END-TO-END OPEN VOCABULARY KEYWORD SEARCH

by

Bolaji Yusuf

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

M.S., Electrical and Electronics Engineering, Boğaziçi University, 2018

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

Graduate Program in Electrical and Electronics Engineering

Boğaziçi University

2024

To Arike and Olayiwola, beloved, beloved

By your eyes, in your voice, I write

ACKNOWLEDGEMENTS

This work was supported by the European Union's Horizon 2020 project No. 870930 - WELCOME, Czech Ministry of Interior project No. VJ01010108 "ROZKAZ", the Scientific and Technological Research Council of Turkey (TUBITAK) under Project 116E076, the Turkish Directorate of Strategy and Budget under the ROYAL Project (2019K12-149250) as well as the Boğaziçi University Research Fund under the Grant Number 16903. Computing on IT4I supercomputer was supported by the Czech Ministry of Education, Youth and Sports through the e-INFRA CZ (ID:90254).

ABSTRACT

END-TO-END OPEN VOCABULARY KEYWORD SEARCH

Keyword search (KWS) solves the problem of searching written queries in spoken documents, thereby facilitating cataloging of and information retrieval from large archives of speech. Conventional keyword search entails transcribing the archive into text with an automatic speech recognition (ASR) system and then searching queries in the resulting transcriptions. Although this approach has grown into a mature and reliable technology, ASR-based cataloging is a complicated undertaking. In this dissertation, we propose a novel end-to-end neural-network-based KWS paradigm in which we replace the graph decoding and symbol matching of ASR-based systems with a simpler search method based on dot products on the outputs of a pair of encoding neural networks. In our method, documents are transformed into sequences in the output space of one encoder, and queries are projected to the same vector space by another encoder. Search is then conducted by comparing, by means of dot products, the query vector to the vectors representing frames of the document. Locations in the document with high dot product similarity to the query are returned as hits and locations with low dot products are ignored as background. We further adopt a multilingual training strategy which increases the viability of the proposed framework for KWS in languages for which low amounts of training data are available. Finally, we propose a scheme which jointly learns the KWS task of searching for text in speech with a task of searching for text in text, with the latter task allowing us to integrate unpaired text into the proposed KWS model. We conduct in-depth experiments across several languages analyzing the various properties of the proposed KWS method and showing its viability for competitive KWS performance both as an alternative and as a complement to ASR-based KWS.

ÖZET

UÇTAN-UCA AÇIK SÖZVARLIKLI ANAHTAR SÖZCÜK ARAMA

Anahtar Sözcük Arama (ASA), yazılı olarak verilen sorgu sözcüklerinin konuşma kayıtlarında aranması problemini çözer ve böylece büyük konuşma arşivlerinden arzulanan bilginin edinilmesini ve bu arşivlerin otomatik olarak kataloglanmasını kolaylaştırır. Geleneksel ASA yöntemi, konuşma arşivinin bir otomatik konuşma tanıma (OKT) sistemi kullanılarak metne çevirilmesi ve ardından elde edilen transkriptlerde sorguları arama sürecini içerir. Bu yaklaşım olgunlaşmış ve güvenilir bir teknolojiye dönüşmüş olsa da, OKT tabanlı kataloglama karmaşık bir işlemdir. Bu tezde, OKT tabanlı sistemlerdeki çizge çözümlemesi ve sembol eşleştirmesini iki adet kodlayıcı sınır ağı çıktısının iç çarpımı tabanlı daha basit bir arama yöntemi ile değiştiren, uçtan uca yapay sınır ağı tabanlı özgün bir ASA yöntemi önerilmektedir. Bu yöntemde, konuşma belgeleri bir kodlayıcı ile vektörel dizilere dönüştürülmekte ve sorgu metinleri de başka bir kodlayıcı vasıtasıyla aynı vektör uzayına yansıtılmaktadır. Sonrasında arama işlemi, sorgu gösterimlerininin konuşma dökümanına ait vektörel dizileri ile iç çarpımlarını karşılaştırarak gerçekleştirilir. Konuşma gösterimleri içinde sorgu gösterimine yüksek iç çarpım skoruna sahip bölgeler isabet olarak değerlendirilir ve düşük iç çarpım değerine sahip yerler ise ilgisiz konuşma olarak değerlendirilerek göz ardı edilir. Ayrıca, kısıtlı eğitim verisine sahip diller için bu tezde önerilen yöntemin uygulanabilirliğini artıran çok dilli bir eğitim stratejisi benimsenmiştir. Son olarak, konuşma içindeki metni arama problemi olan ASA ile metin içindeki metni arama problemi birlikte öğrenen bir yöntem de önerilmiştir; böylece önerilen konuşma kaydı olamayan metinlerin de ASA modeline entegre edilmesi sağlanmıştır. Çeşitli dillerde gerçekleştirilen derinlemesine deneyler ile, önerilen yöntemin birçok özelliği analiz edilmiş ve OKT tabanlı ASA sistemlerine hem alternatif hem de bu sistemlerin eksiklerini giderebilen bir tamamlayıcı olarak güçlü bir ASA sistemi olarak uygulanabilirliği ortaya konmuştur.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$\mathbb{1}$	Indicator function
$\mathbf{e}_{\boldsymbol{q}}$	Embedding of the query $\boldsymbol{q}$ obtained from the query encoder
$\mathbf{H}_{\boldsymbol{X}}$	Embedding of the document $\boldsymbol{X}$ obtained from the document encoder
$\boldsymbol{q}$	Written query comprising of sequence of letters
$\boldsymbol{W}^{text}$	Written document comprising of sequence of letters
$\boldsymbol{X}$	Spoken document comprising of sequence of acoustic features
$\boldsymbol{\theta}$	Parameters of keyword search model
$\boldsymbol{\theta}^{text}$	Parameters of text document pre-encoder
λ	Weight of positive frames in the training objective function
ϕ	Tolerance parameter in the training objective
$\sigma(x)$	Logistic sigmoid function applied to the argument x

LIST OF ACRONYMS/ABBREVIATIONS

ASR	Automatic Speech Recognition
ATWV	Actual Term Weighted Value
BeKWS	Baseline End-to-End Keyword Search
BLSTM	Bidirectional Long Short-Term Memory
BNF	Bottleneck Features
BNTR	Turkish Broadcast News
E2E	End-to-End
FA	False Alarm
FLP	Full Language Pack
FST	Finite State Transducer
GMM	Gaussian Mixture Model
GRU	Gated Recurrent Unit
HMM	Hidden Markov Model
IV	In Vocabulary
JOSTER	Joint Speech and Text Retriever
KWS	Keyword Search
LLP	Limited Language Pack
LM	Language Model
LSTM	Long Short-Term Memory
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
SGD	Stochastic Gradient Descent
STD	Spoken Term Detection
STWV	Supremum Term Weighted Value
TTS	Text-to-Speech

TWV	Term Weighted Value

1. INTRODUCTION

Search lies at the center of much of the world wide web's utility. With the colossal number of documents on the web, and as this number continues to grow, users rely on the existence of search engines which have the capacity to efficiently crawl and index these documents, enabling the user to retrieved desired information from the sea of irrelevant data. The fact that that a large part of the content online is in multimedia form has necessitated research into techniques to allow accurate and efficient search of such content. This thesis focuses on one such technology—Keyword Search (KWS).

KWS, which is also called spoken term detection (STD) in some literature, aims to solve the problem of information retrieval from large archives of speech such as recordings of lectures, broadcast news, podcasts etc. A KWS system takes a user's query—containing one or more words—and returns a list of documents (recordings) in the archive where it hypothesizes that the query was uttered, timestamps for each such hypothesis for ease of browsing, and scores denoting its confidence in each hypotheses so that the hypothesis list can be sorted or filtered by relevance.

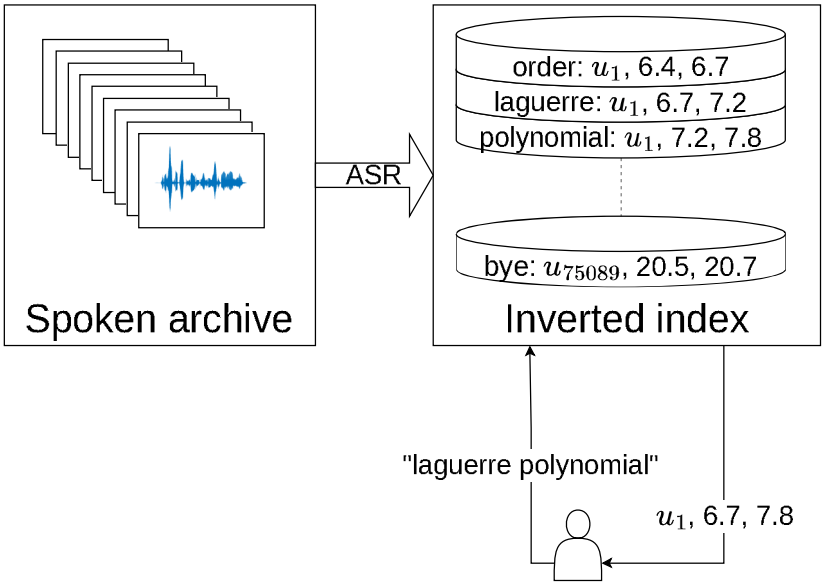

Figure 1.1. Simplified schema of ASR-based keyword search.

Since text is a more amenable modality to work with compared to speech, a natural solution to KWS, illustrated in Figure 1.1, involves first transcribing the speech archive into text form by means of an automatic speech recognition (ASR) system, and then creating an index of the resulting text for search. However, since ASR is inherently an erroneous process, simply indexing one-best transcriptions can result in low recall because any erroneously transcribed terms would become irretrievable. The widely adopted solution to mitigating the impact of such errors is to construct the index from the graph of alternate transcription hypotheses—known as a lattice—or other similar structures from the ASR system, with each entry in the index weighted by the probability assigned to it by the ASR system. Doing so allows terms to be potentially retrieved even if they are not in the single most likely transcription of a particular speech segment. Storing and searching these graph structures—typically implemented as operations on finite state transducer (FST)—incur memory and computational costs which are sub-linear in the total length of the archive, allowing ASR-based KWS to be deployed in large-scale state-of-the-art retrieval systems.

This traditional approach to KWS however has pitfalls that result from of its reliance on an ASR back-end. First, indexing requires decoding with a large vocabulary continuous speech recognition (LVCSR) system, which incurs—in addition to the cost of running the speech through an acoustic encoder—the cost of the costly graph search decoding algorithms required to compute the ASR hypotheses, and obtain timing and confidence information. While there are end-to-end (E2E) ASR methods which feature otherwise simpler training and decoding procedures, they have to resort to the complex graph-based decoding to extract timing information required for KWS.

Furthermore, conventional LVCSR uses word language models (LM) and lexicons which limit the LVCSR output space to the set of valid words. Although this significantly reduces the recognition error rates, it also limits the entries in the resulting index and, ergo, the set of queries that can be retrieved to the set of words that have been seen at training time. Thus, out-of-vocabulary (OOV) queries, i.e., queries which contain words that are not anticipated at training time, cannot be directly retrieved.

This especially poses a problem for agglutinative languages whose rich morpohologies preclude the possibility of full coverage by any word LM. Even for languages with more contrained morphologies, neologisms, technical terms, proper nouns and other named entities constitute a large source of OOV terms, whose influence can be especially outsized in retrieval since they are often the more "interesting" terms to users. For instance, the terms "Fourier transform" and "Lebesgue measure" are far more likely search terms in an archive containing engineering lecture recordings than would be suggested by their frequencies in common parlance and in most training corpora.

Although there are methods for retrieving OOV terms in LVCSR-based KWS such as the use of subword indexes or query expansion, they tend to complicate the retrieval system even further, often leading to different processing pipelines or multiple indexes for in-vocabulary (IV) and OOV queries with the OOV pipeline incurring higher computational cost. For instance, subword indexes are larger than word indexes and are therefore significantly costlier to search through, while query expansion involves simultaneously searching for several "proxy" IV terms deemed acoustically-similar to an OOV term, a process which increases the search cost by a factor proportional to the number of proxies.

These complications open the question of research into KWS systems which can avoid some of the pitfalls of operating downstream of ASR: the cost and complexity of indexing new data, and the inability to handle OOVs naturally without further increasing the complexity and computational cost of search.

In this thesis, we propose an an alternative KWS paradigm, End-to-end Keyword Search, to address the aforementioned issues in conventional KWS. In this paradigm, we rely on the ability of neural networks to model complex interactions to directly learn to search, i.e., we train a neural network to directly predict the locations of queries in spoken archives. A naive way of doing this would be to build a neural classifier of queries. However, such an approach would be limited to classifying queries which are seen during training, making it unsuitable for the inherently open-vocabulary

nature of KWS. We instead propose a vector-space query matching framework. In this framework, we use a neural network to learn a vector transformation such that when a query and a frame of a document are projected by the network to its output vector space, their dot product is high if and only if the frame is part of a span of the document where the query has been uttered.

Our framework, illustrated in Figure 1.2, uses a pair of neural encoders which are trained jointly. The first encoder, the document encoder, is trained to generate embeddings (vector representations) for each time frame of each document in the archive to be searched. This can be viewed as a continuous-space analog of index creation. The second encoder, the query encoder, similarly generates a corresponding embedding for each query. Then the query embedding is compared (by taking dot products) with each document frame embedding, and locations with high similarities to the query are returned as KWS hypotheses.

When training the model, we use word-level time boundary information which we obtain by forced-alignment with a simple Hidden Markov Model (HMM)-based ASR system with Gaussian Mixture Model (GMM) emission probability distribution. While this means that we still rely on an ASR model for training, it is important to note that indexing and search do not involve are entirely vector based and do not involve any complex graph search methods.

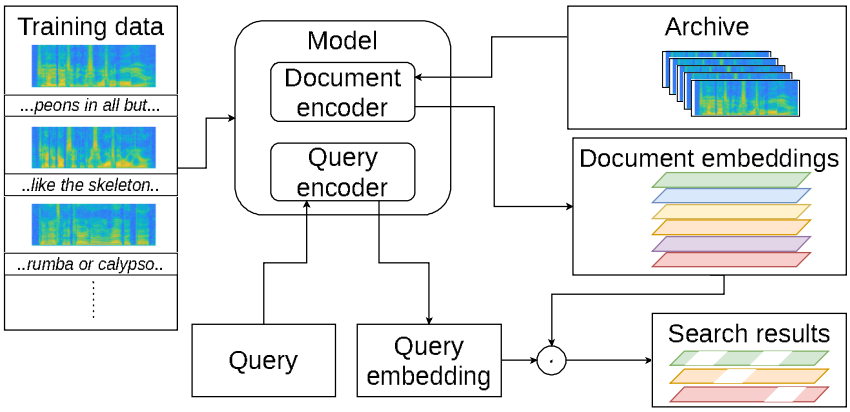

Figure 1.2. E2E KWS.

Overall, we make the following contributions in this thesis:

(i) We introduce a novel open vocabulary end-to-end keyword search model, which replaces discrete indexes with continuous ones and uses dot products to effect index lookup. More concretely, the model contains a pair of neural encoders: a document encoder for generating the vector index and a lightweight query encoder for generating the query encoding which can be searched in the vector index with frame-wise dot products. The dot products at each time frame of the document yields a score which we interpret as log-likelihood ratio between the hypothesis that the query has been uttered in a span including that time frame and the null hypothesis that it has not. Contiguous frames with higher query occurrence probability than null probability are combined and returned as a single hypothesis. This contribution has been published in [1] and [2].

(ii) We propose a training objective function based on modification of the binary cross-entropy objective. The proposed objective improves modeling efficiency through the use of a tolerance term which excludes easily classified locations from the training loss computation, thereby allowing the model's capacity to be focused on more difficult locations. This contribution has been published in [2].

(iii) We explore multilingual training as a way of increasing the effective amount of training data. Noting that several languages can have similar phonetic constructs if not semantic ones, we propose sharing the model across several languages, training it in a multilingual fashion, and then finetuning it with language-specific data from target languages. This contribution has been published in [1].

(iv) We propose joint speech and text retriever (JOSTER), a training scheme in which we modify the E2E KWS model to allow not only text retrieval from speech but also text retrieval from text. By sharing parameters, the text-speech retrieval is able to benefit from text-text retrieval which can be trained on any text corpus. This is analogous to LM training in ASR-based KWS which allows large unpaired text corpora to be used to improve performance, and like LM training, opens the possibility of domain adaptation with unpaired text data. This part of the thesis has been accepted for publication in a future volume

of the IEEE/ACM Transactions on Audio, Speech, and Language Processing (https://doi.org/10.1109/TASLP.2024.3407476).

The rest of the thesis is organized thus:

- In Chapter 2, we provide background information on the problem domain and give a more expansive coverage of extant literature.
- In Chapter 3, we describe the proposed E2E KWS model and the multilingual training scheme, along with experiments analyzing the various components, advantages and disadvantages of the method.
- In Chapter 4, we describe the JOSTER method for integrating unpaired text into E2E KWS training, and provide experimental analyses thereof.
- In Chapter 5, we summarize the findings of this thesis, draw conclusions and outline avenues for future work.

2. BAGKGROUND

The focus of this thesis is using a neural end-to-end keyword search framework. In this chapter, we provide a coverage of existing keyword search literature: both the traditional ASR-based approaches and more recent E2E approaches, and we summarize the key relationship between these techniques and our framework.

2.1. ASR-Based Keyword Search

The dominant KWS paradigm entails using an LVCSR system to transcribe the search archive, construct an inverted index—mapping terms to locations in the speech—on the transcription in the archive, search queries in the inverted index [3–8]. Since ASR inevitable produces some errors, directly indexing one-best transcriptions would make any mistranscribed terms irretrievable.

To alleviate the drop in recall rate that results from transcription errors, it is common practice to index ASR lattices instead. A natural by-product of decoding an HMM-based ASR system, a lattice is a structure [9] which encodes competing ASR hypotheses. It can be efficiently represented as a weighted finite state transducer [10] (depicted in Figure 2.1) with arcs carrying words and their likelihoods as well as connectivity and timing information.

The contents of the lattice are then used to construct an inverted index, which should map any query into the locations that contain it. If only single word queries were allowed, the solution would be to have a map of the arcs in the lattice, with the words as keys and the confidence, utterance and timing information as values. However, this solution is infeasible since queries can contain multiple words. Some prior works use approximate structures such as confusion networks [11] and position-specific posterior lattices [12] in lieu of the full lattice to construct manageable indexes.

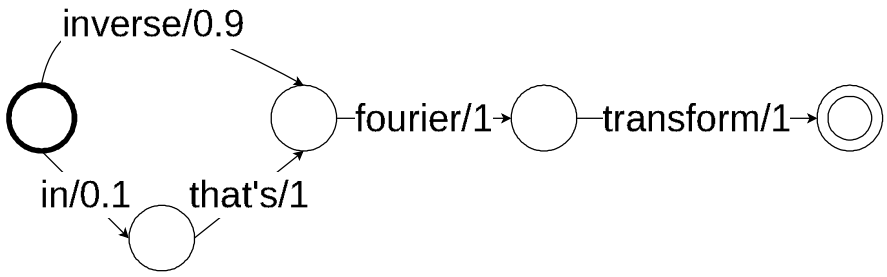

Figure 2.1. Simplified schema of an ASR lattice output.

In [13], a method of indexing the full lattice was proposed which avoids lossy approximations to the lattice. The method involves using a factor-transducer-based inverted index. The factor transducer, depicted in Figure 2.2, is an FST which accepts all factors (i.e., substrings) of a document by augmenting the original arcs in the lattice with empty-label arcs from the initial state to any internal state, and from any internal state to the final state. Interested readers are referred to [13, 14] for more information about factor-transducer-based indexing.

Note that the lattice and factor transducer in Figures 2.1 and 2.2 are only depicted as finite state automata for the sake of simplicity. In reality, they are finite state transducers, i.e., their arcs have not one but two labels: an input and output label. It is the output label which carries the location (utterance identification and timing) information.

Searching for a query in this FST inverted index entails constructing an automaton of the query (depicted in Figure 2.3) and composing [10, 15] it with the index.

