USING CROSSLINGUAL INFORMATION FOR KEYWORD SEARCH IN LOW RESOURCE LANGUAGES

by

Bolaji Yusuf

B.S., Electronics Engineering, Istanbul Kültür University, 2015

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical and Electronics Engineering

Boğaziçi University

2018

ACKNOWLEDGEMENTS

I would like to express my sincerest gratitude to my advisor, Prof. Murat Saraçlar, for his guidance throughout my studies. His vision and patient exhortations coupled with an endless pool of knowledge have been instrumental in making this thesis a reality.

I owe a debt of gratitude to Batuhan Gündoğdu for showing me the ropes; I cannot overstate the impact of his help starting, and his constructive, optimistic advice throughout my M.S. studies.

I am thankful to the friends and colleagues at the BUSIM, WCL and BETA labs, whose Ubuntu implementation has made K-Block an enjoyable experience. I would especially like to thank Can Gürsoy and Alican Gök for their friendship, their hospitality and the endless laughter.

This study was supported in part by the Scientific and Technological Research Council of Turkey (TUBITAK) under Project 116E076.

The data used to conduct the experiments in this study was collected within the IARPA Babel program.

To my father, Olatunji Yusuf, my gratitude has no bound. His inquisitiveness even in excellence and his unflinching acceptance of responsibility have given me a blueprint that I hope to emulate. To my mother, Adenike Yusuf, I give the most thanks. In the strength of her unwavering dedication to compassion and her boundless curiosity, I have had an even loftier ideal to aim for. To both my parents, my walls, am I grateful in more ways than I could fully express for the life I have been provided, and to them, do I dedicate this thesis. Yèyé ni wúrà, baba ni jígí.

ABSTRACT

USING CROSSLINGUAL INFORMATION FOR KEYWORD SEARCH IN LOW RESOURCE LANGUAGES

Keyword search (KWS) is a subtask of spoken content retrieval that aims to solve the problem of locating a written query within a large, unlabeled spoken document. The dominant approach to KWS involves transcribing the document using an automatic speech recognition (ASR) system and conducting the search on indexes obtained from the ASR lattices. The large vocabulary continuous speech recognition (LVCSR) systems used to decode the document typically require enormous amounts of labeled data to give good recognition and, subsequently, search accuracy. Therefore, KWS models built for languages for with relatively little labeled training data need to contend with the deterioration in search performance that accompanies a decline in ASR performance. This deterioration is exacerbated by the increased incidence of search terms that are out of vocabulary (OOV) of the training data. One way of improving KWS performance in such a setting is to leverage information from other languages. In this work, we use a multilingual representation to build a vocabulary agnostic KWS model. The multilingual bottleneck (BN) representation, obtained from a neural network trained on the source languages, is used to train a metric learning based KWS engine in the target languages. Experiments on the low resource datasets from the IARPA Babel Program show the benefits of using the proposed system as an alternative to, or in tandem with, more traditional multilingual models. In an extremely low resource setting, the performance of the proposed system exceed that of the baseline system (also trained with multilingual data). Furthermore, in a milder low resource setting, the proposed system performs better on OOV term retrieval than the baseline. In either setting, we show that combining the results from both systems yields a robustness against OOV terms and better overall performance.

ÖZET

KISITLI KAYNAKLI DİLLERDE ANAHTAR SÖZCÜK ARAMA İÇİN DİLLER ARASI BİLGİ KULLANIMI

Anahtar Sözcük Arama (ASA), metin halinde verilen bir sorgunun büyük, etiketlenmemiş bir konuşma dokümanı içinde bulunmasını amaçlayan, konuşma arama problemlerinin bir alt kümesidir. ASA'ya en hakim yaklaşım, Otomatik Konuşma Tanıma (OKT) sistemleri ile konuşma verisini metne çevirme ve aramayı OKT örülerinden elde edilen dizinler içerisinde gerçekleştirmektir. Bu örüleri elde etmekte kullanılan Geniş Dağarcıklı Sürekli Konuşma Tanıma (GDSKT) sistemleri iyi bir konuşma tanıma, ve dolayısıyla anahtar sozcuk arama performansi icin tipik olara muazzam miktarlarda etiketli veri gerektirmektedir. Bu nedenle, nispeten az etiketli eğitim verisi olan diller için inşa edilen ASA modelleri, OKT performansında yasanan düşüşe eşlik eden arama performansındaki bozulma ile mücadele etmelidir. Bu bozulma, egitim verisinin dagarcigi disinda (DD) kalan terimlerin mevcudiyeti ile daha da kotuye gitmektedir. ASA performansının böyle bir ortamda geliştirilmesinin bir yolu, diğer dillerden gelen bilgilerden yararlanmaktır. Bu çalışmada, dağarcıktan bagimsiz bir ASA modeli oluşturmak için çok dilli bir gösterim kullanmaktayiz. Kaynak diller üzerinde eğitilmiş bir sinir ağından elde edilen çok dilli darboğaz gösterimleri, hedef dillerde uzaklık ölçütü öğrenme temelli ASA motorunu eğitmek için kullanılır. IARPA Babel Programındaki kısıtlı kaynaklı veri kümeleriyle gerçekleştirilen deneyler, önerilen sistemin daha geleneksel çok dilli modellere alternatif olarak veya onlarla birlikte kullanmanın yararlarını ortaya koymuştur. Son derece kısıtlı kaynak ortamında, önerilen sistemin performansı, (yine çok dilli verilerle de eğitilmiş) temel sisteminkini aşmaktadır. Kaynak kısıtlarının daha gevsetildigi bir başka kısıtlı kaynak ortamında ise , önerilen sistemin DD terimlerde temel sistemden daha yüksek başarıma sahip olduğu görülmüştür. Her

iki durumda da, iki sistemin sonuçlarını birleştirmenin daha da iyi bir DD direnci ve yüksek bir genel performans sağladığı gösterilmektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$\mathcal{B}$	Matrix of optimal beginning frames
$d(\mathbf{x}, \mathbf{y})$	Local distance function between $\mathbf{x}$ and $\mathbf{y}$
$\mathcal{D}$	Matrix of accumulated distances
$D_{\hat{\mathbf{\Pi}}}(\mathcal{X},\ \mathcal{Y})$	Total dynamic time warping distance between $\mathcal{X}$ and $\mathcal{Y}$
$\mathcal{H}(\mathbf{x})$	Information entropy of the probability vector x
J_{CE}	Cross entropy cost function
J_H	Entropy cost function
$\mathcal{L}$	Matrix of path lengths
p(x)	Probability density function of the continuous random variable x
P(x)	Probability mass function of the discrete random variable x
$\mathcal{Q}$	Set of queries
t_{beg}	Beginning time of a hypothesized detection
t_{end}	Ending time of a hypothesized detection
β	Miss-false alarm parameter for term weighted value
θ	Term weighted value threshold
$\mathbf{\Pi}$	Dynamic time warping alignment path
$\hat{\mathbf{\Pi}}$	Optimal dynamic time warping alignment path
$\sigma(x)$	Logistic sigmoid function applied to x

LIST OF ACRONYMS/ABBREVIATIONS

ANN	Artificial Neural Network
ASR	Automatic Speech Recognition
ATWV	Actual Term Weighted Value
BN	Bottleneck
Cnxe-min	Minimum Normalized Cross Entropy
Cnxe	Normalized Cross Entropy
DML	Distance Metric Learning
DNN	Deep Neural Network
DTW	Dynamic Time Warping
EDML	Extended Distance Metric Learning
ELR	Extremely Low Resource
FA	False Alarm
G2P	Grapheme to Phoneme
GMM	Gaussian Mixture Model
GPU	Graphical Processing Unit
HAC	Hierarchical Agglomerative Clustering
HMM	Hidden Markov Model
IPA	International Phonetic Alphabet
IV	In Vocabulary
JDML	Joint Distance Metric Learning
KST	Keyword Specific Thresholding
KWS	Keyword Search
LCMA	Length-Constrained Minimum Average
LLP	Limited Language Pack
LR	Low Resource
LSH	Locality Sensitive Hashing
LVCSR	Large Vocabulary Continuous Speech Recognition
MFCC	Mel Frequency Cepstral Coefficients

MTWV	Maximum Term Weighted Value
OOV	Out of Vocabulary
OTWV	Optimum Term Weighted Value
PLP	Perceptual Linear Predictive
QbE	Query by Example
ReLU	Rectified Linear Unit
SDTW	Segmental Dynamic Time Warping
SSDTW	Subsequence Dynamic Time Warping
STD	Spoken Term Detection
STO	Sum-to-one
STWV	Supremum Term Weighted Value
TDNN	Time Delay Neural Network
TWV	Term Weighted Value
WER	Word Error Rate
WFST	Weighted Finite State Transducer

1. INTRODUCTION

The ubiquity of devices capable of recording speech and the ever decreasing cost of storage have resulted in a large trove of spoken data in the form of audiovisual lectures, news broadcasts, podcasts, senate and courthouse recordings, interrogations, call-center recordings, telephone wiretaps etc. The ability to search such archives efficiently would assist significantly in research, media content consumption and law enforcement.

While the search can be, and often is, conducted as text retrieval on the documents' accompanying meta-data, any such search would have to rely on rigorous documentation by the content provider and would probably return an undesirably unwieldy result to the end user especially if the document is too long. Keyword search (KWS) involves trying to find the exact location of a written query of arbitrary length within a spoken document. The KWS system returns a list of locations within the document hypothesized to contain the query term along with a relevance score for each location.

The contemporary approach to KWS involves transcribing the audio archive into words with a large vocabulary continuous speech recognition (LVCSR) system and then conducting textual information retrieval on the transcription. The document is decoded into lattices which are directed acyclic graphs that contain hypothesized transcriptions of the utterance. An inverted index that maps words to their locations in the lattices is then generated to be used for search. Since the document is decoded using a language model with a fixed vocabulary, the words in the index are limited to those in this lexicon. This leads to the problem of out-of-vocabulary (OOV) words. Even with a large lexicon, one would be hard pressed to cover all the words types in a language. Proper nouns, infrequent words, loan words, neologisms and common typing errors as well as morphological inflections in morphologically rich languages are especially difficult to cover. This OOV issue is particularly endemic in low resource languages where the lexicon size is considerably smaller than what would otherwise be expected for an LVCSR system.

Any query term that contains an OOV word is automatically undetectable by an LVCSR based KWS system. This fact is particularly irritating considering that the term can have any number of words only one of which has to be OOV to condemn the whole term. One approach to managing OOV queries has been to construct lattices of subword units such as phones, syllables or morphs [2–4]. The relaxation of the lexical constraint that enables subword lattices to retrieve of OOV terms comes at the cost of increased false alarms by the system; therefore, systems that utilize word lattices for in-vocabulary (IV) term retrieval and subword lattices for OOV retrieval have been proposed [5, 6]. Of course, these amalgam approaches incur the computational cost of multiple decoding passes. Another method of dealing with OOV terms involves lexicon expansion prior to decoding. This involves extending the lexicon with automatically generated pronounciations and thus reducing the OOV rate [7]. A more robust approach which relies less on the ability to accurately predict query words is the proxy keyword method. This involves the substitution of acoustically similar "proxy" words for the OOV ones [8, 9]. The word replacement is carried out with the use of confusion models that facilitate efficient insertion, deletion or substitution of phones to transform the query word into the proxy.

The KWS methodology used in this work is inspired by another branch of speech retrieval called query-by-example (QBE) spoken term detection (STD). In QBE-STD, the task is locating a short spoken query, as opposed to a written one, in a much larger speech document. The template matching techniques developed to to perform QBE-STD have been shown to be serviceable even in zero resource settings and so are independent of the existence of the search term in the training vocabulary.

The main contributions of this thesis are as follows:

(i) We propose a representation of the multilingual bottleneck features that is feasible for dynamic time warping based search. To reduce the deleterious effect of data scarcity to keyword search in low resource languages, we propose using a vocabulary agnostic dynamic time warping system with a representation transferred from other languages. After observing that the crosslingual representation

is not well separable, we propose using a Gaussian mixture model to generate posterior features on which we then conduct the search. Furthermore, we investigate the effects of training the model with data from other languages and show that, in cases of extreme data scarcity, it is the multilingually trained models performs better than the monolingual ones as well as the LVCSR baseline. With enough data in the target language though, the monolingual models we trained performed better than their multilingually trained counterpart.

(ii) We extend the distance metric learning based approach to fully utilize the multilingual representation we have learned. Since the performance of our dynamic time warping model depends on three things, namely; the document representation, the query model and the distortion function used to discriminate between them, we propose a neural network architecture to jointly optimize all three. By prepending a document representation learner (that takes the bottleneck features as input) and a generative query model to a distance metric learner, we obtain a unified framework to optimize our search. The search conducted with this framework performed considerably better than the Gaussian posteriorgram approaches. In a very low resource setting, we show that this method also outperforms the LVCSR baseline, even when the baseline lexicon is augmented considerably. In a more moderate low resource setting, we show that although, compared to the LVCSR baseline, the proposed method performs worse on in-vocabulary terms, it is significantly better on out-of-vocabulary ones and combining the two systems' results results in further gains.

(iii) We propose a method of compressing the features used for search, and thus reduce the memory footprint of the dynamic time warping algorithm. First we modify the metric learner to have intermediate representations that can be interpreted as probability mass functions. By minimizing the entropy of these distributions, we are able to produce a very sparse representation of the features space which is store in memory and used for search. We show that the degradation in performance is minimal at good compression rates.

(iv) We improve upon existing score normalization techniques. Although the raw scores returned by our system have distributions that vary by keywords, we mea-

sure performance using a metric that uses a global threshold to determine the relevance of each result. To meet this need for a normalization of scores across keywords, we propose a parametrized generalization to the median normalization method. The normalization uses median as a "typical" member of the set and normalizes around it. Parameterizing enables us to get the "typical" values at various percentiles and select the one that performs best.

(v) We propose a system combination method based on good separation of relevant and irrelevant scores of each system. Given that the LVCSR baseline and the proposed algorithm perform are trained in very different manners and perform well on different subtasks of the whole task, it is reasonable to expect that a combination of the results from the two would perform better than using either individually. To this end, we propose a fusion based on the normalized cross entropy which measures how-well calibrated a system is, i.e, how well separated are relevant hypothesis from irrelevant ones. For each system, we learn a transformation which, when applied to the scores, minimizes the normalized cross entropy for that system. After transforming the scores in this way, fusion can be done by simple addition without much risk of boosting irrelevant hypotheses from either system. Using a parametrized transformation, we are able to transfer to previously unseen datasets; furthermore, by using only two parameters, we reduce the risk of overfitting to the set on which the parameters are learned. We show that fusing the systems in this manner resulted in gains across languages and system configurations.

The rest of this thesis is organized thus:

- In Chapter 2, some background knowledge about the problem domain is provided.
- In Chapter 3, the models proposed in this thesis are described in depth.
- In Chapter 4, the empirical results and discussion are provided. In addition, the evaluation metrics are discussed.
- In Chapter 5, we draw our conclusions on the thesis.

2. BACKGROUND

The focus of this thesis is using a multilingual representation to bootstrap keyword search in low resource languages in a manner that is insensitive to OOV queries. This chapter provides background information on keyword search systems and the use of multilingual features in speech recognition and retrieval.

