# DOMAIN-SPECIFIC KEYWORD SPOTTING

by

Ali Haznedaroğlu

B.S., Electrical and Electronic Engineering, Bilkent University, 2004

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

Graduate Program in Electrical and Electronic Engineering

Boğaziçi University

2007

# ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor Levent Arslan, for all of his support and guidance not only in this thesis, but in my entire life.

I would like to thank Murat Saraçlar for his great advices, and support, for being in my thesis commitee; and also to Hakan Erdoğan for kindly accepting my invitation to be in my thesis commitee, and for spending his valuable time and energy in it.

I would also like to thank everybody at Sestek, and especially Osman Büyük for all his support in speech recognition stuff.

"A friend is one who believes in you when you have ceased to believe in yourself." Thanks to all my friends, and especially to the ones who were always with me throughout this period: Seçkin, Aysu and Yiğit.

Of course, and mostly, I would like to thank my family, for everything. I can't express my feeling how I wish my aunt, Füsun Sayek, was able to see me getting this M.Sc. degree. All of this is dedicated to her.

# ABSTRACT

## DOMAIN-SPECIFIC KEYWORD SPOTTING

In large vocabulary continuous speech recognition based keyword spotting applications, language modeling directly affects the system performance. In this study, a keyword adapted language model is proposed and a Turkish keyword spotting system is implemented. The proposed language model is compared with two other language models: null-grammar language model as the base model, and a general bigram model. Experiments show that keyword adapted language model gives the best performance in both recall and spotting time. Highest recall rate is 86 per cent. The model that we propose has absolutely increased the recall performance by 4 per cent from the general bigram language model, and by 13 per cent from the null-grammar language model. However, it gives lower precision results than the other systems. Two different methods are used in order to increase the precision of the system. The first method is language model interpolation, which increases the precision, but also decreases the recall. The second method is word insertion penalty adjustment. It is shown that length-adapted adjustment of the word insertion penalty can increase the overall system performance. Finally, a GUI-based computer program that uses the proposed language model is designed and implemented.

# ÖZET

## ALANA ÖZEL ANAHTAR KELİME YAKALAMA

Geniş dağarcıklı sürekli konuşma tanıma bazlı anahtar kelime yakalama sistemlerinde kullanılan dil model, performansa büyük ölçüde etki etmektedir. Bu çalışmada, anahtar kelimelere uyarlı bir dil modeli önerilmiş ve Türkçe bir anahtar kelime yakalama sistemi oluşturulmuştur. Önerilen model iki farklı dil modeliyle karşılaştırılmıştır. Bunlar tekli ve ikili dil modelleridir. Üçüncü ve bizim önerdiğimiz sistem ise, geniş bir Türkçe veritabanından, içinde anahtar kelimelerin geçtiği cümlelerin çekilip, bu cümlelerde en çok geçen kelimelerle oluşturulan dil modelini kullanan, yine SMM bazlı sistemdir. Deneyler sonucunda önerdiğimiz anahtar kelimeye uyarlı dil modeli, diğer modellere göre yakalama oranı ve yakalama süresinde daha iyi performans vermiştir. Ulaşılan en yüksek yakalama oranı yüzde 86'dır. Önerdiğimiz sistem yakalama oranında, ikili dil modeline göre yaklaşık yüzde 4, tekli dil modeline göre de yaklaşık yüzde 13 daha fazla mutlak başarı göstermiştir. Kesinlik kriterinde ise önerilen model diğer iki modelin gerisinde kalmıştır. Kesinliği arttırmak için iki yöntem denenmiştir. Bunlardan ilki dil modeli aradeğerlemesidir. Bu yöntem kesinliği arttırsa da, yakalama oranını düşürmüştür. İkinci yöntem ise kelime ekleme cezasının ayarlanmasıdır. Anahtar kelimelerin uzunluğuna göre ayarlanan kelime ekleme cezasının sistem performansını arttırdığı gözlemlenmiştir.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF SYMBOLS / ABBREVIATIONS

| | |
|---|---|
| $A$ | Acoustic data vector |
| $a_i$ | State transition probability distribution |
| $b_j(\mathbf{y}_t)$ | Observation vector probability distribution in state j |
| $c_i$ | Mel-Frequency Cepstrum Coefficient |
| $H(x)$ | Entropy |
| $w_i$ | The ith word in a word sequence |
| $w_{l,k}$ | The word sequence $w_k\ w_{k+1} \cdots w_{l-1} w_l$ |
| $W$ | Word sequence |
| $\pi_i$ | Initial state distribution |
| $\mu_{jk}$ | Gaussian Mixture Mean Vector |
| $\Sigma_{jk}$ | Gaussian Mixture Covariance matrix |

| | |
|---|---|
| ASR | Automatic Speech Recognition |
| CDR | Call Detail Records |
| CSR | Continuous Speech Recognition |
| DCT | Discrete Cosine Transform |
| HMM | Hidden Markov Model |
| GUI | Graphical User Interface |
| KWS | Keyword Spotting |
| LM | Language Model |
| LVCSR | Large Vocabulary Continuous Speech Recognition |
| MFCC | Mel-Frequency Cepstrum Coefficient |
| OOV | Out of Vocabulary |
| SVM | Support Vector Machine |
| TID | Topic Identification |
| WIP | Word Insertion Penalty |

# 1. INTRODUCTION

"Information is the oxygen of the modern age" once said Ronald Reagan; and it was only the beginnings of the information age when he said this. Today, the situation is more dramatic than that. We live in an age that is totally driven by information.

As the digital revolution is in progress, storage and transmission of information becomes more available everyday. Today, most of the equipment that we use, speaks in just one language, a language that contains just two words: 0 and 1. As this new and digital world is united by one language, communication and exchanging information is easier. Telephones, televisions or other devices can easily connect to computers and exchange information. Internet can connect the computers that are ten thousands of kilometers away. Today, there is more data available than ever, but it is much less than tomorrow.

In spite of its vital role in our lives, information is useless unless it is properly organized, processed, and retrieved. We have large amounts of data in our hands, and as Fred Menger said, "If we torture data sufficiently, it will confess to almost anything". So firstly, we must know how to torture, or more politely, how to treat data, which is more scientifically called "information retrieval".

Information retrieval is searching information in documents. Information can be retrieved from different number of sources, like text, audio, video etc. In this thesis, we focus on a more specialized case, which helps in retrieving information from speech sources. It is called "Keyword Spotting" or KWS in short.

With the improvements in the speech recognition technology, and computer capabilities, speech recognition based applications are used more in practical and commercial applications. One of the popular applications is keyword spotting, which uses a speech recognition engine underneath. The aim of a keyword spotting is identification of keywords from spoken utterances. Keyword spotters are mainly integrated with telephony systems, and used in automated operator services, telephony database queries etc. One very

common use is in call centers, where they are used to retrieve information from the calls, to detect customer satisfaction etc.