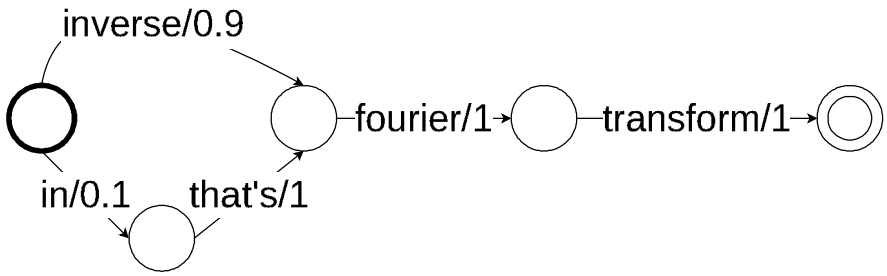

Figure 2.2. Factor transducer constructed from ASR lattice.

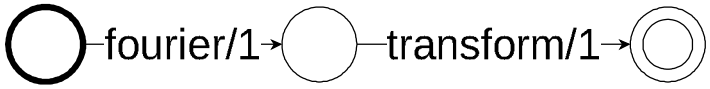

Figure 2.3. KWS query finite state transducer.

2.1.1. Open-Vocabulary ASR-Based Keyword Search

Since ASR language models are trained with a limited number of words due to computational and data availability constraints, the ASR output is limited to the closed vocabulary used in training. However, user queries can—and often do—include words that are not part of this limited vocabulary. Therefore, dealing with the challenge of seamlessly searching such OOV queries has been studied extensively within the context of ASR-based KWS systems. The solutions in literature generally fall into two categories: using subword units and query expansion.

Based on the rationale that the words in a language—even OOV ones—can be composed from a limited set of subword units, using subword-based ASR has been the cornerstone of KWS in morphologically-rich languages [16, 17], as well as open-vocabulary search in other languages [3, 5, 8, 18]. Some works use linguistic units such as syllables, morphs and phones, while others use data-driven units like graphones and multigrams [17, 19, 20]. While subword units do increase the recall of the model, this comes at the cost of larger lattices which are more costly to index and search. Consider, for instance, how the size of the transducer in Figure 2.2) would grow if a syllabic (or worse, phonetic) model were used: the arc labelled “inverse” would be split into “in” and “verse”, “fourier” into “fou”, “ri” and “er”; with each new sub-arc also accepting arcs from the initial state and extending an arc to the final state. The storage size and the cost of graph traversal (in the composition operation) clearly grows significantly. Furthermore, subword search tends to achieve have lower precision for IV queries.

Therefore, it is common to use a hybrid approach where word indexes are used for IV queries and subword indexes are reserved for OOV ones [4,6]. This has the drawback of incurring the cost of double-indexing for every new utterance. This drawback can be

partly reduced by converting word indexes into subword ones for OOV search instead of having a separate subword decoding step [4, 21], an approach limited in that it can only generate phone sequences which are substrings of some IV words.

Query expansion is an alternative approach to subword decoding (which can be viewed as index expansion) for OOV search. Based on the rationale that OOV words are likely to be mistranscribed by the ASR system into similar-sounding IV ones, query expansion involves searching for phrases that are acoustically-similar to the query in order to account for such ASR errors [22–24]. This is typically implemented by composing a query FST with an FST of phone confusions, before composing the expanded query FST with the index. While query expansion can be used with subword indices, it can be leveraged to avoid double indexing by composing the expanded-query FST with the decoder vocabulary, resulting in acoustically-similar IV "proxy" queries which can be searched in a word-based index. Figure 2.4 shows an expanded query transducer for the query in Figure 2.3 with the OOV word "fourier" replaced by similar sounding words "foyer" and "for" with probabilities 0.75 and 0.25 respectively.

Overall, while these approaches alleviate ASR-based KWS' inability to handle OOV queries, they invariably further complicate either the indexing or the search for the already complex ASR-based KWS system. The framework that we propose in this thesis, on the other hand, not only offers simpler indexing and search than ASR-based KWS, it makes no distinction between IV and OOV queries.

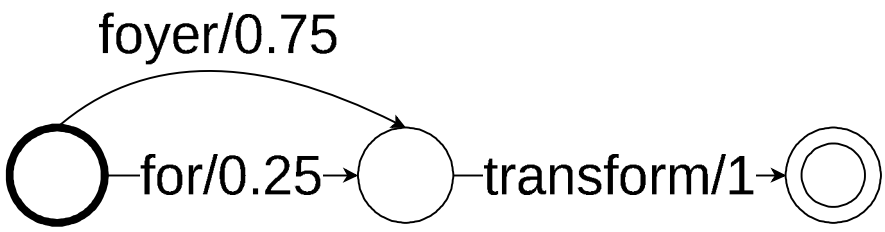

Figure 2.4. Expanded query transducer.

2.2. End-to-End Keyword Search

Leveraging the ability of neural networks to model complex relationships, KWS—which traditionally comprised several disparate, separately-optimized, modules—can now be simplified by formulating and optimizing appropriately designed neural architectures and objectives. One such simplification involves using end-to-end ASR models to construct the KWS index as in [25–28]. While these works simplify ASR training, they still have complex KWS indexing pipelines since the simpler decoding algorithms for end-to-end ASR do not readily yield the timing and confidence information necessary for KWS. Therefore, another direction, in which our work falls, involves training a model able to avoid the ASR decoding entirely while indexing or searching.

The authors of [29] propose a Siamese neural architecture which jointly learns a distance metric for speech documents represented as phone posteriorgrams along with a query representation and conduct search with subsequence dynamic time warping (DTW). The method was extended in [30] to account for query dynamics and in [31] to learn better document representations. While it showed impressive search accuracy, especially for OOV queries, this approach is limited in practice by the significant computational cost of DTW.

The authors of [32] similarly proposed a Siamese architecture which learns text and speech representations for the related task of open-vocabulary keyword spotting.[1] Since the task there does not involve localization of the keywords, there is no associated cost of DTW. In [33], a meta-network was proposed that, for a given query, generates the parameters of a model to classify whether or not a speech segment contains that query. Thus the parameters of the model grow with the number of queries, which makes the model more suitable for limited, but adaptable, query sets as opposed to the unlimited vocabulary as in KWS. Moreover, like the model featured in [32], it also lacks the ability to localize the queries, which makes both approaches unsuitable for

[1]Keyword spotting, as we use it here, involves spotting a limited set of phrases from a closed vocabulary. We note, however, that some other literature use keyword spotting to refer to keyword search of the kind that we tackle.

keyword search where the timestamps of each query's occurrences are required.

In [34], a model was proposed with a pair of encoders for computing fixed-length representations of speech utterances and text respectively, and a feedforward search network classifying whether or not the encoded text occurs in the encoded utterance. While the model was innovative in showing that the open-vocabulary search pipeline can be greatly simplified with this dual-encoder structure, it was limited to the utterance classification task where the probability of existence of a query was artificially set to 0.5 (by sampling positive and negative test utterances with equal probability at test time) and could not work in the highly imbalanced scenario of realistic KWS, where the number of negative trials far outnumber the number of positive ones. Moreover, since the speech encoder outputted a fixed-length representation of each utterance, the model could not temporally localize the keywords, although the authors did experiment with a coarse form of localization by classifying whether the query occurs in the first or second half of the utterance. This method was improved in [35] by using better pretraining objectives, although it still had the limitations of [34] with regards to handling realistic KWS settings.

The most relevant related work to ours are those in [36] and [37] who contemporaneously with us proposed dual encoder architectures capable of open-vocabulary keyword search in realistic settings, complete with the ability to temporally localize the queries. Like us, they ensure that the speech encoders do not lose temporal information, and use forced alignment to obtain the query locations at training time. We shall revisit these works in more detail in the next chapter after we describe our model so that we can point out the similarities and differences of our respective approaches.

2.3. Multilingual Data for Keyword Search

Using multilingual data to improve KWS has been explored in prior work in the context of both ASR-based and ASR-free KWS. In the context of ASR-based KWS systems, a common recipe is to improve the ASR, and consequently KWS, in low-

resource settings by training the acoustic model multilingually [38–42]. This generally entails sharing the lower layers of the acoustic model and using either a shared output layer [43–45] or separate output layers for each language [46–49].

In [31], a joint metric and representation learning method is used to incorporate multilingual bottleneck features into dynamic-time-warping-based KWS. Multilingual bottleneck features and posteriorgrams have also been used for query-by-example (where both query and audio are spoken) with [50–52] or without [53] dynamic time warping. While these works use multilingual data to train the feature extractor for ASR-free search, they do not train the search model itself multilingually.

These methods inspire our use of multilingual training to improve the data efficiency of our proposed framework in low resource settings.

2.4. Unpaired Text for Keyword Search

Due to their modular nature, classical ASR-based methods can naturally incorporate unpaired text. Various works have shown that using external text can significantly improve ASR-based KWS by using such text to augment the ASR pronunciation lexicon [54, 55] and the language model [56, 57]. These works show that the KWS improvements can be substantial even when improvements to the underlying ASR system are less pronounced, due to the effect on rare and out-of-vocabulary (OOV) queries.

Another line of work involves using unpaired text data directly in ASR training. One way of doing so is using a text-to-speech (TTS) system to generate matching speech, and including the resulting paired data as part of ASR training [58–60]. However TTS adds its own significant computational and modeling complexity. Moreover, robust TTS systems are generally a luxury only available for high resource languages. Therefore, joint speech-text models such as MMDA [61], PSDA [62], MUTE-L [63], USTED [64], Textogram [65] and MAESTRO [66] have gained interest as a way of integrating unpaired text into end-to-end ASR to improve performance on ASR, as

well as other downstream tasks such as spoken language understanding [67] and spoken machine translation [68]. These models incorporate unpaired text by treating the entire ASR model as part of a larger multimodal text generator, some of whose parameters can be jointly trained for text-to-text transduction without any explicit TTS synthesis. By improving the underlying ASR system, these methods can plausibly be used to improve ASR-based KWS systems, especially recently proposed KWS systems based on end-to-end ASR [26–28, 69]. However, they cannot work for ASR-free E2E KWS systems which are generally discriminators rather than text generators.

In Chapter 4, inspired by these joint training approaches, we introduce a method for using unpaired text for training E2E KWS methods which possess neither lexicons nor language models by developing a surrogate objective for incorporating text into the discriminative framework ASR-free KWS systems.

3. END-TO-END OPEN VOCABULARY KEYWORD SEARCH

The framework which we propose for end-to-end keyword search involves soft indexing and matching in a vector-space. Where ASR-based KWS methods index a spoken archive by decoding it into a graph of symbolic units and conduct search by matching with a corresponding graph of the query, our approach uses a dense representation in a vector space and conducts search by matching the query with dot products.

3.1. Problem Formulation

We formulate keyword search as the task of classifying whether a keyword occurs at any given location in the document. Given a query phrase $\boldsymbol{q} = \big(q_1, q_2, \ldots, q_K\big)$ where each q_k is a letter, and an utterance represented as a sequence of acoustic features $\boldsymbol{X} = \big(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\big)$, we seek the sequence of occurrence indicators $\boldsymbol{y}(\boldsymbol{q}, \boldsymbol{X}) = (y_1, \ldots, y_N) \in \{0, 1\}^N$ such that

$$y_n = \begin{cases} 1, & \text{if } \boldsymbol{q} \text{ is spoken in } \boldsymbol{X} \text{ in a time span including } n \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

We take a probabilistic view that each y_n is a Bernoulli random variable, which we parameterize with a neural network with parameters $\boldsymbol{\theta}$, so that its probability distribution conditioned on the query and utterance is $P_{\boldsymbol{\theta}}(y_n|\boldsymbol{q}, \boldsymbol{X})$.

To train the model, we assume that we have a dataset of training utterances $\mathcal{X} = \{\boldsymbol{X}^{(1)}, \boldsymbol{X}^{(2)}, \ldots, \boldsymbol{X}^{(S)}\}$ and their transcriptions. We use the phrases in the transcriptions as training query phrases $\mathcal{Q} = \{\boldsymbol{q}^{(1)}, \boldsymbol{q}^{(2)}, \ldots, \boldsymbol{q}^{(L)}\}$. Our aim is then to train the network to minimize the total negative log-likelihood of the occurrence indicators

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{l=1}^{L} \sum_{s=1}^{S} \sum_{n=1}^{N} -\log P_{\boldsymbol{\theta}}\Big(y_n|\boldsymbol{q}^{(l)}, \boldsymbol{X}^{(s)}\Big). \tag{3.2}$$

Note that although, for simplicity, we use N to denote the upper bound of the innermost sum, in reality, it changes per utterance as it is the length of the each utterance, $\boldsymbol{X}^{(s)}$.

We note that training thus requires the indicators for the training data, i.e., it requires knowledge of the beginning and end times of each training query in the utterance which contains it. To get this timing information, we train an HMM-GMM-based ASR system on the training data and use it to do word-level forced alignment.

3.2. Model Definition

Our keyword search model, depicted in Figure 3.1, comprises a recurrent document encoder and a recurrent query encoder. We conduct the search via dot products on the outputs of the encoders. We apply the logistic sigmoid function to the resulting vector of logits to obtain frame-wise posterior probabilities $P_{\boldsymbol{\theta}}(y_n|\boldsymbol{q}, \boldsymbol{X})$ which we post-process to detect the locations of each keyword. Since the document encoder is intended to act as an offline indexer, while the query encoder must be called whenever a query is received, we ensure that the query encoder is a much smaller neural network than the document encoder.

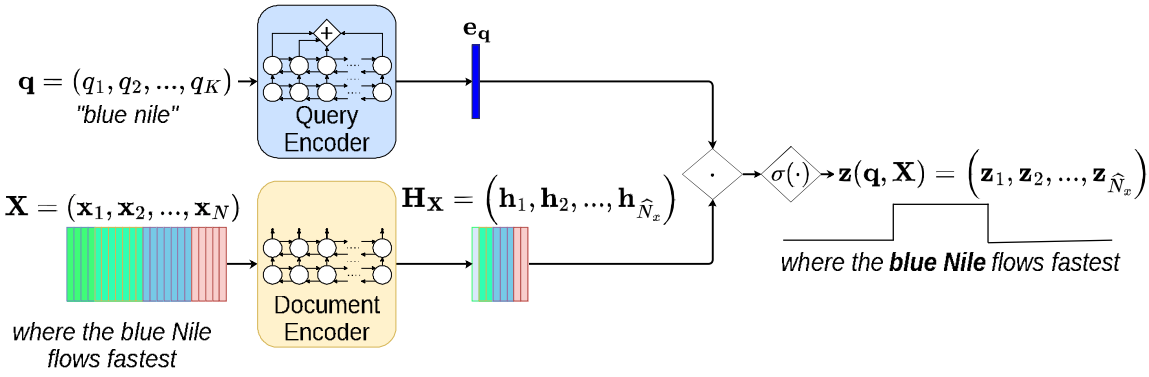

Figure 3.1. Overview of the neural keyword search model.

3.2.1. Document Encoder

The document encoder is a recurrent neural network whose input is the sequence of speech features $\boldsymbol{X}$ of length N. First, $\boldsymbol{X}$ is passed through a stack of bidirectional long short-term memory (BLSTM) layers [70] which output $\boldsymbol{U} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{\hat{N}})$ of length $\hat{N}$. We down-sample the hidden representations between some of the BLSTM layers so that, in general, $\hat{N} < N$. Down-sampling decreases the storage and computational requirements of indexing and search; additionally, we empirically found that because it reduces the duration that higher layers in the network have to process and the amount of information that have to be encoded in the final embeddings, down-sampling makes it easier to model long-range dependencies and, consequently, improves search results.

The final encoder output is then $\boldsymbol{H}_{\boldsymbol{X}} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_{\hat{N}})$, such that for each $\hat{n}$

$$\boldsymbol{h}_{\hat{n}} = \boldsymbol{W}_1 \boldsymbol{u}_{\hat{n}} + \boldsymbol{b}_2, \tag{3.3}$$

where $\boldsymbol{W}_1$ and $\boldsymbol{b}_1$ are the weight matrix and bias vector of a trainable affine transform whose dimensions can be modified to change the dimensionality of each document embedding. The affine transform ensures that the dynamic range of the document embedding can be any real number, and not limited to $(-1, 1)$ by the hyperbolic tangent nonlinearity at the BLSTM's output.

3.2.2. Query Encoder

The query encoder is a recurrent neural network whose purpose is to generate a vector representation of a query which will be similar (in a dot product sense) to the frames of the document embedding $\boldsymbol{h}_{\hat{n}}$ which contain that query and dissimilar for locations which do not. The query encoder's input is the sequence of letters $\boldsymbol{q} = (q_1, \ldots, q_K)$ that constitute the query and its output is a fixed-length (vector) representation $\boldsymbol{e}_{\boldsymbol{q}} \in \mathbb{R}^D$. Note that the query can be a single word or a sequence of words. In the case of multi-word queries, $\boldsymbol{q}$ is a concatenation of the letters in each word with the spaces between them removed. The first layer of the query encoder is

a trainable embedding lookup layer which converts the sequence of letters into a sequence of vectors that are then input into a stack of bidirectional gated recurrent unit (GRU) layers [71]. The GRU outputs another sequence of vectors $\boldsymbol{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_K)$. The final query representation is then computed from the sum of these GRU output vectors along the sequence axis

$$\boldsymbol{e_q} = \sum_{k=1}^{K} \big(\boldsymbol{W}_2 \boldsymbol{v}_k + \boldsymbol{b_2}\big), \tag{3.4}$$

where $\boldsymbol{W}_2$ and $\boldsymbol{b}_2$ constitute an affine transformation with a similar function to that at the output of the document encoder.

We choose different kinds of recurrent neural networks for the document and query encoders (LSTM and GRU respectively) based on the different operational expectations we have for them. We expect the document encoder to be used as an indexer which will be run offline and can therefore be a more computationally expensive network. The query encoder, on the other hand, would be run whenever a user searches the system, and the time it takes for it to run is part of the latency experienced by the user. It is therefore necessary to have it be as lightweight as possible, making the GRU an attractive option since it has fewer parameters and lower computational requirements than an LSTM of the same dimensions. We further reduce this computational cost in our experiments by using a GRU with fewer layers and smaller dimensions than the document encoder's LSTM.

3.2.3. Search Function

We conduct the actual search by multiplying the document encoding, $\boldsymbol{H_X}$, with the query encoding, $\boldsymbol{e_q}$. This matrix-vector product yields a temporal vector of scores on which we apply a logistic sigmoid function to get $\boldsymbol{z}(\boldsymbol{q}, \boldsymbol{X}) = (z_1, \ldots, z_{\hat{N}})$ so that each $z_{\hat{n}} \in (0, 1)$ can be interpreted as the occurrence probability $P_{\boldsymbol{\theta}}\big(y_{\hat{n}} = 1 | \boldsymbol{q}, \boldsymbol{X}\big)$, computed as

$$z_{\hat{n}} = \sigma(\boldsymbol{h}_{\hat{n}}^{\top} \boldsymbol{e_q}). \tag{3.5}$$

Since the document and query representations only interact through this product and are otherwise independent, we can pre-compute and store the encodings of the documents. Consequently, at search time, only the cost of computing the query representation (from the much smaller query encoder) and the cost of the dot product is incurred.

Using dot products as the interaction function between document and query embeddings has the advantage that it can be easily parallelized across utterances, frames and even embedding dimensions. This allows fast search with modern computing hardware (such as multi-core CPUs and GPUs) and software libraries (such as variants of BLAS [72]) which have well-optimized routines for dot product computation. This gives it an advantage other plausible but more complex interaction functions such as multilayer perceptrons or recurrent networks.

This however requires use to make a conditional independence assumption that knowing a single frame of the document encoding is enough to determine the query's existence, i.e., that

$$P_{\boldsymbol{\theta}}\big(y_{\hat{n}}|\boldsymbol{q},\boldsymbol{X}\big) = P\big(y_{\hat{n}}|\boldsymbol{e_q},\boldsymbol{h}_{\hat{n}}\big). \tag{3.6}$$

However, terms typically extend over several frames. For instance, if we set the frame length at the document encoder to be 40ms, then a half-second long term occupies 12 or 13 frames. Therefore, each frame embedding needs to encode information about the frames around it in order to be able to make the decision of whether it and its neighbors contain that query, and the longer the query the more long term information needs to be stored and recalled.

One consequence of this frame-independence assumption is that the document encoder needs to be not just recurrent, but also bidirectional. To see this, consider a document which contains the term "predict", the encodings of each frame corresponding to "pre-" need to be distinguishable from the encodings of "pre-" in, say, "prelude" or "preface", so that those would not be wrongly retrieved. Similarly, the

encodings corresponding to "-sion" in "confusion", need to be distinguishable from those in "television" and "intrusion". Therefore the document encoder needs to be bidirectional because the LSTM's forward direction is needed to disambiguate between shared suffixes, while the reverse direction is necessary to disambiguate shared prefixes. In other words, the reverse direction controls when occurrence probabilities should start spiking and the forward direction controls when they should stop spiking.