2.1. LVCSR based keyword search

LVCSR based KWS systems use a speech recognizer to transcribe the speech into text and then conduct the search on the text. The LVCSR system outputs lattices from which an index is constructed. The query is then searched in this index. Since automatic speech recognition forms the backend for this and most other approaches to KWS systems, it will be described first.

2.1.1. Statistical ASR

The transduction of spoken utterances into linguistic hypotheses is generally done in a statistical manner. The procedure can be described as predicting the most likely word sequence, W, given the observed acoustic sequence, X. The words are generally split into sequences of phones, G e.g. (boş contains the phones b, o and ʃ), which also have to be decoded. The search can be expressed as:

$$\begin{aligned} \hat{W}, \hat{G} &= \arg\max_{W,G} P(W|X) && (2.1) \\ &= \arg\max_{W,G} \frac{P(W)P(G|W)p(X|G)}{p(X)} && (2.2) \\ &= \arg\max_{W,G} p(X|G)P(G|W)P(W) && (2.3) \end{aligned}$$

where $\hat{W}$ and $\hat{G}$ are the estimated word and phone sequences respectively, $p(X|G)$ is referred to as the acoustic model, $P(G|W)$ is referred to as the pronunciation model and $P(W)$ is referred to as the language model [10].

The acoustic sequence, X, is typically a perceptually inspired representation of speech such as perceptual linear prediction (PLP) or mel-frequency cepstral coefficients (MFCC). The acoustic model assigns a probability to each vector in X conditioned on the underlying phone sequence. The language model gives the probability of each word sequence and is typically implemented as a Markov chain from which the probability of a word given a finite history of words can be obtained. The language model serves to limit the search space and enable the distinction of acoustically similar word sequences (such as *Eye two have to oranges* vs *I too have two oranges*) by assigning higher probabilities to syntactically more likely word sequences [11].

Hidden Markov models (HMMs) are commonly used to model the temporal variations in speech. Before the resurgence of neural networks, the observation probability distributions of the HMM states were modeled with Gaussian mixture models (GMMs). While the complexities (and capacities) of earlier artificial neural networks (ANNs) were limited, the proliferation of graphical processing units (GPUs), along with improved training algorithms, has precipitated the development of deeper neural networks with large output layers capable of modeling the context dependent phonemes typically used as HMM states [12].

2.1.2. ASR Output

When decoding an acoustic sequence, an ASR system must search through a large number of possible word sequences. Since keeping track of all such sequences is impractical, unlikely hypothesis are pruned heuristically as the decoder runs [13]. The graph of remaining hypotheses, called a lattice, along with their timing information and scores can be stored for further use.

Although it is possible to construct the search index from one-best ASR output, the recall rate of such an index is likely to be very low for any system with a non-negligible word error rate (WER). For instance, if the ASR mistranscribes “I too have two oranges” as “I do have two oranges”, searching the document for the former in a one-best index would automatically return a false negative. Therefore, it is necessary to

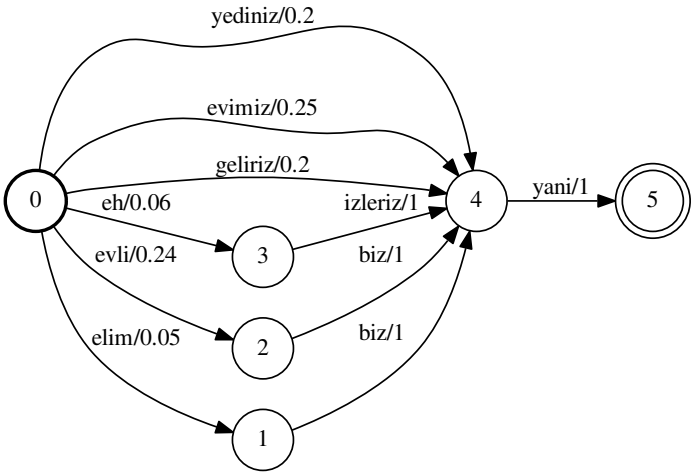

Figure 2.1. An example of an ASR lattice.

index the lattice or some such similar structure. In Figure 2.1, the lattice is represented as a weighted finite state automaton (WFSA) whose arcs carry pairs of word and their score (likelihood). The word sequence hypotheses are given by paths from the initial to final state.

Confusion networks provide an alternative, approximate, representation for the ASR output lattice [14]. Confusion networks are constructed by clustering temporally proximate arcs to form confusion sets. Figure 2.2 shows a confusion network obtained from the lattice in Figure 2.1.

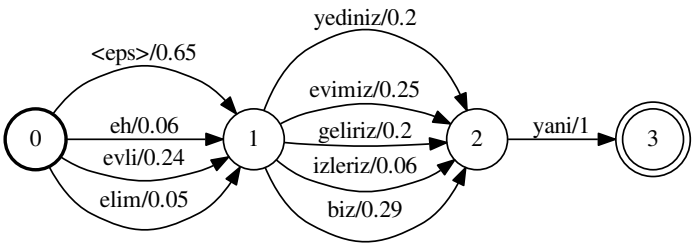

Figure 2.2. A confusion network.

As with the lattices, the ASR transcription hypotheses are represented by paths through the confusion network. Note, however, that equivalent transcriptions are given different scores by the two graphs and they might even result in different best path labels. In particular, the best path through the confusion network in Figure 2.2 ("biz yani") does not even exist in the original lattice. In general, the best path through a confusion minimizes the WER while the best path through a lattice minimizes the sentence error rate. Furthermore, the confusion network introduces hitherto nonexistent paths. This could result in higher recall and lower precision KWS.

2.1.3. Index and Search

The index is a structure that allows efficient mapping from (query) word to location within an utterance. The index contains all the words in LVCSR lattice along with their relevance score.

A timed factor transducer (TFT) converts the lattice weighted finite state transducers (WFSTs) into an inverted index represented as a WFST mapping word to utterance with a three-way weight of beginning time, end time and posterior score.

The search is conducted by composing [15] the query, represented as a WFST with the index, removing ϵ transitions and sorting. This results in an automaton with single-arc paths whose labels carry utterance information and whose weights carry the location and relevance information.

2.2. OOV term retrieval

The word-based LVCSR-KWS approach has the limitation of requiring beforehand a knowledge of all the query words it would handle. Even with a large lexicon, it is virtually impossible to cover all possible words in a language. Query terms that include words outside the lexicon used to train the LVSCR system are referred to as out-of-vocabulary. The presence of OOV terms can significantly hamper the performance of a KWS system especially in low resource settings where the lexicon is severely

limited, and in morphologically rich languages like Turkish, Finnish and Zulu where it is difficult to cover all possible word inflections. Searches for neologisms and loan words such as names of foreign celebrities, dances, cocktails etc. also serve to exacerbate this issue in an increasingly interconnected and multicultural world.

Since the absence of OOV words from the lexicon of an LVCSR system precludes them from appearing in the lattice (and index), it is necessary to develop methods capable of handling the OOV limitations of word lattice based KWS. Some of those methods will be explored in this section.

2.2.1. Subword Units

Words are the most commonly used units for ASR and KWS. For closed vocabulary tasks, it is possible to cover the required vocabulary in training. For open vocabulary tasks, such as search, such complete coverage is virtually impossible. Subword language models improve the coverage and have been shown to improve ASR performance in highly inflective languages.

Several subword units have been proposed for language modeling including linguistic units such as phones, syllables and morphemes. Since linguistically motivated subword units require language specific knowledge that might be expensive to get, data driven subword units such as graphones, particles and morphs have also been studied. The high OOV coverage of subword models generally comes at a cost of precision. This can be explained by the difficulty of recognizing units of such short duration as well as the laxer lexicon constraint [16].

Since word and subword models excel at different subtasks of KWS, hybrid models that take advantage of the merits of both have been proposed such as using joint indexes [17], using word models for in-vocabulary (IV) term retrieval and subword ones for OOV [4, 5].

2.2.2. Lexicon expansion

Using openly available textual sources to augment the LVCSR lexicon and language model can help reduce the OOV rate for low resource languages and, thus, improve the performance of KWS systems [18]. In conjunction with an OOV detection scheme [19], this approach has been shown reduce the OOV rate by up to half [7]. One drawback of using lexicon expansion to counteract the OOV issue is that the efficacy of such an approach is tied to how well the system is able to predict the OOV words used in search and add them to the lexicon.

2.2.3. Proxy keywords

The inability to accurately predict the query terms limits the effectiveness of lexicon expansion methods. One technique that works well without advance knowledge of query terms is the use of acoustically similar "proxy" words [8, 9].

Using a grapheme-to-phoneme (G2P), such as that described in [20], the pronunciation of an OOV word is obtained. A phone confusion model is then used to find IV words with similar pronunciation by substituting, removing or inserting phones. Based on the rationale that the ASR system would transcribe the OOV words as acoustically similar ones, the proxies are then used for search. The whole procedure can be implemented as a series of WFST compositions and optimization, resulting in a modified query WFST which is then composed with the index FST described in Section 2.1.3. With sufficient pruning, the proxy keyword WFST size, consequently, the rate of false alarms due to proxy overgeneration, can be managed. It is also noteworthy that the proxy keyword plugs into a word index without any need to re-decode the document and reconstruct the lattice thus avoiding the computational overhead of hybrid word-subword approaches.

2.2.4. Template matching for keyword search

A recently proposed OOV-robust approach uses the dynamic time warping algorithm to match a representation of the query to one of the document. With a document represented as a phone posteriorgram, a query is generated as a sequence of binary or average phone pseudo-posteriors and an inner-product based similarity is used for the dynamic time warping [21]. In [22], a distance metric learning framework is introduced for as a replacement of the geometric similarities previously used and the average query model is employed. To bypass the make-do average query model, in [23], a query model is learned jointly with the distance metric. The fact that the algorithm matches strictly acoustic sequences results in a better OOV term performance than the LVCSR approach since it employs no language model and has a minimized lexicon dependency; however, this results in lowered IV term performance since the linguistic data is not leveraged.

2.3. Multilingual Networks for Low Resource ASR and KWS

Due to the paucity of transcribed data endemic in low resource languages, the ability to transfer knowledge from other, resource-rich, is an attractive area of research. In particular, the use of multilingually trained neural networks to improve the ASR and KWS in target languages has been studied extensively. Some of the previous work will be summarized here.

One approach to multilingual modeling is to use a neural network whose layers are shared across all languages. The network layer is trained on a phoneme set covering all the languages [24]. The phoneme set is either a concatenation of the phoneme sets of the different languages or one based on external knowledge such as the international phonetic alphabet (IPA) [25]. Finetuning a network trained in this manner on the target language has been shown to improve ASR performance [26].

The multilingual network could also be trained with language specific softmax output layers. In [27], the multilingual network is trained sequentially. After each

language is trained, the softmax layer is removed and replaced with the softmax layer for the next language. This approach allows the network to be trained without having all the data at hand but has the disadvantage of being biased towards later trained languages. Truly multilingual training can be achieved by interleaving the training samples from different languages [28–30]. The hidden layers of the network are shared across the languages while each language has its own softmax output layer. For a training example, only the softmax layer of the pertinent language is activated and the gradients from all other language output layers are set to zero. The multilingual network can also be used as a feature extractor. The idea is to learn a representation that maintains some of the discriminative ability of the neural network. This representation can be the senone or phoneme probabilities obtained from the output of the neural network [31] or the activations of an intermediate layer [32]. In a multilingual framework, unless the target language is part of the multilingual training, using the output probabilities directly typically entails concatenating the output activations for all the languages and performing some dimensionality reduction since the feature size might otherwise be unmanageable. An alternative is to introduce an intermediate, lower dimensional, bottleneck (BN) layer whose activations are used as features for further processing. To counteract the degradation in modeling capacity that results from introducing the BN layer in the middle of the network, a low-rank matrix factorization is used [33]. Extracting the bottleneck features from the low-rank layer also results in richer features since no compressive nonlinearities are applied.

2.4. Spoken Content Retrieval with Dynamic Time Warping

The template matching approach was recently proposed for KWS [21]. Inspired by the success of dynamic time warping (DTW) variants in the QBE-STD task, it has been shown that this approach does not suffer from the degradation that hampers LVCSR-based KWS on OOV queries [34] since the DTW simply compares two acoustic sequences. DTW involves aligning two sequences possibly different lengths and computing a distortion measure between the two. Although it was used with some success in isolated small vocabulary speech recognition applications [10,35,36], DTW has been

largely abandoned for large vocabulary and continuous speech recognition.

The applications of DTW to QBE-STD typically shun cepstral features as their high speaker and environmental variability limit their efficacy in large, speaker independent, retrieval applications. In [37], phone posteriorgrams are used to form both query and document models. The posteriorgram in this case is a time varying phone classification matrix obtained from a supervised deep neural network (DNN). The use of Gaussian posteriorgrams has also been shown to be effective for content retrieval [38]. This approach, particularly suitable for very low resource scenarios, involves training an unsupervised GMM and using the Gaussian occupation probabilities to label the data. Despite not being explicitly trained to classify acoustic units, the Gaussian posteriorgram nevertheless outperform cepstral representations for QBE.

A key difference between the use of DTW for speech recognition and QBE lies in the lengths of the sequences to be warped. In recognition, although the template and word to be classified are seldom the exact same length, it is expected that they are of approximately the same length and so the two sequences are matched from end to end. However, in retrieval applications, one sequence, the query, is much shorter than the other and so trying to match them from end to end is unlikely to succeed since most of the document is irrelevant to the query at hand. Therefore, different modifications to the DTW algorithm have been proposed some of which will be explained below especially as they pertain to our application.

2.4.1. Dynamic Time Warping

The DTW is a dynamic programming algorithm that seeks to align two sequences of varying length. Given two sequences, $\mathcal{X} := (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N)$ and $\mathcal{Y} := \mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_M)$, the algorithm aims to compute an alignment path and a distortion measure between them. A path is a sequence of tuples,

$$\mathbf{\Pi} = \{(n_k, m_k)\},\ k = 1, ..., K \tag{2.4}$$

such that $\mathbf{x}_{n_k}$ is aligned with $\mathbf{y}_{m_k}$. The path is constrained so that:

(i) $\mathbf{\Pi}_1 = (1,1)$ and $\mathbf{\Pi}_K = (N,M)$

(ii) $\mathbf{\Pi}_k - \mathbf{\Pi}_{k-1} \, \epsilon \, \{(0,1),(1,0),(1,1)\}, \; k = 2,...,K.$

The first constraint forces the beginning and end points of $\mathcal{X}$ and $\mathcal{Y}$ to be aligned with each other. The second ensures continuity; no frame in $\mathcal{X}$ or $\mathcal{Y}$ is omitted. Implicitly, the second condition also ensures monotonicity. Under these constraints, a distortion between the two sequences can be computed:

$$D_{\mathbf{\Pi}}(\mathcal{X},\mathcal{Y}) = \sum_{k=1}^{K} d(\mathbf{x}_{n_k}, \mathbf{y}_{m_k}) \tag{2.5}$$

where $d(.,.)$ is a frame-wise cost function which is referred to henceforth as a distance or metric function. It should be noted however, that this function is only a metric in a semantic sense, that is, it assigns small distortion values to similar vectors and high values to dissimilar ones. It does not necessarily satisfy the formal mathematical axioms of metrics.

