Low Resource Keyword Spotting

Guoguo Chen

Department of Electrical and Computer Engineering
Johns Hopkins University
Baltimore, Maryland

Primary Advisor: Sanjeev Khudanpur
Second Advisor: Daniel Povey

December 2014
# Contents

1 **Introduction** ................................................................. 2
  1.1 Problem Statement ............................................................ 2
  1.2 Existing Methods ................................................................ 3
    1.2.1 Query-by-Example Methods ........................................... 3
    1.2.2 Keyword/Filler Methods ............................................... 3
    1.2.3 Large Vocabulary Continuous Speech Recognition Methods ....... 3
  1.3 Low Resource Motivation ..................................................... 4
  1.4 Overview of the Proposal ..................................................... 4

2 **Proxy Keyword Search** ........................................................... 5
  2.1 Motivation ........................................................................... 5
  2.2 Proposed Method .................................................................. 5
  2.3 Preliminary Experiments ...................................................... 7
    2.3.1 Experiment Setup .......................................................... 7
    2.3.2 Initial Results .............................................................. 7

3 **Automatic Lexicon Generation** .................................................. 8
  3.1 Motivation ........................................................................... 8
  3.2 Proposed Method .................................................................. 8
  3.3 Preliminary Experiments ...................................................... 9
    3.3.1 Experiment Setup .......................................................... 9
    3.3.2 Initial Results .............................................................. 9

4 **Deep Neural Network Keyword Spotter** ...................................... 10
  4.1 Motivation ........................................................................... 10
  4.2 Proposed Method .................................................................. 10
  4.3 Preliminary Experiments ...................................................... 11
    4.3.1 Experiment Setup .......................................................... 11
    4.3.2 Initial Results .............................................................. 11

5 **Long Short-Term Memory Feature Extractor** .................................. 12
  5.1 Motivation ........................................................................... 12
  5.2 Proposed Method .................................................................. 12
  5.3 Preliminary Experiments ...................................................... 13
    5.3.1 Experiment Setup .......................................................... 13
    5.3.2 Initial Results .............................................................. 13

6 **Next Step: Phonetic Lexicon Free Keyword Spotting** .................... 14
Chapter 1

Introduction

Keyword spotting (KWS) is a task to automatically detect keywords of interest in continuous speech, which has been utilized in a broad variety of applications. A majority of such applications relate to audio indexing and speech data mining. For example, finding courses related to “speech recognition” from online lecture provider Coursera [14], where large mount of lectures are available in video format, instead of the traditional text format. Another portion of applications that is becoming increasingly important is the wake-up word, where keyword spotting is performed to activate a device or initiate a voice interaction interface. For example, Google’s voice search [55] features the keyword “Okay Google” where users can simply say “Okay Google” to initiate Google’s voice search. Other popular applications include phone call routing, phone call monitoring, voice command, to name just a few. In this proposal, we will explore various techniques to improve keyword spotting performance for audio indexing and wake-up word applications.

1.1 Problem Statement

Keyword spotting can be viewed as a subproblem of automatic speech recognition, where only partial information (the keyword) has to be extracted from speech utterances. While automatic speech recognition has been well formulated mathematically in literature, the formulation of keyword spotting has not been well established. Below we will describe the task in two stages: a spotting stage where the keyword spotter tries to collect the evidence that the keyword is present in the given utterances, and a decision stage where the keyword spotter takes the evidence collected in the spotting stage and makes a decision whether or not the keyword is present in the given utterance.

**Spotting stage.** Suppose $X \in \mathcal{X}$ is the input utterance, where $\mathcal{X}$ is the domain of the speech utterances (say the acoustic feature vectors of the speech signal). For simplicity, we assume $X$ has been truncated so that at most one keyword can appear in it. Suppose $K \in \mathcal{K}$ is the keyword of interest, where $\mathcal{K}$ is the domain of the keyword (say the text format of the keyword), the spotting stage is a function from $\mathcal{X} \times \mathcal{K}$ space to evidence space $\mathbb{R}^d$, parameterized by $\theta_1 \in \Theta_1$:

$$E_K = f_1(X, K; \theta_1)$$

(1.1)