3.3. Model Training

Training the model involves optimizing (3.2). Since this objective function has no closed-form solution but is fully differentiable, we resort to gradient descent which involves initializing the parameters to to some random values $\boldsymbol{\theta} = \boldsymbol{\theta}^0$ and taking steps in the direction of steepest descent of the total negative log-likelihood. However, full gradient descent is impractical because computing the gradient involves summation over all phrases and utterances in the training set. A corpus of S utterances with W words each would have $\mathcal{O}(SW^2)$ elements in the double summation.

We therefore use mini-batch stochastic gradient descent (SGD), where we sample a subset of the total dataset whose gradient approximates the gradient on the whole dataset. At each training step, t, we compute the gradient with respect to the following objective function

$$J_t = \sum_{l=1}^{L_b} \sum_{m=1}^{M} f\Big(\boldsymbol{z}\big(\boldsymbol{q}_{t,l}, \boldsymbol{X}_{t,l,m}\big), \boldsymbol{y}\big(\boldsymbol{q}_{t,l}, \boldsymbol{X}_{t,l,m}\big)\Big), \tag{3.7}$$

where $\{\boldsymbol{q}_{t,1} \dots \boldsymbol{q}_{t,L_b}\}$ is a mini-batch of L_b queries sampled randomly from the set of word unigrams, bigrams and trigrams of the dataset with $L_b \ll L$; $\{\boldsymbol{X}_{t,l,1}, \dots, \boldsymbol{X}_{t,l,M}\}$ is a set of documents sampled from the dataset such that $\boldsymbol{X}_{t,l,1}$ contains $\boldsymbol{q}_{t,l}$ while the other $M-1$ documents are sampled randomly; and $f(\cdot)$ is a modified binary cross-entropy function defined as

$$f(\boldsymbol{z}, \boldsymbol{y}) = -\sum_{n=1}^{\hat{N}} \Big(\mathbb{1}_{z_n > 1-\phi} \cdot (1 - y_n) \log(1 - z_n) + \mathbb{1}_{z_n < \phi} \cdot \lambda \cdot y_n \log z_n \Big), \tag{3.8}$$

where the labels have been down-sampled to match the output frame rate of the document encoder. This objective function extends the binary cross-entropy objective with the hyper-parameters λ and ϕ. When both are set to 1, the loss reduces to frame-wise binary cross-entropy. λ controls the relative importance of positive frames (frames labeled 1) and negative frames (frames labeled 0), i.e., the relative cost of misses to false detections; as λ increases, positive frames labeled contribute more to the total loss. ϕ is a strictness term controlling the sensitivity of the loss function to easily classified frames; frames labeled 1 with sigmoid outputs above ϕ and frames labeled 0 with sigmoid outputs below $1-\phi$ do not contribute to the loss. Having $\phi < 1$ prevents the model from learning to better classify frames that are already well classified at the expense of learning to classify difficult frames.

The training queries are sampled without replacement in a single epoch, i.e., a single cycle through the entire list of training queries. In this cycle, multiple occurrences of the same query are treated as different entries, so that, ultimately, each query is sampled proportionally to how frequently it occurs in the training data.

As stated above, when sampling training documents for each query, we sample only semi-randomly. For each query, we always include the document that contains it as a positive document and randomly sample the other, negative, documents.[2] Although this yields a biased estimate of the gradient, it is necessary from an optimization perspective. Since for any specific query, the overwhelming majority of documents do not contain it, fully random sampling virtually always yields mini-batches with negative query-utterance pairings and, consequently, causes the model to converge to a degenerate solution where it always outputs zero.

Finally, at training step t, the model update is

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^{t} - g(\nabla_{\boldsymbol{\theta}^t} J_t), \tag{3.9}$$

[2]Note that although we refer to the randomly sampled documents as "negative", there is a chance, albeit a small one, that a negative utterance happens to contain the query. When this happens, it is trained as if it were also a positive utterance, i.e., the scores of the frames containing the query are increased while the scores of other frames are decreased.

where $\nabla_{\boldsymbol{\theta}^t} J_t$ is the gradient of the objective function with respect to the current parameters of the model, and $g(\cdot)$ is a function whose specific form depends on the optimizer used. For plain SGD, $g(\cdot)$ would be the identity function, but we use the Adam optimizer [73] in our experiments, so $g(\cdot)$ also depends on moving averages of the first- and second-order trajectories of the gradient. The update procedure is repeated until the objective function stops improving on some held-out portion of the training queries.

3.4. Post-Processing for Keyword Search

After the model is trained, we can search by computing the vector of probabilities $\boldsymbol{z}(\boldsymbol{q}, \boldsymbol{X})$, for a query $\boldsymbol{q}$ and each document $\boldsymbol{X}$ from the archive. Having obtained the vector of probabilities, we still need to post-process them to obtain the timestamps in the document hypothesized to contain the query, and the corresponding confidence scores. The procedure, which is illustrated in Figure 3.2, is as follows:

(i) We zero-out the probabilities (z_n) below some threshold α. This thresholding is a necessary first step because it is otherwise impossible to determine discrete values, $\{0, 1\}$, of y_n since sigmoid outputs are strictly non-zero. α is a hyper-parameter which we set to $\alpha = 0.5$, although we found search performance to be stable for settings of α between 0.4 and 0.7.

(ii) We pick the resulting "islands" of non-zero elements as our system hypotheses. We return the locations of the first and last frame of each island as the timestamps of query *hits* with the median probabilities of the islands serving as the confidence.

(iii) Finally, we prune out spurious hits by discarding hypotheses which are shorter than 40ms×number of letters in the query.

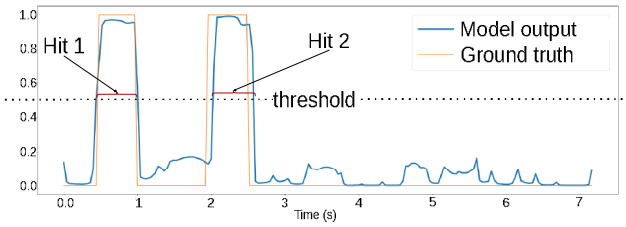

Figure 3.2. Post-processing of model output into KWS hypotheses.

3.5. Multilingual Training

Modern neural networks require large and ever-increasing quantities of data to train. This makes them difficult to apply in low resource settings, such as for languages with scare annotated training resources. As most of the languages spoken in the world fit in this low-resource categorization, it is necessary to explore training strategies which can operate in the low-resource setting. One such strategy is transfer learning, which involves increasing the effective training data by pretraining the neural network on some related data and then finetuning it on the target data.

A popular transfer learning method for speech applications is multilingual pretraining. The crux of multilingual pretraining is the observation that although different languages define different phonemic categories, the sounds that make up those phonemes are often similar or shared across several languages. Therefore, it stands to reason that a neural network trained with large quantities of data from other languages could learn robust representations of human sounds which could then be finetuned to the target language. We hypothesize that although the queries are (language-specific) graphemes, the space to which they are transformed by the query encoder is an acoustic one, i.e., that the search is transformed into the problem of searching for pseudo-speech in actual speech. We further contend that beyond the grapheme representation, the rest of the framework can be shared across various languages and propose to do multilingual training as a way of making our model usable in low-resource settings.

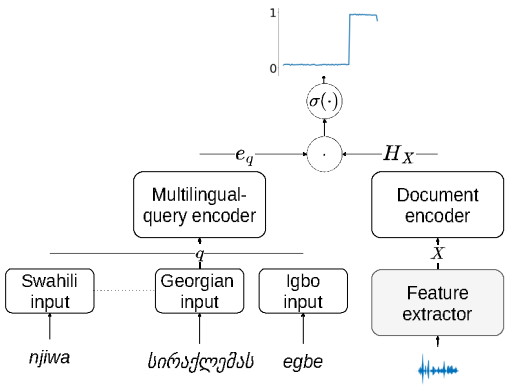

Figure 3.3. Multilingual pretraining of E2E KWS system.

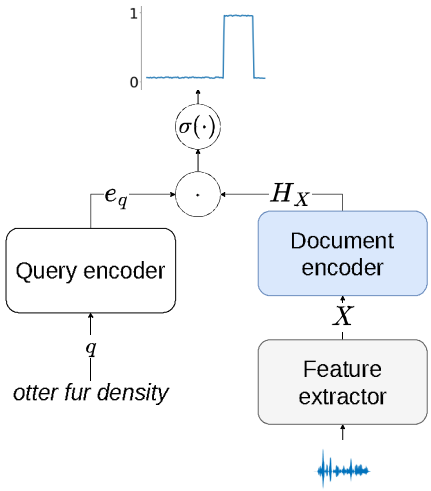

Figure 3.4. Finetuning of E2E KWS after multilingual pretraining.

Our multilingual scheme, depicted in Figure 3.3 and 3.4, involves pretraining the model with a pool of annotated data from several languages, and then finetuning it on the target language. During multilingual training, the entire model is shared by all languages, with the exception of the query encoder's input embedding layer (since the graphemes are language-specific). We train the model with the same objective as in the monolingual setting (as described in Section 3.3). When sampling negative utterances ($\{\boldsymbol{X}_{t,l,2}, \ldots, \boldsymbol{X}_{t,l,M}\}$) for any given training query, $\boldsymbol{q}_{t,l}$, we have the choice of sampling utterances randomly across languages or sampling from within the same language as the query. Empirically, we found no significant difference between either choice so we stick sample negatives across language boundaries for simplicity.

When finetuning to a new language, we transfer only the pretrained document encoder and reinitialize the entire query encoder with random weights, then we train the entire model on the target language's training data. Empirically, we found that transferring the multilingually-pretrained query encoder as well (with the input embedding layer randomly initialized) results in worse KWS performance.

3.6. Discussion of Related Work

As we alluded in Section 2.2, the most relevant related work to ours are the contemporaneous works of [36] and [37] which also use dual-encoder architectures capable of open-vocabulary keyword search in realistic settings, complete with the ability to temporally localize the queries. Like us, they ensure that the speech encoders do not lose temporal information, and use forced alignment to obtain the query locations at training time.

In [36], the authors extend their prior work on closed-set keyword detection and localization [74] to cover open vocabularies. A feedforward network takes the speech and text encodings from a pair of encoders and returns a vector to be compared with the text encoding. Their model has three outputs for each speech segment, a score in $(0, 1)$ indicating the likelihood that the query was spoken and a pair of real numbers denoting the timestamps of the hypothesized hit in the segment. By directly predicting the temporal locations, they avoid the need to have any post processing step. This however comes at the cost of having to run the feedforward network for every query-utterance pair at search time, as opposed to the simpler matrix-vector product that we use for the query-utterance interaction. Therefore, the search can become orders of magnitude slower since each feedforward layer with output dimension of F is a matrix-matrix product which has F times the cost of matrix-vector product.

In [37], the authors propose a very similar structure to our model with the difference that, instead of acoustic features, they use phonetic confusion networks output from an external ASR system as the speech representation. This allows language model information to be indirectly incorporated into the KWS model. However, it also means that creating the document representation requires full ASR decoding. Note that in some of our experiments, we will use acoustic features extracted from neural networks including features from a pretrained ASR model, extracting these features only requires passing data through the neural network—comparable to acoustic modelling—and not the costs and complexities associated with traversing the ASR decoding graph.

3.7. Experiments and Results

In this section, we conduct experiments to analyze various components of the proposed model. First, we describe experiment setup. Then we analyze how various training parameters affect the model performance. Finally, we analyze how the performance of the model changes with different query properties and compare and contrast to how those same properties affect a conventional ASR-based model.

3.7.1. Setup

In this section introduce the setup for our experiments: the dataset on which we experiment, the acoustic feature representations used in our experiments and the metrics which we use to quantify the outcomes of those experiments. Additionally, we provide details on the model and training hyper-parameters for the proposed model.

3.7.1.1. Dataset. We conduct our experiments on the IARPA Babel corpus of conversational telephone speech [75]. The Babel dataset contains recordings of telephone conversations across 26 languages. Of these languages, we select 5 target languages—Assamese, Bengali, Pashto, Turkish and Zulu—on which we evaluate the performance of various models. These languages cover a large geographic span from northern India to South Africa and feature various idiosyncrasies including: the syllabic Assamese and Bengali, the exclusion of vowels from Pashto orthography, the agglutinative morphologies of Turkish and Zulu, and the clicks and tones that feature extensively in Zulu. The wide linguistic coverage of our experiments make it unlikely that any inferences we draw will be hyper-specific to some language or language clusters.

For each language, we use the limited language pack (LLP) training set as the paired training data for training models. The LLP contains 10 hours of training data for each language. Although our models benefit from having larger training sets, this setting mirrors the reality for most languages of the world for which training data is scarce.

Table 3.1. Query distribution per language.

LANGUAGE	ARCHIVE	QUERIES	OOV RATE
Assamese	Dev	1959	30%
	Eval	2709	13%
Bengali	Dev	1966	35%
	Eval	2597	15%
Pashto	Dev	1757	25%
	Eval	2370	14%
Turkish	Dev	283	28%
	Eval	1625	28%
Zulu	Dev	1986	40%
	Eval	1408	27%

Furthermore, the Babel dataset is curated not just for ASR but also KWS with standard query sets for each language and test archives with transcriptions aligned, a standardization which facilitates reproducible KWS experiments.

In addition to the training data, each language has a pair of spoken archives from which we search for various queries. The development (Dev) set contains 10 hours of speech. It is on this set that we tune hyperparameters of our models such as the normalization, fusion parameters and the score thresholds. The evaluation (Eval) set is a five-hour[3] archive on which we solely test our models which we have trained on the training dataset and whose hyperparameters have tuned on the development dataset. Each archive is provided with a list of test queries, some of which are OOV, i.e., they contain words which are not encountered in the training data. Table 3.1 shows the number of queries per language as well as the proportion of queries which are OOV.

For experiments involving multilingual pretraining, we use the LLP data from 17 other Babel languages—totaling around 170 hours—to pretrain the KWS model.

[3]The full Eval set contains around 15 hours of speech per language. However reference annotations required for performance evaluations have been released publicly for only five-hour subsets.

3.7.1.2. Acoustic Features. We consider three types of acoustic features—one spectral and two neural-network-based—as the input document representations:

(i) 13-dimensional Mel-frequency cepstral coefficients (MFCC) with 10ms frames.

(ii) 42-dimensional multilingual bottleneck features (BNF) with frame length of 10ms. Our BNF extractor is a multilingual acoustic model with a time delay neural network [76] architecture which we trained in block-softmax fashion [48] to classify clustered context-dependent triphone states on 19 Babel languages' LLP data.

(iii) 1024-dimensional XLS-R features with frame length of 20ms. The features are extracted from the transformer architecture of [77] which was pretrained on data from 128 languages with the unsupervised Wav2vec2 objective [78]. We extract features from the 15th layer of the 300-million-parameter XLS-R model using the code published with the paper[4] .

3.7.1.3. Metrics. We report results in terms of the variants of the term weighted value (TWV) [79] which are widely used for evaluating KWS performance [79–81]. For each query, a KWS system returns a list of hits, i.e., a list of locations (utterance identifier and timestamps) within the archive purported to contain query, each with an associated confidence score. For any nontrivial test setting, some of these hypotheses are bound to be false, so the system also provides a threshold below which the scores are considered to be irrelevant. In this setting, there are two kinds of errors:

- Misses occur when the system returns no hits for a location which contains the query, or if it does return a hit but with a score lower than the threshold. Note that it is extremely unlikely to get an exact match in timing between the system hits and the reference. Even human annotators might disagree about the exact word boundaries at a resolution smaller than a few tens of milliseconds. Therefore, we follow a convention of accepting a hit if its midpoint is within a 500ms neighborhood of each other of an actual reference location of the queried term.

[4]https://github.com/facebookresearch/fairseq/tree/main/examples/wav2vec/xlsr

- False alarms (FA) occur when the system returns a hit with a score above the threshold at a location which does not actually contain the queried term.

TWV provides a measure of recall and precision at a global, query-independent, threshold. For a set of queries $\mathcal{Q}$ and threshold ξ, the TWV is defined as

$$TWV(\xi, \mathcal{Q}) = 100 \times \Big(1 - \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} (P_{miss}(q, \xi) + \beta P_{FA}(q, \xi))\Big), \tag{3.10}$$

where

$$\beta = \frac{C}{V}\big(\frac{1}{Pr} - 1\big). \tag{3.11}$$

C is the cost of a false alarm, V is the value of a correct detection and Pr is the prior probability assigned to a query in a single trial. Following the NIST STD evaluations [80, 82], we set $\frac{C}{V}$ to 0.1 and Pr to 10^{-4}, yielding $\beta = 999.9$. For each query q, the quantities $P_{miss}(q, \xi)$ and $P_{FA}(q, \xi)$, which respectively denote the probabilities of misses and false alarms at threshold ξ, are defined as

$$\begin{aligned} P_{miss}(q, \xi) &= 1 - \frac{N_{correct}(q, \xi)}{N_{true}(q)} \\ P_{FA}(q, \xi) &= \frac{N_{FA}(q, \xi)}{n_{TPS} \times T - N_{true}(q)}, \end{aligned} \tag{3.12}$$

where

- $N_{correct}(q, \xi)$ is the number of correct hits of q with a confidence score above the threshold ξ.
- $N_{true}(q)$ is the actual number of occurrences of q in the archive.
- $N_{FA}(q, \xi)$ is the number of false detections of q with confidence score above ξ.
- T is the length of the archive in seconds.
- n_{TPS} is the number of trials per second. Following [82], we set $n_{TPS} = 1$.

Overall, we observe that a higher TWV value corresponds to a better system; a perfect system which returns hits in all the correct locations and no false alarms would have a TWV of 100, while a system with no outputs whatsoever (no false alarms and no misses) would have a TWV of 0. Finally, a system with a preponderance of false

alarms could have negative TWV up to -100β.

Since different queries tend to have different score distributions and term weighted value requires setting a single global threshold, it is necessary to normalize scores per query. For this, we use the keyword specific thresholding normalization method [83].

We report the following variations of the TWV to measure different aspects of KWS system performance:

(i) The actual term weighted value (ATWV) is the measure of TWV at the selected threshold. We report ATWV on Eval sets after tuning the threshold on the corresponding Dev set.
(ii) The maximum term weighted value (MTWV) is the value of the TWV at the threshold that maximizes it. Generally, we report MTWV on the Dev sets.
(iii) The optimum term weighted value (OTWV) is the MTWV with a term specific optimal threshold for each query. 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 measures the overall recall rate of the system.

3.7.1.4. Model Configuration. We use a pair of recurrent encoders in our E2E KWS model: an LSTM-based document encoder and a GRU-based query encoder.

The query encoder has a 32-dimensional input embedding layer for converting the (discrete) text query into a sequence of vectors. The embedding layer is followed by two bidirectional GRU layers with 256 units each and a final affine projection layer with 400-dimensional output.

The document encoder has six BLSTM layers with 512 output units each, followed by an affine projection layer with output dimension of 400. We apply Dropout regularization [84] with probability 0.4 between successive BLSTM layers. We down-

sample the output of some of the BLSTM layers so that the each output frame spans a duration of 40ms; when using MFCC or BNF which have 10ms frames at the input, we down-sample after the first and fourth BLSTM layers each time by a factor of two; for XLS-R which has 20ms input frames, we down-sample by a factor of two after the fourth layer.

3.7.1.5. Training Hyper-Parameters. By default, we set the parameters of the training objective (Equations 3.7 and 3.8) to be $\phi = 0.7$, $\lambda = 5$ and $M = 4$. We will analyze the impact of each of these parameters in turn in subsequent sections.

When training, we randomly sample and set aside 10% of the training query n-grams for validation. We use the Adam optimizer [73] for stochastic gradient optimization. We use a mini-batch size (L_b) of 32 for monolingual training and finetuning; for multilingual pretraining, we use a batch size of 256. We start optimization with a learning rate of 2×10^{-4} and halve the learning rate whenever the validation loss stagnates for four epochs (with an epoch defined as a single loop over the training queries). We stop training whenever the validation loss stagnates for 10 epochs.