An optimal alignment path, $\hat{\mathbf{\Pi}}$, is the one that minimizes the distortion between the two sequences:

$$\hat{\mathbf{\Pi}} = \arg\min_{\mathbf{\Pi}} D_{\mathbf{\Pi}}(\mathcal{X},\mathcal{Y}). \tag{2.6}$$

Using dynamic programming, the algorithm is able to avoid the computational cost of computing the score for all possible alignment paths. This is achieved by keeping track of a matrix of minimum accumulated distortion, $\mathcal{D}$; $\mathcal{D}(n,m)$ is the minimum distortion between $\mathcal{X}(1:n)$ and $\mathcal{Y}(1:m)$. Thus the overall dynamic time warping distortion between the two sequences is given by:

$$D_{\hat{\mathbf{\Pi}}}(\mathcal{X},\mathcal{Y}) = \mathcal{D}(N,M). \tag{2.7}$$

for $n = 1$ to N, $m = 1$ to M **do**
  if $n = 1$ **then**
    $\mathcal{D}(n, m) = \sum_{i=1}^{n} d(\mathbf{x}_i, \mathbf{y}_m)$
  else if $m = 1$ **then**
    $\mathcal{D}(n, m) = \sum_{i=1}^{m} d(\mathbf{x}_n, \mathbf{y}_i)$
  else
    $\mathcal{T} = \{(n-1, m-1), (n-1, m), (n, m-1)\}$
    $t = \arg\min_{t \epsilon \mathcal{T}} \mathcal{D}(t)$
    $\mathcal{D}(n, m) = \mathcal{D}(t) + d(\mathbf{x}_n, \mathbf{y}_m)$
  end if
end for

Figure 2.3. Dynamic Time Warping Algorithm.

2.4.2. Segmental Dynamic Time Warping

The segmental dynamic time warping (SDTW) is an altered DTW algorithm designed to align subsegments of two utterances [39]. Consider the sentences:

(i) Very sad, Signor Milo, very sad;
(ii) You must try some artichokes while you are here, Signor Milo.

Since each sentence comprises a different sequence of words, any global alignment between the two is sure to align frames from different words with one another; hence the DTW distortion between the two is virtually meaningless. However, it is clear that the utterances have segments that can be aligned; specifically, both contain the phrase, "Signor Milo." SDTW locates and aligns such similar subsegments.

The first modification to the DTW is the use of paths whose shapes are restricted. SDTW uses a Sakoe-Chiba band [40] to prevent the frames of one utterance from getting too far ahead of those from the other. For a path beginning at (n_1, m_1), the

restriction is:

$$|(n_k - n_1) - (m_k - m_1)| \leq R \; \forall \; (n_k, m_k) \; \epsilon \; \mathbf{\Pi} \tag{2.8}$$

This constraint limits the path two a region of width $2R+1$ that may not contain the coordinate corresponding to the end of both sequences, (N, M). The more significant modification is that several alignment paths are generated by selecting different DTW starting points. Thus, the boundary constraints of the DTW are removed. Equation 2.8 allows the partitioning of the search into grids with by providing a set of intuitive starting points:

$$((2R+1)k+1, 1), \qquad 0 \leq k \leq \left\lfloor \frac{N-1}{2R+1} \right\rfloor \tag{2.9}$$

$$(1, (2R+1)k+1), \qquad 0 \leq k \leq \left\lfloor \frac{M-1}{2R+1} \right\rfloor. \tag{2.10}$$

Within each region a locally optimal alignment and distortion can be computed with the DTW algorithm.

After the set of local alignment paths is computed, each path is shorn of its dissimilar portions. The shearing process can be done by locating the segments of the path with length-constrained minimum average (LCMA) distortion [41] and then extending such segments to include neighboring points with low enough distortion. The length constraint prevents the algorithm from always returning too short segments corresponding to spurious matches between the two sequences.

The SDTW algorithm can be modified for QBE-STD [38]. Since the query sample is fixed, only the utterance is segmented, that is, only the starting points in Equation 2.10 are considered.

2.4.3. Subsequence Dynamic Time Warping

The systems proposed in this thesis run the subsequence dynamic time warping (SSDTW) variation. While the SDTW algorithm discussed in the Section 2.4.2 allows the search of short, query, utterances within longer ones, its path restriction could cause it to fail when the length of the matching segment in the document differs significantly from that of the query. This can be costly when there is a significant mismatch in speaking rates between the document and query speakers; in [42], it was found that for the tendency of speakers to attempt to speak the query clearly and slowly could result in significantly different lengths between the query and a matching segment of the spoken document. SSDTW eschews with the DTW boundary conditions and the diagonal bands in SDTW to conduct an exhaustive search between the query sequence and all subsequences of the document [43, 44].

for $n = 1$ to N, $m = 1$ to M **do**
  if $n = 1$ **then**
    $\mathcal{D}(n, m) = d(\mathbf{x}_n, \mathbf{y}_m)$
    $\mathcal{L}(n, m) = 1$
  else if $m = 1$ **then**
    $\mathcal{D}(n, m) = \sum_{i=1}^{m} d(\mathbf{x}_i, \mathbf{y}_m)$
    $\mathcal{L}(n, m) = n$
  else
    $\mathcal{T} = \{(n-1, m-1), (n-1, m), (n, m-1)\}$
    $t = \arg\min_{t \epsilon \mathcal{T}} \mathcal{D}(t)$
    $\mathcal{D}(n, m) = \frac{\mathcal{D}(t) + d(\mathbf{x}_n, \mathbf{y}_m)}{\mathcal{L}(t)+1}$
    $\mathcal{L}(n, m) = \mathcal{L}(t) + 1$
  end if
end for

Figure 2.4. Subsequence Dynamic Time Warping Algorithm.

Each element, $\mathcal{D}(n, m)$, of the accumulated distortion matrix, tracks the distortion between the query segment, $\mathcal{X}(1 : n)$, and the best most similar subsequence of $\mathcal{Y}$ that ends at m. Akin to the length constraint of the SDTW, SSDTW incorporates

path length normalization to prevent a bias towards short paths.

2.4.4. Dynamic Time Warping Optimization

One of the main drawbacks of DTW approaches is the high computational resource consumption that accompanies them. In general, the search time can be reduced significantly by parallelizing with multiple processor cores [45–47]. Other methods of reducing the computational load of DTW generally involve developing approximations to the original search space on which to run the a quicker search, sometimes at the cost of some retrieval performance.

The segment based DTW is one such method. In this approach, the query and document utterances are segmented into contiguous regions of acoustically frames. The segmentation can be achieved with hierarchical agglomerative clustering (HAC) [48] by iteratively merging neighboring segments that minimize the increase in within-segment variance. Using a segment-wise distance metric, a much faster DTW is then conducted [49]. A second pass with frame level DTW on the hypotheses of the segment-base DTW has been shown to alleviate the loss in retrieval performance [50] that accompanies the segment based dynamic time warping.

Another approach involves the computation of a lower bound for the the DTW distortion. Initially proposed for K-nearest neighbor (KNN) with DTW [51], the idea is to compute a DTW lower bound between a template and different candidate sequences. Afterwards, a fine DTW is run on the candidates in order of their lower bound score. The process is halted when the next candidate has a lower bound that is greater than the K-th smallest DTW score. In [52], this approach was applied to a SDTW-based keyword spotting application. A lower bound for the posteriorgram is computed by using an envelop of the query posteriorgram. It is shown that, with a sufficient value of K, the computational gains are quite significant without much loss in retrieval compared to a baseline SDTW system. Although the use of the KNN lower bound reduces the DTW computations considerably, the estimation of the lower bound itself is rather computationally expensive. In [53], a weaker lower bound is computed that

reduces the lower bound computation at the expense in a relatively higher number of DTW computations. The authors report that this results in an overall speedup even when considering both the lower bound estimation and DTW rescoring. In [54], a more general lower bound is proposed which has the advantage of not being bound to the any one distance measure while also holding up to that in [52] in terms of keyword spotting performance.

A somewhat similar approach is the pre-filtering of the posteriorgram before the DTW [55]. This involves splitting the document into segments and then computing a vector that represents the whole segment; similarly, a single vector is computed for the query and a coarse similarity value is calculated between the two vectors. For posteriorgram, the representation vector can be obtained by taking the average of the frames within the segment and the normalized inner product is used to compute the similarity. Any segment whose dissimalirity exceeds a threshold is eliminated from further search. If a segment is within the desired similarity, it is extended temporally to include neighboring frames and then a SSDTW fine search is conducted to compute the refined similarity score as well as the hypothesized temporal location of the query.

Another way to reduce the computational cost of DTW, especially the local distance computations, is to index the document frames. The work in [56] provides a framework for doing this using locality sensitive hashing (LSH) [57, 58]. Using a random matrix, the frames of the utterances are projected to a different subspace and thresholded at zero to get a bit signature so that the cosine similarity can be approximated by a function of the Hamming distance. The document bit signatures are then sorted lexicographically and the nearest neighbors of each query frame bit signature can be obtained with a binary search [59]. The bits are permuted for a number of times and the nearest neighbor is computed for each; this increases the likelihood of finding the actual nearest neighbors since the lexicographic sort inappropriately inflates the importance of the initial bits. For sufficiently similar frames, the temporal neighbors are also compared resulting in a sparse similarity matrix with diagonal-shaped regions indicating matches between the utterances. Thereafter, a fast line search for syllable-length diagonals [60] is run followed by SDTW to get refined search results.

3. KEYWORD SEARCH WITH MULTILINGUAL REPRESENTATION

In this thesis, we utilize the template matching approach to keyword search in low resource languages. We follow this method in order to alleviate the effects of linguistic (pronunciation lexicon, language model) paucity in such a scenario. Using a transferred representation, we are able to leverage acoustic knowledge from other languages in building the model for the language of interest. Thus, the effects of both acoustic and linguistic scarcity are reduced. Our approach involves conducting DTW-based KWS on various transformations of the multilingual BN representation.

3.1. Multilingual Bottleneck Features

The bottleneck features are generated from the activations of a hidden layer of a multilingual acoustic model deep neural network with a bottleneck layer that is significantly smaller than surrounding layers and a non squashing activation function to prevent excessive gradient loss. This generator network is trained with MFCC features from several languages to predict language-specific senone labels. The hidden layers are shared across the languages up till the bottleneck layer; on top of these shared layers, a language specific hidden layer and a language specific softmax output layer are added. The training involves pooling the features from the source languages along with their force-aligned labels and interleaving them in batches from different languages. Each training batch contains only features from one language whose output layer is active while the gradients from the other languages' output layers are set to zero. By training in this way, we are able to obtain a representation that is language independent and compact while also keeping some of the discriminative power of the posterior features.

The BN extractor used in this work is a time-delay neural network (TDNN) with rectified linear units (ReLU) as activation functions. The use of TDNN layers up to the BN layer allows the incorporation of longer temporal contexts and by sub-sampling [61],

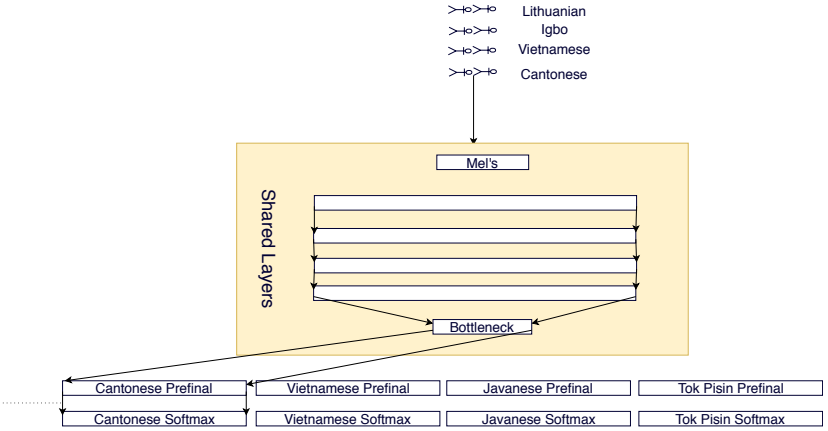

Figure 3.1. Multilingual bottleneck neural network.

a lot of the latency associated with recurrent architectures is avoided.

3.2. Query Modeling for Search

Since the query is provided in text, a different modality from the document, conducting DTW based search requires that the query be transduced to the same modality. This is done by converting the query into a sequence of phonemes using the Sequitur G2P toolkit [20] trained with the limited lexicon. Each phoneme is then decomposed into sub-phoneme states (five states for silence and three for all other phonemes). Although the sequence of phones can be used, the use of these sub-phonemic states allows for finer discrimination in the dynamic time warping. Using the Viterbi forced alignment of the available transcribed training data, a set of clusters along with duration statistics can be collected for each state. For each state, a centroid vector representative of its cluster of features is computed. These states are repeated (based on their average duration in the training data) and concatenated to represent the query.

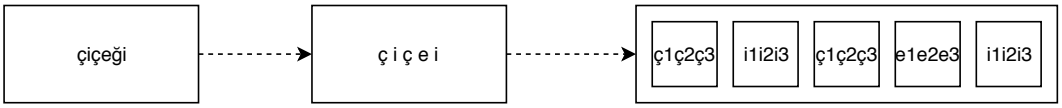

Figure 3.2. A simple query model.

When using the raw BN features for search, we use the cosine distance,

$$d_{cos}(\mathbf{x}, \mathbf{y}) = 1 - \frac{\mathbf{x}^T \mathbf{y}}{||\mathbf{x}||.||\mathbf{y}||} \tag{3.1}$$

which measures the angular distance between two vectors and a query vector that minimizes the total cosine distance within each state. Given a state cluster, $\mathcal{C}$, the class centroid, m, can be computed by minimizing:

$$\begin{aligned} J_{\mathcal{C}} &= \sum_{\mathbf{x}\epsilon\mathcal{C}} d(\mathbf{x}, \mathbf{m}) \\ &= |\mathcal{C}| - \sum_{\mathbf{x}\epsilon\mathcal{C}} \frac{\mathbf{x}^T \mathbf{m}}{||\mathbf{x}||.||\mathbf{m}||} \\ &= |\mathcal{C}| - \sum_{\mathbf{x}\epsilon\mathcal{C}} \frac{\mathbf{x}^T \hat{\mathbf{m}}}{||\mathbf{x}||} \text{ where } \mathbf{m} = \alpha\hat{\mathbf{m}},\ ||\hat{\mathbf{m}}|| = 1. \end{aligned}$$

The objective function becomes

$$J_{\mathcal{C}} = |\mathcal{C}| - \sum_{\mathbf{x}\epsilon\mathcal{C}} \frac{\mathbf{x}^T \hat{\mathbf{m}}}{||\mathbf{x}||} + \lambda(\hat{\mathbf{m}}^T \hat{\mathbf{m}} - 1).$$

By taking the gradient of $J_{\mathcal{C}}$ with respect to $\hat{\mathbf{m}}$ and setting it to zero, we get:

$$\begin{aligned} \hat{\mathbf{m}} &= \frac{1}{\lambda} \sum_{\mathbf{x}\epsilon\mathcal{C}} \frac{\mathbf{x}}{||\mathbf{x}||} \text{ and} \\ \mathbf{m} &= \frac{\alpha}{\lambda} \sum_{\mathbf{x}\epsilon\mathcal{C}} \frac{\mathbf{x}}{||\mathbf{x}||}. \end{aligned}$$

Since the magnitude of $\mathbf{m}$ has no bearing on the distance, i.e. $d_{cos}(\mathbf{x}, \mathbf{y}) = d_{cos}(\beta\mathbf{x}, \gamma\mathbf{y})$ for all positive scalars β, γ, the scale, α/λ can be set to 1, resulting in:

$$\mathbf{m} = \sum_{\mathbf{x}\epsilon\mathcal{C}} \frac{\mathbf{x}}{||\mathbf{x}||}. \tag{3.2}$$

Thus the centroid for each state can be computed by taking the mean of the normalized examples of that state in the training data.