where $E_K \in \mathbb{R}^d$ is a vector of evidence that keyword $K$ appears in the given utterance $X$. A common choice of evidence includes the posterior probability $P(K|X)$, the beginning time of keyword $t_{K_b}$ and the end time of keyword $t_{K_e}$. Therefore $E_K = [P(K|X), t_{K_b}, t_{K_e}] \in \mathbb{R}^3$. Other auxiliary evidence includes prior probability of keyword $P(K)$, duration of keyword $T_K$, etc.
Decision stage. The decision stage takes the evidence $E_K$ from the spotting stage, and gives a decision whether or not the keyword $K$ appears in the speech utterance $X$, along with the position of the keyword. Let the output $O_K$ be a triplet $O_K = (Yes/No, t_K^b, t_K^e)$, the decision stage is a function from $E_K$ to $O_K$, parameterized by $\theta_2 \in \Theta_2$:

$$O_K = f_2(E_K; \theta_2)$$

Combining the above two stages, the task of keyword spotting is then to learn a composite function $f_{KWS} = f_2 \circ f_1$ parameterized by $\theta = (\theta_1, \theta_2) \in \Theta = (\Theta_1, \Theta_2)$ so that the detections given by the function

$$O_K = f_{KWS}(X, K; \theta)$$

generate as little loss as possible, according to some pre-defined quality measure.

1.2 Existing Methods

Over the past 40 years, a lot of techniques have been proposed for keyword spotting. Below we try to summarize the literature in 3 categories.

1.2.1 Query-by-Example Methods

Query-by-example (QbyE) methods are among the earliest attempts for keyword spotting [4], and the name query-by-example is self-explaining: examples of keywords, usually exist in audio format, are used to spot the keyword. QbyE methods typically consist of two steps: a template representation step where audio examples of the keyword are represented as templates in certain format (posterior features, lattices, etc), and a template matching step where templates are compared with the target speech utterances which have been processed in the same format. Over the past decades, research focus of QbyE has been primarily on novel template representation methods [20, 80, 28, 25, 57, 46, 71, 66, 38], while most of those methods use some or other variant of dynamic time warping (DTW) [51] for template matching [7, 80, 79].

1.2.2 Keyword/Filler Methods

The keyword/filler method sometimes is also known as acoustic keyword spotting [42], which models keyword and non-keyword (filler) explicitly in parallel using subword units. At spotting stage, target utterances are aligned with both the keyword model and the filler model, and decisions will be made based on the alignment cost. In [50, 77, 78, 49, 76], Hidden Markov Models (HMMs) are used to model both keywords and fillers, and spotting is done by searching through a decoding graph where keywords and fillers appear parallelly. The latter research in this category more or less follows this framework, with focus on filler word modeling [32, 44, 58, 65] and advanced scoring procedure [61, 74, 2, 35]. Discriminative training methods have also been explored in this context to directly optimize the keyword spotting performance [62, 48].

1.2.3 Large Vocabulary Continuous Speech Recognition Methods

Large vocabulary continuous speech recognition (LVCSR) methods have been extensively used recently for applications such as audio indexing and speech data mining. In this case, speech utterances are transcribed into words with LVCSR systems, which will be further indexed for search [21]. The 1-best transcription from LVCSR systems usually contains errors which hurts the spotting stage. For better performance, confusion network [43, 23] or lattice [67, 75, 47] are commonly generated instead for indexing [1, 53, 6, 13]. One
drawback of LVCSR based keyword spotting methods is that the vocabulary of the LVCSR system is usually pre-defined, therefore if a keyword is out-of-vocabulary (OOV) for the LVCSR system, there is no way that the system can return anything for that keyword. Several techniques have been proposed to overcome the OOV problem, including subword modeling [45, 63], fuzzy search [12, 54], etc.