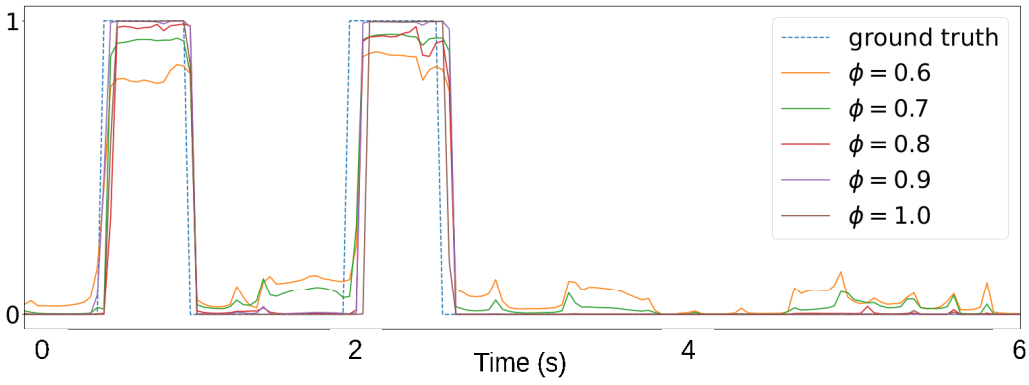

Figure 3.5. Outputs from the KWS model for an example query-utterance pair from the Turkish development set under various settings of the training objective tolerance (ϕ).

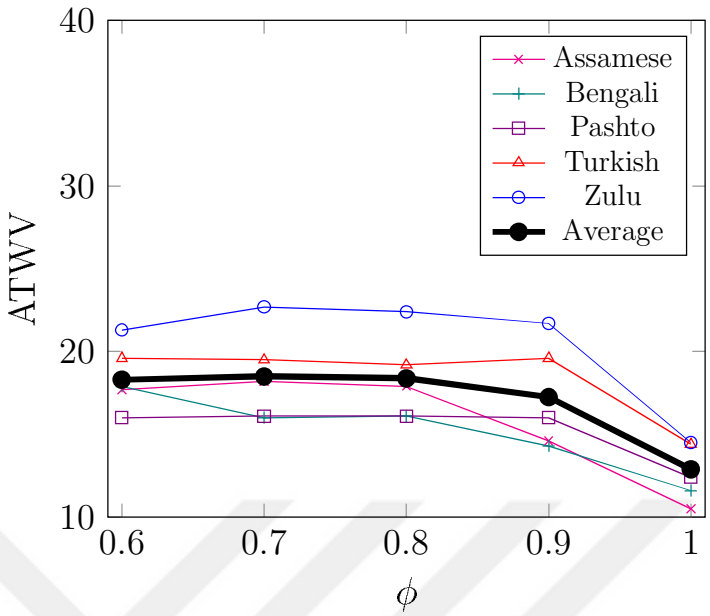

Figure 3.6. Variation of ATWV with ϕ on the evaluation sets.

3.7.2. Effect of Loss Function Parameters

In this section, we analyze the impact of the hyper-parameters of the loss function, ϕ and λ from Equation 3.8. Recall that λ is the weight given to positive training frames (frames which contain the training phrase), while ϕ controls the allowable margin beyond which no loss is incurred. When both values are set to 1, the objective becomes the classic binary cross-entropy objective. All models here use bottleneck features as input without any pretraining or speed perturbation. First we vary the tolerance (ϕ) of the loss function with λ fixed to 5. Figure 3.5 depicts the output, $\boldsymbol{z}(\boldsymbol{q}, \boldsymbol{X})$, of models trained with various values of ϕ on an example from the Turkish dev set. All settings of ϕ track the correct shape, outputting high values where the ground truth, $\boldsymbol{y}(\boldsymbol{q}, \boldsymbol{X})$, is 1 and low values where the ground truth is 0. However, the degrees with which they do so vary, with increasing ϕ unsurprisingly resulting in more extreme separation of positives from negatives.

Figure 3.6 shows the ATWV as ϕ changes. We observe that ATWV shows little variation with the choice of ϕ except for $\phi = 1$ where we observe significant ATWV degradation, implying that the exact value of ϕ is not crucial so long as it is not 1.

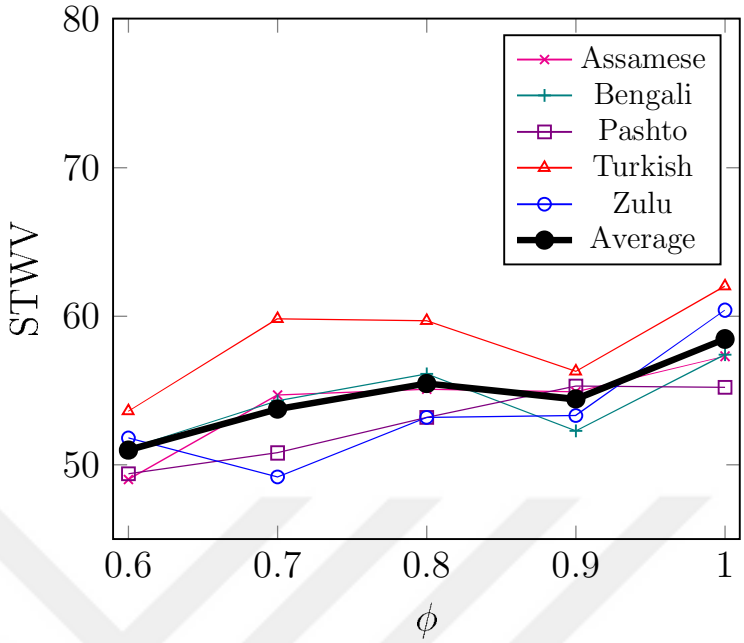

Figure 3.7. Variation of STWV with ϕ on the evaluation sets.

Figure 3.7 shows the variation of STWV (which solely measures recall) with ϕ. We find that setting higher values of ϕ generally increases the STWV and may be preferred for applications requiring higher recall at the expense of precision.

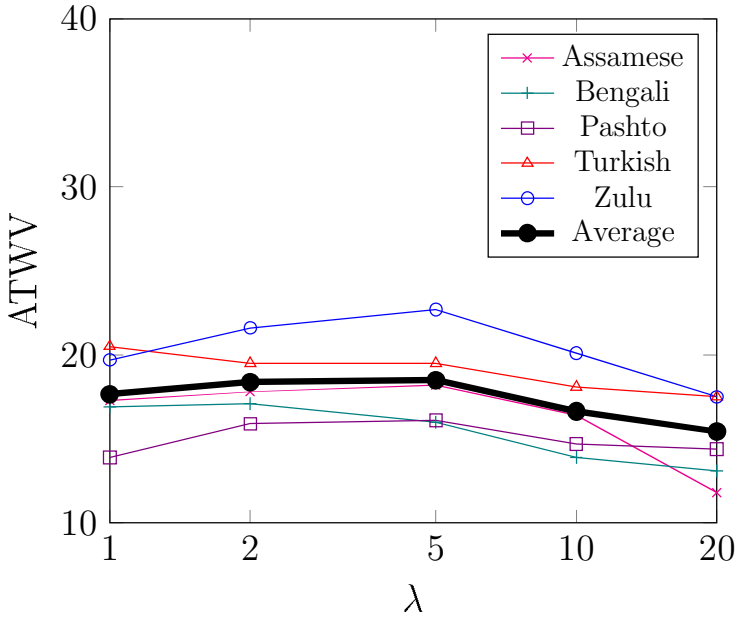

Figure 3.8. ATWV on the evaluation sets as the weight of positive frames λ in the training objective is varied.

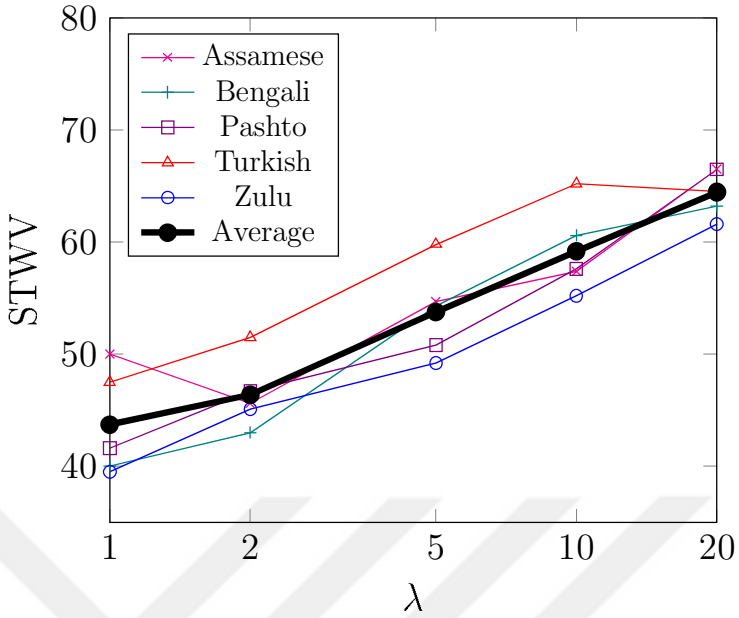

Figure 3.9. STWV on the evaluation sets as the weight of positive frames λ in the training objective is varied.

Figures 3.8 and 3.9 show the TWV as λ is varied with ϕ fixed to 0.7. The ATWV increases with λ until around 5 where it peaks, and then falls off as λ is further increased. This indicates that the decrease in precision that accompanies increasing λ starts to hurt the overall ATWV. Thus, the correct setting of λ seems tied to the relative costs of false alarms to misses and must be set based on the task at hand.

However, we note that the recall, as measured by STWV increases monotonically with λ. This behavior is expected as increasing λ makes the training objective prioritize detecting positive locations over suppressing negative ones. Thus, a higher setting of λ is more appropriate when recall is paramount e.g. when another, higher-precision, model will be used to rescore the hits.

3.7.3. Effect of the Ratio of Negative to Positive Training Utterances

For each training phrase, we sample M utterances, one of which is the utterance from which the current training phrase is taken. The other $M-1$ "negative" utterances are sampled randomly. In the experiments so far, we have set $M = 4$.

Figure 3.10 and 3.11 show the result of varying M in BNF-based models. Increasing M has two obvious effects on the optimization; it reduces the approximation error due to the sampling process and it inadvertently up-weights the contribution of negative samples to the loss function (effectively, it reduces λ).

We argue that the ATWV improvements we observe in Figure 3.10 are a result of the first effect, because, as we have already seen in Section 3.7.2 (and Figure 3.8), decreasing λ does not improve the ATWV; furthermore, doubling (or even quadrupling) it does not degrade the term weighted value to the extent that reducing the number of negatives by setting $M = 2$ does.

However, the second effect has an impact on STWV, which, as Figure 3.11 shows, decreases strictly as M increases. This is due to a "broken-clock" effect, where models trained with lower M (similar to models trained with smaller λ) have higher recall simply by virtue of returning far more hits, whether spurious or correct, as a consequence of seeing a lower number and diversity of negative training utterances.

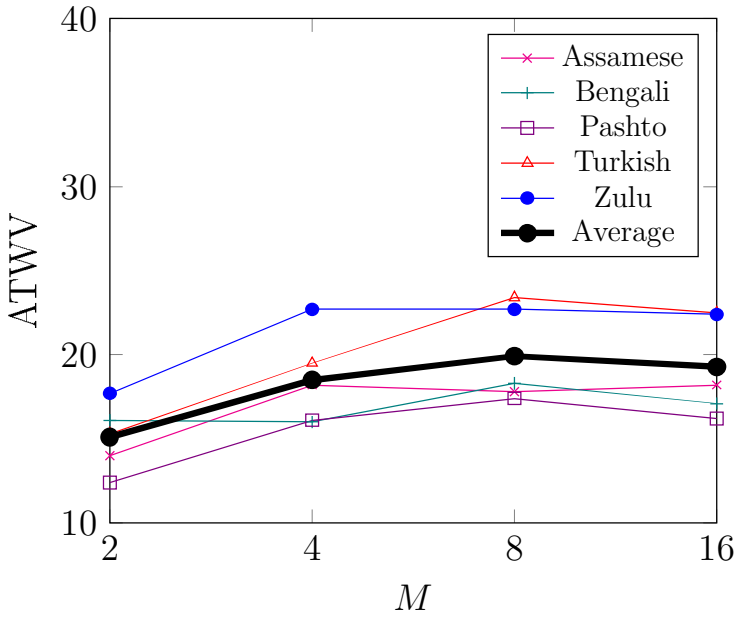

Figure 3.10. ATWV on the evaluation sets as the ratio of negative training utterances is varied.

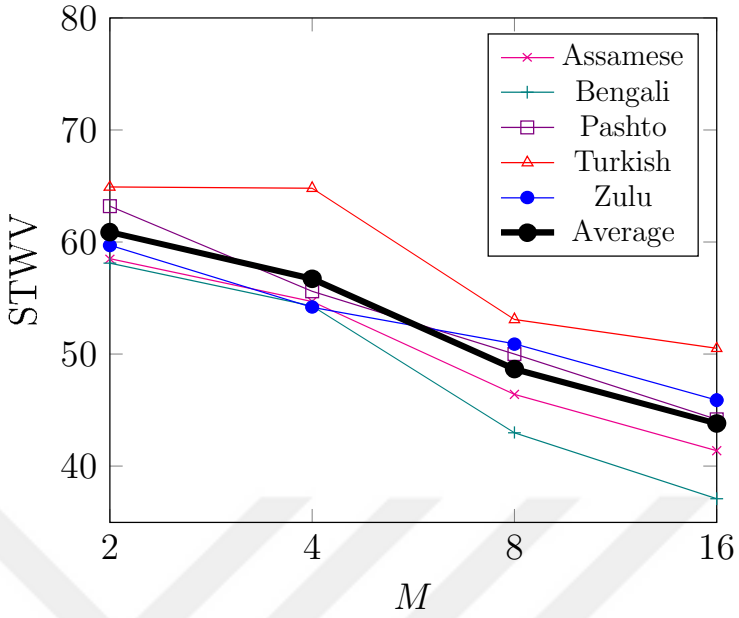

Figure 3.11. STWV on the evaluation sets as the ratio of negative training utterances is varied.

Finally, we see that with $M = 8$, we get a good enough approximation and that increasing M further does not lead to significant improvements in ATWV–even slightly degrading the performance for Bengali and Pashto.

3.7.4. Effect of Down-Sampling

Our document encoder features a pair of down-sampling steps after the first and fourth BLSTM layers. Thus, the output document encodings are down-sampled by a factor of 4, resulting in smaller document storage and computational requirements. In this section, we analyze the effect of the total amount of down-sampling on keyword search performance.

Figure 3.12 shows the ATWV as the total down-sampling factor varies between 1, 2, 4, 8 and 16. We note that the ATWV improves as we introduce down-sampling, decreases slightly as the down-sampling factor is increased from 2 to 4, and starts to degrade upon further down-sampling. With a down-sampling factor of 16, the ATWV becomes almost equal to not having any down-sampling at all.

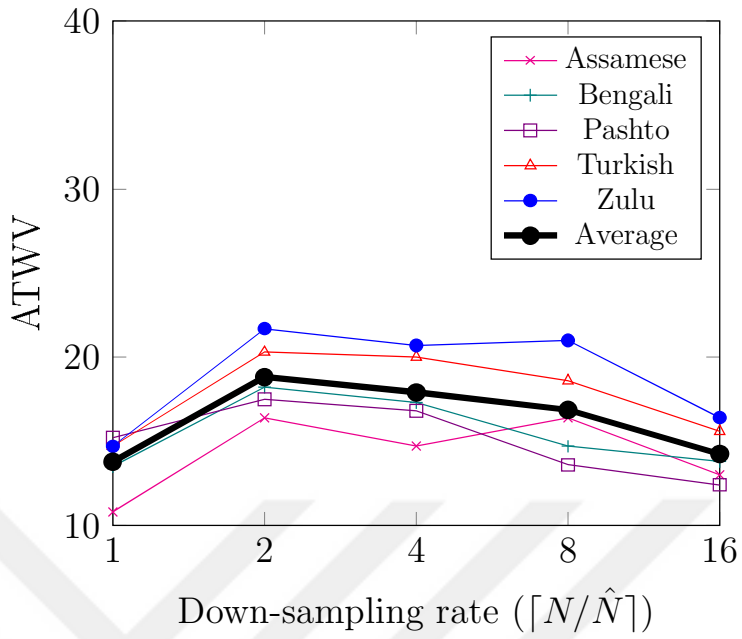

Figure 3.12. ATWV on the evaluation sets as the down-sampling rate is varied.

Overall, we infer that having down-sampling—within some bounds—does not just maintain accuracy but actually improves it, while also being faster. We ascribe this improvement to the difficulty of learning very long range dependencies within the model encodings. For instance, to correctly localize a query of length 500ms at a frame rate of 10ms, each BLSTM hidden state would have to contain information spanning at least 50 frames; after down-sampling by a factor of 2, this burden reduces to 25 frames etc. Beyond the optimal down-sampling rate, the search fidelity starts to degrade, ostensibly due to the excessive loss in resolution.

These arguments are supported by the results in Figure 3.13 which shows the ATWV improvement or degradation for queries of different length as the down-sampling rate is varied. The systems with down-sampling generally perform better as the query length increases supporting the hypothesis that without down-sampling, the system struggles to model long-term dependencies. On the other hand, as the rate of down-sampling is increased, the model struggles with detecting shorter queries (see for instance the performance with down-sampling rate of 16), supporting the idea that loss of resolution eventually limits feasible amount of down-sampling.

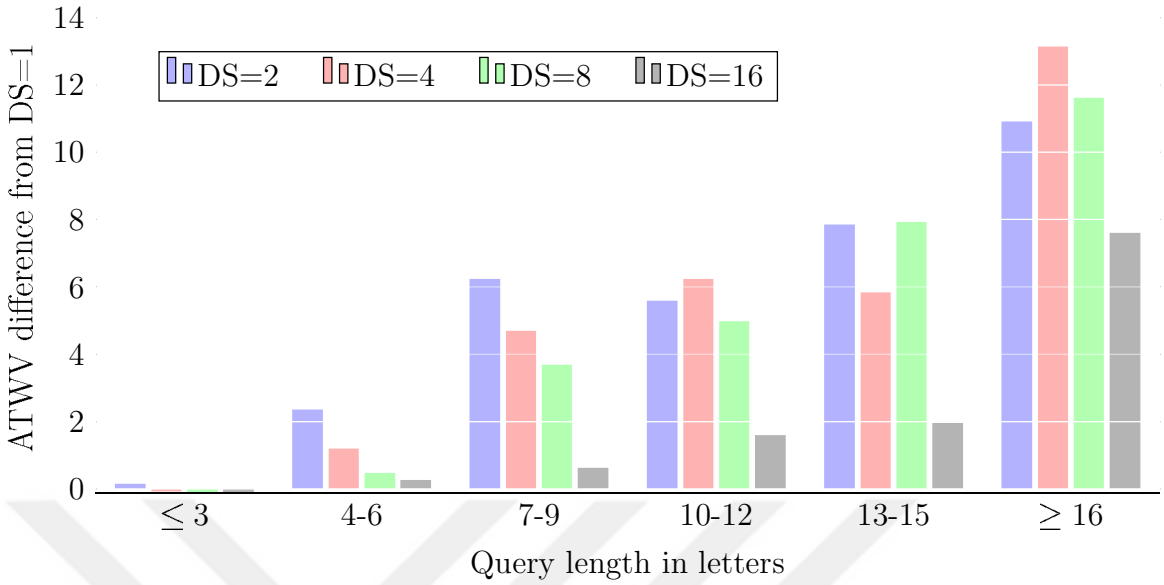

Figure 3.13. Average difference in ATWV of systems with various down-sampling factors when compared to the system with no down-sampling.

3.7.5. Impact of Multilingual Pretraining

In this section, we experiment with methods to increase the effective training data by using speed perturbation [85] and multilingual pretraining. As is common practice in speech recognition, for speed perturbation, we create two extra copies of the training data by perturbing the speaking rates by factors of 0.9 and 1.1 respectively.

First, Table 3.2 shows the effect of speed perturbation on KWS performance. In the "System" column of the table, "Raw" denotes models trained without speed perturbation or multilingual pretraining, "+SP" denotes models trained with the speed perturbation on the target language, and "+Pretrain" indicates multilingual pretrained models which are finetuned with speed perturbation on the target language.

We observe that speed perturbation leads to significant improvements regardless of input feature, with much higher relative improvements on MFCC. Therefore, in subsequent experiments, we use speed perturbation by default.

Table 3.2. TWV of E2E KWS on the Babel Dev and Eval sets with various features and augmentations.