Although the research on keyword spotting has been done for many years, practical KWS applications are recently emerging, as need for them, and their performance increase. So, the need for better KWS systems still exists.

## 1.1. Problem Definition

Keyword spotting deals with detecting a set of selected keywords from unconstrained speech, using a speech recognition engine. Like most of the today's conventional speech recognizers, keyword spotters are mostly Hidden Markov Model (HMM) based where small speech units called phonemes are acoustically modeled using HMMs. Apart from acoustic modeling, in this thesis we investigate the effects of language modeling on the spotting performance, and we will propose a new keyword-adapted language model. Keyword spotting will be done on Turkish recordings.

Various keyword spotting approaches can be found in the literature. Two main approaches are word recognition based KWS, and large vocabulary continuous speech recognition (LVCSR) based KWS. In word recognition based KWS, only the keywords are modeled in detail, and a generic model is used for the non-keywords. In LVCSR based KWS, both the keywords and the non-keywords in the recognition vocabulary are modeled in great detail, and the input speech is transcribed into text. Then a textual search is performed to detect the keywords.

In this thesis, we follow the LVCSR based approach. LVCSR based KWS systems have the advantage of good detection rates, and they allow fast and easy research of the keywords as the speech is transcribed. Null-grammar language models are mainly used in LVCSR systems. We propose a new, keyword-adapted language model for KWS in this thesis. A keyword-specific text is created from a general text-corpus, and then this text is used to train the bigram keyword-adapted language model. Performance of the KWS that uses our proposed model will then be compared to the performance of two other

implemented spotters: one using a null-grammar language model (our base model), and the other one using a general bigram language model.

A GUI-based computer program will also be designed and implemented as a practical application that does KWS using our proposed keyword-adapted language model.

## 1.2. Previous Research

### 1.2.1. Early KWS Approaches

As it is mainly derived from automatic speech recognition (ASR), KWS has followed a similar development process as ASR. Early KWS approaches include dynamic programming based word template matching [1]. Word template matching is the simplest of the speech recognition approaches, in which the input speech is matched to previously determined patterns or templates. Words in the recognition vocabulary are converted into number of templates by spectrum analysis, and each template and the input speech sequence are divided into frames, which are used as the basic matching units. A distance metric is used to measure the similarity between the input and template frames, and the evaluation of the input speech is done by adding up the distances between the input frames and the template frames. Nearest template sequence to the input sequence can be found by searching over all possible template sequences, and dynamic programming is used in the search stage to efficiently find the best sequence of templates for the input speech [2]. Best sequence will have the smallest distance score to the input sequence.

### 1.2.2. CSR based KWS Approaches

Improvements in the statistical approaches to continuous speech recognition (CSR) make these approaches also popular in KWS. Statistical methods such as dynamic time warping are used in some works [3], but as also in CSR, Hidden Markov Models (HMMs) are the most popular approach in KWS. The work of Rohlicek, *et al*, is the first attempt for HMM based KWS. In this work [4] a KWS system that uses HMM is presented, and the

effects of covariance structure of the Gaussian models, feature sets and feature transformations on KWS performance are investigated. Whole-word models are used to represent the keywords, and an alternative model is used to represent non-keywords. It is concluded that including derivatives and the $0^{th}$ cepstrum coefficient in the feature set improves the spotting performance, but using full or diagonal covariance matrix for Gaussian models does not affect the spotting performance as much.

In his work [5], Rose presents a CSR based KWS system that uses HMMs. Mainly; dealing with non-keyword speech by the use of filler models is investigated. By using HMM based filler models, acoustical information over different speakers and word contexts can be assimilated, so that using HMMs for the arbitrary non-keywords speech is advantageous. Keywords are modeled with triphones, which are context dependent sub-word units, and both monophone and triphone models are trained for the filler models. Spotting performance is better for the simpler monophone filler models, and it is found that training the filler models from transcribed speech is beneficial. Dealing with linear channel effects is also investigated in this work. It has been shown that using an acoustic class dependent spectral normalization can compensate the channel effects caused by inter speaker and speaking medium variability. Thus, it increases the spotting performance.

In [6], a trial of a KWS technology has been done on customer responses over a long distance network. Tests show that if the customer speaks in isolation, an HMM-based isolated word recognizer gives almost perfect results of 99.3 per cent recognition rate. However, it's seen that, only 20 per cent of the utterances spoken by the customer have the desired keywords, and the rest is non-keywords. This work describes the modifications that made to an HMM based connected speech recognizer to handle KWS in unconstrained speech. The approach is based on creating statistical models for the non-keywords and background together with the keywords, and then find best hypothesis by using a connected word recognizer. The work is also concentrated on what type of speech data can be used to train the garbage models, and it is shown that only a few number of garbage HMMs are enough to achieve high performances. 95 per cent correct recognition rate is reached for 5 keywords, when the customer speaks short utterances like "I want to make a collect call", where "collect" is the keyword. This shows that KWS technology can be used in telecommunication services such as automatic operator services. In [7], some

improvements are made to optimize this system for telecommunication applications. Also a brief comparison for different KWS applications has been made.

Weintraub's work [8] can be considered as the first real LVCSR based KWS system as it uses a bigram language model. In this work, benefits of using language model in KWS, a method for increasing the figure-of-merit of the system and a method of rapid porting to new keyword vocabularies is investigated. Two different KWS topologies are tested. First topology, which is a traditional CSR solution, uses a fixed vocabulary of the keywords and the N most common words in the task, which is a traditional CSR solution. In the second topology, a background model is added to the keyword list, so that parts of the speech are transcribed as background. It's seen that adding a background word model decreases the recognition rate, as some keywords are transcribed as background.

Weintraub's work is the first KWS application that uses a bigram language model. Bigram language model probabilities are estimated using Katz's back-off. The recognizer is forced to choose among the allowable words, which are the keywords and the most common words in the task. In this topology, non-keywords are better modeled than the previous approaches, and it yields a significantly improved KWS performance. 27.1 per cent increase in FOM is obtained when this topology is used in place of the previous topologies which use only garbage models to represent non-keywords. Also an algorithm that smoothes the bigram language probabilities is introduced. This algorithm provides reduction in the test set perplexity.

### 1.2.3. Adaptive Language Modeling

In this work, adaptive language modeling will be used in KWS as the language model will be adapted to the domain of the selected keywords. Adaptive language models are used in previous CSR applications, and they improve the performance over the static N-gram models.

N-grams are very simple static language models which only consider the n-1 previous words while predicting the next word. N-grams are very sensitive to the domain

of their training text corpus. Adaptive language modeling uses contextual and longer distance information together with the static N-gram models to improve the performance. It is the adaption of the language model to the domain of the current document.

