of 0 for all keywords. This normalization technique becomes useful in that it prevents the scores taking too small numbers, where this is the case for STO.

Another very pertinent score normalization technique that the conventional LVCSR-based KWS systems use called the *Keyword Specific Threshold (KST) Normalization.* In KST, the main idea is to optimize the TWV metric given in (11). However, the actual number of occurrences ($N_{target}(term)$) of a keyword is not known by the system developer. The estimation of $N_{target}(term)$ is done by the summation of each hypothesis. Since in the LVCSR-based approach the lattices yield posterior probabilities of the keyword existing in a certain utterance, the summation of these probabilities, i.e. scores, gives an estimate of $N_{target}(term)$. The optimal threshold for the KST normalization is then obtained using this estimate.

System Fusion

Fusion of heterogeneous systems has been studied and used in many detection and verification applications. The main idea is to utilize superiorities of one system to compensate for the impotencies of others in different areas. The fusion can take place at the search part, using different structures and levels, yet generally the fusion is applied to the system scores of different systems. One general fusion methodology is called the *majority voting,* in which when majority of the systems have high scores for the same hypothesis, the weighted average of their scores is used as a single output of the fusion system. This methodology is very useful for elimination of the false alarms.

MAIN FOCUS OF THE CHAPTER

This chapter aims to address the task of keyword search primarily for the low resource languages in order to compensate for the impotencies of the LVCSR systems trained on little data. As the main search methodology, the fruitful technique of frame-based QbE spoken term detection scheme is applied and it is extended to the keyword search problem. With this, the primary intention is to release the necessity for a high performance decoder and propose a new method for the retrieval of the terms out of the vocabulary of such an LVCSR system. This proposed methodology already carries the problems to be addressed in the task. For one, the frame-based QbE systems, which follow the sDTW recipe, obtains frame representations from the document (utterances) and the spoken queries. The methodology of this chapter also obtains such frame representations from the utterances using *deep neural networks* and uses the *posteriorgram* representations to be used in the DTW search. The query, however, which is given in text form in this task, has to be handled separately. Hence, the fist problem is modeling the text queries as *pseudo posteriorgrams*.

In QbE, the frames of the posteriorgrams are posterior vectors, obtained generally from neural network, hence in sDTW, as the distance metric, versions of *cosine distance* is used. However, in the methodology proposed in this chapter, the queries are modeled artificially and the cosine distance may not be the best similarity measure to be used between posterior vectors of the utterance. Hence, the second problem to be addressed is to learn a similarity measure to be used in accordance with the artificially modeled query frames.

In LVCSR based systems, the output scores of the hypotheses are posterior probabilities and an estimation of the actual occurrence number is feasible making it possible to use a normalization like the KST normalization. Furthermore, the number of hypotheses is small compared to the sDTW based search systems, where it is basically possible to propose a similarity score for any subsequence of the utterance. Because of this reason, a sum-to-one normalization is directly feasible for LVCSR based systems, where in sDTW based search systems; a pruning of unlikely subsequences is necessary to keep the STO-normalized scores in a big enough range. This pruning threshold is another optimization parameter to make the problem harder than it is. The third problem to be addressed is an suitable normalization scheme for the novel approach proposed in this chapter.

The fusion of heterogeneous systems is a non-trivial subject. For the *majority voting* to work, the score ranges of the two systems have to be aligned. However, the scoring methodologies of the LVCSR based systems and the sDTW-based systems are completely different, where the former outputs probability values for each hypothesis and the latter assigns similarity values through accumulated DTW distances for each hypothesis. A generally applied solution is to use STO normalization before combination so that the relative significances of each hypothesis have the same meaning across systems. However, this approach brutally cancels out the benefits of the individual normalization being studied as the third problem. Furthermore, as mentioned before, the number of hypotheses to be normalized in the sDTW based approach is significantly higher than the LVCSR based systems making it impossible to apply a proper STO normalization. Hence, the fourth problem is to work on a suitable fusion methodology to utilize powerful parts of each of the LVCSR and sDTW based systems.

In summary, the problems to be addressed for the methodology proposed in this chapter can be enumerated as follows:

Problem One: Modeling the text keyword into a proper frame level representation suitable for template base search.

Problem Two: Learning an optimal similarity measure to be used in sDTW for these representations.

Problem Three: Finding a proper score normalization technique to work properly with the sDTW based KWS output.

Problem Four: An efficient system fusion methodology for the combination of the sDTW based and LVCSR based systems.