1.3 Low Resource Motivation

Despite of the fact that keyword spotting has been an active research topic for over 40 years, the low resource aspect of keyword spotting only catches attention recently. In academia, recent research activities such as the Johns Hopkins University’s 2012 summer workshop on zero resource speech technologies [31] and IARPA’s Babel program [29] propel the development of KWS technologies in low language resource conditions, whereas in industry, we also see rising demand of KWS technologies in low computation conditions [9, 10]. Those technologies are essential in certain applications. For example, as for year 2014, USC Shoah Foundation covers audio-visual testimonies from survivors and other witnesses of the Holocaust in 61 countries and 39 languages [69], and providing searching capacity for those testimonies requires substantial KWS technologies in low language resource conditions, as for most languages, speech recognition resources are not as rich as that for English. For another example, hands-free communication with mobile devices now becomes popular, where a KWS algorithm listens to the audio consistently and wakes up the device when pre-defined keywords are uttered (also known as wake-up words). This requires the KWS algorithm to run in a computation constrained condition, while being highly accurate, otherwise the KWS algorithm will quickly drain out the device’s battery.

LVCSR based KWS methods have been a major focus for KWS development in the past decade, due to their superior performance. However, those methods have a well known disadvantage: any words that are not defined in the LVCSR’s vocabulary will not have a chance to be found in the search collection, even if they do exist. This out-of-vocabulary (OOV) keywords problem becomes extremely severe when the training language resource is limited. For example, in [11] the author report that OOV keyword rate hits as high as 50% when they train their LVCSR system on a 10 hours training set. In this proposal, we will stay within the LVCSR framework when developing KWS techniques in low language resource conditions. We try to improvement the KWS performance by proposing new techniques that can handle OOV keywords in LVCSR based KWS methods.

LVCSR based KWS methods are also relatively expensive in terms of computation, when compared to other KWS methods, as full speech recognition will have to be performed in order to spot a keyword in the search collection. For example, for a single discriminatively trained subspace Gaussian mixture model (SGMM) system as described in [68, 39], the decoding of 25 hours of data takes 88.7 CPU hours with Intel Xeon Processor E7-4830 CPUs, not to mention the indexing and searching time. This type of methods are therefore not suitable for computation constrained devices such as mobile phones and tablets. In this proposal, we also propose KWS techniques that can work in low computation resource conditions, with the potential to serve as wake-up words for mobile devices.

1.4 Overview of the Proposal

This proposal explores the low resource keyword spotting problem in two directions: low language resource and low computation resource. In Chapter 2 and Chapter 3, techniques are proposed to improve keyword spotting performance in low language resource condition, with application to audio indexing. In Chapter 4 and Chapter 5, novel keyword spotting methods are developed for computation constrained devices such as mobile phones, with application to wake-up word. In Chapter 6 we briefly wrap up the proposal and list potential areas to explore for next step of research.
Chapter 2

Proxy Keyword Search

Large vocabulary continuous speech recognition (LVCSR) based keyword spotting (KWS) methods typically yield decent performance, but they also come with the well known disadvantage that only pre-defined vocabulary words can be found. This disadvantage becomes especially severe when the LVCSR training resource is limited. For example, one task in IARPA’s Babel program only uses 10 hours of training data for a particular language, and the keywords OOV rate reaches as high as 50% \[12\]. In this chapter, we propose a simple but effective weighted finite state transducer (WFST) based framework to handle the out-of-vocabulary (OOV) problem of LVCSR based keyword spotting methods, without re-training the system.

2.1 Motivation

Techniques have been proposed to tackle the OOV problem of LVCSR based keyword spotting methods. One way to minimize the OOV problem is to preemptively expand the the LVCSR lexicon. In other words, one adds automatically generated pronunciations of a large number of words in the LVCSR lexicon before lattice generation and indexation. In \[11\] it is shown that if one can anticipate the OOV keywords ahead of time, such a method leads to remarkable improvement in KWS performance. However, advance knowledge of all possible keywords is rarely the typical operating condition for KWS systems.

Another way to handle OOV keywords is via sub-word units such as phones, syllables or word-fragments. A sub-word index is created by either generating a sub-word lattice \[53, 59, 41\], or by converting a word lattice into a sub-word lattice with or without the use of an appropriate phone confusion matrix \[8\]. OOV keywords are represented as sequences of sub-word units, and matched against the sub-word index. Two sets of indices, therefore, have to be stored: one for in-vocabulary (IV) keywords and the other for out-of-vocabulary (OOV) keywords.