		Dev MTWV			Eval ATWV		
Language	System	MFCC	BNF	XLS-R	MFCC	BNF	XLS-R
Assamese	Raw	6.1	16.3	26.4	6.6	18.2	25.1
	+SP	6.8	18.8	31.2	8.9	21.4	31.3
	+Pretrain	**13.2**	**23.2**	**33.8**	**14.3**	**23.4**	**33.8**
Bengali	Raw	5.2	16.4	29.8	5.8	16.0	27.4
	+SP	8.9	20.5	32.3	8.8	21.1	30.8
	+Pretrain	**13.9**	**23.5**	**37.4**	**13.0**	**22.4**	**33.5**
Pashto	Raw	4.9	11.7	22.4	6.9	15.2	26.3
	+SP	7.9	14.5	25.9	10.6	18.1	28.8
	+Pretrain	**9.4**	**16.3**	**27.8**	**11.4**	**20.3**	**32.2**
Turkish	Raw	18.1	29.1	41.2	7.9	17.8	34.8
	+SP	24.4	31.0	44.2	13.5	23.7	36.8
	+Pretrain	**31.6**	**37.6**	**47.2**	**18.2**	**24.7**	**40.3**
Zulu	Raw	4.7	16.4	31.3	6.6	17.9	28.9
	+SP	9.9	23.5	35.8	12.3	25.3	34.1
	+Pretrain	**15.9**	**26.4**	**38.0**	**17.7**	**26.8**	**36.0**
Average	Raw	7.8	18.0	30.2	6.8	17.0	28.5
	+SP	11.6	21.7	33.9	10.8	21.9	32.4
	+Pretrain	**16.8**	**25.4**	**36.9**	**14.9**	**23.5**	**35.1**

Table 3.2 also shows the effect of multilingual pretraining on KWS performance. Here, in order to limit computational costs, we only use speed perturbation when finetuning, and not when pretraining. We observe significant improvements on the baseline, on top of the improvements from speed perturbation, regardless of input feature type. This is despite the fact that both the BNF and XLS-R already contain multilingual information.

3.7.6. Performance of Queries with Various Properties

In this section, we analyze the performance of the proposed model on queries of various properties. Specifically, we analyze how the performance of the model changes depending on whether the query in question is in-vocabulary or out-of-vocabulary, as well as how the performance changes with length of the query. We compare the result to how the same factors affect conventional ASR-based systems.

Table 3.3. Evaluation set IV and OOV term weighted values for the proposed system, a hybrid ASR-based KWS system and the fusion of both systems.

		ATWV			OTWV		
Language	System	IV	OOV	All	IV	OOV	All
Assamese	ASR-based	34.6	16.2	32.2	50.1	29.2	47.4
	Proposed	23.6	19.0	23.0	40.9	33.5	39.9
	ASR-based + Proposed	**40.7**	**26.1**	**38.7**	**55.8**	**44.5**	**54.3**
Bengali	ASR-based	35.8	27.8	34.6	50.8	39.0	49.0
	Proposed	26.3	22.0	25.6	41.1	37.4	40.5
	ASR-based + Proposed	**41.1**	**33.8**	**40.0**	**56.4**	**50.5**	**55.5**
Pashto	ASR-based	35.0	12.7	31.9	51.5	24.4	47.8
	Proposed	22.0	13.5	20.9	39.3	28.4	37.8
	ASR-based + Proposed	**38.0**	**18.9**	**36.2**	**55.1**	**36.2**	**52.5**
Turkish	ASR-based	46.2	27.9	40.5	62.5	41.9	56.7
	Proposed	29.0	26.7	28.4	46.1	45.4	45.9
	ASR-based + Proposed	**50.3**	**35.0**	**46.6**	**65.9**	**54.0**	**62.6**
Zulu	ASR-based	37.6	23.5	33.8	50.8	36.5	47.0
	Proposed	27.9	25.3	27.2	41.3	39.7	40.9
	ASR-based + Proposed	**41.3**	**33.2**	**39.7**	**56.3**	**48.8**	**54.3**
Average	ASR-based	37.8	21.6	34.6	53.1	34.2	49.6
	Proposed	25.8	21.3	25.0	41.7	36.9	41.0
	ASR-based + Proposed	**42.3**	**29.4**	**40.2**	**57.9**	**46.8**	**55.8**

For this comparison, we use BNF as the E2E KWS model input with speed perturbation and multilingual pretraining, and we increase the number of training negative utterances by setting $M = 8$. For the ASR-based system, we build a TDNN-based [76] hybrid ASR model.

In order to have a fair comparison, the TDNN acoustic model is pretrained on the same data we use for multilingual pretraining of our model, and finetuned on each target language with speed perturbation applied for both pretraining and finetuning. We use a word-subword hybrid index where IV queries are searched in a word-based index while OOV queries are searched in a syllabic one. Both the word and the subword lattices are obtained using respective word and syllabic trigram Kneser-Ney-smoothed [86] language models which we found to perform better than higher order language models in this low-resource setting. We use the official lexicon provided in the Babel corpus to get IV word pronunciations and to train a Sequitur grapheme-to-phoneme converter [87] for generating OOV pronunciations.

3.7.6.1. In-Vocabulary vs Out-Of-Vocabulary Queries. Table 3.3 shows the performance of the ASR-based KWS system and proposed system on IV and OOV queries. We find that the proposed model has much smaller discrepancy between IV and OOV performance than the ASR-based system. With the exception of Pashto, the difference in between IV and OOV queries is always under 5 ATWV points.

In terms of ATWV, the proposed system slightly outperforms the ASR-based baseline system for OOV queries but lags it significantly for IV queries. This is somewhat expected, especially in the low-resource settings, as the baseline has a strong guide for IV queries in the word language model, an advantage that is diminished for OOV queries, even with a subword language model and index.

We also report the OTWV, which slightly favors the proposed model compared to the baseline. For IV terms, the relative average degradation between the proposed system and the baseline is reduced from 32% in ATWV to 21% OTWV. For OOV

terms, the relative improvement is increased from par to 8%. This suggests that part of the performance difference is due to the score normalization being better suited to the baseline rather than qualitative differences in the models.

Finally, we report the results of fusion, where we combine the hitlists from both systems by weighted summation of scores with weights tuned on the development sets. We find that the performance of the baseline is significantly improved by score fusion; around 12% on IV and 36% on OOV ATWV, with similar improvements in OTWV. This underscores the potential benefit of deploying both systems in tandem where computationally feasible. Note that we use BNF as our E2E models' acoustic features in the comparisons in this section. The equivalent model with XLS-R features outperforms the ASR-based system on all queries by an average of +2.6 ATWV (and constitutes, to our knowledge, state-of-the-art performance on this dataset). However, we do not use it in the following comparisons these as we believe the BNF model, which does not include any more external pretraining data than the ASR system, makes for a fairer comparison than XLS-R.

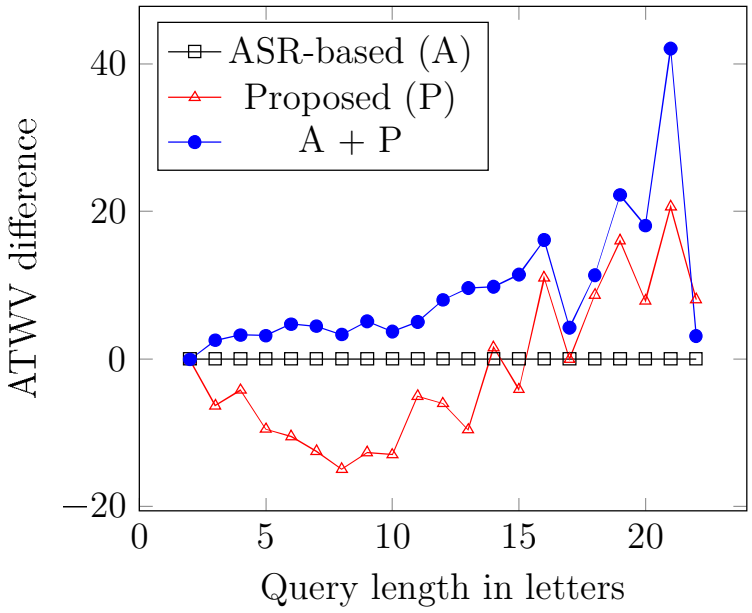

Figure 3.14. Average difference in ATWV of various systems when compared to the ASR-based baseline as query length varies.

3.7.6.2. Query Length. We have seen that while our approach does not distinguish much between IV and OOV queries, its IV performance trails that of a strong ASR-based KWS baseline. To find the root of this difference, we compare the performance of the systems on queries of various length.

Figures 3.14 and 3.15 show the average (across languages) difference in ATWV between the proposed model and the baseline. Negative values indicate query lengths for which the baseline is better and positive values indicate query lengths for which the proposed system is better. Note that although it is not conveyed in these figure, all systems—including the baseline—have better performance as the query length increases. Even so, we find that in general, the baseline performs better for shorter queries, both systems perform comparably for mid-length queries, and the proposed system performs better for long queries. Finally, we observe that fusion outperforms the baseline regardless of query length. These results are not surprising as short queries are both easier to miss and easier to falsely spot, and the ASR-based system being able to leverage contextual information provided by the language model helps it better find short queries.

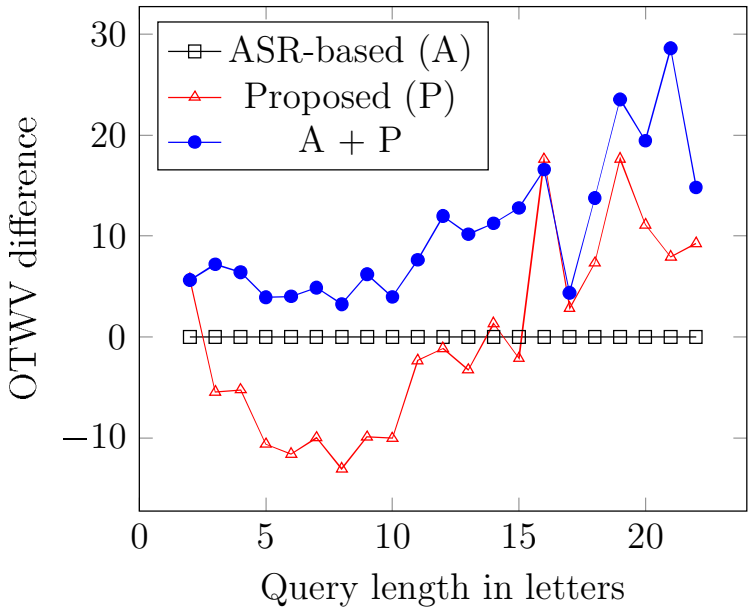

Figure 3.15. Average difference in OTWV of various systems when compared to the ASR-based baseline as query length varies.

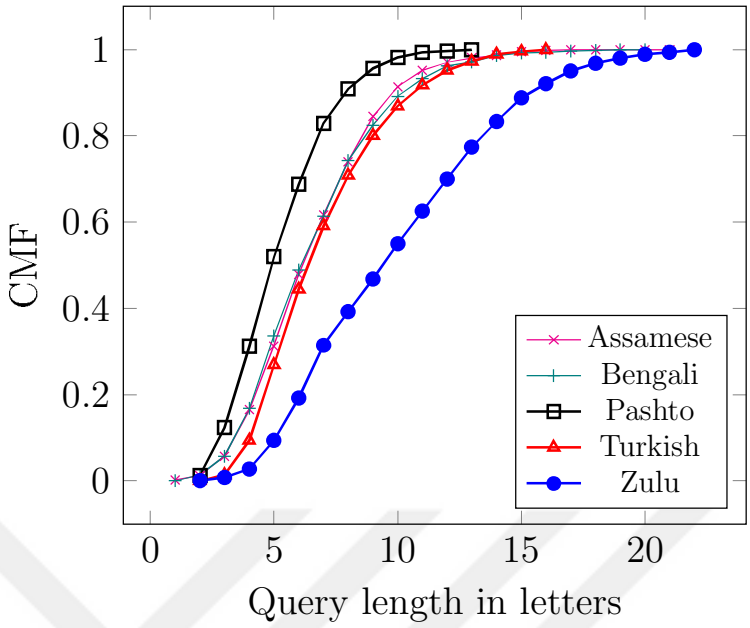

Figure 3.16. Cumulative distributions of query lengths for each evaluation set.

Figure 3.16 shows the distribution of query lengths per language. As the figure shows, the queries in the Pashto evaluation set are quite short, with over 50% of queries being less than 5 letters long. This partly explains why our model performs worse for Pashto than the other languages.

4. JOINT SPEECH AND TEXT RETRIEVAL

Keyword search, like other speech applications, benefits significantly from linguistic information. Consider a search for the query "eyeful" in an archive containing an utterance whose transcription is "I fully agree with you on this, Doctor Adigun". If the search were conducted based solely on acoustic characteristics, the beginning of the utterance "I full-" would be wrongly returned as a hit. The only way to disambiguate in such situations is through linguistic context, e.g., the knowledge that "Eyefully agree with..." is syntactically implausible.

The paired speech-text data used to train the KWS model provide a crucial source of this linguistic information. However, if we trained solely to such paired data, we would be ignoring a readily-available source of linguistic information, namely unpaired text corpora. Since these text corpora typically dwarf the size of paired corpora, they cover larger vocabularies with more expansive linguistic constructs that could significantly benefit KWS. Furthermore, incorporating text from an unpaired text corpus which covers a domain for which no paired training data is readily available could make the improve the KWS system's ability to accurately index and search archives in that domain.

The probabilistic formulation of ASR provides a principled mechanism for incorporating such unpaired text into ASR-based KWS models, namely the language model. The unpaired distribution can be used to train a better, possibly more domain-appropriate, LM, i.e., to better estimate the prior probability distribution over words which, when combined with the likelihood model (a neural network), yields a better estimate of the posterior distribution of words given audio.

The E2E KWS framework which we have introduced in Chapter 3 simplifies the KWS pipeline by directly modelling probabilities of query locations without going through intermediate step of modelling probabilities of words. However, due to this

loss of modularity, it cannot incorporate linguistic knowledge in the form of an LM. Therefore, we propose an alternative approach of integrating linguistic knowledge from external text corpora into the model.

In the proposed text integration framework, we generalize the location prediction paradigm to text modality. In addition to training the model to search for text in speech, we simultaneously train it to search for text in text. Doing so allows us to better learn from text-only corpora the linguistic contexts that can support various queries.

4.1. Baseline End-to-End Keyword Search

This section recapitulates the E2E KWS model described in Chapter 3—which we refer to as Baseline End-to-End Keyword Search(BeKWS) in this chapter. BeKWS—depicted in Figure 4.1—is a model trained to predict the probabilities of a query occurring in each frame of a spoken document.

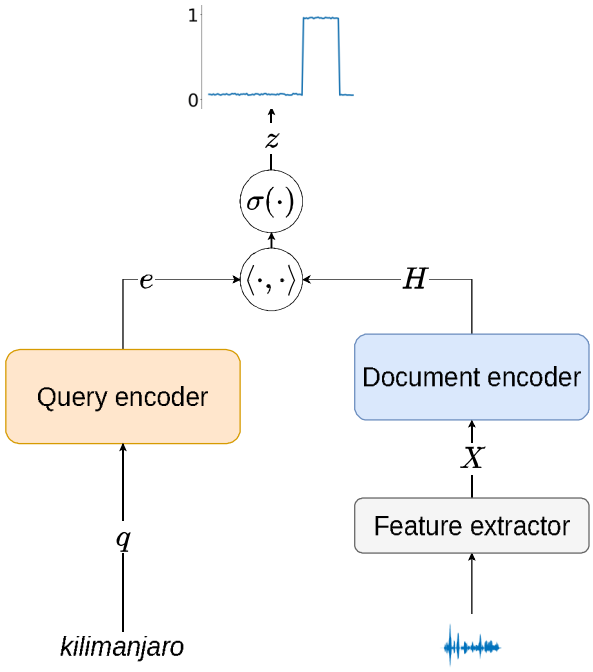

Figure 4.1. Simplified schema of the BeKWS model.

For a (possibly multi-word) query $\boldsymbol{q} = (q_1, q_2, \ldots, q_K)$ comprising a sequence of K letters and a document $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$ comprising a sequence of N acoustic frames, the model is used to predict the sequence $\boldsymbol{y}(\boldsymbol{q}, \boldsymbol{X}) = (y_1, \ldots, y_N)$, where each $y_n \in \{0, 1\}$ is a binary random variable indicating the existence of the query, i.e.,

$$y_n = \begin{cases} 1, & \text{if } \boldsymbol{q} \text{ is spoken in } \boldsymbol{X} \text{ in a time span including } n \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

The model has parameters $\boldsymbol{\theta}$ comprising a document encoder and a query encoder. Given the query $\boldsymbol{q}$ and the document $\boldsymbol{X}$, the model outputs the sequence $\boldsymbol{z}(\boldsymbol{q}, \boldsymbol{X}) = (z_1, \ldots, z_{\hat{N}})$ of query occurrence probabilities where

$$z_{\hat{n}} = \sigma\Big(\boldsymbol{h}_{\hat{n}}^{\top} \boldsymbol{e}_{\boldsymbol{q}}\Big), \tag{4.2}$$

where $\boldsymbol{h}_{\hat{n}}$ is the $\hat{n}$th frame of $\boldsymbol{H}_{\boldsymbol{X}}$, a down-sampled representation of $\boldsymbol{X}$ computed by the document encoder; $\boldsymbol{e}_{\boldsymbol{q}}$ is the vector representation of the query computed by the query encoder, and $\sigma(\cdot)$ is the logistic sigmoid function. Thus, $z_{\hat{n}}$ can be interpreted as $P_{\boldsymbol{\theta}}\big(y_{\hat{n}} = 1 | \boldsymbol{q}, \boldsymbol{X}\big)$.

Given a training dataset, $\mathcal{X}$, containing a set of spoken utterance and their transcriptions, the model is trained by stochastic gradient descent to minimize the negative log-likelihood of the indicators

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{\boldsymbol{q} \in Q} \sum_{\boldsymbol{X} \in \mathcal{X}} \sum_{n} -\log P_{\boldsymbol{\theta}}\Big(y_n | \boldsymbol{q}, \boldsymbol{X}\Big), \tag{4.3}$$

where the training queries are taken from $\mathcal{Q}$, the set of all unigrams, bigrams, trigrams in the transcripts of $\mathcal{X}$, and the word-level timestamps required for training are obtained by forced-alignment with an HMM-GMM-based ASR system trained on the KWS training data.

4.2. Joint Speech and Text Retriever

The approach we propose for incorporating unpaired text into E2E KWS, Joint Speech and Text Retriever(JOSTER)—depicted in Figure 4.2—involves modifying the

document encoder of BeKWS to accept not just acoustic inputs but also textual ones, so that queries can be retrieved from not just spoken documents but also written ones.

To achieve this, we introduce a text document pre-encoder with trainable parameters, $\boldsymbol{\theta}^{text}$. The pre-encoder transforms a written sentence into a sequence in the space of acoustic features. The resulting sequence is fed into the document encoder as if it were a sequence of actual acoustic features in which a query's location will be searched.

The pre-encoder's input is a written document presented as a sequence $\boldsymbol{W}^{text} = (w_1, \ldots, w_N)$ of N containing letters $\{w_i\}$ with no spaces between them. We mask and repeats the letter in the document before inputting them into the pre-encoder.

First, we mask the sentence to obtain $\tilde{\boldsymbol{W}}^{text} = (\tilde{w}_1, \ldots, \tilde{w}_N)$, where

$$\tilde{w}_n = \begin{cases} \_, & \text{with probability } \pi, \\ w_n, & \text{with probability } 1 - \pi, \end{cases} \tag{4.4}$$

where _ is a special mask symbol. The masking acts as a form of regularization, à la Dropout, on the input sequence.