METHODOLOGY

The solution to the first problem follows the recipe introduced in (Gündoğdu et al, 2016). In the beginning, of the system set-up, there is a set of telephone transcribed telephone conversations to train a DNN to obtain the posteriorgrams. After training the network until convergence, this DNN is used to obtain document posteriorgrams from the telephone conversations, on which the desired keyword search is to be conducted. The posteriorgram representations and the phone-frame alignments depicting which frame belongs to which phone are also obtained using this DNN. The flowchart of obtaining the posteriorgrams and the alignments are depicted in Figure 10.

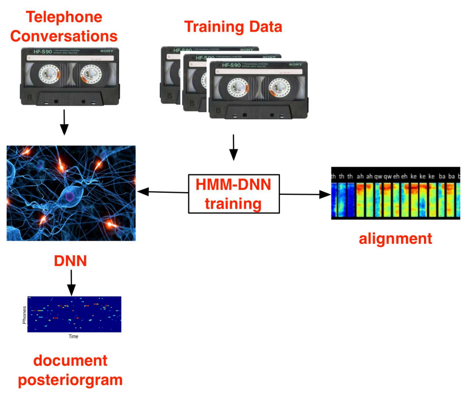


Figure 10 The flowchart for obtaining the document posteriorgrams and the training set frame-phone alignments.

From, the training set alignments, the estimations of statistics for phonemes can be obtained. Two of these statistics are the average phoneme posterior vectors depicting a way of inter-phone confusions introduced by the DNN and the average durations of each phoneme. Using these statistics, the pseudo query models can be models can be obtained with two methods. The text query is fist converted into sequences of phonemes depicting its pronunciation using a pronunciation lexicon for IV words, and a grapheme-to-phoneme conversion system for OOV words. Each phoneme in the language will basically have an index number for which the DNN outputs a posterior probability value. The initial step for obtaining index numbers from a keyword is depicted in Figure 11.

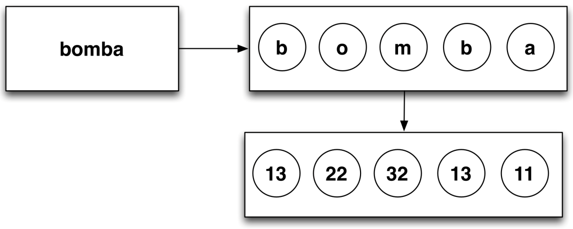


Figure 11: The methodology for obtaining indices from text queries.

From the indices, the statistics obtained from the training alignment are used and they are modeled as pseudo posteriorgrams. In the first *query modeling* methodology, one-hot vectors are used to represent phonemes. A one-hot vector is a vector where only one of the indices is 1 (the one depicted by the phoneme number in the pronunciation) and the other indices are zero. Each phoneme is repeated by the number of their average durations to model the phoneme durations in the artificial query model posteriorgram. This methodology is called *binary query modeling.* The other query modeling technique uses phoneme averages obtained from the training alignment instead of one-hot vectors to model the confusions. The second technique is called *average query modeling.* Sample query models obtained from the keyword *"arma"* using these query modeling techniques are demonstrated in Figure 12.

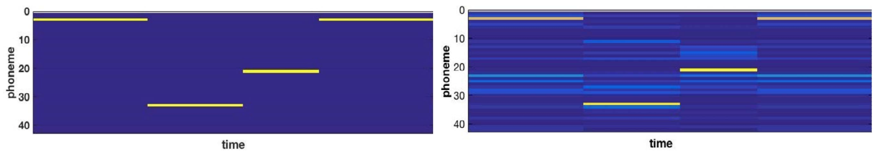


Figure 12: Pseudo Query Models obtained using binary query modeling (left) and average query modeling (right) for the sample keyword "arma".

Source (Gundogdu & Saraclar, 2017b)

Once the pseudo posteriorgrams are obtained for the keywords, the sDTW algorithm is used to get similarity scores for every subsequence of the utterance. These similarity scores are calculated from the average accumulated distance through the alignment path:

$$score = 1 - \frac{1}{|\Phi|}\sum_{k=1}^{T} d(\mathbf{q}_{i_k}, \mathbf{x}_{j_k}) \tag{13}$$

Similarity Measure Optimization

As it is given in (13), system similarity scores for the hypotheses offered by the proposed system is as relevant to the distance measure ($d(\mathbf{q}_i, \mathbf{x}_j)$) as it is to the alignment algorithm (Φ). Several distance measures can be used for this. In a generic distance measure, denoted $d(\mathbf{x}, \mathbf{y})$, the distance value between vectors **x** and **y** are desired to be high, if they are "far" or "dissimilar"; and small if they are "close" or "similar". For example, in the Euclidean space where the **x** and **y** values denote location information, the Euclidean distance measure (14) would be proper. On the other hand, for the posterior features being used in QbE, cosine distance (15) and logarithmic cosine distance (16) are more appropriate to use, because of their geometric and probabilistic interpretations.