3.3. Gaussian Posteriorgram for Bottleneck Features

Gaussian mixture models are used to represent features with high variability and have been used to generate posteriorgrams for spoken term detection [38]. In our preliminary experiments, we found that when the raw BN features are used to represent the document, their high variance made it difficult obtain an effective query model. To deal with this, we train an unsupervised GMM with k-means initialization [62] for each language on the bottleneck features of its training data. From the GMM, posteriorgrams are generated and used to represent the search document.

The Gaussian posteriorgram is a sequence of vectors that represent the occupation probabilities of the Gaussian densities. For an acoustic sequence, $\mathcal{X} := (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N)$, its Gaussian posteriorgram, $\mathcal{G} := (\mathbf{g}_1, \mathbf{g}_2, ..., \mathbf{g}_N)$. The dimensionality of each $\mathbf{g}_i$ is the number of Gaussian densities, m, in the underlying GMM. Formally, each $\mathbf{g}_i$ is such that:

$$\begin{aligned} \mathbf{g}_{ij} &= P(D_j|\mathbf{x}_i) \\ &= \frac{p(\mathbf{x}_i|D_j)P(D_j)}{\sum_{k=1}^{m} p(\mathbf{x}_i|D_k)} \end{aligned} \tag{3.3}$$

where D_j and D_k are the j-th and k-th Gaussian densities in the GMM respectively.

We also investigate the impact of using bottleneck features from the source languages to bootstrap the GMM training. We train a multilingual unsupervised GMM with the source languages' training data and use it to generate posteriorgrams for search in the target languages. Although this multilingual representation would be desirable for multilingual search with spoken queries, the absence of a language independent orthographic system means the queries in KWS are necessarily language dependent and so a more language specific document representation is more relevant to our task. Therefore, for each of the target languages, we build another system by performing a rapid finetuning of the multilingual GMM using that language's training data to better represent the language specific variabilities. From the language specific GMMs, posteriorgrams are generated and used to represent the search document.

The query is represented as a pseudo-posteriorgram as described in Section 3.2. The state centroids are computed by taking the average of the class examples in the training set. The nonnegative and probabilistic nature of the Gaussian posteriorgram allows us to use the log-cosine similarity measure,

$$d_{log}(\mathbf{x}, \mathbf{y}) = -\log(\delta + \frac{\mathbf{x}^T\mathbf{y}}{||\mathbf{x}||.||\mathbf{y}||}) \tag{3.4}$$

where δ is a small positive number added to ensure that the operand of the logarithm function is strictly positive. The log-cosine, in a sense, gives the log-likelihood of the two frames being compared originating from by the same Gaussian density.

3.4. Dynamic Time Warping for Keyword Search

Given a query vector sequence, $\mathcal{X} := (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N)$, the subsequence dynamic time warping algorithm is used to find matching subsequence of the document vector sequence, $\mathcal{Y} := (\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_N)$. The algorithm, as applied to KWS, is shown in Figure 3.3. To avoid the need for backtracing to find the beginning of the path, a matrix, $\mathcal{B}$, of the starting points of each subsequence is stored in memory along with the matrix of accumulated distances, $\mathcal{D}$ and the matrix of subsequence lengths, $\mathcal{L}$.

From the last row of the matrix of accumulated distances, $\mathcal{D}$, a vector, $\mathbf{u}$, of the raw scores of the subsequences most similar to the query ending at each frame of the document can be computed.

$$\mathbf{u}(m) = 1 - \frac{\mathcal{D}(N, m)}{L(N, m)}. \tag{3.5}$$

for $n = 1$ to N, $m = 1$ to M **do**
  if $n = 1$ **then**
    $\mathcal{D}(n, m) = d(\mathbf{x}_n, \mathbf{y}_m)$
    $\mathcal{L}(n, m) = 1$
    $\mathcal{B}(n, m) = m$
  else if $m = 1$ **then**
    $\mathcal{D}(n, m) = \sum_{i=1}^{m} d(\mathbf{x}_i, \mathbf{y}_m)$
    $\mathcal{L}(n, m) = n$
    $\mathcal{B}(n, m) = 1$
  else
    $\mathcal{T} = \{(n-1, m-1), (n-1, m), (n, m-1)\}$
    $t = \arg\min_{t \epsilon \mathcal{T}} \mathcal{D}(t)$
    $\mathcal{D}(n, m) = \frac{\mathcal{D}(t) + d(\mathbf{x}_n, \mathbf{y}_m)}{\mathcal{L}(t)+1}$
    $\mathcal{L}(n, m) = \mathcal{L}(t) + 1$
    $\mathcal{B}(n, m) = \mathcal{B}(t)$
  end if
end for

Figure 3.3. Subsequence Dynamic Time Warping Algorithm for Keyword Search.

From this, the best score above a predefined threshold is returned as a "hit" for the search. The location of hit and score are obtained thus:

$$\begin{aligned} t_{end} &= \arg\max_{t} \mathbf{u}(t) \\ t_{beg} &= \mathcal{B}(N, t_{end}) \\ \text{hit} &= \mathcal{Y}(t_{beg} : t_{end}) \\ \text{score} &= \mathbf{u}(t_{end}). \end{aligned} \tag{3.6}$$

Multiple hits for the query can be found by recursively removing a sufficient neighborhood of the hit, $(t_{beg} : t_{end})$, from $\mathbf{u}$ and repeating Equation 3.6 until a stopping criterion is satisfied.

3.5. A Unified Framework for Optimizing the Search Path

By observing the search algorithm and equations 3.5 and 3.6, we can see that:

$$\text{score} = \frac{1}{\text{length}(\hat{\mathbf{\Pi}})} \sum_{(n,m)\epsilon\hat{\mathbf{\Pi}}} d(\mathbf{x}_n, \mathbf{y}_m) \tag{3.7}$$

where $\hat{\mathbf{\Pi}}$ is the path of the optimal matching subsequence. From this, we observe that the score (along with the path) is dependent on:

- The document representation, $\{\mathbf{y}_m\}$,
- the query model, $\{\mathbf{x}_n\}$ and
- the distortion function $d(.,.)$ used.

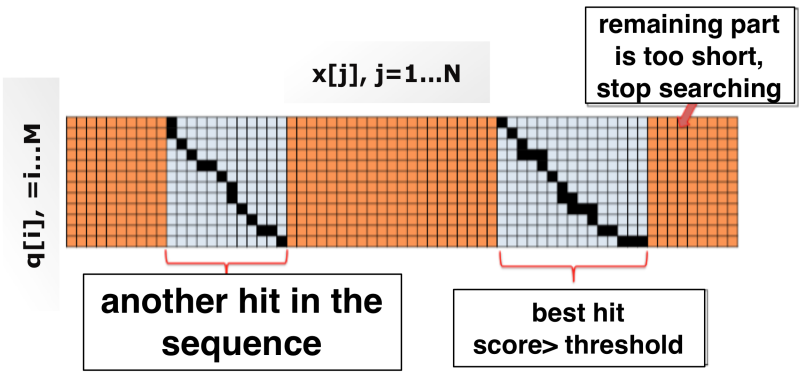

Figure 3.4. Subsequence dynamic time warping for finding multiple hits [1].

We propose using a unified framework to optimize each of these implemented as an extended distance metric learning (EDML) neural network. The EDML network is based on the joint optimization of three parts:

(i) The distance learner learns a distance function called the sigma distance [22] based on a weighted inner product between its inputs. Given a query vector, $\mathbf{x}$ and a document vector, $\mathbf{y}$, the distance function, parametrized by its weight matrix, $\mathbf{W}$, and bias value, c, is given by:

$$d_\sigma(\mathbf{x}, \mathbf{y}) = \sigma(\mathbf{x}^T\mathbf{W}^T\mathbf{W}\mathbf{y} + c) \tag{3.8}$$

where $\sigma(.)$ is the logistic sigmoid function. The parameters of the network (W and c) are optimized to discriminate between sub-phoneme states. In the ideal case, this network outputs:

$$\hat{d}_{\sigma}(\mathbf{x}, \mathbf{y}) = \begin{cases} 0, & \text{if class}(\mathbf{x}) = \text{class}(\mathbf{y}) \\ 1, & \text{if class}(\mathbf{x}) \neq \text{class}(\mathbf{y}). \end{cases} \tag{3.9}$$

(ii) The document representation is learned with a series of nonlinear neural network layers. The part of the network responsible for this takes a bottleneck feature vector, $\mathbf{b}$, as input and outputs $\mathbf{y}$. Since the bottleneck representation is quite rich with classes that are not linearly separable, this part of the network serves to simplify the representation and obtain more easily separable class representations.

$$\mathbf{y} = \mathcal{F}(\mathbf{b}). \tag{3.10}$$

The function ($\mathcal{F}$) is simply a hierarchy of stacked ReLU layers.

(iii) The third part of the network generates the state models. It's input is a one-hot vector representing the state being modeled which is transformed into the query model, $\mathbf{x}$. It is parametrized by a single matrix, $\mathbf{V}$ so that:

$$\mathbf{x} = \mathbf{V}\mathbf{o} \tag{3.11}$$

where $\mathbf{o}$ is a vector whose entries are all zero except for the entry corresponding to the active state which is set to one [23, 34]. Since only one element of $\mathbf{o}$ is active at a time, the query templates are the columns of the matrix $\mathbf{V}$.

The query and document representation learners are prepended to the distance learner as shown in Figure 3.5 to form a unitary network to be optimized.

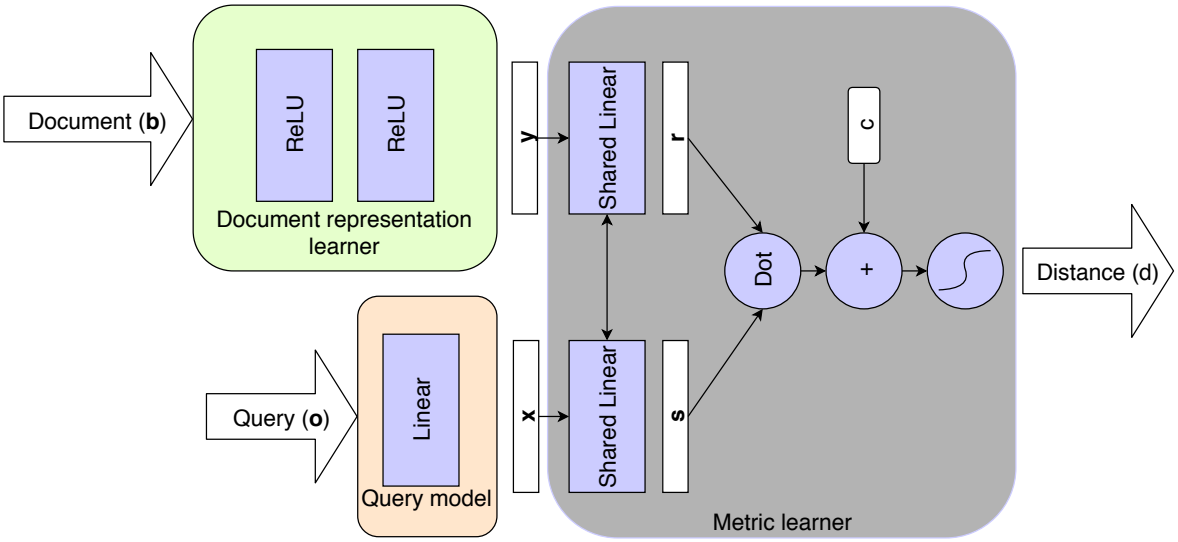

Figure 3.5. The extended distance metric learning network.

3.5.1. Training the Extended Distance Metric Learner

The training of the whole network is done using the backpropagation algorithm. The forward pass takes mini-batches of pairs of inputs, $(\mathbf{b}, \mathbf{o})$, and performs the operations in Equations 3.11, 3.10 and 3.8 to obtain the dissimilarity between the representations $(\mathbf{y}, \mathbf{x})$ of the inputs. From this the objective function is computed. We use the cross-entropy between the output dissimilarity, $d_\sigma(\mathbf{x}, \mathbf{y})$, and the actual dissimilarity, $\hat{d}_\sigma(\mathbf{x}, \mathbf{y})$, defined in Equation 3.9. The objective function per sample is defined as:

$$J_{CE} = -\hat{d}_\sigma(\mathbf{x}, \mathbf{y}) \log d_\sigma(\mathbf{x}, \mathbf{y}) - (1 - \hat{d}_\sigma(\mathbf{x}, \mathbf{y})) \log(1 - d_\sigma(\mathbf{x}, \mathbf{y})). \tag{3.12}$$

The gradients of the parameters with respect to this objective function are computed and the parameters are updated with the Adam optimization method [63].

Certain considerations are made with regards to the choice of the sample pairs. If the samples are chosen randomly, the network will be biased heavily towards outputting values close to 1 since there are far more adversaries than there are samples of the same class for any choice of $\mathbf{b}$. For instance, a language with 39 non-silence phonemes has 122 (3 for each phoneme, 5 for silence) states. If these classes are equally represented

and the network outputs a distance of 1 regardless of its inputs, the expected accuracy:

$$\bar{A} = 121/122 \approx 99\%.$$

Clearly, this behavior is not desirable. Another, similar, issue results from the unequal representation of the classes in the training set. For example, analysis of the data shows that the silence states account for about 30% of all the training frames. A naive training sample selection biases the network to model well represented sample classes well at the expense of other classes.

To deal with these imbalances, we use a class sampling strategy to select the training frames. The procedure can be summarized thus at a sample level:

(i) Sample two distinct classes from the set of sub-phoneme states with one-hot representations $\mathbf{o}^1$ and $\mathbf{o}^2$.
(ii) Sample two feature vectors $\mathbf{b}^1$ and $\mathbf{b}^2$ belonging to the classes of $\mathbf{o}^1$ and $\mathbf{o}^2$ respectively.
(iii) Train the network with the sample pairs: $(\mathbf{b}^1, \mathbf{o}^1), (\mathbf{b}^1, \mathbf{o}^2), (\mathbf{b}^2, \mathbf{o}^1)$ and $(\mathbf{b}^2, \mathbf{o}^2)$.

By repeating this procedure iteratively, we can train the network with samples of approximately equal class and similarity representation.