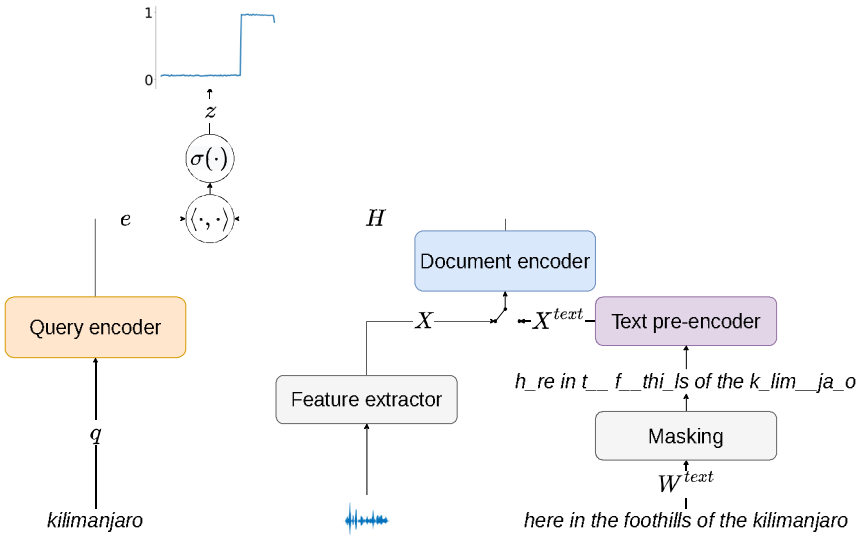

Figure 4.2. Proposed JOSTER model.

After masking, we incorporate a rudimentary duration model transforming the input to $\hat{\boldsymbol{W}}^{text} = (\hat{w}_1, \ldots, \hat{w}_N)$ where each $\hat{w}_n$ is obtained by simply repeating $\tilde{w}_n$ ρ times. For instance, the phrase $\boldsymbol{W}^{text} = \texttt{thecat}$ might be converted by masking to $\tilde{\boldsymbol{W}}^{text} = \texttt{t\_ec\_t}$, and then, if $\rho = 2$, to $\hat{\boldsymbol{W}}^{text} = \texttt{tt\_\_eecc\_\_tt}$.

This final representation is then input into the text pre-encoder—a recurrent neural network with an embedding lookup input layer with parameters $\boldsymbol{\theta}^{text}$—to obtain $\boldsymbol{X}^{text}$, which can then be used in the same way as document in BeKWS. Note that $\boldsymbol{X}^{text}$ depends on both the written document and the parameters of the neural network, and would perhaps more correctly be denoted as $\boldsymbol{X}^{text}(\boldsymbol{W}^{text}; \boldsymbol{\theta}^{text})$. However, here and henceforth, we omit this dependency from the notation for better readability.

As explained in Section 4.1, for spoken documents, we seek the sequence of occurrence indicators, $\boldsymbol{y}(\boldsymbol{q}, \boldsymbol{X})$, which is defined by whether the query is spoken at a time span of the document. For written documents, the we seek the corresponding sequence, $\boldsymbol{y}(\boldsymbol{q}, \boldsymbol{X}^{text})$ which is similarly defined by whether the query occurs exactly at a given location. For each letter in the document, the occurrence indicator is 1 if it occurs as part of a subsequence of words which are identical to the query; otherwise it is 0. In the example above, with document sentence $\boldsymbol{W}^{text} = \texttt{thecat}$ and $\hat{\boldsymbol{W}}^{text} = \texttt{tt\_\_eecc\_\_tt}$, for a query $\boldsymbol{q} = \texttt{cat}$,

$$\boldsymbol{y}(\boldsymbol{q}, \boldsymbol{X}^{text}) = 000000111111. \tag{4.5}$$

For any query $\boldsymbol{q}$, we can use Equation 4.2 to get the occurrence probabilities $\boldsymbol{z}(\boldsymbol{q}, \boldsymbol{X})$ and $\boldsymbol{z}(\boldsymbol{q}, \boldsymbol{X}^{text})$, corresponding respectively to the sequence of probabilities $\left\{P_{\boldsymbol{\theta}}\left(y_{\hat{n}}|\boldsymbol{q}, \boldsymbol{X}\right)\right\}_{\hat{n}=1}^{\hat{N}}$ and $\left\{P_{\boldsymbol{\theta}}\left(y_{\hat{n}}|\boldsymbol{q}, \boldsymbol{X}^{text}\right)\right\}_{\hat{n}=1}^{\hat{N}}$, with the parameters $\boldsymbol{\theta}$ shared by both the text-speech retriever and the text-text retriever.

4.3. Training

The JOSTER training procedure closely follows the BeKWS training procedure described in Section 3.3, with modifications made to accommodate the differences between the models.

We train the model jointly on a paired speech-text dataset, $\mathcal{X}$, and an unpaired text-only one, $\mathcal{X}^{text}$, using gradient descent. At each training step, t, we sample between the datasets with $50:50$ probability. When the paired dataset is sampled, then the training step is identical to that described in Section 3.3. When the unpaired dataset is sampled we compute gradients with the analogous objective

$$J_t = \sum_{l=1}^{L_b} \sum_{m=1}^{M} f\Big(\boldsymbol{z}\big(\boldsymbol{q}_{t,l}, \boldsymbol{X}^{text}_{t,l,m}\big), \boldsymbol{y}\big(\boldsymbol{q}_{t,l}, \boldsymbol{X}^{text}_{t,l,m}\big)\Big), \tag{4.6}$$

where $\{\boldsymbol{q}_{k,1} \dots \boldsymbol{q}_{k,L_b}\}$ is a mini-batch of L_b queries sampled randomly from the set of unigrams, bigrams and trigrams of $\mathcal{X}^{text}$; $\{\boldsymbol{X}^{text}_{k,l,1}, \dots, \boldsymbol{X}^{text}_{k,l,M}\}$ are the pre-encoder outputs from $\{\boldsymbol{W}^{text}_{k,l,1}, \dots, \boldsymbol{W}^{text}_{k,l,M}\}$, a set of documents sampled from the dataset such that $\{\boldsymbol{W}^{text}_{k,l,1}\}$ contains $\boldsymbol{q}_{k,l}$ while the other $M-1$ documents are sampled randomly; and $f(\cdot)$ is the same modified binary cross-entropy function used in BeKWS

$$f(\boldsymbol{z}, \boldsymbol{y}) = -\sum_{n=1}^{N} \Big(\mathbb{1}_{z_n > 1-\phi} \cdot (1 - y_n) \log(1 - z_n) + \mathbb{1}_{z_n < \phi} \cdot \lambda \cdot y_n \log z_n \Big). \tag{4.7}$$

The parameters $\boldsymbol{\theta}$ and $\boldsymbol{\theta}^{text}$ are trained with a stochastic gradient-based optimizer. This entails initializing them with random parameters and the updating them iteratively, as in Equation 3.9, in the direction of steepest gradient descent.

4.4. Keyword Search with Joint Speech and Text Retriever

After the model is trained, we discard the text pre-encoder as it is not required for KWS. Thus, for indexing and search, JOSTER becomes effectively identical to BeKWS. Finally, we post-process the output of the query and document encoders for KWS using the procedure described in Section 3.4.

Table 4.1. Statistics of the training text corpora.

Language	Assamese	Bengali	Pashto	Turkish	Zulu
LLP vocabulary size	7661	7993	6186	10110	13764
FLP vocabulary size	22033	24339	17640	38311	54295
LLP OOV rate (%)	12.6	13.1	11.4	19.2	20.3
FLP OOV rate (%)	1.6	2.1	2.3	6.5	6.7

4.5. Experiments and Results

In this section, we conduct experiments to analyze various aspects of the proposed JOSTER model. First, we describe the experiment setup including datasets, input features, metrics and model configuration. Next we present a macro KWS performance comparison between JOSTER and BeKWS. Then we analyze the effect of the text representation hyperparameters on search performance. Afterwards, we conduct experiments to understand how JOSTER achieves its improvements with analyses of the effect of the size and choice of unpaired text, the performance difference on various kinds of queries and, finally, the effect of the domain of the unpaired text on KWS performance.

4.5.1. Experimental Setup

The setup of the experiments in this chapter follows those in the previous chapter (Section 3.7.1) with a few differences, which we highlight in this section.

4.5.1.1. Datasets. We conduct the bulk of our experiments on the IARPA Babel corpus described in Section 3.7.1.1. As with the experiments in Chapter 3, we use the limited language pack which contains about 10 hours of training data per language as the paired training data. In addition, we use the text from the full language packs (FLP) as the unpaired text for each language. These contain 5-6 times as many sentences per language as the LLP subsets.

We also use the same 10-hour Dev sets and a 5-hour Eval sets, with the same query sets. Table 4.1 gives a summary of the text data for each language including the size of the paired text lexicon, the size of the unpaired text lexicon, and the proportion of evaluation queries which are OOV with respect to each text source. We note that Turkish and Zulu, both agglutinative languages, have larger vocabulary sizes and higher OOV rates.

In addition to these, we use the LLP data from 19 other languages of the Babel corpus (about 190 hours in total) for multilingual pretraining of the KWS model—which we showed to significantly improve KWS performance for BeKWS in Chapter 3—in order to measure whether and how well the proposed method can be used to improve a strong multilingually-pretrained KWS baseline.

In addition, we experiment with data from the Turkish Broadcast News (BNTR) in the multi-domain experiments of Section 4.5.8, and we use the Librispeech corpus of read speech [88] in Section 4.5.9 where we compare with a TTS-based text augmentation scheme.

4.5.1.2. Acoustic Features. We mainly use features from the pretrained 300 million parameter XLS-R model [77] as the acoustic input to our KWS system. In Section 4.5.2, we also report results using multilingual BNF as an alternative acoustic input, giving us another axis along which to analyze the performance of the proposed joint training method. See Section 3.7.1.2 for more information about these acoustic features.

4.5.1.3. Metrics. We use the same TWV metrics as in the previous chapter as the main metrics for measuring system performance. In addition, we report Detection Error Tradeoff (DET) curves in Section 4.5.2. The DET curves show a plot of miss probabilities vs false alarm probabilities for a KWS system, giving a more holistic view of keyword search performance. KWS systems with DET curves closer to the lower-left corner of the plot have better false alarm to miss trade-offs and are thus considered

better.

4.5.1.4. Model Configuration. We base the architecture of our model on the architecture described in Section 3.7.1.4. The only difference is the text pre-encoder, for which use a configuration comprising an 32-dimensional embedding layer for computing vector representations of each grapheme in the written document, followed by a single BLSTM layer with 512 output units per direction.

The other difference is that here, we use the same amount of down-sampling (a single down-sampling by a factor of 2 after the fourth encoder BLSTM layer) in the document encoder for both the BNF and XLS-R input. Therefore, the frame length at the encoder output for XLS-R features remains 40ms while the BNF encoder output has 20ms frames.

4.5.1.5. Training Hyper-Parameters. For the text-document representation, we set the masking probability to $\pi = 0.3$ and the duration to $\rho = 2$. For the training loss function, we set the positive frame weight to $\lambda = 5$, the tolerance parameter to $\phi = 0.7$ and the number of training utterances per query to $M = 4$.

For JOSTER training, we randomly sample and set aside 10% of the training query n-grams from the paired data for validation and we use the Adam optimizer for stochastic gradient optimization. We use a mini-batch size (number of queries per training step) of 32 for monolingual training and finetuning; for multilingual pretraining, we use a batch size of 256. We start optimization with a learning rate of 2×10^{-4} and halve the learning rate whenever the validation loss stagnates for four epochs and stop training whenever the validation loss stagnates for 10 epochs. This is nearly identical to BeKWS training, with the exception that, since training involves sampling batches uniformly between the paired and much larger unpaired datasets, here we define an epoch as 10000 training steps instead of the size of either dataset.

Table 4.2. TWV comparison between the BeKWS and JOSTER models.

		BNF		XLS-R		XLS-R*	
Language	System	Dev	Eval	Dev	Eval	Dev	Eval
Assamese	BeKWS	17.3	17.9	26.4	25.1	34.0	34.0
	JOSTER	**22.5**	**23.5**	**30.2**	**30.5**	**37.9**	**37.6**
Bengali	BeKWS	18.4	17.0	29.8	27.4	35.1	34.2
	JOSTER	**24.8**	**23.1**	**34.2**	**32.7**	**40.9**	**38.7**
Pashto	BeKWS	13.5	16.3	22.4	26.3	29.9	33.4
	JOSTER	**14.9**	**18.5**	**25.9**	**30.4**	**31.5**	**35.2**
Turkish	BeKWS	29.2	21.6	41.2	34.4	46.0	42.0
	JOSTER	**35.3**	**26.8**	**46.6**	**39.2**	**48.6**	**43.8**
Zulu	BeKWS	21.4	22.5	31.3	28.9	39.8	36.2
	JOSTER	**25.9**	**25.7**	**39.0**	**35.8**	**44.4**	**42.2**
Average	BeKWS	20.0	19.1	30.2	28.4	37.0	36.0
	JOSTER	**24.7**	**23.5**	**35.2**	**33.7**	**40.7**	**39.5**

4.5.2. Performance Comparison to Baseline End-to-End Keyword Search

In this section, we compare the performance of JOSTER to BeKWS across languages and feature kinds. Table 4.2 shows the term weighted values for each of the five test languages.

For the baseline (BeKWS), as already noted in Section 3.7.5, XLS-R features significantly outperform BNF across languages—with a difference on average of +10.2 MTWV on the Dev sets and +9.3 ATWV on the Eval sets.

Furthermore, the XLS-R* setting—which uses speed perturbation, multilingual pretraining of the document encoder (Section 3.5) and increased the number of training utterances per query, with M increased from 4 to 8—improves the BeKWS performance is increased by an additional +6.8 Dev MTWV and +7.6 Eval ATWV on average across

languages, showing that BeKWS with XLS-R features can be improved with multilingual KWS pretraining despite the fact that XLS-R feature extractor is a multilingually trained neural network.

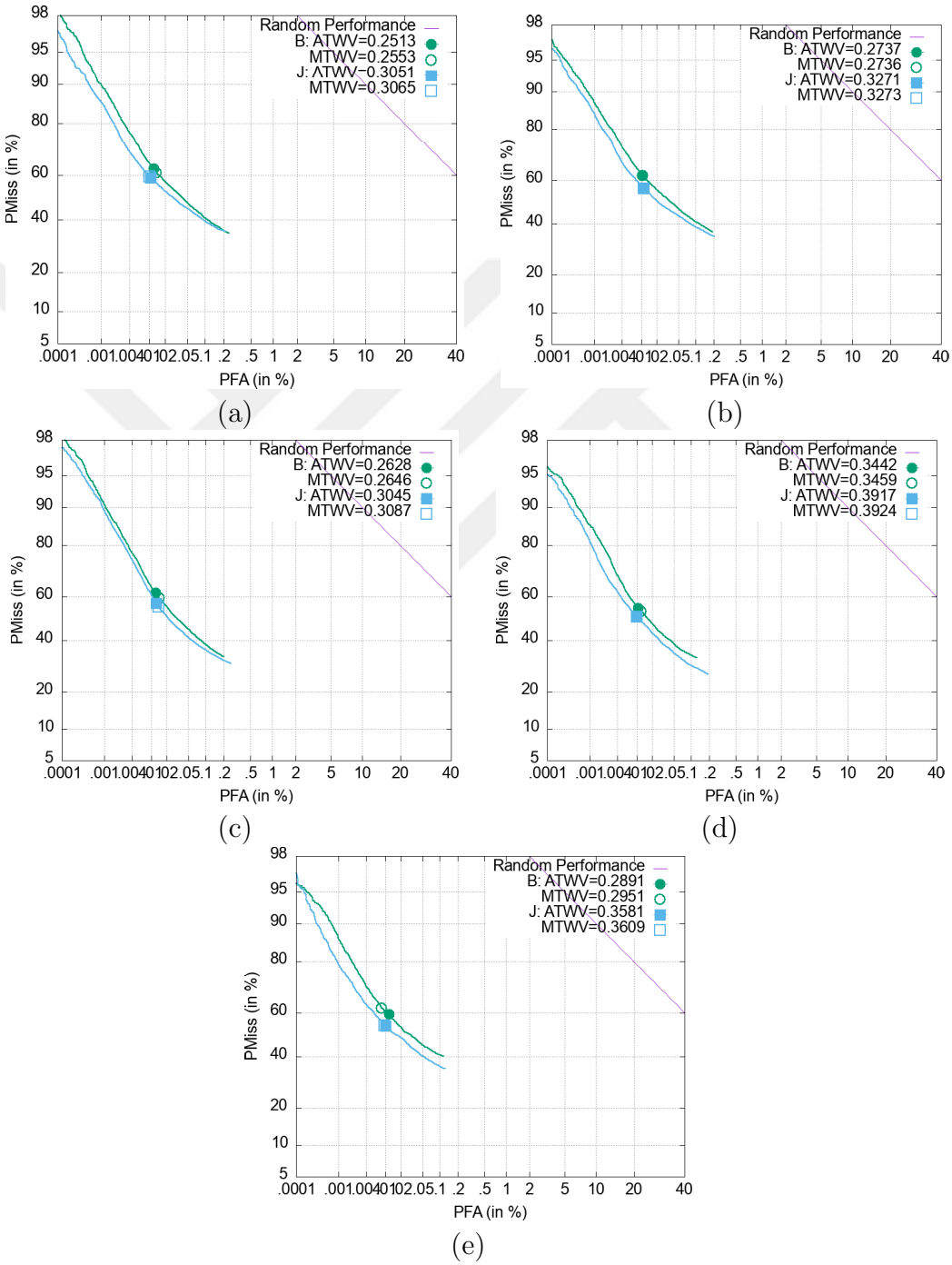

Figure 4.3. DET plots showing the evolution of misses and false alarms on the (a) Assamese, (b) Bengali, (c) Pashto, (d) Turkish and (e) Zulu evaluation sets for BeKWS (B) and JOSTER (J).

We find that JOSTER *invariably* improves the TWV by considerable margins compared to BeKWS in each setting (BNF, XLS-R, XLS-R*). For BNF, the improvements across languages average +4.7 for Dev set MTWV and +4.4 for Eval set ATWV. For raw XLS-R features, the respective improvements increase slightly to +5 and +5.3. When finetuning the multilingually pretrained model with XLS-R features, we get average improvements of +3.7 and +3.5 by using JOSTER instead of BeKWS. Note that we use the same multilingual model—which is trained without unpaired text—to initialize the document encoders for both BeKWS and JOSTER.

The DET plots in Figure 4.3 provide an even more comprehensive picture of the performance difference. In each test language, JOSTER outperforms BeKWS across virtually all operating points of the plots; i.e., at any given recall rate, JOSTER incurs fewer false alarms than BeKWS, further strengthening the significance of JOSTER's performance superiority.

4.5.3. Text Pre-Processing

As described in Section 4.2, when computing the representation of written documents during training, we first mask with probability $\pi = 0.3$ and repeat each token $\rho = 2$ times. In this section, we quantify the significance of these choices on retrieval performance.

Figure 4.4 shows the MTWV as π is varied with ρ fixed to 2. We find that even without masking (at $\pi = 0$), JOSTER already outperforms BeKWS by +3.5 MTWV. This runs counter to our original intuition that without masking, retrieval from written sentences could be too trivial to aid learning. Nevertheless, setting π to 0.15 further improves MTWV by an average of +1.9. Increasing π further starts to worsen performance. We note that although MTWV varies with the masking rate, only at extreme values ($\pi > 0.9$) does it get worse than the baseline, indicating that the joint training is robust across a large range of π. We surmise from these results that having unpaired text input is crucial, and the masking acts as extra regularization

in the vein of Dropout.

Figure 4.5 shows the performance of JOSTER as ρ is varied with π fixed to 0.3. JOSTER outperforms the baseline across all the settings of ρ we tried. Although the average MTWV at $\rho = 2$ is better than the MTWV at $\rho = 1$, by 1.4, the latter may still be preferred as the computational cost of the text document pipeline increases linearly with ρ. Finally, we consider a more involved duration model (denoted $\bar{\rho}$ in the figure), where we set ρ for each letter to be its average duration—estimated by forced-alignment with graphemic HMMs trained for each language. We find that this added complexity yields no TWV improvements. In fact, it generally degrades performance compared to fixed duration with $1 \leq \rho \leq 4$.

Overall, we note that although both parameters can change the performance of the system, the variance is low enough that JOSTER still outperforms BeKWS over large ranges of either parameter.