$$d_{Euc}(\mathbf{x}, \mathbf{y}) = \sqrt{\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - 2\mathbf{x}^T\mathbf{y}} \tag{14}$$

$$d_{\cos}(\mathbf{x}, \mathbf{y}) = 1 - \frac{\mathbf{x}^T\mathbf{y}}{\|\mathbf{x}\|.\|\mathbf{y}\|} \tag{15}$$

$$d_{\log}(\mathbf{x}, \mathbf{y}) = -\log\left(\frac{\mathbf{x}^T\mathbf{y}}{\|\mathbf{x}\|.\|\mathbf{y}\|}\right) \tag{16}$$

This methodology, on the other hand, learns models the queries artificially making these distance metrics no longer appropriate for the new model. Hence, the second problem of learning a new similarity measure that would work better than the other ones is going to be addressed here. For this, the following Siamese neural network model is proposed.

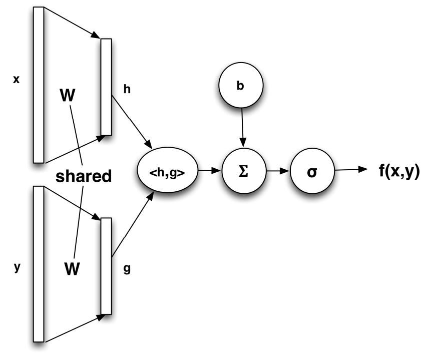


Figure 13 Siamese neural network based similarity measure learning scheme.

Source : (Gundogdu & Saraclar, 2017a)

Hence, once the parameters are learned the new similarity measure will be :

$$f(\mathbf{x},\mathbf{y})=\sigma\left(\mathbf{x}^T\mathbf{W}^T\mathbf{W}\mathbf{y}+b\right) \tag{17}$$

where, $\sigma(z)$ is the sigmoid function defined as

$$\sigma(z)=\frac{1}{1+e^{-z}} \tag{18}$$

In this model, the parameters to be optimized are as follows:

- **W :** Shared weight matrix, projecting the vectors onto another space.
- b **:** The bias value to take the new inner product to the range where the sigmoid non-linearity works better.

This similarity learning algorithm, first proposed in (Gundogdu & Saraclar, 2017), uses all pairs of frames in the training set alignment and finds the **W** and b values such that the total cross entropy cost is minimum.

$$J(\mathbf{W},b)=\sum_t -r_t\log(f_t)-(1-r_t)\log(1-f_t) \tag{19}$$

where f_t is $f(\mathbf{x}_t,\mathbf{y}_t)=\sigma\left(\mathbf{x}_t^T\mathbf{W}^T\mathbf{W}\mathbf{y}_t+b\right)$, the sigmoid similarity between tth (x,y) pair in the training set, and r_t is the label or the ground truth of this pairs. Since it is desired that the frames belonging to the same class (*friends*) give high similarity and ones belonging to different classes (*foes*) give low similarity, the following labels are used in the training.

$$r_t=\begin{cases}1, \text{ if } \mathbf{x} \text{ and } \mathbf{y} \text{ are friends}\\ 0, \text{ if } \mathbf{x} \text{ and } \mathbf{y} \text{ are foes}\end{cases} \tag{20}$$

For this cost function, the training was conducted using the online (stochastic) gradient descent. The gradient update rules for the parameters can be found as follows:

$$\begin{aligned}\mathbf{W} &= \mathbf{W} + \eta(r-f)(\mathbf{x}\mathbf{y}^T + \mathbf{y}\mathbf{x}^T) \\ b &= b + \mu(r-f)\end{aligned} \tag{21}$$

The similarity measure optimization problem can also be incorporated with the first problem of query modeling. Recently in (Gundogdu & Saraclar, 2017b), authors proposed such a scheme by adding on more layer to the Siamese neural network and having an asymmetric structure.

Score Normalization for the New Approach

The KWS approach being used in this chapter is very similar to the QbE-STD approaches except from the query modeling and similarity learning novelties. Hence, in addition to classical normalization techniques, this new methodology allows for new normalization techniques, as well.