We would like to have a shared index for both IV and OOV keywords, without re-training the LVCSR system at the spotting stage. In this chapter, a weighted finite state transducer (WFST) based framework is proposed to generate acoustically similar IV keywords (referred to as proxy keywords) for OOV keywords at spotting stage, and those proxy keywords are used for spotting instead of the original OOV keywords, against the original word level index. We compare our method with a standard phone index, and show that our approach achieves better performance, without generating additional indices.

2.2 Proposed Method

Let $K$ represent a finite-state acceptor for an OOV keyword, and $L_2$ a finite state transducer for the pronunciation of the OOV keyword; e.g. pronunciations hypothesized via the joint-sequence model implemented in Sequitur software \[3\]. Let $E$ be an edit-distance transducer that maps a phone sequence to any other phone
sequence with costs estimated from a phone confusion matrix. Let $L_1$ denote the pronunciation lexicon of the LVCSR system. Our WFST procedure for generating a proxy keyword $K'$ may be described as,

$$K' = \text{Project}\left(\text{ShortestPath}\left(K \circ L_2 \circ E \circ (L_1^*)^{-1}\right)\right) .$$  \hspace{1cm} (2.1)

**Generation of $L_2$.** We use the G2P software Sequitur [3] to generate pronunciations for OOV keywords. The G2P model is trained with $L_1$, which we also use for LVCSR training.

**Edit-distance Transducer $E$.** Phone confusion pairs are obtained by decoding the LVCSR’s training data, and aligning the decoded output with reference. Those collected confusion pairs are then used to estimate phone confusion probabilities, with which we build the standard edit-distance transducer $E$, as shown in Figure 2.1.

![Figure 2.1: Example of a phone confusion encoding transducer $E$.](image)

**Modified Edit-Distance Transducer $E'$** To make search-by-word-proxy closer to phonetic search (improving recall), while still retaining lexical constraint on the phone sequence of $K'$, we create a modified edit-distance transducer $E'$ (Figure 2.2) which allows low cost insertions and deletions at word boundaries. Word proxies are thus generated as,

$$K' = \text{Project}\left(\text{ShortestPath}\left(K \circ L_2 \circ E' \circ (L_1^*)^{-1}\right)\right) .$$  \hspace{1cm} (2.2)

![Figure 2.2: Modified phone confusion encoding transducer $E'$, with freer edits permitted at the keyword-boundary.](image)
Generation of Phone Proxies. Since languages usually have a closed phone set, a KWS system based on a phone index can be considered as open vocabulary. Furthermore, as claimed in [34], adding phone confusion to either the index or the phoneme representation $K \circ L_2$ of the (OOV) keyword can help improve KWS performance. To generate such phone proxies in our framework requires only a minor modification of (2.1) as,

$$K'' = \text{Project} \left( \text{ShortestPath} \left( K \circ L_2 \circ E \right) \right). \quad (2.3)$$

Impact of Language Model Score. The proxy generation process (2.2) takes acoustic confusion into account, so that occurrences of $K'$ in the index are good candidates for actual occurrences of $K$. But the word lattices from which the index was created contain both acoustic and language model scores. The language model score for $K'$ is arguably not appropriate for comparing/sorting these occurrences. We will also evaluate the impact of retaining/discarding this score.

2.3 Preliminary Experiments

<table>
<thead>
<tr>
<th>Cond.</th>
<th>Index source</th>
<th>Proxy type</th>
<th>Uses $E$ or $E'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Phone lattice</td>
<td>Phone ($K''$)</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Phone lattice</td>
<td>Phone ($K''$)</td>
<td>Yes ($E$)</td>
</tr>
<tr>
<td>3</td>
<td>Word lattice</td>
<td>Word ($K'$)</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Word lattice</td>
<td>Word ($K'$)</td>
<td>Yes ($E'$)</td>
</tr>
<tr>
<td>5</td>
<td>Word lattice w/ no LM scores</td>
<td>Word ($K'$)</td>
<td>Yes ($E'$)</td>
</tr>
</tbody>
</table>

Table 2.1: Experiment conditions

Figure 2.3: ATWV versus the number of proxies per keyword.