In Jelinek's work [30], language model is adapted to the current document using the partially dictated document, which is called caching. N most recently dictated words are used to build unigram, bigram, and trigram frequency distributions, and then these distributions are linearly smoothed to obtaion the dynamic tirgam model. Then this dynamic model is interpolated with the static trigram model to obtain the adaptive language model, which yields 23 per cent decrease in perplexity.

Another approach that uses the partially dictated document is the work by Pietra, *et al* [31]. In this work, a unigram distribution is calculated from the partially dictated document, the using the discriminant information distance measure the closest n-gram distribution to the static model is found, and used. Partially dictated document can also be used in the identifying the domain of the speech, and then the domain-specific language model can be used. In this case, the unigram distribution of the dictated speech is compared with the predefined domains' distributions to determine the domain of the speech.

Taking the related words, or the *triggers,* into account can be another adaptive approach. Some words that occured in the document increase the occurance probability of their related words. Triggers can again be used to determine the domain of the speech. A trigger-based adaptive language model approach can be found in [32].

In a more recent work [33], a topic-adaptive language model that is created by using a multi class support vector machine (SVM) - based topic classifier has been developed for a broadcast news transcription system. First, the universal language model is used for topic clasification, and based on the result of the classification, an on-topic language model, that was classified using a SVM, is selected, and interpolated with the universal model.

### 1.2.4. Turkish KWS

Works of Duran [9], and Yapanel [10] are two examples of Turkish KWS works. In [9], context independent phones as garbage models are examined, and their performances are evaluated. Three different spotters using different HMM-based topologies are tested. The first spotter uses a triphone model of the keyword, and one garbage model; the second spotter uses triphone and monophone model of the word; and the third spotter combines them by using triphone and monophone model of the word and one garbage model. The reasoning for using the last model is as follows: If a keyword is modeled both by triphones and monophones, its triphone model should get the best recognition score if the keyword exits. If the keyword doesn't exist, its monophone model or the garbage model should get the highest recognition score. This is because monophone models have greater variances and they can model all phonemes more generally. It's seen that, the combined model gives better recognition results, and it is concluded that context independent monophones are good candidates for the garbage models.

In [10], spotters using different garbage models are constructed, and their performances are evaluated. Constructed spotters are as follows: spotter with one garbage model, spotter with phone class garbage models a spotter with context independent garbage models. In the first spotter, the recognition network is constructed from the keywords, a garbage model, and a silence background model. As the garbage model will take lower likelihood scores than the keywords, an additional bonus is added to it. In the second spotter, number of the garbage models is increased to increase the detail of garbage modeling. In this spotter, phones are clustered as nasals, glides, etc, and different garbage models are built for each of them. So now the recognition network contains the keywords plus a number of garbage models. As the acoustic detail of the system increases with the usage of more garbage models, this spotter is expected to improve the performance of the first spotter. Third spotter uses garbage models which consist of context-independent monophones as in [5]. Best results are obtained for the third spotter which uses context-independent monophones.

In [10], a LVCSR based keyword spotter is also implemented and tested. In this spotter, a null-grammar recognition network is built for the keywords, and all the words in

the test set. As the acoustic detail of the system is higher than all of the previous systems, this system gives the best spotting results, and Yapanel conclude that no other KWS system can outperform an LVCSR based KWS system. Although its good recognition rates, this spotter is found to be unpractical because if its long computation time.

### 1.2.5. Turkish LVCSR Researches

As an LVCSR based Turkish KWS system will be implemented in this thesis, it's good to examine the previous researches on Turkish LVCSR systems. Because of the agglutinative nature of the Turkish language, vocabulary explosion and large number of out-of vocabulary words cause problems in Turkish LVCRS systems. In [11], properties of the Turkish language are examined. Then some speaker-independent Turkish LVCSR experiments is done, and a morpheme based solution is proposed to overcome the OOV problem. Although OOV rate decreases in the morpheme-based approach, word error rate (WER) increases.

In [12], four language models are compared for Turkish LVCSR. Compared models are whole-word model, morpheme-based model, stem-ending-based model, and syllable-based model. These models are compared according some criteria like their recognition performance, OOV rate, unit decomposition difficulty, sensitivity to context etc. In spite of their easy decomposition, and low OOV rate, syllable models are found to be useless in Turkish speech recognition because of their very low recognition rate.

In [13], a Turkish broadcast news dictation system is designed and different recognition units are proposed. Words are parsed into stems, endings, and morphemes as in [12] and these units are tested to overcome the high OOV rate problem. Two different language models are tested. First one is a word-based model, in which the whole words are the lexicon entries. Other model is a combined model, in which three models are combined. First model in the combination is the whole-word model that was explained previously. Second model is a morpheme-based model in which the words are decomposed into their stems and morphemes, and these units are taken as the lexicon entries. Third model is a stem-ending-based model in which the words are decomposed into their stems

and endings, and these units are taken as the lexicon entries. Combined model decreases the OOV rate, but its WER is higher than the whole-word model.

As smaller sub-word units result in acoustic confusability, in [14], an algorithm that uses longer sub-word units is proposed. Half-words and full words are chosen as the recognition unit, and it is seen that half-word bigram language models yields a significant reduction in the WER compared to the whole-word bigram language model. Also a Turkish language constraint, called vowel harmony, is incorporated into the language model with a weighted finite state machine and it is combined with a trigram language model. WER further decreases when this combination is used.

### 1.2.6. Other Researches That Include KWS

Now we shall examine research in some other areas that include KWS. Topic identification (TID) is a task in which keyword spotters are often used. TID is a classification problem of the topic in a document. With the digital storage of the TV or radio broadcast, TID from speech segments becomes an important task, which needs to be done rapidly and easily.

In [15], a KWS based topic recognizer for Japanese TV news is implemented. An HMM based keyword spotter is used with two linguistic constraints. First constraint allows any phoneme sequences and in the second constraint it is supposed that the last recognized word is a keyword. Keywords are chosen regarding to their mutual information about a topic, and their length. As shorter words can be inserted more easily with small acoustic similarities, longer words can be better spotted. In this application 3000 nouns which are longer than 6 phonemes are chosen as the keywords. TID is done by calculating the topic possibilities in terms of the spotted keyword's acoustic likelihood score, and the topic probability of the word. Bias of the topics in the keyword set is removed to neutralize the effect of wrongly inserted keywords. TID rate of 66.5 per cent is obtained if the identified topic is included in the top three topics.

In [16], some experiments are done on the Switchboard Corpus [17] using various TID approaches. Both CSR and KWS are investigated in this work and it is concluded that using KWS in TID applications gives better results. This work also deals with the selection of the keywords for the TID task, and it's showed that rather than selecting the keywords only from a previously transcribed text, it's better to select the keywords from the output of the keyword spotter. A keyword that is selected from a text can be a bad choice for the keyword spotter as its detection rate may be low, or it may result in high number of false alarms because of its acoustic properties. Selecting the keywords from the output of the spotter overcomes these problems.