3.5.2. EDML-based Keyword Search

The EDML network provides a unified way to optimize the query, document and the distortion function to run the dynamic time warping. To reduce latency, the vectors obtained by passing the document features and state one-hot vectors through the network up till the final dot product are stored. For the entire document BN representation, $\mathcal{B} := (\mathbf{b}^1, \mathbf{b}^2, ..., \mathbf{b}^N)$, we compute and store another sequence, $\mathcal{R} := (\mathbf{r}^1, \mathbf{r}^2, ..., \mathbf{r}^N)$ where:

$$\mathbf{r}^i = \mathbf{W}\mathcal{F}(\mathbf{b}^i). \tag{3.13}$$

For the set of phone states, with one-hot representations $\mathcal{O} = \{\mathbf{o}^j\}$, we compute and store another set of representations, $\mathcal{S} := \{\mathbf{s}^j\}$ such that:

$$\mathbf{s}^j = \mathbf{W}(\mathbf{V}\mathbf{o}^j). \tag{3.14}$$

The query is represented as a concatenation of the pertinent states from $\mathcal{S}$ and the SSDTW search is conducted with $\mathcal{R}$ as the document and the distortion function:

$$d_\Sigma(\mathbf{s}^j, \mathbf{r}^i) = \sigma(\mathbf{s}^j.\mathbf{r}^i + c). \tag{3.15}$$

3.6. Low Entropy Representation for Keyword Search

One drawback of DTW based systems is the memory consumption that accompanies them. In addition to the intermediate memory costs of running the search, there is also the cost of storing the document representation. This is especially an issue when deploying the system in a keyword search server where the document must be stored in memory permanently as it would be impractically slow to load it from the disk each time a user queries the system. In this section, we propose a modification to the EDML network described in Section 3.5 that allows us to compress the document representation significantly.

A softmax is added to the distance metric matrix. This results in a new document representation, $\tilde{\mathcal{R}} := (\tilde{\mathbf{r}}^1, \tilde{\mathbf{r}}^2, ..., \tilde{\mathbf{r}}^N)$ where:

$$\tilde{\mathbf{r}}^i = softmax(\mathbf{r}^i)$$

and a new set of phone state representations $\tilde{\mathcal{S}} := \{\mathbf{s}^j\}$ where

$$\tilde{\mathbf{s}}^j = softmax(\mathbf{s}^j).$$

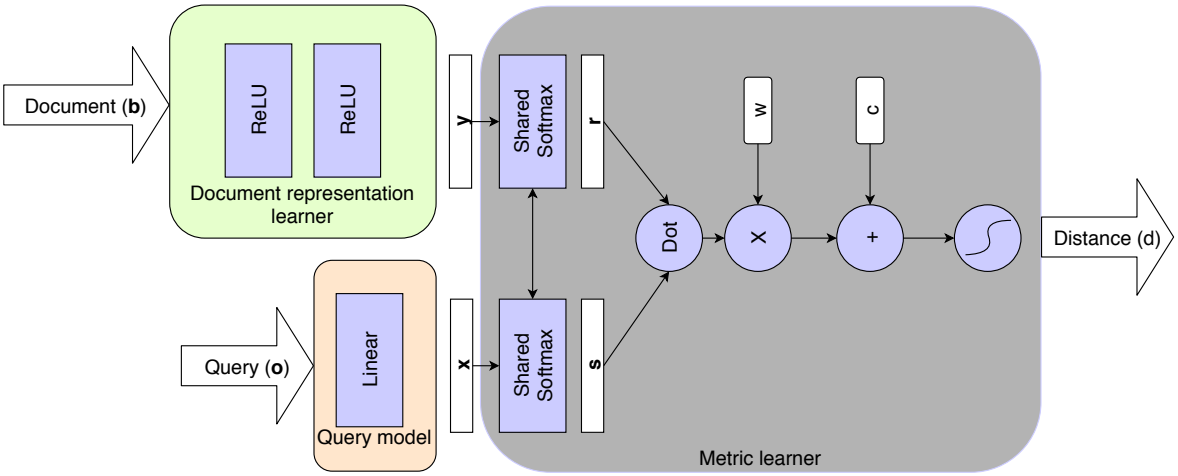

Figure 3.6. The EDML network modified for sparsity.

In this representation, each frame can be interpreted as a vector of the probability values of a categorical distribution. By minimizing the entropy of this categorical distribution, we can get sparser frame representation which have a much lower memory footprint than the original representation. The loss function of the EDML network from Equation 3.12 is modified to effect this sparsity objective.

$$\begin{aligned} J_T =& J_{CE} + \lambda J_H \\ &= J_{CE} + \lambda(\mathcal{H}(\mathbf{r}) + \mathcal{H}(\mathbf{s})) \\ &= -\hat{d}_\Sigma(\mathbf{r},\mathbf{s}) \log d_\Sigma(w\mathbf{r},\mathbf{s}) - (1-\hat{d}_\Sigma(\mathbf{r},\mathbf{s})) \log(1-d_\Sigma(w\mathbf{r},\mathbf{s})) \\ &\quad - \lambda \sum_{i=1}^{n} (\mathbf{r}_i \log \mathbf{r}_i + \mathbf{s}_i \log \mathbf{s}_i) \end{aligned} \tag{3.16}$$

where $\mathcal{H}(.)$ is the information entropy, n is the dimensionality of $\mathbf{r}$ and $\mathbf{s}$, and λ is a hyperparameter that determines the relative weighting between the discriminative objective and the sparsity objective. $d_\Sigma(.,.)$ is defined in Equation 3.15 and

$$\hat{d}_\Sigma(\mathbf{r},\mathbf{s}) = \begin{cases} 0, & \text{if class}(\mathbf{r}) = \text{class}(\mathbf{s}) \\ 1, & \text{if class}(\mathbf{r}) \neq \text{class}(\mathbf{s}). \end{cases} \tag{3.17}$$

Note that since $\mathbf{r}$ and $\mathbf{s}$ are vectors that sum up to one, their dot product is a value in $[0, 1]$. Passing this value directly through a the final sigmoid nonlinearity produces a

number within a maximum range of 0.245 (when $c = 0.5$). Therefore, we scale the dot-product with a weight, w, before adding the bias term and applying the nonlinearity. By using a large positive weight with a negative bias or a large negative weight with a positive bias, we can use utilize the full range of the sigmoid activation.

3.7. Score Normalization

A keyword search system is expected to return some spurious hypotheses. Hence, it is necessary to have a threshold of scores below which hypothesis are pruned away. Since the query terms are not known beforehand, the threshold must be global (irrespective of query). However, different queries return hypotheses with different score distributions; this fact makes it rather impossible to get a threshold for the raw scores that returns a list of hypotheses with a good balance of misses and false alarms for any given keyword without first normalizing to ensure that the effective dynamic ranges of the scores are comparable across keywords. Figure 3.7 shows the unnormalized score histograms of five randomly chosen keywords from the Turkish language KWS system.

Given a query, q, and $S_q = \{s_q\}$, the set of its hypothesized hits, the normalized score ($\tilde{s}_q$) of each raw score, s_q is a function of the entire set.

One method of score normalization for keyword search is the sum-to-one (STO) normalization [64]. In this method, each score is normalized by the sum of all scores returned for the query:

$$\tilde{s}_q^{STO} = \frac{s_q}{\sum_{s' \epsilon S_q} s'}. \tag{3.18}$$

STO has the effect of reducing the scores of queries with many putative hits while enhancing those of terms with few hits. For our system, which returns thousands of hits per query, this has an effect of significantly reducing the dynamic ranges of the scores. A global threshold computed for such a set of scores would be unstable and difficult to transfer to a dataset other than the one on which it was originally learned

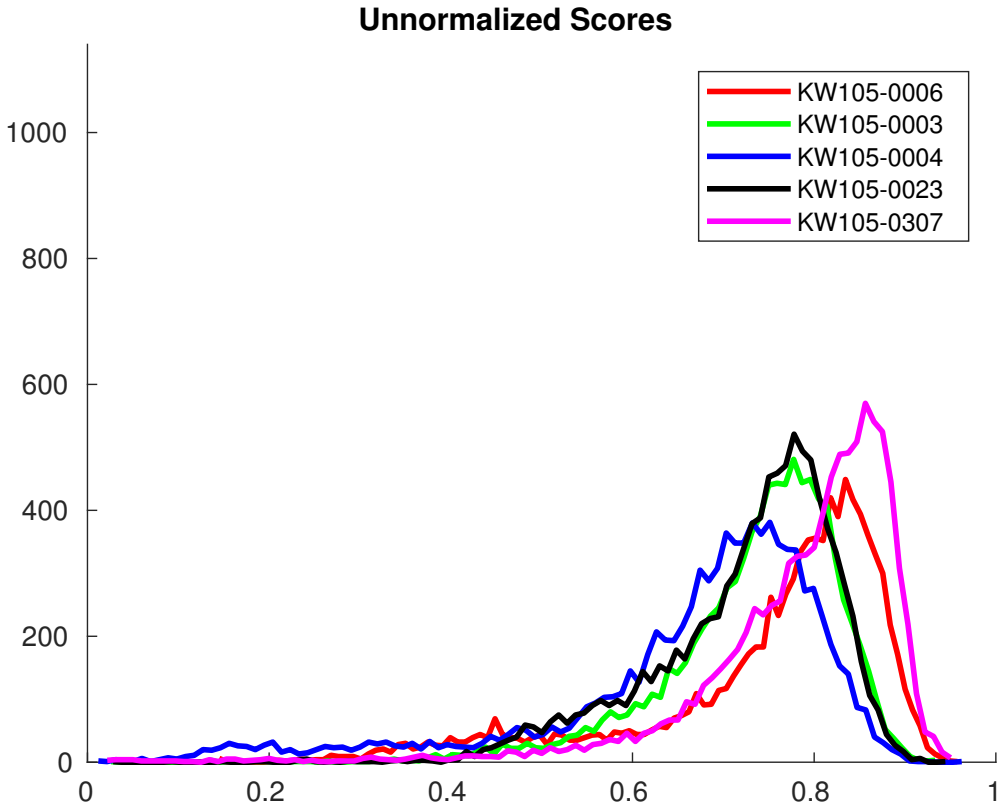

Figure 3.7. Histograms of unnormalized keyword scores.

because it would be very sensitive to slight numerical variations.

The keyword-specific thresholding (KST) is another method that has been used successfully in normalization of scores for LVCSR based keyword search. By formulating the problem as a likelihood ratio test, a threshold that maximizes the term weighted value (TWV) is computed for each keyword. To get a global threshold, the scores of each query can be weighted by a ratio of the target threshold to the computed threshold. The threshold computation requires a knowledge of the prior probability of the keyword which is obviously not available beforehand; therefore, an integration over the posterior probabilities is used to estimate the priors. If the raw scores are posterior probabilities, then the prior can be estimated by summing over the raw scores of a query. In our system, however, the scores are DTW distortions which are not directly interpretable as posterior probabilities and so cannot be used in the prior probability estimation required for KST.

Since we have numerous scores, we are able to compute low-variance estimates of statistics of the score distributions. With these statistics, we can find what a "typical" element of the set of scores is expected to be. A family of normalization schemes have been proposed that use this along with information about the spread of the distribution. The z-norm [65] works by subtracting the sample mean from each score and then dividing by the sample standard distribution.

$$\tilde{s}_q^z = \frac{s_q - \text{mean}(S_q)}{\text{std}(S_q)}. \tag{3.19}$$

The z-norm is sensitive to irrelevant scores since it uses the mean to represent a typical member of the set. Moreover, it is based on an implicit assumption that the scores have a Gaussian distribution. In [66], it is claimed that the distribution of scores returned by a DTW-based STD system are actually asymmetric with a long trailing left trail and a steeper tail right of the mode (this can also be seen visually by inspecting the scores histograms in Figure 3.7). The authors proposed another normalization scheme, called m-norm, in which scores below the mode, considered, to be irrelevant are ignored from the standard deviation computation. If we define another set of the scores, $S_q^\zeta := \{s_q \, \epsilon \, S_q : s_q > \zeta\}$, then the m-norm normalized scores are given by:

$$\tilde{s}_q^m = \frac{s_q - \text{mode}(S_q)}{\text{std}(S_q^{\text{mode}(S_q)})}. \tag{3.20}$$

The b-norm proposed in [67] uses the median in stead of the mode to represent the typical element of the set. Since the sample mode can only really be computed with histogram bins, changing the sizes and locations of the bins results in variations in the mode. Therefore, the mode computation is not entirely dependent on just the set of scores. The sample median, on the other hand, is computationally dependable and has no axes of variation outside the members of the set. The b-norm normalized

scores are given by:

$$\tilde{s}_q^b = \frac{s_q - \text{median}(S_q)}{\text{std}(S_q^{\text{median}(S_q)})}. \tag{3.21}$$

In this work, we propose using a generalized version of the b-norm. If we define the η-th percentile of the score distribution as the smallest number, $\phi(S_q, \eta)$, such that $\eta|S_q| < \phi$. The scores normalize with the generalized b-norm are given by:

$$\tilde{s}_q^{c(\eta)} = \frac{s_q - \phi(S_q, \eta)}{\text{std}(S_q^{\phi(S_q,\eta)})}. \tag{3.22}$$

The η parameter controls how much of the original set is considered relevant. While the

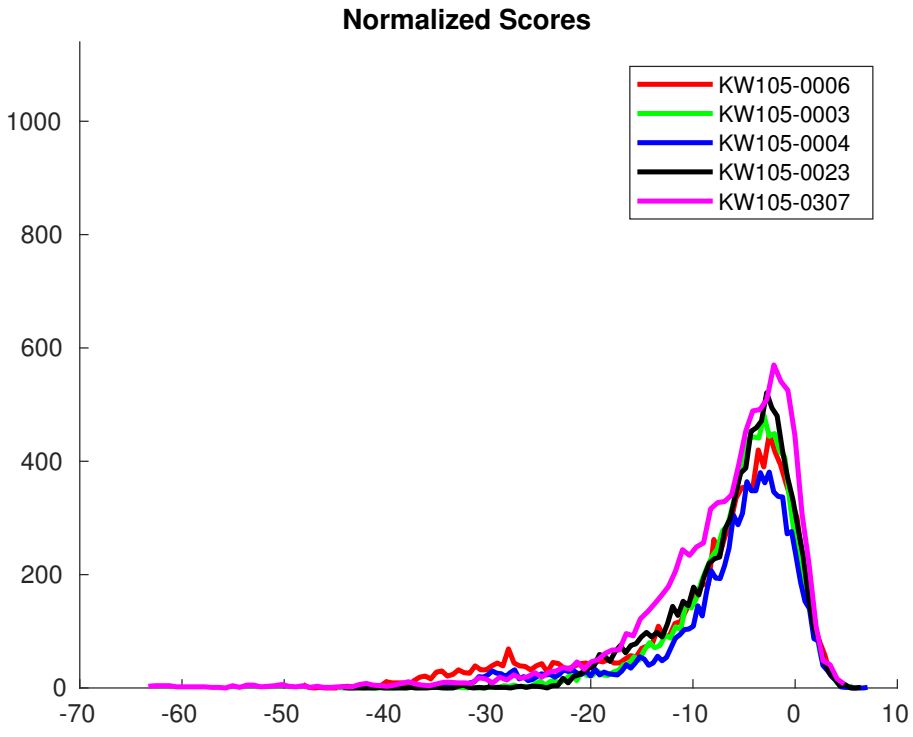

Figure 3.8. Histograms of keyword scores normalized with the generalized b-norm.

original b-norm can be obtained by setting $\eta = 0.5$, we found a value of $\eta = 0.9$ to be optimal. In addition to the aligning the distribution of scores, normalizing with such a high percentile enables us to prune more irrelevant hypotheses and prevent any further

processing on them. Another somewhat subtle advantage of the high percentile results from the standard deviation computation; since the standard deviation is computed on scores above the percentile, setting a higher threshold gives on a smaller standard deviation. Dividing by a smaller standard deviation results in a higher dynamic range of normalized scores for which a more stable global relevance threshold can be set.