2.3.1 Experiment Setup

Our dataset is Tagalog\(^1\), released by IARPA’s Babel program in Sprint 2013. We take the 10 hours of development data as our search collection. IV keywords are directly searched from the word index, OOV keywords are either searched with their word proxies (Equation 2.2) from the word index, or with their phone proxies (Equation 2.3) from the phone index. For details of our KWS pipeline, please refer to [12].

2.3.2 Initial Results

We conduct KWS experiments using keyword proxies for the 670 OOV keywords in 5 different conditions shown in Table 2.1. The first two conditions designate phone-based search without and with the use of phone proxies (Equation 2.3), the next two conditions designate word-based search without and with the use of word proxies (Equation 2.2), and the last designates word-based search with word proxies after ignoring language model scores in the lattices.

ATWV [19] performance in all 5 numbered conditions are plotted in Figure 2.3 as a function of the number of proxies. Initial results show that our proposed proxy method yields better performance than the standard phone index approach, and the fact that word index contains language model information does not impact much in terms of ATWV performance.

\(^1\)Language collection release babel106b-v0.2g-sub-train
Chapter 3

Automatic Lexicon Generation

This chapter also investigates the out-of-vocabulary (OOV) keyword problem of large vocabulary continuous speech recognition (LVCSR) based keyword spotting methods. Instead of trying to tackle the problem at the spotting stage, as we propose in Chapter 2, here we explore ways to fix the problem before training the LVCSR system. Given a limited number of vocabulary words in a certain language, we create a long list of hallucinated words and append them to the existing LVCSR vocabulary. Experiments suggest that the augmented vocabulary, with the potential to cover more actual words, helps with keyword search performance, although its impact on LVCSR’s word error rate is modest.

3.1 Motivation

In [11], the authors investigate the value of lexicon in terms of keyword spotting performance when the language resource is constrained. They show that if an additional list of words can be obtained, adding those words to the LVCSR vocabulary can yield remarkable improvement in terms of ATWV [19], although the impact on LVCSR’s word error rate is small.

In a real low language resource condition, however, such list of words is difficult to obtain. In this chapter, we propose a technique to automatically generate words from a limited size lexicon. We show that although only a small portion of the hallucinated words will be actual words in that given language, they do improve keyword spotting performance.

3.2 Proposed Method

Automatic Lexicon Generation. Our proposed lexicon generation procedure works as follows. First, for each word in the original lexicon, we take its pronunciation, and treat it as a sentence. We then build an ARPA language model for those sentences, and generate a large number (two million) of hallucinated “sentences” with the trained language model. We compute the language model probabilities for the hallucinated “sentences”, remove those that are already in the original lexicon, and keep one million best ones, which will become the pronunciations of augmented lexicon entries.

A lexicon entry needs a spelling as well as a pronunciation. We use the g2p tool from Sequitur [3] in reverse to produce the most likely spellings for each pronunciation (“sentence” as described above). We reverse the original lexicon by taking each lexicon entry, e.g. hi ʰ i y and reversing it to hi y ʰ i. It is not done exactly this way because we want iy to appear as a single symbol on the left, rather than as a sequence of two symbols. In fact, we map the phones to ASCII symbols first. After reversing the original lexicon, we train the g2p model, which can predict the spelling from the pronunciation.
Table 3.1: ATWV performance of using the original lexicon, using the original lexicon plus proxy keyword search, using the augmented lexicon, and using the augmented lexicon plus the proxy keyword search

<table>
<thead>
<tr>
<th>Language</th>
<th>Original</th>
<th>Original + Proxy</th>
<th>Augmented</th>
<th>Augmented + Proxy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Haitian</td>
<td>0.3804</td>
<td>0.4136</td>
<td>0.3966</td>
<td>0.4325</td>
</tr>
<tr>
<td>Lao</td>
<td>0.3648</td>
<td>0.3881</td>
<td>0.3625</td>
<td>0.3874</td>
</tr>
<tr>
<td>Assamese</td>
<td>0.2183</td>
<td>0.2423</td>
<td>0.2282</td>
<td>0.232</td>
</tr>
<tr>
<td>Bengali</td>
<td>0.2220</td>
<td>0.2539</td>
<td>0.2309</td>
<td>0.2773</td>
</tr>
<tr>
<td>Zulu</td>
<td>0.1041</td>
<td>0.1499</td>
<td>0.1462</td>
<td>0.2213</td>
</tr>
</tbody>
</table>

Combining the generated “sentences” from the language model, and the outputs from the reversed g2p, we create lexicon entries for the hallucinated words, which have a potential to be actual words in a given language.