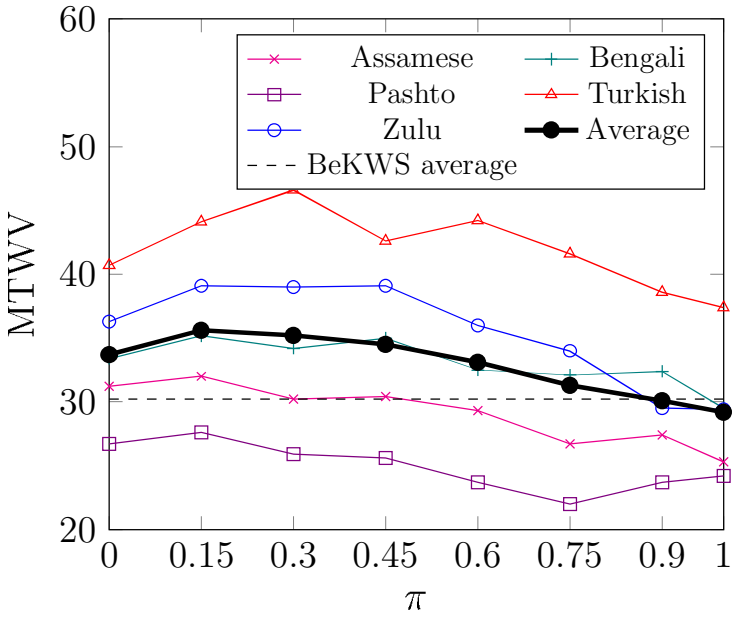

Figure 4.4. MTWV on the development sets as the masking rate of text documents is varied.

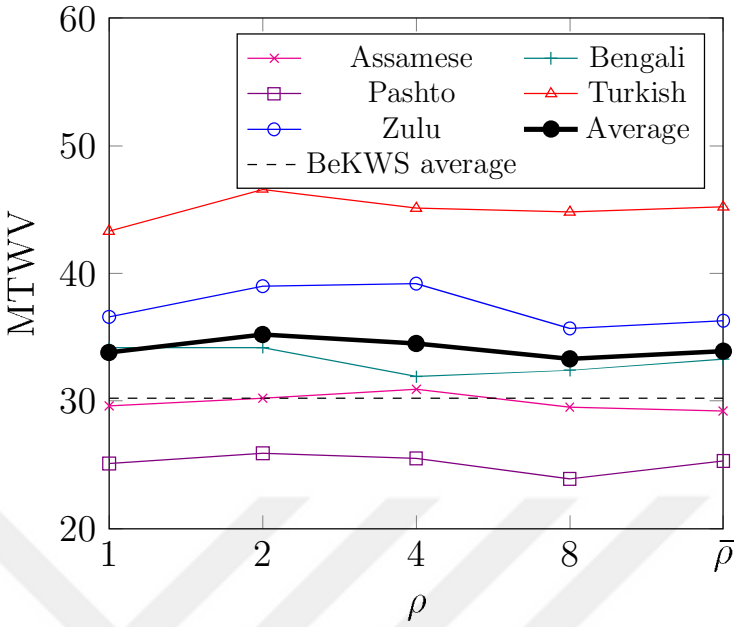

Figure 4.5. MTWV on the development sets as the duration of each letter in text documents is varied.

4.5.4. Number of Negative Utterances per Training Step

In this section, we measure the impact of the number of negative examples in each training step. Instead of fixing $M = 4$ for both paired and unpaired batches, we vary them in turn:

- $M(audio)$: We set $M = 4$ for unpaired batches and vary it between 2, 4 and 8 for paired batches.
- $M(text)$: We set $M = 4$ for paired batches and vary it between 2, 4 and 8 for unpaired batches.

Figure 4.6 shows the impact of these variations. In both cases, increasing M increases the MTWV, although $M(audio)$ has higher impact compared to $M(text)$. This however comes at the cost of increased compute and memory cost for each training step.

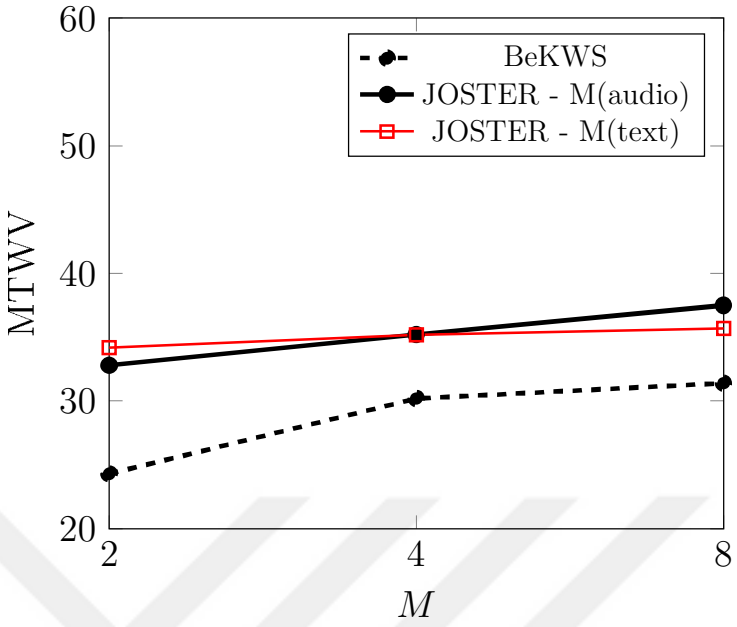

Figure 4.6. Average MTWV on the development sets as the number of negative utterances per training step is varied.

4.5.5. Number of Shared Layers

So far, the we have fed the output of the text pre-encoder into the document encoder. In this section, we experiment with feeding the text pre-encoder into intermediate layers of the document encoder, so that the number of layers shared between the paired and unpaired objectives is varied and the pre-encoder learns to generate features in the LSTM hidden space instead of the input feature space.

Figure 4.7 shows that the Dev set MTWV generally improves as more layers are shared. It is particularly noteworthy that when no layers are shared by the two modalities' document encoders the MTWV is almost identical to the BeKWS MTWV, even though the query encoder shared as always. This indicates that the performance improvements result from using the unpaired text to improve the (acoustic) representations learned by the document encoder rather than simply having more text data for training the query encoder.

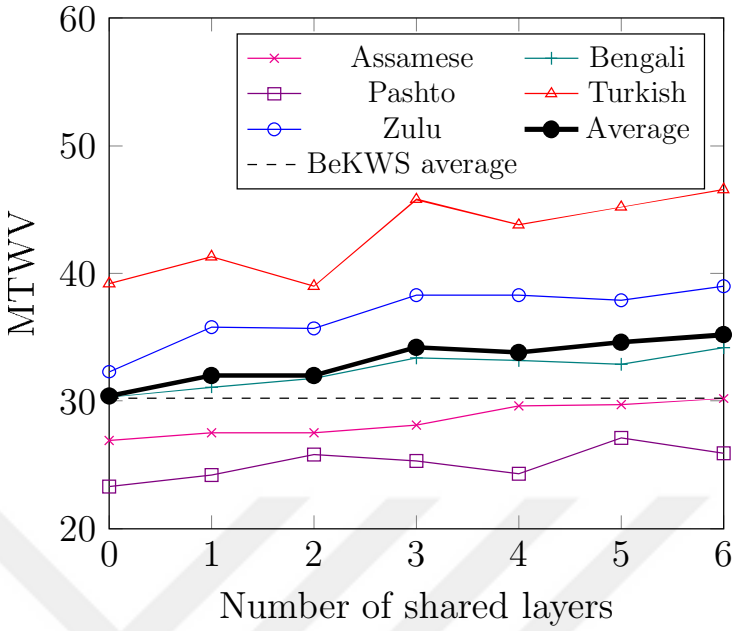

Figure 4.7. MTWV on the development sets as the number of layers shared between the two training tasks is varied.

4.5.6. Size of Unpaired Text

In this section, we measure the impact as we change the amount of unpaired text used for the auxiliary text-to-text task and report the results in Figure 4.8. First, we compare using the FLP text as has been done so far to using the LLP text (denoted as LLP and FLP respectively in the figure), i.e., using the transcripts of the paired data as the "unpaired" text. The LLP text performs significantly worse than using the FLP text and, in three of the five languages, worse even than BeKWS.

Next, to test how much of this degradation is due to data size and how much of it results from using the same text as the paired data, we create three random subsets of the FLP text (with the LLP text excluded) each with the same number of sentences as the LLP text. We report the MTWV of the best (S-max) and worst (S-min) performing of these splits for each language. We observe that even the best split performs much worse than the full FLP indicating the size of the augmentation text matters. However, we also observe that even the worst random split outperforms the LLP text, indicating that textual diversity is also crucial.

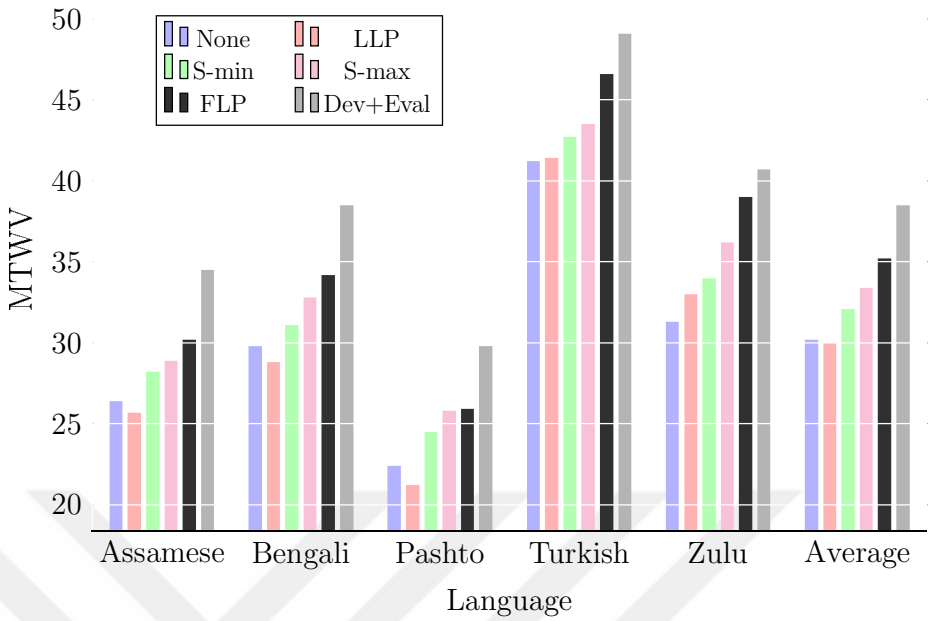

Figure 4.8. MTWV on the development sets as the size and composition of unpaired text is varied.

Finally, we report a topline (Dev+Eval) where we use the text from the transcriptions of the Dev and Eval sets as the unpaired text for training, and find that, unsurprisingly, it outperforms using even the larger FLP text. While it is unrealistic to assume that the transcription of the test set can be obtained beforehand, this shows that further improvements can be obtained if it can be somewhat anticipated.

4.5.7. In-Vocabulary and Out-of-Vocabulary Queries

In this section, we quantify how much improvement we get on various queries depending on whether or not they exist in the unpaired text.

Figure 4.9 shows the ATWV difference between JOSTER and BeKWS across languages for queries that are:

(i) OO: OOV with respect to both the KWS LLP training data and the FLP text.
(ii) OI: OOV with respect to the KWS LLP training data but in the FLP vocabulary.
(iii) II: In-vocabulary with respect to both the LLP training data and the FLP text.

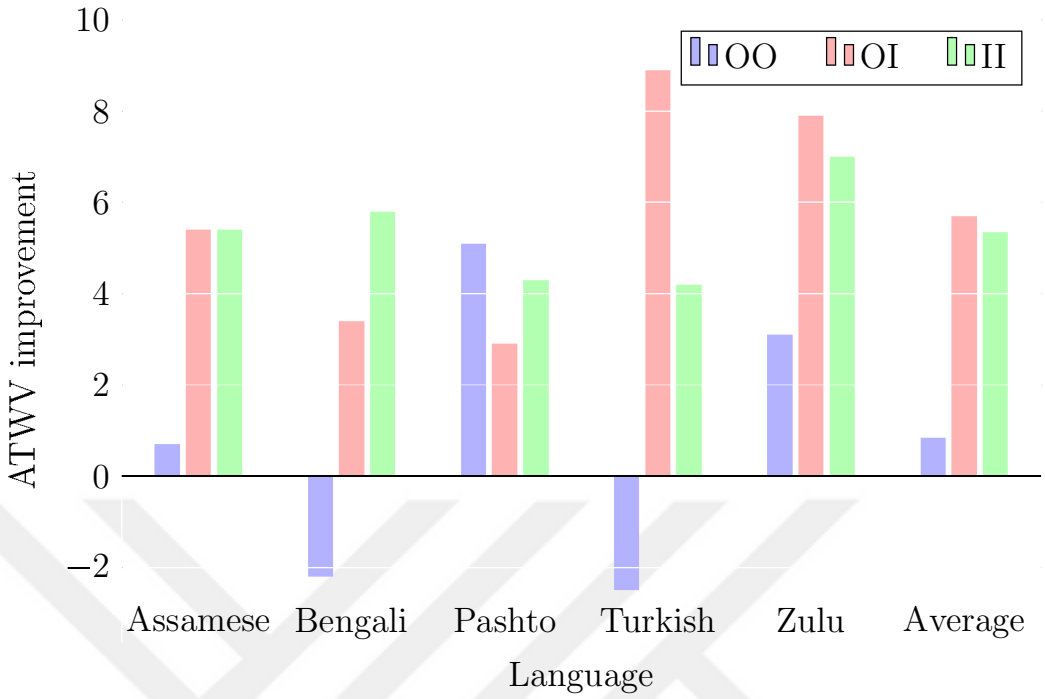

Figure 4.9. ATWV differential between the proposed system and the baseline on different subsets of the eval sets' queries.

The worst average improvements over BeKWS (+0.84 on average ATWV) are achieved for OO queries (which form a minute proportion of all queries as shown in the OOV-F column of Table 4.1), with performance even degrading for two of the five languages. For OI and II queries, we get consistent significant improvements (+5.7 and +5.3 average ATWV respective improvements).

Overall, we infer that JOSTER improves the document encoder's representation of words which are in the augmentation text regardless of whether or not they actually exist in spoken form in the paired data.

4.5.8. Performance in Mismatched Domain Setting

So far, we have trained and tested exclusively with Babel data. In this section, we experiment with Turkish data from various domains with various configurations of paired and unpaired data. The objective for doing so is twofold:

(i) To what extent is the matching domain necessary? In other words, can we improve the TWV in one domain by using text from a different domain?

(ii) Is text-only domain adaptation possible? Given paired training data in one domain and a test set in another domain, how much can we improve the test set performance using unpaired text from the target domain?

To answer these questions, we conduct experiments on two Turkish language datasets. In addition to the Turkish Babel dataset used in previous experiments, we use Turkish Broadcast News (BNTR) for KWS training and testing.

To match the training data size of the Babel LLP corpus, we use a 10-hour subset of BNTR from the VOA channel[5] for training. We select two 10-hour subsets from the remaining BNTR data as Dev and Eval sets. Since the BNTR dataset has no official keyword lists, we randomly select 1500 queries composed of equal proportions of unigrams, bigrams and trigrams for each of the Dev and Eval sets with OOV rates of 11.7% and 6.5% respectively. We experiment with three text corpora for unpaired training: Babel FLP text, BNTR—unpaired text from the Broadcast News dataset totalling around 180 hours and text from Turkish Wikipedia. The first two allow us to measure the impact of using text from the test domains, while Wikipedia stands as a control corpus.

Table 4.3 shows the results of training with various configurations of paired and unpaired data on the different test sets. First, we note that the BNTR results are generally better than the Babel ones, which is to be expected as the former corpus contains news recordings of professional newscasters while the latter contains conversational speech recorder over telephone channels.

For each test set, we get improvements by using JOSTER regardless of the unpaired text used for joint training. However, we get the largest improvements when we use text from the target domain to augment training.

[5]https://catalog.ldc.upenn.edu/LDC2012S06

Table 4.3. TWV on the Turkish Babel and Broadcast News datasets as the paired and unpaired training data are varied.

	Training data		Test set			
	Paired	Unpaired	BNTR		Babel	
System	speech	text	Dev	Eval	Dev	Eval
BeKWS	Babel	-	55.8	56.1	41.2	34.4
JOSTER	Babel	Babel	64.1	64.1	46.6	39.2
JOSTER	Babel	Wikipedia	68.5	67.6	41.2	37.5
JOSTER	Babel	BNTR	74.6	74.1	46.5	40.8
BeKWS	BNTR	-	84.8	86.1	24.2	20.5
JOSTER	BNTR	Babel	86.5	87.8	29.2	27.4
JOSTER	BNTR	Wikipedia	87.6	88.6	26.1	25.7
JOSTER	BNTR	BNTR	89.0	89.8	28.3	25.6

In the cross-domain setting where we train with the paired Babel data and test on BNTR, JOSTER with the Babel FLP text improves the Dev MTWV and Eval ATWV by +8.3 and +8.0 respectively compared to BeKWS. Augmenting with Wikipedia text results in further +4.4 and +3.5 Dev and Eval improvements compared to using the Babel FLP text. Finally, using BNTR (target domain) unpaired text provides further improvements of +6.1 and +6.5 compared to Wikipedia. This final result cuts the gap to a topline of using BNTR data for training by 65% and 60% on the Dev and Eval sets respectively.

In the converse cross-domain setting (training with BNTR paired data and testing on Babel), we observe a similar trend where JOSTER using Wikipedia improves on the performance of BeKWS, with further performance gains obtained from using BNTR text, and the best performance resulting from using Babel text. We note, however, that the performance improvements are not as dramatic in this case—likely due to the difficulty of transferring the BNTR-trained model to the difficult acoustic conditions of the Babel data.

Table 4.4. TWV on the Libri-Light corpus.

		Clean		Other	
System	Unpaired text	Dev	Test	Dev	Test
BeKWS	-	73.2	72.7	62.2	62.3
JOSTER	Wikipedia	79.2	79.8	67.8	69.5
JOSTER	Librispeech-100	82.6	83.0	71.7	72.7
TTS	Librispeech-100	85.0	84.9	67.8	67.8

Finally, when training and testing within the same domain, we observe that JOSTER generally improves the TWV compared to BeKWS even with unpaired text from other domains. This holds even for BNTR which already has a high baseline performance.

Overall, these results add an extra dimension to the results so far, showing that the proposed joint training method performs well, not just across languages—as shown in previous sections—but also across domains within the same language. Furthermore, they suggest that training JOSTER with unpaired text from a domain most improves search performance on test sets in that domain, providing a good alternative when domain-specific data is limited.

4.5.9. Comparison with Text-to-Speech-Based Text Augmentation

In this section, we compare our proposed method with TTS-based unpaired text integration, where we use an off-the-shelf TTS model to synthesize speech for the unpaired text and train with the resulting data. Here, we experiment with English language corpora due the difficulty of obtaining high-quality open-source TTS systems for other languages. Specifically, we use the 10-hour Libri-light corpus [89] as the paired KWS training data and the 100-hour Librispeech training set [88] as the unpaired data for JOSTER and TTS. For JOSTER, we also experiment with text from Wikipedia. We use the Coqui xTTS system [90] for speech synthesis. We test on the standard

Librispeech test splits (Dev-clean, Test-clean, Dev-other and Test-other) with around 1300 randomly generated queries for each test split.

Table 4.4 shows the results of these experiments. JOSTER, even with Wikipedia text, improves across all dev and test sets, and further, although the best performance is achieved when the in-domain Librispeech-100 text is used. Similarly, using TTS for data augmentation significantly improves KWS compared to BeKWS.

Compared to JOSTER, we note that TTS performs better on the "clean" test sets and performs worse on the more acoustically-challenging "other" sets. This highlights a difference between the two approaches. JOSTER, being text-based, is more channel-agnostic and is more influenced by linguistic similarities between the unpaired text and the target. TTS, on the other hand, is also influenced by channel match between the output of TTS (which is typically clean speech by design) and the test audio. Although, the impact of TTS augmentation on KWS could plausibly be improved by augmentation with artificially generated noise, reverberations or room impulse responses, an in-depth exploration of TTS-based augmentation is out of the scope of this paper. Moreover, these add extra complications which JOSTER does not have.

5. CONCLUSION

In this thesis, we have presented a novel end-to-end framework for keyword search. We introduce a dual-encoder architecture which converts the KWS problem into a vector matching problem, with vector representations of queries compared with frame-wise representations of documents. Thus, we avoid the cumbersome step of ASR in cataloging and searching speech archives. For training the proposed architecture, we propose an objective function based on a modification of the binary cross-entropy objective. We show that the proposed method is not only efficient but also an effective KWS system which achieves competitive performance with an ASR-based system.

Next, noting that end-to-end frameworks like ours tend to be data intensive, we adopt multilingual pretraining as a strategy for improving performance in low-resource settings. We pool data from several languages for pretraining—effectively increasing the size of the model's training data—and then finetune on target languages. We show that such multilingual pretraining leads to improved search performance across all tested languages and acoustic features even when such acoustic features already have a multilingual provenance.