In [18], another TID system on TV broadcast news is implemented. The system uses an HMM based keyword recognizer which employs a normalized Viterbi algorithm that computes the word likelihoods by just forward passing to decrease the computation time. TID is done by integrating the acoustic keyword probabilities and a priori knowledge probability which is called topic contribution of a keyword.

Another research area that includes KWS is audio or video indexing. With the development in the multimedia applications, amount of audiovisual content increases day by day, and this content needs good indexing for fast and easy retrieval. This indexing can be done by KWS. In [19], a video soundtrack indexing solution by using HMM based KWS is proposed, and some constraints for the application are identified. In [20], several approaches to overcome the OOV problem in spoken document indexing are investigated.

### 1.3. Practical Applications

Although KWS systems are used in various types of applications like topic identification, or audiovisual indexing as discussed before, there are two main markets that KWS is used. These are call center and homeland security markets [21]. These two markets will be briefly discussed.

### 1.3.1. KWS in Call Center Applications

Countless number of calls are made in call systems per day. In modern call centers, these calls are digitally recorded and stored to mine some valuable information from the calls afterwards. This valuable information can be the customer satisfaction about a certain service, or it can give an idea about the quality assurance process of the call center. Call agents should be perfectly trained as they interact with the customers, and their behaviors to the customers should be monitored perfectly.

Most of the information about the calls come from the Call Detail Records (CDR) which contains information about the source/destination of the call, duration of the call, the amount billed for the call etc. Although the calls are digitally stored, it's too hard to dig information from their contents manually. Monitoring of the calls by humans is very time-consuming and expensive. Humans can only monitor some of the randomly selected calls, but the number of these monitored calls is only a very small ratio of the total number of calls.

By using KWS, all of the calls can be monitored and important information can be retrieved easily. Suppose a call center that receives customer calls about a service, and observes the customer satisfaction about the service. Let's say that the customers will be classified into two satisfaction groups: the ones who are happy about the service and the ones who are complaining about the service. So we have two concepts, and first we should choose keywords related to these concepts. As an example, Table 1.1 shows these two concepts and a number of keywords for each of them.

| Happy | Complainant |
|---|---|
| Works fine | Complaint |
| Great service | Want to quit |
| Useful | Waste of money |
| Thank you | Damn It |

Table 1.1. Concepts and Keywords

Using KWS, we can automatically detect these keywords in the calls, and we can determine the customer satisfaction level according to which concept has the most number of detected words. Some weights can be given to the keywords to make better and more reasonable decisions. For example “want to quit” will be weighted more heavily than “thank you”, so that a call in which the customer wants to quit the service, and ends the conversation with “thank you” will be classified as a complaint call.

### 1.3.2. KWS in Homeland Security Applications

Government investments in homeland security are raising everyday. Fear of new terror attacks forces governments to invest money up to 72 billion dollars a year [21]. KWS allows searching of some keywords, like the name of the terrorist groups etc, in telephone calls to decide which calls should be monitored more carefully by human listeners.

## 1.4. Outline of the Thesis

This thesis is organized as follows: In Chapter 2, a brief speech recognition overview is presented. In Chapter 3, language models that are used in LVCSR based KWS are discussed, and the proposed model is presented. Chapter 4 describes the framework for

the experiments, and the KWS experiments are presented in Chapter 5. The conclusion is made and some future work is discussed in the final chapter. Screen captures of the GUI-based computer program that is implemented is given in Appendix A.

# 2. SPEECH RECOGNITION

## 2.1. Introduction

Speech recognition is the basis of today's conventional keyword spotting applications. Generally, output of a speech recognition module is processed to find the searched keywords. Therefore in this chapter, fundamentals of speech recognition will be briefly explained. Diagram of a typical speech recognizer is shown in Figure 2.1.

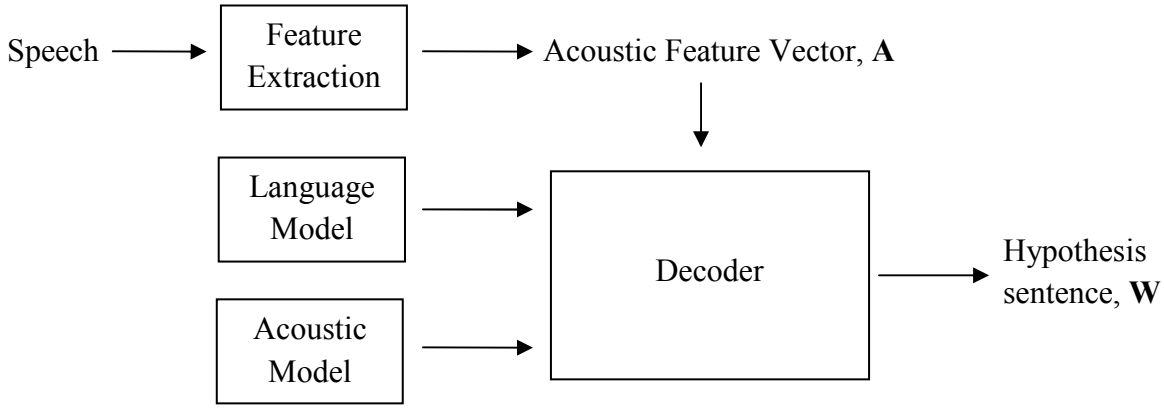


Figure 2.1. Block Diagram of a Speech Recognizer.

Speech recognition is the process of converting an acoustic signal into written form. The main idea behind speech recognition is human-machine interaction. Unlike the easiness of human-human interaction by means of speech, human-machine interaction is not very easy; unfortunately, speech recognition is a difficult task. Intense research on this topic has been done since 1970's, and further research and development still continues. Today, speech recognition applications are widely used in different tasks like interactive voice response (IVR), automatic dictation and so on.

Mathematically speaking, speech recognition problem is formulated as [22]:

$$\hat{W} = \arg\max_{W} P(W \mid A) \tag{2.1}$$

where $\hat{W}$ represents the hypothesis word sequence, and $A$ is the given acoustic observation vector. Our goal is to find the most probable word sequence given the acoustic vector. Applying Bayes' Rule and knowing that probability of $A$ does not change for each word sequence, (2.1) becomes:

$$\hat{W} = \arg\max_{W} P(A \mid W)P(W) \tag{2.2}$$

There are two probabilities in (2.2); $P(A \mid W)$ is the probability of generating the acoustic vector $A$, given the word sequence $W$, and $P(W)$ is the probability of generating the word sequence $W$. These probabilities constitute two essential parts of a speech recognizer, and their accurate models are crucial to the recognition performance. $P(A \mid W)$ is the basis of acoustic modeling, and $P(W)$ is the basis of language modeling. These models will be discussed later in this chapter, but first of all we must convert our acoustic signal into a more compact form, which is feature extraction.