Language Model for Hallucinated Words. It is shown in [11] that adding a language model for the augmented words usually yields much better keyword spotting performance than merely adding those words to the LVCSR vocabulary. Our hallucinated words, however, does not come with a language model training corpus. One way to solve this is to treat those hallucinated words as OOVs. We can estimate the OOV mass probability from a heldout set, and then assign unigram probability to the hallucinated words. Another way to tackle this involves class based language model [5, 72, 52]: probabilities of hallucinated words can be estimated from the actual words that are in the same class.

Combination with Proxy Keywords. Proxy keyword search [12] tries to solve the OOV problem of LVCSR based keyword spotting methods by adding phonetic confusions at the spotting stage: it generate acoustically similar words and use them as proxies of the original OOV word when searching. The proposed automatic lexicon generation method, on the other hand, adds the phonetic confusions at lexicon level, which should be complementary to proxy keywords. Besides, the increased number of vocabulary words also boost the number of proxy candidates for OOV keywords, which usually leads to improvement in keyword spotting performance.

3.3 Preliminary Experiments

3.3.1 Experiment Setup

We evaluate the impact of the automatically generated lexicon in the IARPA Babel Program (IARPA-BAA-11-02) framework, which has released conversational telephone speech corpora for several languages. In this study, we measure our system performance on Haitian\(^1\), Lao\(^2\), Assamese\(^3\), Bengali\(^4\) and Zulu\(^5\). We take the limited language pack (LimitedLP), which contains a 10 hour of training data, to simulate the low resource constrain. For details of our KWS pipeline, please refer to [11].

3.3.2 Initial Results

We take the 10 hour development set as our search collection. ATWV [19] performance of using the original lexicon, using the original lexicon plus proxy keyword search, using the augmented lexicon, and using the augmented lexicon plus the proxy keyword search are shown in Table 3.1. Note that proxy search is applied to the keywords that are out-of-vocabulary with respect to the original lexicon. Table 3.1 indicates that our proposed method gives positive improvement for four out of five languages.

---

\(^1\)Language collection release IARPA-babel1201b-v0.2b.
\(^2\)Language collection release IARPA-babel1203b-v3.1a.
\(^3\)Language collection release IARPA-babel1102b-v0.5a.
\(^4\)Language collection release IARPA-babel1103b-v0.4b.
\(^5\)Language collection release IARPA-babel1206b-v0.1e.
Chapter 4

Deep Neural Network Keyword Spotter

Large vocabulary continuous speech recognition (LVCSR) based keyword spotting (KWS) methods typically yield decent performance, but they are also computationally expensive [9, 10] since end-to-end speech recognition is performed in order to spot the keywords, thus not suitable when computation is constrained. In this chapter we describe a lightweight keyword spotting (KWS) system based on deep neural networks (DNNs), which is intended for wake-up word applications on computation constrained devices.

4.1 Motivation

Thanks to the rapid development of smartphones and tablets, interacting with technology using voice is becoming commonplace. For example, Google offers the ability to search by voice [55] on Android devices and Apple’s iOS devices are equipped with a conversational assistant named Siri. This increasing trend of voice interaction also sparks the demand of KWS algorithms on mobile devices: the device listens to the environment continuously and wakes up the voice interaction interface if a specific keyword is uttered, thus providing a fully hands-free experience. LVCSR based KWS methods are typically highly accurate when the training resource is adequate, but running LVCSR systems on mobile devices continuously are usually not practical as performing end-to-end speech recognition involves speaker adaptation, discriminative features, and model transformations [60], which are computationally expensive. We therefore propose a lightweight DNN based keyword spotting method (referred to as Deep KWS) that is suitable for mobile devices.