The best performing normalization techniques for the proposed approach have been the versions of b-norm proposed in (Gündoğdu, & Saraçlar, 2017c). The distribution of scores per each keyword approximates a Gaussian distribution with a longer left tail, due to the fact that more number of telephone conversations is irrelevant than relevant. So the number of low scores is inherently higher than the number of high scores (explaining the longer left tail). Furthermore, the score distribution of some other keyword over the same audio archive will have different statistics, but a similar distribution. Hence, it is desired to modify the scores for all keywords so that they have the same relative meaning. This is very important in terms of operation with a single global threshold and fusion with other systems. The unnormalized distribution for 5 different keywords scores using the proposed retrieval approach is shown in Figure 14 (top-left). The sum-to-one and histogram equalization normalizations, along with the b-norm normalized distribution are shown in the same Figure on top-right, bottom-left and bottom-right, respectively

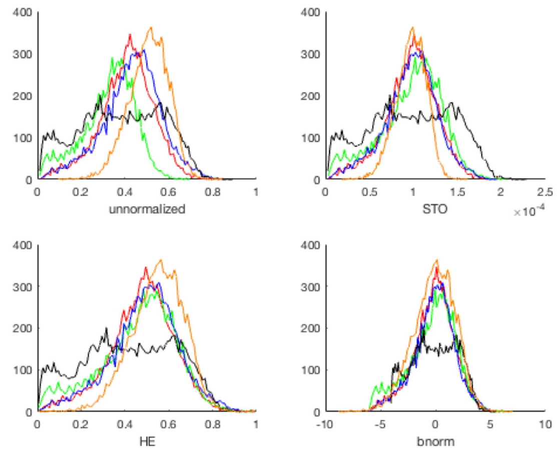


Figure 14 Effect of score normalization on system scores for 5 different keywords: x-axes denote the score of a detection, where y-axes denote the number of detections with this score. Curves with different colors denote different keywords.

Source : (Gundogdu & Saraclar, 2017c)

For a specific keyword, If the vector of scores of the hypotheses obtained from the search audio is denoted **s**, the b-norm (and versions of it), proposed in (Gündoğdu, & Saraçlar, 2017c) can be described as seen in (22).

$$\mathbf{s}_{norm} = \frac{\mathbf{s} - \text{median}(\mathbf{s})}{\hat{\sigma}(\mathbf{s})} \tag{22}$$

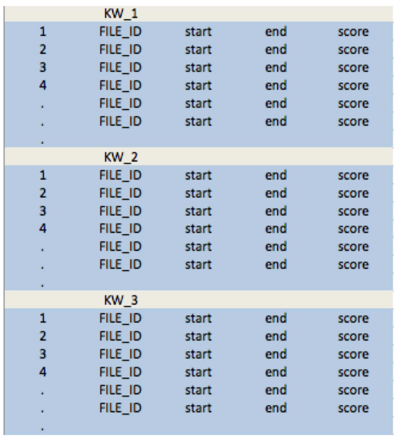
where, $\hat{\sigma}(\mathbf{s})$ is the standard deviation calculated over all scores greater than some chosen vicinity of the median of the distribution.

Fusion of sDTW and LVCSR-based Systems

The current system fusion methodologies implemented for KWS proved inefficient for the combination of LCVSR and sDTW based systems. The reason is that the LVCSR-based systems use KST normalization while the best practiced normalization technique for the sDTW-based systems is the b-norm (median normalization). The existing system fusion techniques are mainly dependent on taking the score ranges of the two systems to the same values by applying sum-to-one normalizations, while incorporating the information about the number of hits per system and per query. However, the application of sum-to-one normalization on top of the b-norm simply eliminates its efficiency and makes the proposed systems just STO normalized.

In order to utilize the dynamic power of the new normalization techniques, the combination procedure is vectorized and a time efficient algorithm was implemented. Each system output is a list that consists of the locations and scores of each hypotheses per keyword as shown in (23) and visualized in Figure 15.

$$\begin{gathered} sys(i) = \{kw(j) : \{file(k), start(k), end(k), score(k)\}\} \\ i = 1 \ldots \# systems,\ \ j = 1 \ldots \# keywords,\ \ k = 1 \ldots \# hits(i, j) \end{gathered} \tag{23}$$



	KW_1			
1	FILE_ID	start	end	score
2	FILE_ID	start	end	score
3	FILE_ID	start	end	score
4	FILE_ID	start	end	score
.	FILE_ID	start	end	score
.	FILE_ID	start	end	score
.				
	KW_2			
1	FILE_ID	start	end	score
2	FILE_ID	start	end	score
3	FILE_ID	start	end	score
4	FILE_ID	start	end	score
.	FILE_ID	start	end	score
.	FILE_ID	start	end	score
.				
	KW_3			
1	FILE_ID	start	end	score
2	FILE_ID	start	end	score
3	FILE_ID	start	end	score
4	FILE_ID	start	end	score
.	FILE_ID	start	end	score
.	FILE_ID	start	end	score
.				