4. EXPERIMENTS AND RESULTS

In this chapter, we describe the experiments performed to demonstrate the efficacy of the methods described in the previous chapter. First we describe the dataset on which the experiments are run and the metrics used to measure the system performance. For each of the proposed models, we run experiments for four languages. In each language, we run a pair of experiments to observe the performance of the models at different levels of data scarcity and compare the results to a competitive baseline system trained on the same data.

4.1. Dataset

The test experiments are performed on the limited language pack (LLP) data from the IARPA Babel Program [68]. We divide the languages available to us into two groups. Our target languages, on which we evaluate our models, are Pashto [1] , Turkish [2] , Zulu [3] and Kazakh [4] . The 19 other languages in the program are used as the source languages and a pool of their LLP training data is used train the multilingual bottleneck representation extractor as well as the multilingual GMM.

Each LLP has ten hours of transcribed conversational telephone speech. For each language, we train an HMM-GMM with speaker adaptive training (SAT) on the LLP and use the Viterbi algorithm to obtain frame level forced alignments. In the source languages, these alignments are used as labels for training multilingual TDNN training. In the target languages, the alignments are used to learn the query model and duration statistics. We run two sets of experiments for the target languages; in the low resource (LR) setting, we train with the LLP while in the extremely low resource (ELR) setting, we train with only a one-hour subset of the LLP.

[1] babel104-v0.4bY (dev:kwlist3, evalpart1:kwlist4)
[2] babel105b-v0.4 (dev:kwlist, evalpart1:kwlist2)
[3] babel206b-v0.1e (dev:kwlist3, evalpart1:kwlist4)
[4] babel302b-v1.0a (dev:kwlist, evalpart1:kwlist4)

In addition to the training data, each language has a pair of spoken documents on which we conduct the search with the models built with the training data. The development (dev) document has ten hours of speech with the necessary resources to evaluate system performance. It is on this document that we tune the hyperparameters of our models such as the normalization and fusion parameters, the score thresholds, the sparsity weights and even which model to keep using. The evaluation (evalpart1) document is the five-hour document on which we purely test our models which we have trained on the training dataset and tuned on the development dataset. Although the length of the full evaluation document provided in the Babel program is about fifteen hours for each language, the information required for evaluation is openly available only for a five-hour subset. Even though the number of query terms used in these experiments vary by language, the number of evaluation terms is consistently higher than the number of development terms; in a real use case, the number of user provided queries is expected to eventually outgrow the number of queries used in system development. Table 4.1 shows the number of keywords in each test for the LR setting as well as the percentage of keywords that are OOV.

Table 4.1. Query distribution per language in the low resource setting.

LANGUAGE	DOCUMENT	QUERIES	OOV RATE (%)
Pashto	Dev	2065	29
	Eval	4203	23
Turkish	Dev	307	29
	Eval	3171	38
Zulu	Dev	2000	40
	Eval	3310	34
Kazakh	Dev	4171	26
	Eval	4533	35

4.2. Evaluation Metrics

Evaluation metrics are necessary to measure the performance of a model. A good metric simulates how good an end user would consider the model to be. In this thesis, we use the term weighted value metrics to measure system performance. In addition to the TWV, we also discuss the normalized cross entropy (Cnxe) in this section. While we do not directly use the Cnxe to measure system performance, it is instrumental to our system fusion methodology.

4.2.1. Term Weighted Value

A KWS system returns a list of hypothesized hits which are (query, location-in-document) pairs along with their respective confidence scores. Some of these are bound to be false, so it is necessary for the system to provide a threshold below which the scores are considered to be irrelevant. In this framework, there are two kinds of errors possible:

- Misses occur when a query is not found in a document even though it exists or if the query is found but it's score is lower than the threshold. Given the sheer unlikelihood of getting an exact match in timing between the system hits and the reference, a hit is considered to match a reference location if they are within a (500ms) neighborhood of each other.
- False alarms (FA) occur when a query is returned at a location where it does not actually occur in the reference with a confidence score above the selected threshold.

The term weighted value provides a measure of recall and precision at a global threshold. If different queries terms use different thresholds, these thresholds must be computable and transformable to the global threshold using only the information available at search time. Given a set of terms, $\mathcal{Q} = \{q\}$ and a threshold, θ, the TWV

is defined thus:

$$TWV(\theta, \mathcal{Q}) = 1 - \frac{1}{|\mathcal{Q}|}\sum_{q\epsilon\mathcal{Q}}(P_{miss}(q,\theta) + \beta P_{fa}(q,\theta)) \tag{4.1}$$

where

$$\beta = \frac{C}{V}(\frac{1}{Pr_q} - 1). \tag{4.2}$$

C is the cost of a false alarm, V is the value of finding a term correctly and Pr_q is the prior probability of a query. Following the 2006 NIST STD evaluations [69], the ratio $\frac{C}{V}$ is set to 0.1 and Pr_q is set to 10^{-4} resulting in a value of 999.9. By observing, Equation 4.1, we see that a system that returns all the correct hits and no false alarms would have a TWV of 1 while one that returns not hits and no false alarms would have a TWV of 0; furthermore, it is possible to get negative TWVs up to $-\beta$. For each query, q, the following are defined:

$$\begin{aligned} P_{miss}(q,\theta) &= 1 - \frac{N_{correct}(q,\theta)}{N_{true}(q)} \\ P_{fa}(q,\theta) &= \frac{N_{fa}(term)}{n_{TPS}.T - N_{true}} \end{aligned} \tag{4.3}$$

where:

- $N_{correct}(q,\theta)$ is the number of true detections of q with a confidence score above θ.
- $N_{fa}(q,\theta)$ is the number of false detections of q with a confidence score above θ.
- $N_{true(q)}$ is the actual number of occurrences of q in the document. T is the length of the document in seconds. n_{TPS} is the number of trials per second. Since such a value does not really exist for continuous speech, it is arbitrarily set to 1.

We define a few variations of the TWV to measure different aspects of KWS system performance. These are:

(i) The actual term weighted value (ATWV) is the measure of TWV at the selected threshold. The ATWV is the closest representation of the goodness of the system to an end user.

(ii) The maximum term weighted value (MTWV) is the value of the TWV at the threshold that maximizes it. Generally, a development set is used to get the threshold that gives the MTWV and this threshold is used for further search.

(iii) The optimum term weighted value (OTWV) is the MTWV with a term specific optimal threshold for each term. The OTWV shows how well the scores are ordered for each query.

(iv) The supremum term weighted value (STWV) is the TWV if the cost of false alarms is considered to be zero. It is a measure of the overall recall rate of the system.

4.2.2. Minimum Normalized Cross Entropy and System Fusion

The normalized cross entropy (Cnxe) is another metric that ha been used in spoken term detection. Originally proposed for speaker recognition [70] and language recognition [71], it has also been used to evaluate QbE-STD [72]. Where TWV measures the performance of the KWS system using hard decisions at a specific threshold, Cnxe also measures how well calibrated the scores of a system, i.e. how well separated the relevant hypotheses are from the irrelevant ones. A well calibrated system has scores that can be interpreted as log-likelihood ratios. Such a system would have scores of ∞ for targets and $-\infty$ for non-target sections of the document and a Cnxe value of 0. A non-informative system would have a Cnxe value of 1 and the higher Cnxe values indicate severe system mis-calibration.

By optimizing the Cnxe of a system, we can get good score separation between relevant parts of the document and irrelevant parts. Such a system would be robust to changes in the operating point and threshold. In this way, optimizing the Cnxe is very likely to lead to better TWV values.

The minimum normalized cross entropy (Cnxe-min) is an analogue to the MTWV for Cnxe. To compute Cnxe-min, we recalibrate the scores use a simple affine transform on all the scores $\{x\}$:

$$\hat{x} = ax + b$$

and optimize the Cnxe by conducting an exhaustive search over different values of a and b. Note that although lower values for the Cnxe can be computed with the pooling of adjacent violators algorithm [73, 74], we stick to the affine transform since it is parametric; in this way, we can learn these parameters on the development set and transfer them to the evaluation set. Additionally, since there are only two parameters, there is little risk of overfitting the development set.

We do not use the Cnxe-min to measure our system performance, rather we use it as a way to learn the weights for system fusion. Given a system that we want to combine with another, transforming it using the Cnxe-min parameters ensures that there is better separation between the relevant and irrelevant scores. Therefore, when we sum the transformed scores of the systems, we are less likely to increase the false alarms of either one unless one of systems is overwhelmingly bad on its own.

4.3. Baseline System

We train a baseline model that uses the contemporary LVCSR based KWS approach. To have a fair comparison, we also incorporate the multilingual information in training the baseline acoustic model. We truncate the multilingual TDNN at the bottleneck layer (the last layer that is shared across the source languages), and for each of the target languages, we append a ReLU layer followed by a softmax output layer and finetune the entire network on that language's training data while using a lowered learning rate for the transferred layers.

An index is then constructed from the LVCSR lattices and used to search for the queries. OOV words in the queries are handled with the proxies [8, 9] and the scores

Table 4.2. Query distribution per language in the extremely low resource setting.

LANGUAGE	DOCUMENT	QUERIES	OOV RATE (%)
Pashto	Dev	2065	65
	Eval	4203	62
Turkish	Dev	307	68
	Eval	3171	81
Zulu	Dev	2000	84
	Eval	3310	76
Kazakh	Dev	4171	62
	Eval	4533	80

are normalized with the KST method [75].

When the amount of training data is decreased, the size of the pronunciation lexicon decreases and the OOV rate increases. This can be seen by comparing Tables 4.1 and 4.2. We see that although the keywords do not change, the percentage of keywords that are out-of-vocabulary is considerably higher in the ELR setting than in the LR one. Therefore, in the ELR setting, we have a second baseline system that uses pronunciation lexicon of words in the whole LLP instead of just the one-hour subset. This allows us to gauge how much of improvement we get from the scarcity of acoustic data and how much from the scarcity of linguistic data. The second baseline simulates a method like the lexicon expansion method [7] which works on the premise that it is easier to obtain the pronunciation of words in a language from external sources than it is to get more transcribed speech data.

4.4. Preliminary Experiments

Before undertaking the expensive task of training the multilingual BN representation for keyword search, we train a monolingual BN extractor with a 42-dimensional

bottleneck layer using just the Turkish language. Using the monolingual BN representation, we run a series of KWS experiments using query models with a single class per phoneme.

Using the average of the BN features of each class as our query model and the cosine similarity, we conduct the search on the Turkish development set. While we obtain a positive STWV (recall) from this, our OTWV was very low and our MTWV zero. From this we see that while the BN features do have discriminative information, the classes are not well separated enough to be able to get a well ordered set of hypotheses. Therefore, it is necessary to transform the features to a space where separation can be more easily achieved.

By training a GMM with 1024 mixtures and using its posteriorgram, we get a representation that is quite flattened compared to the bottleneck features and for which the normalized inner product based similarities have been shown to be effective [39]. When we use the cosine similarity we see that at the expense of a slight reduction in STWV, we get an increased OTWV and a positive MTWV. Using the log-cosine distance function, we get an increase across all metrics.

All our query models so far have been based on the use of an average representation of each phonemic class. While this minimizes the total within-class distortion, it does not account for separation between class and so it requires having a document representation in which the different classes are already somewhat well linearly separated. To avoid this dependency, we use the joint distance metric learning (JDML) [23, 34]. JDML learns a query model for a representation that has low within class distance and a good separation of classes along with the distance function to use. This way, we are able to increase the TWV metrics even beyond what we get with the posteriorgram with the log-cosine distance. However, when we train the JDML on the Gaussian posteriorgram, the results degrade compared to the log-cosine distance.

While JDML learns the distance metric and query model, it is still bound to the document representation that we feed it. It is not hard to imagine that with a

Table 4.3. Results on Turkish dev set with monolingual BN representation

DISTANCE	METRIC	RAW BN	GAUSSIAN POST.
Cosine	MTWV	0	0.0228
	OTWV	0.0552	0.1150
	STWV	0.4361	0.4195
Log-cosine	MTWV	-	0.1035
	OTWV	-	0.2618
	STWV	-	0.5819
JDML	MTWV	0.1164	0.0961
	OTWV	0.2809	0.2444
	STWV	0.6278	0.5676
EDML	MTWV	0.1809	0.1036
	OTWV	0.3288	0.2627
	STWV	0.6492	0.5992

better document representation, the network can more efficiently discriminate between classes. With this rationale, we propose the EDML system described in Chapter 3. Using the EDML, we are able to achieve a considerable increase in all TWV metrics for the raw BN features over the JDML system. The improvement is more modest for Gaussian posteriorgrams; in fact, the EDML results for the Gaussian posteriorgrams are worse than the JDML for raw BN features.

From this experiments, we learn that, with enough babysitting, the bottleneck representation is feasible for DTW based keyword search. In addition, we learn that although the Gaussian posteriorgrams are better than the raw bottleneck features if used directly, they are not as good for the distance metric learning based approaches. This might be due to the fact that the frame embeddings for DTW obtained from the metric learner are of a much smaller dimension than the Gaussian posteriorgrams. This is a practical consideration as the frame embeddings need to be stored in memory

during search. While the Gaussian posteriorgrams are sparse and can thus be stored efficiently in memory, the same thus not hold for the frame embeddings. Storing ten hours of $1024 - D$ frame embeddings in half-precision floating point would require:

$$10 * 360000 * 4 * 1024 \; bytes \approx 15 \text{ GB}.$$

All experiments reported henceforth use the multilingual BN representation. These subsequent experiments do not include any further attempt to use the bottleneck features directly.

4.5. Extremely Low Resource Experiments

For each of the target languages, we train our models with only a one-hour subset of the LLP training data and evaluate the performance of the systems trained thus.

4.5.1. Development set

We train two baseline systems, B and B* with one-hour and ten-hour lexicons respectively. The term weighted value splits (by IV and OOV terms) are shown in Table 4.4. Note that although the MTWV performance of the two systems on IV and OOV terms are similar, the overall performance of B* is considerably better than that of B since the OOV rate of B is significantly higher (see Tables 4.1 and 4.2).

We train a GMM (G_S) from scratch using only the bottleneck features of the one hour data from each of the target languages. Seeing as this did not perform well, we look to use the data from the source languages to bootstrap the GMM training.

Another GMM (G_M) is trained from the source languages and used directly to generate the document representation. The only knowledge used from the target languages comprises the alignments used to compute the averages for the query models. Even so, the Turkish posteriorgrams obtained in this way outperform the monolingual GMM trained on ten hours of Turkish language data (Table 4.3). Moreover, in the

Table 4.4. ELR dev set baseline MTWV results

LANGUAGE	B			B*		
	All	IV	OOV	All	IV	OOV
Pashto	0.0598	0.1458	0.0083	0.1233	0.1533	0.0449
Turkish	0.1037	0.3180	0.0166	0.2258	0.2867	0.0992
Zulu	0.0384	0.1905	0.0096	0.1472	0.2323	0.0253
Kazakh	0.0718	0.1854	0.0110	0.1321	0.1762	0.0185
Average	0.0684	0.2099	0.0114	0.1571	0.2121	0.0463

Table 4.5. MTWVs for ELR Gaussian posteriorgram on the dev set

LANGUAGE	G_S	G_M	G_F
Pashto	0.0653	0.0576	0.0516
Turkish	0.0003	0.1529	0.0002
Zulu	0.0057	0.1062	0.0032
Kazakh	0.0709	0.0799	0.0501
Average	0.0356	0.0992	0.0263

ELR setting, these posteriorgrams outperform both the baseline (B) and the language specific Gaussian posteriorgrams (G_S).