## 2.2. Feature Extraction

In order to process speech, we must first convert it into an appropriate form, i.e. the speech signal must be parameterized into vectors. Parameterization of the speech waveform is done in the feature extraction (or front-end parameterization) part of a speech recognizer. There are many different feature extraction techniques, but here, only the most common one used in speech recognition applications will be discussed.

Speech signal can be virtually decomposed into a source passed through a linear time-varying filter. The source represents air flows at the vocal cords, while the filter represents the vocal tract resonances changing over time. Classification of the speech units are mostly dependent on the filter's characteristics, so that speech recognizers estimate these filter characteristics rather than the source. This is an all-pole linear filter and its parameters can be calculated by LPC analysis.

Although the spectral characteristics of speech signal vary over time, we can assume that speech signal is stationary over an interval of few milliseconds. So in our calculations, we perform short-time spectral analysis. We first divide the signal into overlapping portions and assume that each portion is stationary. The spacing between each overlapping portion is generally 10 ms, and the portions are of length 25 ms. We multiply each short time frame with a Hamming window to get rid of the artificial high frequencies. Applying Fourier transform to each windowed frame, and discarding the harmonics will give us the LPC parameters of the filter, but here we take a different approach related to our speech perception, which is Filter-Bank analysis.

Our main organ of hearing, cochlea at the inner ear, acts a filter-bank, and processes the vibrations of different frequencies separately at different banks. So Filter-Bank analysis is a very popular approach in speech recognition. Additionally, positioning these frequency bands logarithmically on the mel scale approximates the auditory system's response much more closely, making the mel-frequency cepstral coefficients (MFCCs) a very good choice for the front-end parameters of a speech recognizer.

MFCCs are computed by first mapping the log amplitudes of the spectrum obtained by the Fourier transform onto the Mel-scale using triangular overlapping windows shown in Figure 2.2, and the relation between linear frequency and mel scale is given in (2.3). Other than approximating the frequency resolution of the human ear, it was shown that using the male scale improves the recognition accuracy [23].

$$Mel(f) = 2595 * \log(1 + \frac{f}{700}) \tag{2.3}$$

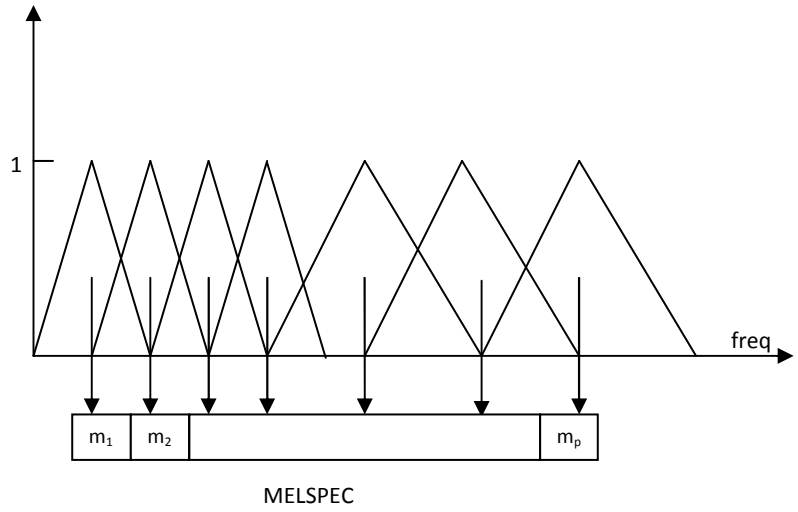


Figure 2.2. Mel Scale Filter-bank

Then discrete cosine transform (DCT) is applied to the resulting log filter-bank coefficients, compressing the spectral information into lower order ones, and also decorrelating them. DCT is formulated as:

$$c_i = \sqrt{\frac{2}{N}} \sum_{j=1}^{N} \cos\{m_j (\frac{i\pi}{N}(j-0.5))\} \quad \text{i} = 1,...,\text{M} \tag{2.4}$$

where N is the number of filter-bank channels, and M is the number of DCT coefficients. In our case, we take M in (2.4) as 12, and also add the $0^{th}$ cepstral coefficient, which represents the energy of the frame. We also perform cepstral mean normalization.

Although it is assumed that neighboring speech frames are uncorrelated, because of the human vocal tract's anatomy, this assumption is very poor. Instead, neighboring sounds are very much correlated. To overcome this problem, we can append the first and second order differentials of the static coefficients to our vectors. In our work only the first order differentials are appended.

## 2.3. Acoustic Modeling

In acoustic modeling, our aim is to find the probability of observing the acoustic vector $A$, on the condition that the word sequence $W$ was uttered. That is $P(A|W)$ in (2.2). Modern speech recognizers are based on Hidden Markov Models (HMMs). HMMs are both good at modeling the short-time stationary characteristics of speech signals, and they are also very simple and computationally feasible to use. In the next subsection, HMMs will be discussed briefly.

### 2.3.1. Hidden Markov Models

Hidden Markov Models (HMMs) are introduced in the late 1960's, but it was 1980's when they have become increasingly popular especially in speech recognition [24]. HMMs are stochastic signal models, meaning that they try to characterize only the statistical properties of the signal. Stochastic models assume that the signal can be well characterized as a parametric random process whose parameters can be determined in a precise manner.

An HMM is a finite-state machine, which can be thought as a vector sequence generator. At each time unit, *t*, a new state, *j*, is entered, and a vector, $\boldsymbol{y}_t$, is generated with probability density $b_j(\boldsymbol{y}_t)$. The transition form state *i* to state *j* is also probabilistic, and its probability is given by $a_{ij}$. So an HMM is a double probabilistic model. As the new state that is entered depends only on the previous state and the transition probability between them, these models display the Markovian property, and called "Markov" models. They are called "Hidden" because only the observation sequence $\boldsymbol{Y}$ is known, while the underlying state sequence is unknown, or hidden. HMMs have 3 parameters:

- State transition probability distribution, A = $\{a_{ij}\}$,
- Observation vector probability distribution in state j, B= $\{ b_j(\boldsymbol{y}_t)\}$
- Initial state distribution, $\pi = \{\pi_i\}$

These parameters can be estimated from a training set, by using Forward-Backward and then Baum-Welch algorithms. Baum-Welch algorithm is a very simple, efficient, and computationally feasible way of estimating the model parameters, making the HMMs very popular in speech recognition technology.

### 2.3.2. Acoustic Modeling Using HMMs

HMMs can be used to model whole words as in Isolated Word Recognition applications. But in large vocabulary speech recognition applications, we have so many different words to recognize, and to model each of them accurately, we need lots of training data for each, which makes use of whole word models practically impossible. Instead, we decompose the words into smaller subword units and train an HMM for each of the subwords. Most popular subwords used in speech recognition are phones and triphones. Phones are the smallest speech units, and triphones are obtained by concatenating three phones.