Figure 15: A sample KWS system output scheme for evaluation.

The fusion algorithm aims to find the overlaps between score-lists of different systems for the same keyword and combine the scores of the overlapping hypotheses after some normalization. This chapter introduces a time efficient combination scheme for combination of heterogeneous systems. Each system is represented with a sparse 3-D matrix:

$$\mathbf{SYS}\{i\} \in \Re^{K \times U \times F} = \mathbf{SYS}\{i\}(j, file(k), start(k):end(k)) = score(k) \tag{24}$$

where K is the number of keywords, U is the number if files (utterences), and F is the maximum frame number for which the systems propose a hypothesis. The filling of the sparse matrix is explained in (24) and visualized on Figure 16.

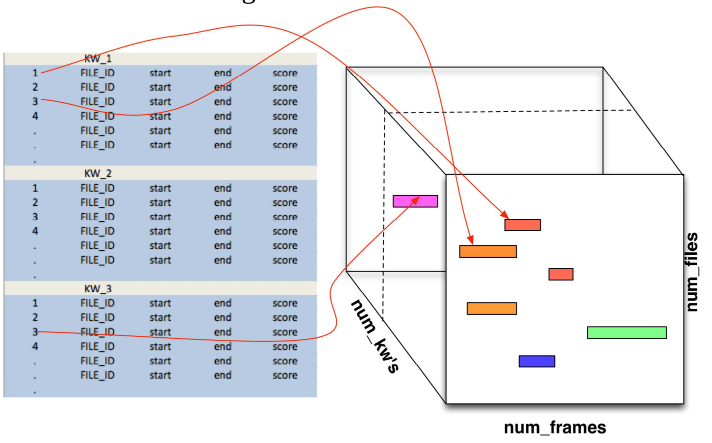


Figure 16: Methodology to model the system output as a sparse cubic structure.

Since the LCVSR based system is KST normalized, their scores range between 0-1 and the most confident hypotheses, which are for in-vocabulary terms, have scores close to 1. On the other hand, b-normalized sDTW based ssytems will have larger values for confident hits, due to the division by the standard variation. It is desirable to “trust” the LVCSR based systems for their hypotheses for the in-vocabulary words. Hence, the ranges of the two matrices are equalized by simply multiplying the LCVSR based system with the global maximum of the sDTW based system matrix.

$$\mathbf{SYS}_{norm}\{LCVSR\} = \mathbf{SYS}\{LCVSR\} \times \max(\max(\max(\mathbf{SYS}\{sDTW\}))) \tag{25}$$

Now that the whole scheme is vectorized and equalized, the overlap search is simply done via taking the weighted sum of the sparse system matrices (Figure 17) .For the experiments of this chapter, authors took equal weights for the systems. (26).

$$\mathbf{Fusion} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{SYS}\{i\} \tag{26}$$

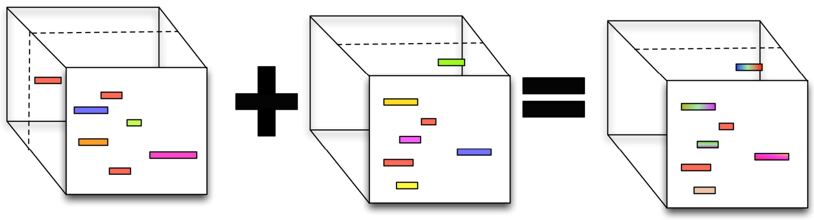
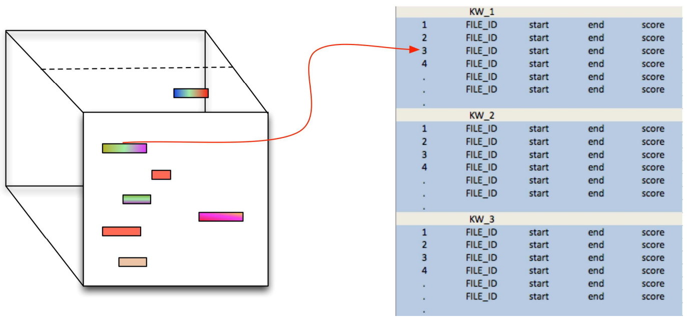


Figure 17: Procedure to obtain the fusion matrix cube from the two systems.