We further investigate the use of the multilingual GMM as an initialization for the final GMM (G_F). Starting with the pretrained GMM, we proceed to run the expectation maximization algorithm using only the data from the target language. The posteriorgrams generated in this way yield similar performance to those from G_S.

Our final model for the ELR setting is the EDML trained using only the bottleneck and alignments of the ELR subset. This outperformed all the GMM-based models

Table 4.6. MTWVs for best ELR systems on dev set

LANGUAGE	B*	EDML	B*+EDML	GAIN(%)
Pashto	0.1233	0.1104	0.1727	40
Turkish	0.2258	0.2737	0.3541	57
Zulu	0.1472	0.2319	0.2617	78
Kazakh	0.1321	0.1482	0.2021	53
IV Average	0.2121	0.1791	0.2713	28
OOV Average	0.0463	0.2176	0.2039	340
Average	0.1571	0.1910	0.2477	58

as well as both baselines. As the best of our systems, we kept it for fusion and eval set experiments. We combine its results with those of the B* baseline using the Cnxe-min parameters to transform each system's scores. The EDML results along with the fusion results are shown in Table 4.6.

4.5.2. Evaluation set

We use the thresholds learned on the development set to evaluate the B* and EDML systems on the evaluation set. The fusion is done with the Cnxe-min parameters learned on the development set as well. When we fuse the baseline results with the EDML, we obtain an average ATWV improvement of 52%. One thing to notice (and this also holds for the dev set experiments) is that the average OOV ATWV of the fused system is worse than that of the EDML by itself. Therefore, a better combination strategy would have been to use the fusion for IV terms and the EDML alone for OOV terms. These results are shown in Table 4.6.

4.6. Low Resource Experiments

In the LR experiments, the entirety of the LLP training data of each target language is used to train its KWS model.

Table 4.7. ATWVs for best ELR systems on eval set

LANGUAGE	B*	EDML	B*+EDML	GAIN(%)
Pashto	0.1398	0.1037	0.1881	35
Turkish	0.1600	0.1836	0.2300	44
Zulu	0.1557	0.2131	0.2560	65
Kazakh	0.0870	0.0733	0.1294	49
IV Average	0.1784	0.1383	0.2136	20
OOV Average	0.0030	0.1586	0.1257	4090
Average	0.1356	0.1434	0.2009	48

4.6.1. Development Set

In the LR setting, we have two baseline systems, B and B′. For each target language, B is trained in the same way as in the ELR setting except that the entirety of the LLP data is used to finetune the multilingual network for LVCSR decoding and search. B′ is trained with the EDML on posterior features from a monolingual DNN. Comparison with B shows the effect of using the proposed models as opposed to an LVCSR-based KWS system while comparison to B′ shows the impact of the transferred crosslingual representation.

As in the ELR setting, here also, we train three GMM posteriorgram generator for each target language. The first (G_S) is trained using only the bottleneck features in the target language; the second (G_M) is trained with the features from the source languages and is truly multilingual except for the query model which is necessarily language specific; the final GMM (G_F) uses G_M as an initialization for the language specific GMMs. In the LR dev-set experiments, we see that G_M, which was the best in the ELR setting is not nearly as good as G_S. In fact, even after further optimizing using language specific data to get G_F, G_S is still the superior model. These results are shown in Table 4.9.

Table 4.8. LR dev set baseline MTWV results

LANGUAGE	B			B′		
	All	IV	OOV	All	IV	OOV
Pashto	0.2621	0.3199	0.0931	0.1207	0.1106	0.1549
Turkish	0.4614	0.5813	0.1865	0.2783	0.2937	0.2440
Zulu	0.2912	0.3961	0.1335	0.2057	0.1684	0.2671
Kazakh	0.3359	0.4159	0.1229	0.1259	0.1127	0.1540
Average	0.3377	0.4283	0.1340	0.1827	0.1714	0.2050

Table 4.9. MTWVs for LR Gaussian posteriorgram on the dev set

LANGUAGE	G_M	G_S	G_F
Pashto	0.0739	0.1004	0.0901
Turkish	0.1577	0.2488	0.2134
Zulu	0.1245	0.1812	0.1583
Kazakh	0.0827	0.1477	0.1128
Average	0.1097	0.1695	0.1437

In addition to these Gaussian posteriorgram models, we train an EDML model for each of the target languages. The MTWV scores obtained from this are considerably better than those obtained from the Gaussian and the supervised monolingual phone posteriorgram (B′). In fact, if we the compare B′ column in Table 4.10 to the EDML column in Table 4.6, we see that even with one hour of data in the target language, our average MTWV with the multilingual representation is slightly better than what we get from monolingual training with ten hours of training data in the target language.

While we are unable to directly outperform the "B" baseline on all keywords in the LR setting with the EDML, we get a much better OOV performance and on fusion,

Table 4.10. MTWVs for best LR systems on dev set

LANGUAGE	B′	B	EDML	B+EDML	GAIN(%)
Pashto	0.1207	0.2621	0.1432	0.3052	16.4
Turkish	0.2783	0.4614	0.3320	0.5095	10.4
Zulu	0.2057	0.2912	0.2823	0.3753	28.9
Kazakh	0.1259	0.3359	0.2012	0.3789	12.8
IV Average	0.1714	0.4283	0.2260	0.4391	2.5
OOV Average	0.2050	0.1340	0.2747	0.2906	116.9
Average	0.1827	0.3377	0.2397	0.3922	16.2

we get an MTWV improvement of 15% on all keywords.

We have claimed in Section 3.2 that using three state phone representations is better than using a single state per phone because it provides finer discrimination for DTW search. To verify this claim, we run train an EDML model using only a single state for each phonemic classes. From Table 4.11 we see that the use of three-state phones gives an average relative MTWV gain of 6.7%.

Table 4.11. Dev set MTWV of single-state and three-state phone models for EDML

LANGUAGE	Single State	Three States	GAIN(%)
Pashto	0.1376	0.1432	4
Turkish	0.3295	0.3320	0.7
Zulu	0.2580	0.2823	9
Kazakh	0.1776	0.2012	13
Average	0.2257	0.2397	6.7

4.6.2. Evaluation Set

Using the parameters learned on the dev set, we conduct experiments whose results are shown in Table 4.12. As expected, since we the EDML system does not use the lexicon or language model, it performs worse than the baseline on IV terms, but its OOV term retrieval performance is significantly better.

Table 4.12. ATWVs for best LR systems on eval set

LANGUAGE	B′	B	EDML	B+EDML	GAIN(%)
Pashto	0.1180	0.3141	0.1540	0.3320	5.7
Turkish	0.1778	0.3665	0.2307	0.4267	16.4
Zulu	0.1956	0.3032	0.2552	0.3693	21.8
Kazakh	0.0654	0.2729	0.1251	0.3085	13.0
IV Average	0.1360	0.3810	0.1815	0.3936	3.3
OOV Average	0.1473	0.0971	0.2241	0.2422	149.4
Average	0.1392	0.3142	0.1912	0.3591	14.3

4.7. KWS with Low Entropy Features

The experiments described in this section are based on the modified EDML described in Section 3.6. Using various values of the entropy weight, we obtain different representations of varying sparsity and conduct our DTW based search on them. We measure the sparsity in terms of the average frame entropy of the spoken document and the density of the document. The density is measured as the ratio of the number of elements greater than some threshold (we use 10^{-4}) to the total number of elements. The density is linearly proportional to the amount of memory required to store the document.

4.7.1. Development Set

We conduct experiments using three values of λ $(0.01, 0.1$ and $1)$ and compare these to the default case of $\lambda = 0$. By increasing λ, we are able to get much sparser representations. Using $\lambda = 0.1$, we get an average compression ratio of $3.85 : 1$ without much loss in TWV compared to the original EDML (see Table 4.10). Further increasing the entropy weight to $\lambda = 1$ gives a compression ratio of $14.3 : 1$ with a relative average MTWV drop of about 19%.

Table 4.13. Dev set MTWV for low-entropy EDML with different values of λ

LANGUAGE	$\lambda = 0$	$\lambda = 0.01$	$\lambda = 0.1$	$\lambda = 1$
Pashto	0.1420	0.1494	0.1478	0.0967
Turkish	0.3177	0.3301	0.3399	0.2883
Zulu	0.2664	0.2730	0.2735	0.2465
Kazakh	0.1755	0.1753	0.1827	0.1437
Average	0.2254	0.2320	0.2360	0.1938

Table 4.14. Dev set sparsity metrics

LANGUAGE	Entropy				Density			
λ	0	0.01	0.1	1	0	0.01	0.1	1
Pashto	3.547	3.343	3.020	1.833	0.485	0.740	0.172	0.045
Turkish	3.153	3.468	3.209	2.241	0.602	0.385	0.279	0.070
Zulu	3.651	3.440	3.381	2.235	0.450	0.361	0.297	0.070
Kazakh	3.627	3.590	3.430	2.498	0.410	0.409	0.296	0.095
Average	3.495	3.460	3.260	2.202	0.487	0.474	0.261	0.070

Tables 4.13 and 4.14 show the retrieval and compression performance at different values of λ respectively. The same information is summarized in Figure 4.1

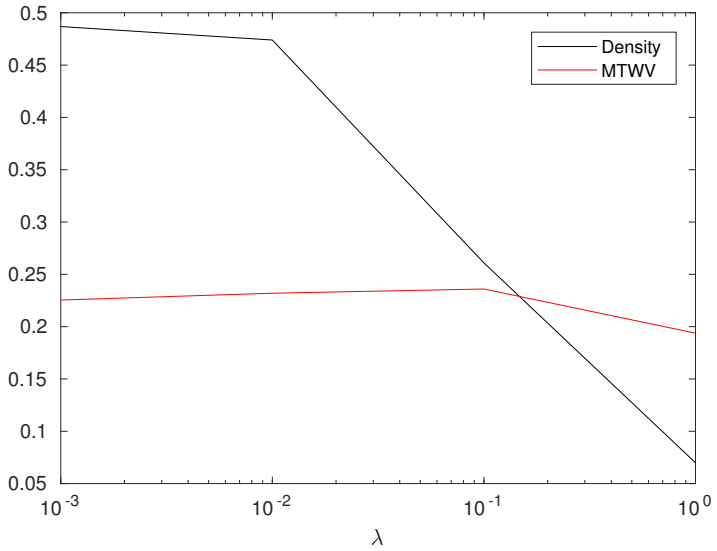

Figure 4.1. Evolution of density and MTWV with change in λ.

4.7.2. Evaluation Set

In the eval set we use the low entropy EDML representations with hyperparameters learned from the dev set. While there is some degradation in ATWV compared to the regular EDML, it is not overly pronounced until we use a very high value of λ (which of course gives very high compression rates).

Table 4.15. Eval set ATWV for low-entropy EDML with different values of λ

LANGUAGE	$\lambda = 0$	$\lambda = 0.01$	$\lambda = 0.1$	$\lambda = 1$
Pashto	0.1478	0.1597	0.1510	0.1124
Turkish	0.2299	0.2295	0.2216	0.2038
Zulu	0.2375	0.2486	0.2450	0.2360
Kazakh	0.0885	0.1018	0.1009	0.0714
Average	0.1759	0.1849	0.1796	0.1559

Table 4.16. Eval set sparsity metrics

LANGUAGE	Entropy				Density			
λ	0	0.01	0.1	1	0	0.01	0.1	1
Pashto	3.545	3.356	3.024	1.833	0.484	0.745	0.172	0.045
Turkish	3.151	3.469	3.210	2.240	0.602	0.385	0.279	0.070
Zulu	3.657	3.452	3.389	2.235	0.451	0.362	0.297	0.070
Kazakh	3.628	3.590	3.429	2.497	0.410	0.409	0.296	0.095
Average	3.496	3.466	3.263	2.201	0.487	0.475	0.261	0.070

Tables 4.15 and 4.16 show the ATWV and retrieval performance of the system on the eval set. Figure 4.2 summarizes the same information.

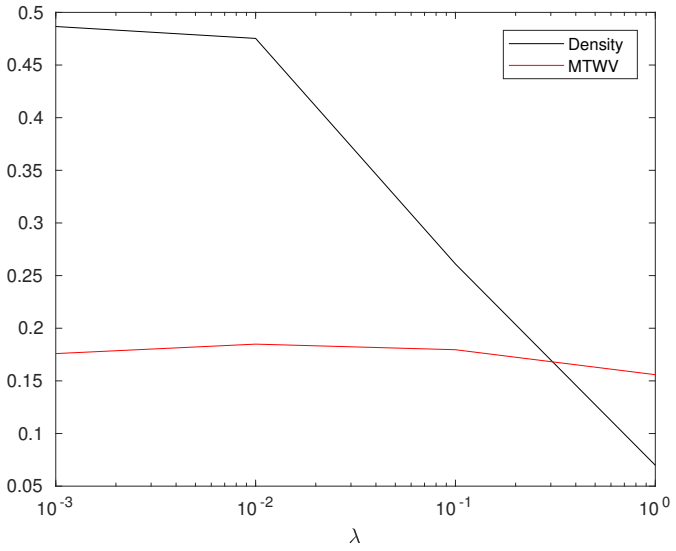

Figure 4.2. Evolution of eval set density and ATWV with change in λ.

5. CONCLUSIONS

In this thesis, we tackle some of the issues associated with keyword search in data-scarce settings, namely the paucity of acoustic and linguistic data required for building an LVCSR based KWS engine. To deal with the acoustic data scarcity, we propose bootstrapping with data from other languages; to reduce the impact of the paucity of linguistic data, we propose models based on dynamic time warping template matching since it is less dependent on linguistic data and so is less sensitive to the scarcity of such. Where the LVCSR system requires external assistance to deal with out-of-vocabulary keywords, the proposed models does not differentiate between IV and OOV terms.

The crosslingual information is extracted by training a multilingual neural network with a low dimensional bottleneck layer whose activations are used as features in the search. Although the source languages used in training the multilingual representation do not include the target languages on which the actual search is conducted, the representation proves to be useful for search in the target language nevertheless. By transforming bottleneck features in various ways, we are able to obtain different representations of the search document. Since the queries are in a different modality from the document, we transform them into sequences of sub-phonemic states which we then artificially model to resemble the document and then conduct the keyword search.

Finding that the bottleneck features obtained from the multilingual neural network do not lend themselves easily to query modeling, we generate a new document representation by training unsupervised GMMs with the bottleneck features; these we use to generate Gaussian posteriorgrams and form queries by taking an average of samples of each class in the training data. We propose using posteriorgrams obtained from target language or multilingual (from the source languages) Gaussian mixture models. When the amount of available training data in the target language is extremely limited, we find that the multilingual Gaussian posteriorgrams outperform those obtained

from the GMMs trained on the target languages' bottleneck features. As the amount of target language data is increased though, the posteriorgrams generated from GMMs trained on target languages' data perform much better than those generated from the multilingual GMMs.

We also propose a model (EDML) that simultaneously learns a document representation, a query model for that representation and a distance function for these representations. Using this model, we are able to far exceed the performance of the Gaussian posteriorgrams. In an extremely low resource setting where only one hour of training data is used from each target language, EDML surpasses the LVCSR baseline even when the baseline is trained with an augmented lexicon of ten hours. In the milder low resource setting where ten hours of training data is used per target language though, EDML is only able to outperform the baseline on OOV terms. When the detection results of EDML are combined with those from the baseline, an overall ATWV increase of about 14% is observed including a 149% improvement on OOV terms.