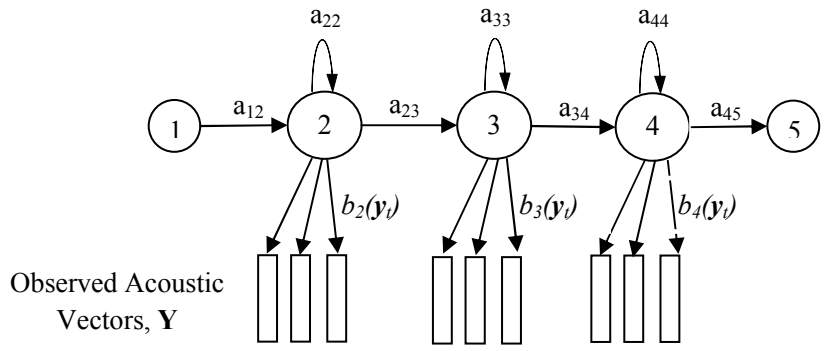


Figure 2.3. HMM Phone Model

Figure 2.3 shows a simple left-to-right HMM with 3 emitting states. This topology is very popular in speech recognition to model the phonemes, and it's also used in this thesis. It's called left-to-right because the state transitions can only occur from a left state to its right-hand state. The entry and exit states, numbered 1 and 5 respectively, are provided to join different models, creating a composite HMM. Exit state of a phone can be merged with the entry state of another phone.

At each time unit, HMM changes to a new state and generates a new acoustic vector according to the output distributions, $b_j(\boldsymbol{y}_t)$. The transition probabilities, $a_{ij}$, model the durational variability of the speech, while the output distributions model the spectral variability. In order to model the spectral variability of the speech accurately, output distribution function is very crucial. Unlike the early recognition systems which use discrete probability functions, modern systems use continuous-density distribution functions, and the most common choice is multivariate Gaussian-mixtures which can be formulated as:

$$b_j(o) = \sum_{k=1}^{M} c_{jk} N(o, \mu_{jk}, \Sigma_{jk}) \tag{2.5}$$

where $c_{jk}$ is the mixture coefficient for the $k^{th}$ mixture in state j, and $N$ is a Gaussian density function with mean vector $\mu_{jk}$ and covariance matrix $\Sigma_{jk}$.

Acoustic characteristics of phones largely depend on their context. Its neighboring phones cause large variations on the pronunciation of a phone. For this reason, context dependent triphones are commonly used. In this case, every phone has a distinct HMM model for every unique pair of its preceding and following neighbors. These context dependent triphone models perform much better than the context independent phone models in recognition applications.

One problem in using triphones is that there are many of them. For example, in English, the number of phones typically ranges from 40 to 45, so there are at least $40^3$ triphones. Not all of the triphones occur in the language due to some constraints, but the number of remaining triphones is still very big. So we have too many parameters to train, which is very hard to handle computationally. To overcome this problem we use state tying, in which acoustically indistinguishable states are tied together. After tying, several states share distributions, decreasing the number of the parameters to train. Phonetic decision trees are used to determine which states to tie.

## 2.4. Language Modeling

Statistical language modeling deals with finding $P(W)$ in (2.2), which is the probability of generating the word sequence $W$. In other words, we model the probability distribution $P(s)$ over all possible sentences, $s$ [25]. Language modeling got its start from speech recognition and it is being used in many different applications like machine translation, optical character or handwriting recognition, and many more. Language modeling is data-driven, i.e. in order to estimate the probability distributions, we need training data, which is text. In the next subsection, we will briefly introduce a language model that is used nearly in all of today's speech recognition applications, which is N-Grams.

### 2.4.1. N-Grams

N-grams are used in most of today's speech recognizers, and they use word history to estimate the probability of generating word sequences. N-grams make no use of linguistic rules, they only concentrate on local dependencies. $P(W)$ in (2.2) can be written as a product of conditional probabilities:

$$P(W) = P(w_1, w_2, ..., w_n) = \prod_{i=1}^{n} P(w_i \mid w_1, ..., w_{i-1}) \tag{2.6}$$

where $w_i$ is the i$^{th}$ word in the sequence. An n-gam models the language (or simply the word sequence $W$) as a Markov source of order n-1. So (2.6) becomes:

$$P(W) \approx \prod_{i=1}^{n} P(w_i \mid w_{i-N+1}, ..., w_{i-1}) \tag{2.7}$$

Most common n-grams used in speech recognition are trigrams (n = 3), and bigrams (n = 2). N-Grams can be easily estimated from simple frequency counts which can be directly computed from text-data as:

$$P(w_i \mid w_{i-N+1},...,w_{i-1}) = \frac{C(w_{i-N+1},...,w_{i-1},w_i)}{C(w_{i-N+1},...,w_{i-1})} \quad (2.8)$$

where $C(w_{i-N+1},...,w_{i-1},w_i)$ represents number of times $w_{i-N+1},...,w_{i-1},w_i$ appears in the training data, and $C(w_{i-N+1},...,w_{i-1})$ represents the number of times $w_{i-N+1},...,w_{i-1}$ appear.

As we increase the value of N, the accuracy of the n-gram model increase. This means that if we have enough training data, trigrams perform better than bigrams. N-grams strongly depend on training data, and unfortunately data sparseness is a big problem. For example, online Turkish dictionary of Turkish Language Institution has 104.481 different words, so there are $104.481^3$ potential trigrams, which is nearly equal to $10^{15}$. Still, this number is not even close to the exact number of triphones as Turkish is an agglutinative language. Thus, many of the potential trigrams will never appear in the training data, and many of them will appear very few times, making the estimation of (2.8) very poor. A sentence that never appeared in the training set, can appear in the test set. $P(W)$ will be zero in this case, and the word sequence $W$ will never be considered as a possible hypothesis. So whenever the probability $P(W)$ is zero, there is the risk of a recognition error. To overcome this problem, "Smoothing" is done on the language model, adjusting the probability estimations to produce more robust probabilities for previously unseen data.

Smoothing techniques tend to make distributions flatter. Low probabilities such as zero probabilities are raised, and high probabilities are lowered. Generalization capability of a language model also improves by smoothing. One very popular smoothing technique in speech recognition is "Katz Backoff". It is based on the Good-Turing estimation principle, and it is very easy to implement.

In Good-Turing estimation, n-grams are partitioned into groups depending on their frequency of occurrence in the training set. Then smoothing will be based on the n-gram frequency. Katz smoothing is an extension of the Good-Turing estimation and it combines the high order n-grams with low order n-grams. If the frequency of an n-gram is below certain threshold, it will be backed-off to the (n-1)-gram model. Backing-off continues

until an lower order n-gram that exceeds the frequency threshold is reached. Lower order n-gram probability should be scaled by a back-off weight in order to ensure that all higher order n-gram probabilities for a given word history sum up to one. With α as the normalizing backoff weight, and *T* as the frequency threshold, Katz backoff can be formulated as:

$$\mathrm{P}_{\mathrm{Katz}}(w_i \mid w_{i-N+1},...,w_{i-1}) = \begin{cases} P(w_i \mid w_{i-N+1},...,w_{i-1}) & if \quad C(w_{i-N+1},...,w_{i-1},w_i) > T \\ \alpha.P_{Katz}(w_i \mid w_{i-N+2},...,w_{i-1}), & otherwise \end{cases} \tag{2.9}$$

Beside n-grams, several other language models have been studied till today, including tree-based models, trellis models and so on. These models yielded only small improvements at considerable computational costs, so n-grams still dominate the speech recognition applications.