The fusion system matrix, which is a sum of sparse matrices, will again be sparse matrix. The ultimate list of system hypotheses upon the fusion is then extracted from this matrix by non-zero segments. Due to averaging, non-overlapping hypotheses will be suppressed. These also correspond to hypotheses where the minority rooted for. For other hypotheses, with majority votes, the overlapping part will have a relatively high score. Due to the fact that different systems will report different start and end locations for the same hypothesis, the scores for those will vary along frames on the combination matrix. The colors of the separated areas in Figure 16 correspond to different scores described by (24). Hence the fusion matrix on Figure 17 may have varying range of score values for each separated hypothesis. During the extraction of the hypothesis list from the matrix, their mean is taken for each separated hypothesis in the sparse cube, so that a measure of the overlap ratio is also integrated in the proposed combination methodology. This procedure is visualized in Figure 18.



	KW_1			
1	FILE_ID	start	end	score
2	FILE_ID	start	end	score
3	FILE_ID	start	end	score
4	FILE_ID	start	end	score
.	FILE_ID	start	end	score
.	FILE_ID	start	end	score
.				
	KW_2			
1	FILE_ID	start	end	score
2	FILE_ID	start	end	score
3	FILE_ID	start	end	score
4	FILE_ID	start	end	score
.	FILE_ID	start	end	score
.	FILE_ID	start	end	score
.				
	KW_3			
1	FILE_ID	start	end	score
2	FILE_ID	start	end	score
3	FILE_ID	start	end	score
4	FILE_ID	start	end	score
.	FILE_ID	start	end	score
.	FILE_ID	start	end	score
.				

Figure 18: The system output for evaluation is obtained from the fusion cube, by taking the means of the sparse locations recording their addresses.

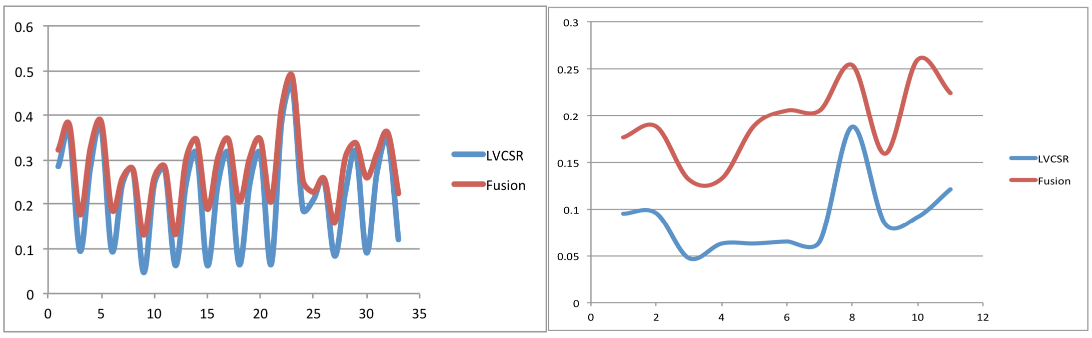
EXPERIMENTAL RESULTS

The techniques proposed in this chapter were tested on varying parts of 45 hours of telephone conversations from three low-resource languages. The keywords of interest were simulated to be taken from random phrases or words from each language. The term weighted value performances were calculated for the LVCSR-based baseline system and the fusion with the sDTW-based system. The initial experiments conducted on each language with the 10-hour telephone conversation data served as system development to learn the optimal operation thresholds for the detection. Hence, for the development experiments MTWV scores were calculated. On the last part of the experiments for each language, with the 5-hour telephone conversation recordings, both the ATWV and MTWV scores were calculated using the optimal decision thresholds learned in the fist part. The list of experiments can be seen in Table 1.

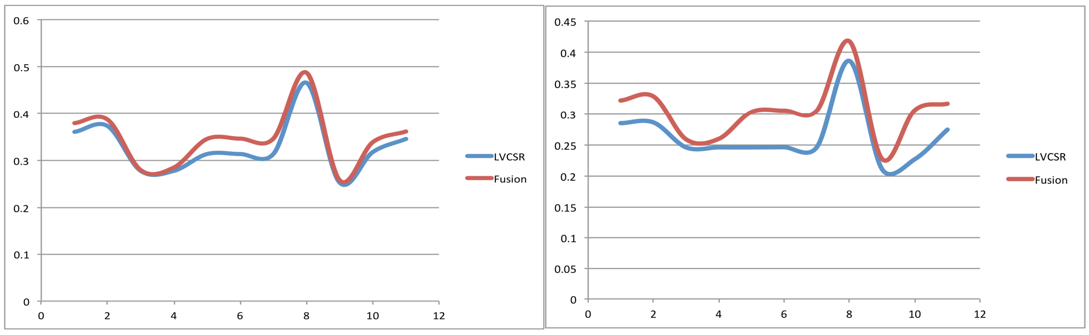
Table 1. List of Experiments.