Finally, we present a joint speech and text retriever for improving E2E KWS by learning from unpaired text. While training the KWS model to retrieve text queries from spoken documents, we simultaneously train it to retrieve text queries from written documents. Since this latter objective only requires text, it allows us to train on large text corpora which we would otherwise be unable to utilize. We show that this joint training objective improves KWS across several languages, acoustic features and domains by improving the document representations of the terms in the unpaired corpora. We further show that by jointly training on paired data from one domain and unpaired text from another, we can obtain significant improvements in search performance in the latter domain, demonstrating the utility of joint training for text-only domain adaptation.

We see several avenues for future work to build on the work presented in this thesis:

(i) Our framework uses a dot product as the only interaction function between document and query embeddings. While this allows very efficient search, it also places a heavy burden on the document embeddings by forcing them to encode enough information about their neighborhoods to confidently make search decisions. Future work could improve the search accuracy by incorporating (possibly recurrent) decoders that allow more complex interaction between the query and document embeddings. To mitigate the resulting explosion in computational cost, a two-pass approach could be utilized, whereby a first pass based on the efficient method we have presented here retrieves a list of hypotheses from the full archive and the costly second-pass would only be run on this much reduced set.

(ii) Although we have limited our explorations so far to KWS, having a vector-space framework allows more intricate retrieval that would not be feasible in an ASR-based system. Slight modifications of the proposed network and objective function can allow other forms of spoken content retrieval such as (possibly multilingual) query-by-example, or searching by speaker or language identity within the same model.

(iii) The JOSTER method which we proposed for integrating unpaired text into training could be extended to improve end-to-end training of other spoken retrieval tasks, such as spoken question-answering, for which the size of extant text training data far exceeds the size of available paired speech data.

REFERENCES

1. Yusuf, B., J. Černocký and M. Saraçlar, "End-to-End Open Vocabulary Keyword Search With Multilingual Neural Representations", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, Vol. 31, No. 08, pp. 3070–3080, 2023.

2. Yusuf, B., A. Gok, B. Gundogdu and M. Saraclar, "End-to-End Open Vocabulary Keyword Search", *Proceedings of Interspeech*, pp. 4388–4392, Brno, Czechia, 2021.

3. Ng, K. and V. W. Zue, "Subword-Based Approaches for Spoken Document Retrieval", *Speech Communication*, Vol. 32, No. 3, pp. 157–186, 2000.

4. Saraclar, M. and R. Sproat, "Lattice-Based Search for Spoken Utterance Retrieval", *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pp. 129–136, Boston, Massachusetts, USA, 2004.

5. 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, Amsterdam, The Netherlands, 2007.

6. Szöke, I., M. Fapšo and L. Burget, "Hybrid Word-Subword Decoding for Spoken Term Detection", *Proceedings of The 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 42–48, Singapore, 2008.

7. Chelba, C., T. J. Hazen and M. Saraçlar, "Retrieval and Browsing of Spoken Content", *IEEE Signal Processing Magazine*, Vol. 25, No. 3, pp. 39–49, 2008.

8. Karakos, D. and R. Schwartz, "Subword and Phonetic Search for Detecting Out-of-Vocabulary Keywords", *Proceedings of Interspeech*, pp. 2469–2473, Singapore,

2014.

9. Huang, X., A. Acero, H.-W. Hon and R. Reddy, *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*, Prentice Hall PTR, USA, 2001.

10. Mohri, M., F. Pereira and M. Riley, "Weighted Finite-State Transducers in Speech Recognition", *Computer Speech & Language*, Vol. 16, No. 1, pp. 69–88, 2002.

11. 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.

12. Chelba, C., J. Silva and A. Acero, "Soft Indexing of Speech Content for Search in Spoken Documents", *Computer Speech & Language*, Vol. 21, No. 3, pp. 458–478, 2007.

13. Can, D. and M. Saraçlar, "Lattice Indexing for Spoken Term Detection", *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 19, No. 8, pp. 2338–2347, 2011.

14. Mohri, M., P. Moreno and E. Weinstein, "General Suffix Automaton Construction Algorithm and Space Bounds", *Theoretical Computer Science*, Vol. 410, No. 37, pp. 3553–3562, 2009.

15. Allauzen, C., M. Mohri and M. Saraclar, "General Indexation of Weighted Automata: Application to Spoken Utterance Retrieval", *Proceedings of the Workshop on Interdisciplinary Approaches to Speech Indexing and Retrieval at HLT-NAACL 2004*, pp. 33–40, Association for Computational Linguistics, 2004.

16. Parlak, S. and M. Saraclar, "Performance Analysis and Improvement of Turkish Broadcast News Retrieval", *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 20, No. 3, pp. 731–741, 2012.

17. Turunen, V. T. and M. Kurimo, "Speech Retrieval from Unsegmented Finnish Audio Using Statistical Morpheme-Like Units for Segmentation, Recognition, and Retrieval", *ACM Transactions on Speech and Language Processing*, Vol. 8, No. 1, pp. 1:1–1:25, 2008.

18. Su, H., V. T. Pham, Y. He and J. Hieronymus, "Improvements on transducing syllable lattice to word lattice for keyword search", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4729–4733, 2015.

19. Arisoy, E., D. Can, S. Parlak, H. Sak and M. Saraclar, "Turkish Broadcast News Transcription and Retrieval", *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 17, No. 5, pp. 874–883, 2009.

20. He, Y., P. Baumann, H. Fang, B. Hutchinson, A. Jaech, M. Ostendorf, E. Fosler-Lussier and J. Pierrehumbert, "Using Pronunciation-Based Morphological Subword Units to Improve OOV Handling in Keyword Search", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, Vol. 24, No. 1, pp. 79–92, 2016.

21. Karakos, D., I. Bulyko, R. Schwartz, S. Tsakalidis, L. Nguyen and J. Makhoul, "Normalization of Phonetic Keyword Search Scores", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7834–7838, Florence, Italy, 2014.

22. Can, D., E. Cooper, A. Sethy, C. White, B. Ramabhadran and M. Saraclar, "Effect of Pronounciations on OOV Queries in Spoken Term Detection", *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3957–3960, Taipei, Taiwan, 2009.

23. Chen, G., O. Yilmaz, J. Trmal, D. Povey and S. Khudanpur, "Using Proxies for OOV Keywords in the Keyword Search Task", *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 416–421, Olomouc, Czechia, 2013.

24. 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", *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 464–469, 2013.

25. Bai, Y., J. Yi, H. Ni, Z. Wen, B. Liu, Y. Li and J. Tao, "End-to-end Keywords Spotting Based on Connectionist Temporal Classification for Mandarin", *10th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, pp. 1–5, Tianjin, China, 2016.

26. Rosenberg, A., K. Audhkhasi, A. Sethy, B. Ramabhadran and M. Picheny, "End-to-End Speech Recognition and Keyword Search on Low-Resource Languages", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5280–5284, New Orleans, LA, USA, 2017.

27. Shi, G.-X., W.-Q. Zhang, G.-B. Wang, J. Zhao, S.-Z. Chai and Z.-Y. Zhao, "Timestamp-Aligning and Keyword-Biasing End-to-End ASR Front-End for a KWS System", *EURASIP Journal on Audio, Speech, and Music Processing*, Vol. 2021, No. 1, pp. 1–14, 2021.

28. Yang, R., G. Cheng, H. Miao, T. Li, P. Zhang and Y. Yan, "Keyword Search Using Attention-Based End-to-End ASR and Frame-Synchronous Phoneme Alignments", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, Vol. 29, No. 1, pp. 3202–3215, 2021.

29. 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.

30. Gundogdu, B., B. Yusuf and M. Saraclar, "Generative RNNs for OOV Keyword Search", *IEEE Signal Processing Letters*, Vol. 26, No. 1, pp. 124–128, 2018.

31. Yusuf, B., B. Gundogdu and M. Saraclar, "Low Resource Keyword Search with Synthesized Crosslingual Exemplars", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, Vol. 27, No. 7, pp. 1126–1135, 2019.

32. Sacchi, N., A. Nanchen, M. Jaggi and M. Cernak, "Open-Vocabulary Keyword Spotting with Audio and Text Embeddings", *Proceedings of Interspeech*, pp. 3362–3366, Graz, Austria, 2019.

33. Bluche, T. and T. Gisselbrecht, "Predicting Detection Filters for Small Footprint Open-Vocabulary Keyword Spotting", *Proceedings of Interspeech*, pp. 2552–2556, Shanghai, China, 2020.

34. Audhkhasi, K., A. Rosenberg, A. Sethy, B. Ramabhadran and B. Kingsbury, "End-to-End ASR-Free Keyword Search From Speech", *IEEE Journal of Selected Topics in Signal Processing*, Vol. 11, No. 8, pp. 1351–1359, 2017.

35. Zhao, Z. and W.-Q. Zhang, "End-to-End Keyword Search Based on Attention and Energy Scorer for Low Resource Languages", *Proceedings of Interspeech*, pp. 2587–2591, Shanghai, China, 2020.

36. Fuchs, T. S., Y. Segal and J. Keshet, "CNN-Based Spoken Term Detection and Localization without Dynamic Programming", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6853–6857, Toronto, Canada, 2021.

37. Švec, J., L. Šmídl, J. V. Psutka and A. Pražák, "Spoken Term Detection and Relevance Score Estimation Using Dot-Product of Pronunciation Embeddings", *Interspeech*, pp. 4398–4402, Brno, Czechia, 2021.

38. Knill, K. M., M. J. F. Gales, S. P. Rath, P. C. Woodland, C. Zhang and S. . Zhang, "Investigation of Multilingual Deep Neural Networks for Spoken Term Detection", *IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 138–

143, Olomouc Czechia, 2013.

39. Tüske, Z., P. Golik, D. Nolden, R. Schlüter and H. Ney, "Data Augmentation, Feature Combination, and Multilingual Neural Networks to Improve ASR and KWS Performance for Low-Resource Languages", *Proceedings of Interspeech*, pp. 1420–1424, Singapore, 2014.

40. Cui, J. *et al.*, "Multilingual Representations for Low Resource Speech Recognition and Keyword Search", *IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 259–266, Scottsdale, AZ, USA, 2015.

41. Karafiát, M., M. K. Baskar, P. Matějka, K. Veselý, F. Grézl, L. Burget and J. Černocký, "2016 BUT Babel System: Multilingual BLSTM Acoustic Model with I-Vector Based Adaptation", *Proceedings Interspeech*, pp. 719–723, Stockholm, Sweden, 2017.

42. Sercu, T., G. Saon, J. Cui, X. Cui, B. Ramabhadran, B. Kingsbury and A. Sethy, "Network Architectures for Multilingual Speech Representation Learning", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5295–5299, New Orleans, LA, USA, 2017.

43. Thomas, S., S. Ganapathy and H. Hermansky, "Cross-Lingual and Multi-Stream Posterior Features for Low Resource LVCSR Systems", *Proceedings of Interspeech*, pp. 877–880, Chiba, Japan, 2010.

44. 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", *Proceedings of Interspeech*, pp. 2586–2589, Portland, OR, USA, 2012.

45. Vu, N. T., D. Imseng, D. Povey, P. Motlicek, T. Schultz and H. Bourlard, "Multilingual Deep Neural Network Based Acoustic Modeling for Rapid Language Adapta-

tion", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7639–7643, Florence, Italy, 2014.

46. Scanzio, S., P. Laface, L. Fissore, R. Gemello and F. Mana, "On the Use of a Multilingual Neural Network Front-End", *Proceedings of Interspeech*, pp. 2711–2714, Brisbane, Australia, 2008.

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

48. Veselý, K., M. Karafiát, F. Grézl, M. Janda and E. Egorova, "The Language-Independent Bottleneck Features", *IEEE Workshop on Spoken Language Technology (SLT)*, pp. 336–341, Miami, FL, USA, 2012.

49. 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, Vancouver, Canada, 2013.

50. Rodriguez-Fuentes, L. J., A. Varona, M. Penagarikano, G. Bordel and M. Diez, "High-Performance Query-by-Example Spoken Term Detection on the SWS 2013 Evaluation", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7819–7823, Florence, Italy, 2014.

51. Szöke, I., M. Skácel, L. Burget and J. Černocký, "Coping with Channel Mismatch in Query-by-Example - BUT QUESST 2014", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5838–5842, South Brisbane, Australia, 2015.

52. Chen, H., C.-C. Leung, L. Xie, B. Ma and H. Li, "Multitask Feature Learning for Low-Resource Query-by-Example Spoken Term Detection", *IEEE Journal of*

Selected Topics in Signal Processing, Vol. 11, No. 8, pp. 1329–1339, 2017.

53. Ram, D., L. Miculicich and H. Bourlard, "Neural Network Based End-to-End Query by Example Spoken Term Detection", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, Vol. 28, No. 1, pp. 1416–1427, 2020.

54. Can, D. *et al.*, "Web Derived Pronunciations for Spoken Term Detection", *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, p. 83–90, New York, NY, USA, 2009.

55. Chen, G., S. Khudanpur, D. Povey, J. Trmal, D. Yarowsky and O. Yilmaz, "Quantifying the Value of Pronunciation Lexicons for Keyword Search in Low Resource Languages", *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8560–8564, Vancouver, Canada, 2013.

56. Gandhe, A., L. Qin, F. Metze, A. Rudnicky, I. Lane and M. Eck, "Using Web Text to Improve Keyword Spotting in Speech", *IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 428–433, Olomouc, Czechia, 2013.

57. Mendels, G., E. Cooper, V. Soto, J. Hirschberg, M. J. Gales, K. M. Knill, A. Ragni and H. Wang, "Improving Speech Recognition and Keyword Search for Low Resource languages Using Web Data", *Proceedings of Interspeech*, pp. 829–833, Dresden, Germany, 2015.

58. Rossenbach, N., A. Zeyer, R. Schlüter and H. Ney, "Generating Synthetic Audio Data for Attention-Based Speech Recognition Systems", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7069–7073, IEEE, Barcelona, Spain, 2020.

59. Wang, G., A. Rosenberg, Z. Chen, Y. Zhang, B. Ramabhadran, Y. Wu and P. Moreno, "Improving Speech Recognition Using Consistent Predictions on Syn-

thesized Speech", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7029–7033, Barcelona, Spain, 2020.

60. Baskar, M. K., L. Burget, S. Watanabe, R. F. Astudillo and J. H. Černocký, "EAT: Enhanced ASR-TTS for Self-Supervised Speech Recognition", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6753–6757, Toronto, Canada, 2021.

61. Renduchintala, A., S. Ding, M. Wiesner and S. Watanabe, "Multi-Modal Data Augmentation for End-to-end ASR", *Proceedings of Interspeech*, pp. 2394–2398, Hyderabad, India, 2018.

62. Wiesner, M., A. Renduchintala, S. Watanabe, C. Liu, N. Dehak and S. Khudanpur, "Pretraining by Backtranslation for End-to-End ASR in Low-Resource Settings", *Proceedings of Interspeech*, pp. 4375–4379, Graz, Austria, 2019.

63. Wang, P., T. N. Sainath and R. J. Weiss, "Multitask Training with Text Data for End-to-End Speech Recognition", *Proceedings fo Interspeech*, pp. 2566–2570, Brno, Czechia, 2021.

64. Yusuf, B., A. Gandhe and A. Sokolov, "USTED: Improving ASR with a Unified Speech and Text Encoder-Decoder", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8297–8301, Singapore, 2022.

65. Thomas, S., B. Kingsbury, G. Saon and H.-K. J. Kuo, "Integrating Text Inputs For Training and Adapting RNN Transducer ASR Models", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8127–8131, Singapore, 2022.

66. Chen, Z., Y. Zhang, A. Rosenberg, B. Ramabhadran, P. J. Moreno, A. Bapna and H. Zen, "MAESTRO: Matched Speech Text Representations through Modality Matching", *Proceedings of Interspeech*, pp. 4093–4097, Incheon, Korea, 2022.

67. Thomas, S., H.-K. J. Kuo, B. Kingsbury and G. Saon, "Towards Reducing the Need for Speech Training Data to Build Spoken Language Understanding Systems", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7932–7936, Singapore, 2022.

68. Tang, Y., J. Pino, X. Li, C. Wang and D. Genzel, "Improving Speech Translation by Understanding and Learning from the Auxiliary Text Translation Task", *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pp. 4252–4261, Bangkok, Thailand, 2021.

69. Huang, R., M. Wiesner, L. P. Garcia-Perera, D. Povey, J. Trmal and S. Khudanpur, "Building Keyword Search System from End-To-End Asr Systems", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, Rhodes, Greece, 2023.

70. Hochreiter, S. and J. Schmidhuber, "Long Short-Term Memory", *Neural Computation*, Vol. 9, No. 8, pp. 1735–1780, Nov. 1997.

71. Cho, K., B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation", *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar, 2014.

72. Dodson, D. S., "Corrigendum: Remark on "Algorithm 539: Basic Linear Algebra Subroutines for FORTRAN Usage"", *ACM Transactions on Mathematical Software*, Vol. 9, No. 1, p. 140, Mar. 1983.

73. Kingma, D. P. and J. Ba, "Adam: A Method for Stochastic Optimization", *Proceedings of the 3rd International Conference on Learning Representations, ICLR*, San Diego, CA, USA, 2015.

74. Segal, Y., T. S. Fuchs and J. Keshet, "SpeechYOLO: Detection and Localization of Speech Objects", *Proceedings Interspeech*, pp. 4210–4214, Graz, Austria, 2019.

75. Harper, M., "IARPA Babel Program", 2014, `https://www. iarpa. gov/index. php/research-programs/babel`, accessed on April 25, 2024.

76. Peddinti, V., D. Povey and S. Khudanpur, "A Time Delay Neural Network Architecture for Efficient Modeling of Long Temporal Contexts", *Proceedings of Interspeech*, pp. 3214–3218, Dresden, Germany, 2015.

77. Babu, A. *et al.*, "XLS-R: Self-supervised Cross-lingual Speech Representation Learning at Scale", *Interspeech*, pp. 2278–2282, 2022.

78. Baevski, A., Y. Zhou, A. Mohamed and M. Auli, "Wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations", *Advances in Neural Information Processing Systems*, Vol. 33, pp. 12449–12460, Vancouver, Canada, 2020.

79. NIST, "The Spoken Term Detection (STD) 2006 Evaluation Plan", 2014, `https://catalog.ldc.upenn.edu/docs/LDC2011S02/std06-evalplan-v10.pdf`, accessed on April 25, 2006.

80. NIST, "OpenKWS14 Keyword Search Evaluation Plan", 2014, `http://www.nist.gov/itl/iad/mig/upload/KWS14-evalplan-v11.pdf`, accessed on April 25, 2024.

81. Anguera, X., L.-J. Rodriguez-Fuentes, A. Buzo, F. Metze, I. Szöke and M. Penagarikano, "QUESST2014: Evaluating Query-by-Example Speech Search in a Zero-Resource Setting with Real-Life Queries", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5833–5837, South Brisbane, Australia, 2015.

82. 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, Amsterdam, The Netherlands, 2007.

83. 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", *Proceedings of Interspeech*, pp. 314–317, Antwerp, Belgium, 2007.

84. Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", *Journal of Machine Learning Research*, Vol. 15, No. 56, pp. 1929–1958, 2014.

85. Ko, T., V. Peddinti, D. Povey and S. Khudanpur, "Audio Augmentation for Speech Recognition", *Proceedings of Interspeech*, pp. 3586–3589, Dresden, Germany, 2015.

86. Ney, H., U. Essen and R. Kneser, "On Structuring Probabilistic Dependences in Stochastic Language Modelling", *Computer Speech & Language*, Vol. 8, No. 1, pp. 1–38, 1994.

87. Bisani, M. and H. Ney, "Joint-Sequence Models for Grapheme-to-Phoneme Conversion", *Speech communication*, Vol. 50, No. 5, pp. 434–451, 2008.

88. Panayotov, V., G. Chen, D. Povey and S. Khudanpur, "Librispeech: an ASR Corpus Based on Public Domain Audio Books", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5206–5210, South Brisbane, Australia, 2015.

89. Kahn, J. *et al.*, "Libri-Light: A Benchmark for ASR with Limited or No Supervision", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7669–7673, Barcelona, Spain, 2020.

90. Eren, G. and The Coqui TTS Team, "Coqui TTS", 2021, `https://github.com/coqui-ai/TTS`, accessed on April 25, 2024.