### 2.4.2. Language Model Evaluation

Language models are evaluated mostly over test sets, and the main quality measure is the test set perplexity. Perplexity is calculated from entropy which measures the uncertainty of an event. For a random variable X ranging over χ, entropy is defined as:

$$H(x) = -\sum_{x\varepsilon\chi} p(x)\log p(x) \tag{2.10}$$

where *p(x)* is the probability function. In language modeling, *p(x)* is not known, and cross-entropy becomes more useful. Cross entropy can be calculated as:

$$Cross \quad Entropy(p, p_m) = -\sum_{D} p(D)\log p_m(D) \tag{2.11}$$

where $p_m$ is a model of $p$, and $D$ is the new data sample. Then, perplexity is:

(2.12)

$$Perplexity(p, p_m) = 2^{CrossEntropy(p,pm)}$$

Perplexity is the average branching factor according to the language model that is used. It measures the goodness of a language model, and lower perplexity means a better one. If the perplexity is high, the recognizer should consider a large number of choices after recognizing the pervious word. Although lower perplexity usually gives better recognition results, this is not always true and there are plenty of counterexamples.

## 2.5. Decoding

In decoding, word sequence $\hat{W}$ that maximizes (2.2) is found. This is a search problem on a large HMM network which is formed using both acoustic and language models. Viterbi algorithm is commonly used in speech recognizer decoders to search and find the best path. Viterbi is a breadth-first search algorithm, in which all hypotheses are pursued in parallel.

For a speech input of a specific length, every path from the start node to the exit node of the network is a possible hypothesis $\hat{W}$. Decoder finds the paths through the network that have the highest log probabilities. For every frame of the speech input, an HMM changes its state according to the state transition probabilities, and emits an output vector which is generated with some probability density. A path's total log probability is found by summing the state transition log probabilities, and the output log probabilities of each emitting state. Within HMM transition are determined by the acoustic model likelihoods, and word transitions are determined by the language model likelihoods as in Figure 2.4.

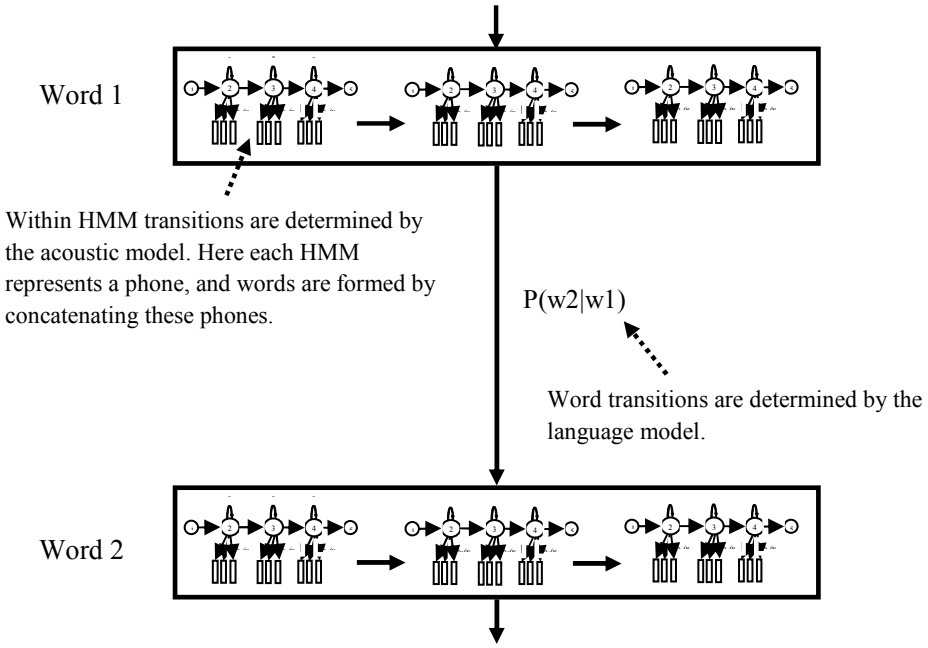


Figure 2.4. A Section of an HMM Branch

The path having the highest log probability is found by *Token Passing* algorithm which is an alternative formulation of the Viterbi algorithm. In token passing, a token is initially placed at the start node, and then it is copied to all the connecting nodes as each acoustic vector is input. At each vector input, likelihood scores are updated, and finally at the end node, the token with the highest likelihood score is chosen as the most likely path, or the word sequence hypothesis.

As we have a very large search space in speech recognition, token passing is computationally very costly and is also very time consuming. To overcome this problem, pruning of the search space is essential and *beam search* is used. At every frame or vector input, token with the best score is chosen, and any token whose score is more than a beam-width less than this best score is destroyed, and its path is pruned.

# 3. LANGUAGE MODELING IN LVCSR BASED KWS

Previously used language models in LVCSR based KWS applications are the null-grammar [10], and bigram models [8]. In this thesis, three LVCSR based spotters that use different language models will be implemented and their performances will be compared. Null-grammar language model will be our base model. Second model will be a general bigram language model, and lastly we will propose a new, keyword-adapted bigram model.

## 3.1. Null-Grammar Language Model

Null-Grammar language model is a very relaxed model, in which every speech unit in the vocabulary can be followed by another with equal probability. In fact, there is no affect of this language model in the recognition process. As language model likelihoods are equal for every pair of speech units, they don't affect the recognition network.

We will use whole words as the speech units in all of the language models. Unlike English, Turkish is an agglutinative language, and word-based Turkish LVCSR systems perform poorly than the corresponding English systems because of their high out-of-vocabulary (OOV) rate [11]. Morpheme-based or stem-ending-based models may be used in Turkish LVCSR systems to overcome this OOV problem [13]. Our word spotter will also be LVCSR-based, but as we only search for some very specific words, OOV problem does not seem that drastic, and using word-based language models seems appropriate. Figure 3.1 illustrates a word-based model.