4.2 Proposed Method

The proposed Deep KWS framework is illustrated in Figure 4.1. The framework consists of three major components: (i) a feature extraction module, (ii) a deep neural network, and (iii) a posterior handling module. The feature extraction module (i) performs voice-activity detection and generates a vector of features every frame (10 ms). These features are stacked using the left and right context to create a larger vector, which is fed as input to the DNN. We train a DNN (ii) to predict posterior probabilities for each output label from the stacked features. These labels can correspond to entire words or sub-words for the keywords. Finally, a simple posterior handling module (iii) combines the label posteriors produced every frame into a confidence score used for detection.

Feature Extraction. To reduce computation, we use a voice-activity detection system [27] and only run the KWS algorithm in voice regions. For the voice regions, we generate acoustic features based on 40-dimensional log-filterbank energies computed every 10 ms over a window of 25 ms. 30 frames past frames and 10 future frames are used as context.
Deep Neural Network. For the proposed Deep KWS, the labels of the DNN can represent entire words or sub-word units in the keyword/key-phrase. We report results with full word labels, as these outperform sub-word units. These labels are generated at training time via forced alignment using our 50M parameter LVCSR system [30]. Our neural network is trained using the DistBelief framework [15].

Posterior Handling. Raw posteriors from the neural network are noisy, so we first do a smoothing step, by averaging the current posterior with the previous $n_{\text{smooth}}$ frames. We then take the maximum value for each label within a sliding window, except the garbage label, and compute the confidence score by taking a geometric mean of those maximum values.

4.3 Preliminary Experiments

4.3.1 Experiment Setup

We evaluate the proposed method on a set of 10 keywords, and report the performance in the form of a modified version of receiver operating characteristic (ROC) curves [9]. Our baseline is a standard keyword/filler HMM KWS system, but in contrast to previous work [50, 73], our implementation uses a DNN to compute the HMM state densities. Details of the keywords, dataset and baseline can be found in [9].

4.3.2 Initial Results

Figure 4.2 demonstrate the performance of the proposed Deep KWS system and the baseline system. “VS” in the parentheses means that the system is trained on general voice search data, “KW” means the system is trained with the keyword specified data, whereas “VS+KW” means both voice search and keyword specified data sets are used. From Figure 4.2 it is clear that the proposed Deep KWS outperforms the baseline HMM KWS system even when it is trained with less data.
Chapter 5

Long Short-Term Memory Feature Extractor

This chapter also focuses on wake-up word applications for computation limited devices. But unlike the method described in Chapter 4, where keywords cannot be changed after training the system, this chapter proposes a novel feature extraction technique for speech signals, and employs the query-by-example (QbyE) keyword spotting framework, making it possible to let the users define their own keywords.

5.1 Motivation

With the growing popularity of voice control in mobile devices, the need for high performance, small footprint, and low computational cost keyword spotting (KWS) methods is becoming increasingly important \cite{9}. In such applications, KWS usually serves as a frontier of voice search: it listens to the audio continuously and initiates the voice search if a specific keyword is detected, thus, providing a fully hand-free experience when interacting with devices.

A common use is to have a pre-defined keyword to activate devices. For example, Google’s voice search \cite{55} uses the phrase “Okay Google” to initiate the search interface and Apple’s conversational assistant Siri features the keyword “Hey Siri”. However, this general phrase makes the experience less personal, and usually requires additional speaker identification if the user does not want others to easily activate their device.

Query-by-example (QbyE) KWS methods detect a keyword by comparing the example keywords with the test audio, thus defining a new keyword only requires providing examples of this new keyword, which is favorable for user specified KWS systems. In this chapter, we stay in the QbyE framework, and propose a novel long short-term memory network based feature extractor that is suitable for on-device applications.