Exp.No.	**Language**	**Keyword list size**	**Telephone conversation length**	**Keywords in the vocabulary?**
1	Turkish	307	10 hours	mixed (IV and OOV)
2	Turkish	88	10 hours	Only OOV
3	Turkish	219	10 hours	Only IV
4	Turkish	3171	5 hours	mixed (IV and OOV)
5	Turkish	1216	5 hours	Only OOV
6	Turkish	1955	5 hours	Only IV
7	Pashto	2065	10 hours	mixed (IV and OOV)
8	Pashto	600	10 hours	Only OOV
9	Pashto	1465	10 hours	Only IV
10	Pashto	4203	5 hours	mixed (IV and OOV)
11	Pashto	971	5 hours	Only OOV
12	Pashto	3232	5 hours	Only IV
13	Zulu	2000	10 hours	mixed (IV and OOV)
14	Zulu	806	10 hours	Only OOV
15	Zulu	1194	10 hours	Only IV
16	Zulu	3310	5 hours	mixed (IV and OOV)
17	Zulu	1112	5 hours	Only OOV
18	Zulu	2198	5 hours	Only IV

The MTWVs (for development and evaluation experiments) and the ATWVs (for the evaluation experiments), were plotted on Figure 19(a). It can be seen that the fusion system performance (denoted with red) is always better than the LVCSR-based system (denoted with blue). The valleys on the both lines correspond to the experiments with OOV-only experiments (e.g. 2, 5, 6 etc.) and the peaks correspond to IV only experiments (e.g. 3, 6, 9 etc). Obviously, the both systems perform better when the keyword is in the vocabulary of the LVCSR system (IV). On the other hand, it can be observed that the fluctuation between IV and OOV performance is much lower for the fusion system. The system performances for the OOV-only experiments are seen in Figure 19 (b). The improvement of the fusion system can more clearly be seen when the OOV-only experiments are considered. The fusion system performance is almost always more than twice of the baseline system. Although, for the IV-only experiments, the LVCSR based system is expected to perform at a satisfactory level, the experiments show that the fusion with the sDTW-based system provides an improvement on all experiments, as can be seen in Figure 19 (c). The experiments conducted over a mixed set of IV and OOV terms are more akin to realistic, real-world scenarios. The clear performance improvement that can be seen in Figure 19 (d) is the demonstration of this experiment set.



(a) (b)



(c) (d)

Figure 19: TWV performances evaluated on the experiments conducted: x-axes denote set of experiments, and y-axes denote the MTWV performance in the experiment. (a) All experiments given in Table 1, (b) OOV-only experiments, (c) IV-only experiments, (d) mixed (IV and OOV) experiments.

CONCLUSION AND FUTURE RESEARCH DIRECTIONS

In this chapter, a similarity optimization based keyword search technique is introduced and the novel query modeling, score normalization, and system fusion techniques were presented. These techniques were tested on several experimental set-ups on three different languages and various different keyword-lists provided by the IARPA Babel Program. It was shown that, this approach not only provide a new an powerful method for the retrieval of OOV terms, but also improves the conventional LVCSR-based systems' performances over all experiments.

As a future research direction, the authors recommend modeling the query using generative recurrent neural networks. Instead of modeling the phonemes by one-hot vectors (or with fist order statistics) and applying the same phoneme model through its estimated duration, a properly trained generative recurrent neural network would model a more realistic query posteriorgram.

ACKNOWLEDGMENT

This study was supported in part by the Intelligence Advanced Research Projects Activity

(IARPA) via Department of Defense U.S. Army Research Laboratory (DoD/ARL) contract number W911NF- 12-C-0012 and the Scientific and Technological Research Council of Turkey (TUBITAK) under Project 116E076. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government. The authors would like to thank Mr. Bolaji Yusuf and Mr. Recep Doga Siyli for their invaluable help in conducting the experiments.

REFERENCES

Albrecht, T., & Muller, M. (2009). Dynamic Time Warping (DTW). *Information Retreival for Music and Motion*, 70-83.