Like other systems that use the dynamic time warping algorithm, the EDML-based KWS system has a high memory cost. In addition to the intermediate memory usage incurred during search, there is a permanent cost associated with keeping the document in memory. To alleviate this cost, we propose an alteration to the EDML that biases it towards outputting sparser frame embeddings. With this modification, we're able to reduce the memory footprint considerably at the cost of some retrieval performance.

Since our system performance is evaluated on a single score threshold for all queries, it is necessary to ensure that the detection score distributions are normalized across keywords. We propose a generalization of an existing normalization technique which, in addition to normalizing the keywords, prunes out more irrelevant hypotheses and gives a higher dynamic range of normalized scores. With this high dynamic range, thresholds are less sensitive to slight numerical perturbations in the hypotheses' score.

That the fusion of multiple systems, especially those that perform complementary functions, leads to better performance is a well established fact in machine learning in general, and keyword search in particular. We propose a fusion methodology that works by re-calibrating the scores of each system with a transformation that maximizes the separation between spurious hypotheses and true ones. After this, a simple addition is enough to combine the scores from the systems.

Further work will mostly involve reducing the computational cost of running the proposed systems. This will involve using approximations that allow the pruning away of irrelevant swathes of the document before running the fine dynamic time search. For instance, by using the lower bound dynamic time warping approaches or the pre-filtering method on the low entropy frame embeddings from the sparse EDML, further gains in processing time could be obtained. Additionally, a recurrent neural network summarizer could be trained with which a quick comparison between the query and portions of the document could be made before then running the full DTW.

REFERENCES

1. Sarı, L., B. Gündoğdu and M. Saraçlar, "Fusion of LVCSR and posteriorgram based keyword search", pp. 824–828.

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

3. Logan, B., J.-M. Van Thong and P. J. Moreno, "Approaches to reduce the effects of OOV queries on indexed spoken audio", *IEEE Transactions on Multimedia*, Vol. 7, No. 5, pp. 899–906, 2005.

4. Saraçlar, M. and R. Sproat, "Lattice-based search for spoken utterance retrieval", *HLT-NAACL 2004: Main Proceedings*, Vol. 51, pp. 129–136, 2004.

5. Szöke, I., M. Fapšo and L. Burget, "Hybrid word-subword decoding for spoken term detection", *The 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 42–48, 2008.

6. Mamou, J., B. Ramabhadran and O. Siohan, "Vocabulary independent spoken term detection", *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 615–622, 2007.

7. Gandhe, A., L. Qin, F. Metze, A. Rudnicky, I. Lane and M. Eck, "Using web text to improve keyword spotting in speech", *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pp. 428–433, IEEE, 2013.

8. Chen, G., O. Yilmaz, J. Trmal, D. Povey and S. Khudanpur, "Using proxies for OOV keywords in the keyword search task", *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pp. 416–421, Dec 2013.

9. Saraclar, M., A. Sethy, B. Ramabhadran, L. Mangu, J. Cui, X. Cui, B. Kingsbury and J. Mamou, "An empirical study of confusion modeling in keyword search for low resource languages", *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), Dec 2013*, 2013.

10. Rabiner, L. and B.-H. Juang, *Fundamentals of Speech Recognition*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

11. Huang, X., A. Acero and H.-W. Hon, *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*, Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edn., 2001.

12. Hinton, G., L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups", *IEEE Signal Processing Magazine*, Vol. 29, No. 6, pp. 82–97, 2012.

13. Abdou, S. and M. S. Scordilis, "Beam search pruning in speech recognition using a posterior probability-based confidence measure", *Speech Communication*, Vol. 42, No. 3-4, pp. 409–428, 2004.

14. Mangu, L., E. Brill and A. Stolcke, "Finding consensus in speech recognition: word error minimization and other applications of confusion networks", *Computer Speech & Language*, Vol. 14, No. 4, pp. 373–400, 2000.

15. Mohri, M., F. Pereira and M. Riley, "Weighted finite-state transducers in speech recognition", *Computer Speech and Language*, Vol. 16, No. 1, pp. 69–88, 2002.

16. Chelba, C., T. J. Hazen and M. Saraclar, "Retrieval and browsing of spoken content", *IEEE Signal Processing Magazine*, Vol. 25, No. 3, pp. 39–49, 2008.

17. Hori, T., I. L. Hetherington, T. J. Hazen and J. R. Glass, "Open-vocabulary spoken utterance retrieval using confusion networks", *IEEE International Conference on*

Acoustics, Speech and Signal Processing (ICASSP), Vol. 4, pp. 73–76, IEEE, 2007.

18. Can, D., E. Cooper, A. Ghoshal, M. Jansche, S. Khudanpur, B. Ramabhadran, M. Riley, M. Saraclar, A. Sethy, M. Ulinski *et al.*, "Web derived pronunciations for spoken term detection", *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 83–90, ACM, 2009.

19. Qin, L. and A. Rudnicky, "OOV word detection using hybrid models with mixed types of fragments", *Interspeech*, pp. 2450–2453, 2012.

20. Bisani, M. and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion", *Speech communication*, Vol. 50, No. 5, pp. 434–451, 2008.

21. Gündoğdu, B., L. Sarı, G. Çetinkaya and M. Saraçlar, "Template-based keyword search with pseudo posteriorgrams", *24th Signal Processing and Communications Applications Conference (SIU)*, pp. 973–976, IEEE, 2016.

22. Gündoğdu, B. and M. Saraclar, "Distance Metric Learning for Posteriorgram Based Keyword Search", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5660–5664.

23. Gündoğdu, B. and M. Saraçlar, "Similarity Learning Based Query Modeling for Keyword Search", *Interspeech*, pp. 3617–3621, 2017.

24. Thomas, S., S. Ganapathy and H. Hermansky, "Cross-lingual and multistream posterior features for low resource LVCSR systems", *Interspeech*, pp. 877–880, 2010.

25. Vu, N. T., D. Imseng, D. Povey, P. Motlicek, T. Schultz and H. Bourlard, "Multilingual deep neural network based acoustic modeling for rapid language adaptation", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7639–7643, IEEE, 2014.

26. Vu, N. T., W. Breiter, F. Metze and T. Schultz, "An Investigation on Initialization Schemes for Multilayer Perceptron Training Using Multilingual Data and Their Effect on ASR Performance", *Interspeech*, pp. 2586–2589, 2012, `http://repository.cmu.edu/lti`.

27. Ghoshal, A., P. Swietojanski and S. Renals, "Multilingual training of deep neural networks", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7319–7323, IEEE, 2013.

28. Scanzio, S., P. Laface, L. Fissore, R. Gemello and F. Mana, "On the use of a multilingual neural network front-end", *Interspeech*, pp. 2711–2714, 2008.

29. Veselý, K., M. Karafiát, F. Grézl, M. Janda and E. Egorova, "The language-independent bottleneck features", pp. 336–341, 2012.

30. Heigold, G., V. Vanhoucke, A. Senior, P. Nguyen, M. Ranzato, M. Devin and J. Dean, "Multilingual acoustic models using distributed deep neural networks", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8619–8623, IEEE, 2013.

31. Hermansky, H., D. P. Ellis and S. Sharma, "Tandem connectionist feature extraction for conventional HMM systems", *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 3, pp. 1635–1638, IEEE, 2000.

32. Grézl, F., M. Karafiát and M. Janda, "Study of probabilistic and Bottle-Neck features in multilingual environment", *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pp. 359–364, 2011.

33. Sainath, T. N., B. Kingsbury, V. Sindhwani, E. Arisoy and B. Ramabhadran, "Low-rank matrix factorization for Deep Neural Network training with high-dimensional output targets", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6655–6659, 2013, `http://ieeexplore.ieee.org/document/6638949/`.

34. Gündoğdu, B., B. Yusuf and M. Saraçlar, "Joint Learning of Distance Metric and Query Model for Posteriorgram-Based Keyword Search", *IEEE Journal of Selected Topics in Signal Processing*, Vol. 11, No. 8, pp. 1318–1328, 2017.

35. Brown, M. and L. Rabiner, "Dynamic time warping for isolated word recognition based on ordered graph searching techniques", *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 7, pp. 1255–1258, May 1982.

36. Aradilla, G., J. Vepa and H. Bourlard, "Using posterior-based features in template matching for speech recognition", *Interspeech*, pp. 2570–2573, 2006.

37. Hazen, T. J., W. Shen and C. White, "Query-by-example spoken term detection using phonetic posteriorgram templates", *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pp. 421–426, 2009.

38. Zhang, Y. and J. R. Glass, "Unsupervised spoken keyword spotting via segmental DTW on Gaussian posteriorgrams", *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pp. 398–403, 2009.

39. Park, A. S. and J. R. Glass, "Unsupervised pattern discovery in speech", *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 16, No. 1, pp. 186–197, 2008.

40. Sakoe, H. and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 26, No. 1, pp. 43–49, 1978.

41. Lin, Y.-L., T. Jiang and K.-M. Chao, "Efficient algorithms for locating the length-constrained heaviest segments with applications to biomolecular sequence analysis", *Journal of Computer and System Sciences*, Vol. 65, No. 3, pp. 570–586, 2002.

42. Lee, L.-s., J. Glass, H.-y. Lee and C.-a. Chan, "Spoken content retrieval: Beyond

cascading speech recognition with text retrieval", *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, Vol. 23, No. 9, pp. 1389–1420, 2015.

43. Müller, M., *Information Retrieval for Music and Motion*, Springer-Verlag, Berlin, Heidelberg, 2007.

44. Anguera, X. and M. Ferrarons, "Memory efficient subsequence DTW for Query-by-Example Spoken Term Detection", *IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, July 2013.

45. Gajjar, M. R., R. Govindarajan and T. Sreenivas, "Online unsupervised pattern discovery in speech using parallelization", *Interspeech*, pp. 2458–2461, 2008.

46. Srikanthan, S., A. Kumar and R. Gupta, "Implementing the dynamic time warping algorithm in multithreaded environments for real time and unsupervised pattern discovery", *2nd International Conference on Computer and Communication Technology (ICCCT),*, pp. 394–398, IEEE, 2011.

47. Zhang, Y., K. Adl and J. Glass, "Fast spoken query detection using lower-bound dynamic time warping on graphical processing units", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5173–5176, IEEE, 2012.

48. Qiao, Y., N. Shimomura and N. Minematsu, "Unsupervised optimal phoneme segmentation: Objectives, algorithm and comparisons", *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pp. 3989–3992, IEEE, 2008.

49. Chan, C.-a. and L.-s. Lee, "Unsupervised spoken-term detection with spoken queries using segment-based dynamic time warping", *Interspeech*, pp. 693–696, 2010.

50. Chan, C.-a. and L.-s. Lee, "Integrating frame-based and segment-based dynamic time warping for unsupervised spoken term detection with spoken queries", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5652–5655, May 2011.

51. Keogh, E. and C. A. Ratanamahatana, "Exact indexing of dynamic time warping", *Knowledge and Information Systems*, Vol. 7, No. 3, pp. 358–386, Mar 2005.

52. Zhang, Y. and J. R. Glass, "An inner-product lower-bound estimate for dynamic time warping", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5660–5663, 2011.

53. Zhang, Y. and J. Glass, "A piecewise aggregate approximation lower-bound estimate for posteriorgram-based dynamic time warping", *Interspeech*, pp. 1909–1912, 2011.

54. Yang, P., L. Xie, Q. Luan and W. Feng, "A tighter lower bound estimate for dynamic time warping", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8525–8529, 2013.

55. Cetinkaya, G., B. Gundogdu and M. Saraclar, "Pre-filtered dynamic time warping for posteriorgram based keyword search", *Proceedings of the IEEE Workshop on Spoken Language Technology (SLT)*, pp. 376–382, Dec 2016.

56. Jansen, A. and B. Van Durme, "Efficient spoken term discovery using randomized algorithms", *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pp. 401–406, 2011.

57. Indyk, P. and R. Motwani, "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality", *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 604–613, 1998.

58. Charikar, M. S., "Similarity estimation techniques from rounding algorithms", *Pro-*

ceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC), pp. 380–388, ACM, 2002.

59. Jansen, A. and B. V. Durme, "Indexing raw acoustic features for scalable zero resource search", *Interspeech*, pp. 2466–2469, 2012.

60. Jansen, A., K. Church and H. Hermansky, "Towards spoken term discovery at scale with zero resources", *Interspeech*, pp. 1676–1679, 2010.

61. Peddinti, V., D. Povey and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts", *Interspeech*, pp. 3214–3218, 2015.

62. Anguera, X., "Speaker independent discriminant feature extraction for acoustic pattern-matching", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 485–488, IEEE, 2012.

63. Kingma, D. P. and J. Ba, "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.

64. Mamou, J., J. Cui, X. Cui, M. J. Gales, B. Kingsbury, K. Knill, L. Mangu, D. Nolden, M. Picheny, B. Ramabhadran *et al.*, "System combination and score normalization for spoken term detection", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8272–8276, 2013.

65. Montague, M. and J. A. Aslam, "Relevance score normalization for metasearch", *Proceedings of the 10th International Conference on Information and Knowledge Management*, pp. 427–433, 2001.

66. Szoke, I., L. Burget, F. Grezl, J. H. Cernocky and L. Ondel, "Calibration and fusion of query-by-example systems—BUT SWS 2013", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7849–7853, 2014.

67. Gündoğdu, B. and M. Saraçlar, "Novel score normalization methods for keyword search", *25th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, 2017.

68. Harper, M., *IARPA Babel program*, 2014, `https://www.iarpa.gov/index.php/research-programs/babel`, accessed at June 2018.

69. Fiscus, J. G., J. Ajot, J. S. Garofolo and G. Doddingtion, "Results of the 2006 spoken term detection evaluation", *Proceedings of the ACM SIGIR Workshop on Searching Spontaneous Conversational Speech*, pp. 51–57, 2007.

70. Brümmer, N. and J. Du Preez, "Application-independent evaluation of speaker detection", *Computer Speech & Language*, Vol. 20, No. 2-3, pp. 230–275, 2006.

71. Brummer, N. and D. A. V. Leeuwen, "On calibration of language recognition scores", *2006 IEEE Odyssey - The Speaker and Language Recognition Workshop*, pp. 1–8, June 2006.

72. Rodriguez-Fuentes, L. J. and M. Penagarikano, *MediaEval 2013 Spoken Web Search Task: System Performance Measures*, Tech. Rep. TR-2013-1, Department of Electricity and Electronics, University of the Basque Country, 2013.

73. Brummer, N. and J. d. Preez, "The PAV algorithm optimizes binary proper scoring rules", *arXiv preprint arXiv:1304.2331*, 2013.

74. Proenga, J., A. Veiga and F. Perdigão, "Query by example search with segmented dynamic time warping for non-exact spoken queries", *23rd European Signal Processing Conference (EUSIPCO)*, pp. 1661–1665, 2015.

75. Miller, D. R., M. Kleber, C.-L. Kao, O. Kimball, T. Colthurst, S. A. Lowe, R. M. Schwartz and H. Gish, "Rapid and accurate spoken term detection", *Interspeech*, pp. 314–317.