5.2 Proposed Method

The general idea of the proposed method is to embed audio segments of varying lengths into a fixed-dimensional representation. Given the success of deep learning \cite{26}, and the power of LSTMs for sequence modeling \cite{22}, we choose an LSTM to learn this embedding. The LSTM is attractive because the state of the LSTM can encode information about past history, and intuitively after processing a complete audio segment, this LSTM state encodes information about the complete sequence. This idea was motivated by face verification work in \cite{64}, with the key difference in our work being that we use the LSTM state to embed a fixed-length representation of a variable length input sequence, as opposed to having a fixed-length input.
representation and thus using a convolutional neural network.

Our proposed LSTM KWS system is illustrated in Figure 5.1. In the enrollment phase, an utterance is spoken three times. For each utterance, the activations from the last hidden layer of the LSTM are calculated per frame, and the last \( k \) activations are used to create a fixed feature vector \( f \). Note that since we have three enrollment templates, we can keep all the three as separate templates, or average them into one single vector.

At runtime, another LSTM feature vector is generated in the same way for the sliding window, and Cosine distance is used to measure the similarity between the keyword template(s) and the sliding window. Decisions are made based on the similarity score.

![Figure 5.1: Framework of the LSTM KWS system.](image)

5.3 Preliminary Experiments

5.3.1 Experiment Setup

We evaluate the proposed method on a set of 8 keywords, and report the performance in the form of a modified version of receiver operating characteristic (ROC) curves [9]. Our baseline is a standard phone posteriorgram + dynamic time warping (DTW) QbyE method [80, 25], where phone posteriorgrams are generated by either a feed forward deep neural network (DNN) or a long short-term memory (LSTM) network. For details of the keywords, dataset and baseline, please refer to [10].

5.3.2 Initial Results

Figure 5.2 shows the performance of the proposed LSTM Feature Extractor and 2 DTW KWS systems. “Babble” in the parentheses means we manually add 10dB of bable noise to the evaluation set, whereas “Clean” means no artificial noise is added. Under both the clean and noisy conditions, we can see that the Phone LSTM + DTW (Babble) system is better than the Phone DNN + DTW (Babble) system, while the LSTMFeat Extractor (Babble) system outperforms both of them.

![Figure 5.2: LSTM feature extractor v.s. baseline](image)
Chapter 6

Next Step: Phonetic Lexicon Free Keyword Spotting

The techniques we propose in Chapter 2 and Chapter 3 both rely on an expert created phonetic lexicon. First of all, the underlying large vocabulary continuous speech recognition (LVCSR) system on top of which we build the keyword spotting (KWS) system typically requires a phonetic lexicon. Second, phonetic lexicon is also used in Chapter 2 to generate proxy keywords for OOV keywords and in Chapter 3 to hallucinate new words. This is however a little bit ironic in a practical low language resource condition: the creation of a phonetic lexicon in a certain language usually requires a large amount of time of a skilled linguist, while such expert is scarce oftentimes for a low resource language. We therefore would like to explore LVCSR based KWS techniques without an expert created phonetic lexicon as the next step of our research. We will assume that a small amount (say 10 hours) of transcribed speech data will be available, which can be collected relatively easily without linguist effort, and we will investigate the following directions.

**Grapheme systems.** Grapheme systems have been introduced to automatic speech recognition (ASR) due to the lack of phonetic pronunciation lexicons in certain tasks [56, 33, 17, 18, 16]. In such systems, graphemes are used as subword units and lexicons are simply the orthographic transcription of the words. Those techniques have also been applied to LVCSR based KWS systems [70, 24]. Typically grapheme based systems yield worse word error rate than phonetic lexicon based systems, but it will be interesting to see how such systems perform for KWS in a low language resource condition.

**Data-driven lexicon discovery.** Data-driven methods have also been explored to discover pronunciation lexicon automatically [36, 37, 40]. In [37], a hierarchical Bayesian model is proposed to learn the phonetic inventory and the letter-to-sound (L2S) mapping jointly. In [40], a seed lexicon is required to train a grapheme-to-phoneme (G2P) model, which will then be used to create pronunciations for other words that are not in the seed lexicon. All those work tries to create a pronunciation lexicon from the data without expert effort. We will try data-driven methods to create pronunciation lexicon for our KWS systems as well, to minimize linguist effort for KWS methods in low language resource condition.
References