Borges, J. L. (1998). *The library of Babel.* Collected fictions.

Can, D., & Saraclar, M. (2011). Lattice indexing for spoken term detection. *IEEE Transactions on Audio, Speech, and Language Processing, 19*(8), 2338-2347.

Chen, G., Yilmaz, O., Trmal, J., Povey, D., & Khudanpur, S. (2013, December). Using proxies for OOV keywords in the keyword search task. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on* (pp. 416-421). IEEE.

Fiscus, J. G., Ajot, J., Garofolo, J. S., & Doddingtion, G. (2007). Results of the 2006 spoken term detection evaluation. In *Proc. sigir* (Vol. 7, pp. 51-57).

Gandhe, A., Qin, L., Metze, F., Rudnicky, A., Lane, I., & Eck, M. (2013, December). Using web text to improve keyword spotting in speech. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on* (pp. 428-433). IEEE.

Garcia, A., & Gish, H. (2006, May). Keyword spotting of arbitrary words using minimal speech resources. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*(Vol. 1, pp. I-I). IEEE.

Gündoğdu, B., & Saraçlar, M. (2017a). Distance Metric Learning for Posteriorgram Based Keyword Search. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP) New Orleans, USA.*

Gündoğdu, B., & Saraçlar, M. (2017b). Similarity Learning Based Query Modeling for Keyword Search. *Proc. Interspeech 2017*, 3617-3621.

Gündoğdu, B., & Saraçlar, M. (2017c). Novel score normalization methods for keyword search. In *Signal Processing and Communications Applications Conference (SIU), 2017 25th* (pp. 1-4). IEEE.

Gündogdu, B., Sarı, L., Çetinkaya, G., & Saraçlar, M. (2016). Template-Based Keyword Search with Pseudo Posteriorgrams. In *2016 24th Signal Processing and Communications Applications Conference (SIU)* (pp. 973-976).

Haidt, J. (2006). *The happiness hypothesis: Finding modern truth in ancient wisdom*. Basic Books.

Harper, M. (2014). *IARPA babel program. https://www. iarpa. gov/index. php/research-programs/babel.*

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A. R., Jaitly, N., & Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine, 29*(6), 82-97.

Huang, X., Acero, A., Hon, H. W., & Foreword By-Reddy, R. (2001). *Spoken language processing: A guide to theory, algorithm, and system development*. Prentice hall PTR.

Hundt, C., Schmidt, B., & Schömer, E. (2014, September). Cuda-accelerated alignment of subsequences in streamed time series data. In *Parallel Processing (ICPP), 2014 43rd International Conference on* (pp. 10-19). IEEE.

Karakos, D., & Schwartz, R. (2014). Subword and phonetic search for detecting out-of-vocabulary keywords. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Lee, L. S., Glass, J., Lee, H. Y., & Chan, C. A. (2015). Spoken content retrieval—beyond cascading speech recognition with text retrieval. *IEEE/ACM Transactions on Audio, Speech, and Language Processing, 23*(9), 1389-1420.

Lee, S. W., Tanaka, K., & Itoh, Y. (2005, March). Combining multiple subword representations for open-vocabulary spoken document retrieval. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on* (Vol. 1, pp. I-505). IEEE.

Mohri, M., Pereira, F., & Riley, M. (2002). Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, *16*(1), 69-88.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE, 77*(2), 257-286.

Rodriguez-Fuentes, L. J., & Penagarikano, M. (2013). MediaEval 2013 spoken web search task: system performance measures. *n. TR-2013-1, Department of Electricity and Electronics, University of the Basque Country*.

Saraclar, M., & Sproat, R. (2004). Lattice-based search for spoken utterance retrieval. *Urbana*, *51*, 61801.

Sarı, L., Gündoğdu, B., & Saraçlar, M. (2015). Fusion of LVCSR and Posteriorgram Based Keyword Search. *INTERSPEECH 2015*, 824-828.

Sakoe, H., & Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing, 26*(1), 43-49.

Shen, W., White, C. M., & Hazen, T. J. (2009). *A comparison of query-by-example methods for spoken term detection*. MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB.

Van Thong, J. M., Moreno, P. J., Logan, B., Fidler, B., Maffey, K., & Moores, M. (2002). Speechbot: an experimental speech-based search engine for multimedia content on the web. *IEEE Transactions on multimedia*, *4*(1), 88-96.

Wang, D., Frankel, J., Tejedor, J., & King, S. (2008, March). A comparison of phone and grapheme-based spoken term detection. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on* (pp. 4969-4972). IEEE.