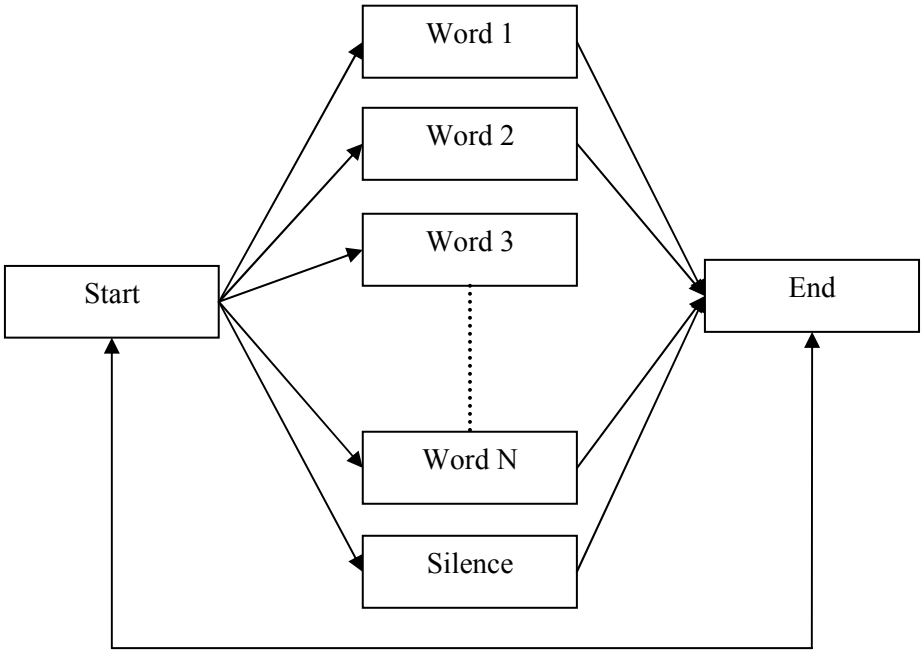


Figure 3.1. Word-based Language Model

Choosing appropriate lexicon size, and the appropriate lexicon words has a big impact on the spotting performance. For each language model, we will use varying lexicon sizes of 100, 500, 2000, and 5000 words. The most frequent words in the training database plus the keywords are chosen as the lexicon entries for the null-grammar language model. Recognition dictionary is constructed by dividing each lexicon entry into its monophones and appending a short pause that represents the gaps between consecutive words in continuous speech. An example dictionary entry is:

*kelime  k e l i m e*

Keyword spotter that uses null-grammar language model will be our base system. Performance analysis of the other systems that use different language models will be based on this system's performance.

## 3.2. Bigram Language Model

In the null-grammar language model case, we assume that every word in the lexicon can follow any other word with equal probability. This means that if we have 1000 words in the lexicon, probability of any word following any other word is 0.001.

If we tighten this assumption, and consider that words follow each other with probabilities depending on their relative frequencies on a certain text. This text is our training-corpus, and looking at the conditional probability of a word given the previous words from this text, we come up with the N-gram language model. An n-gram language model approximates the probability of a word given the previous *n-1* words.

Although n-grams make no use of linguistic rules and only concentrate on local dependencies, they are a very popular choice in speech recognition applications. Nearly all of today's speech recognizers use n-grams, or more namely bigrams and trigrams. Bigram models approximate the probability of a word given only the previous word, and trigram models approximate the probability given two previous words. As HTK only supports bigram language models, we will use bigram models in this study. Bigrams can be advantageous in word spotting applications as their computation time is less than trigram models. Using bigram models in LVCSR based KWS was first proposed in [8], and results showed that bigram language model improves the recognition performance.

## 3.3. Keyword Adapted Language Model

Language models are very sensitive to the text on which they are trained. Changes in the genre or the topic of the training text dramatically affect the recognition performance. A language model that is trained with radiology reports gives approximately 95 per cent recognition rate on transcription of the radiology reports [26], while its recognition rate dramatically lowers on transcription of daily news. To get better recognition results, topics of the training text and test data should somewhat be common.

Bigram language model that will be used in our experiments is trained from a text-corpus which was collected in 4 different topics: political news, economy news, sports news, and some general writings. This is a very general language model, and it would probably give good LVCSR results if it is used with an adequate lexicon. But for our case, this model is far from being optimal. We are only looking for some specific keywords, and training the language model from a keyword or task adapted text corpus will be the optimal case.

Here is what we do to make our language model more keyword or task specific: We again use the same text corpus but now we specialize it to the keyword. First, we detect the sentences containing the keyword from the large text corpus, and then pull them out to form a new and keyword specific text. Formation of the keyword specific text is illustrated in Figure 3.2.

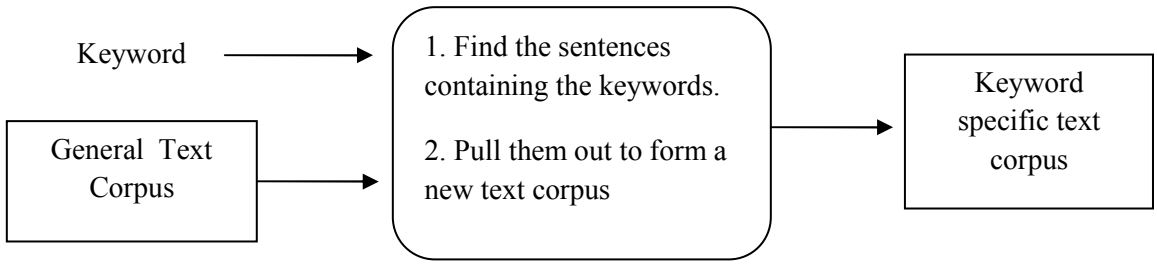


Figure 3.2. Formation of the Keyword-Specific Text Corpus.

Lexicon is built from the most frequent words in the keyword-specific text corpus. This makes our language model not only keyword specific, but also task-specific. For example, for the keyword “fatura” (means “bill” in Turkish), 10 of the most frequent words in the 500-word lexicon has the stem “öde” (means “to pay” in Turkish). So a word spotter that uses this language model becomes optimally adjusted for a telephony costumer service that deals with the payment of the bills.

### 3.3.1. Keyword Adapted Language Modeling For the Newly Introduced or Sparse Keywords

In keyword spotting applications, it is possible for the keywords to be newly introduced words such as a new company or institution name, or infrequent personal names for which no language model could be built because of the data sparseness in the training text. In this case one possible solution is using a null-grammar language model with the keywords included in the lexicon. But as language modeling is expected to improve the spotting performance, we should include better language models in our spotter.

In order to build a language model for the newly introduced or sparse keywords, we again use the big-text corpus. Newly introduced words mainly come from few categories such as personal names, company or institution names, or geographical names. For these categories we form synthetic texts using the big-text corpus and train the language models from these synthetic texts.

To form the synthetic texts, we start with certain popular person, company, and place names, and pull the sentences containing them form the big text corpus to form separate text corpus for each of the three categories. These chosen words can be called as category-indicator words. This procedure is same as building keyword-specific text corpus as in the previous section, but this time we form category-specific texts instead of keyword-specific texts. When we come up with a keyword that is absent or very sparse in the big text corpus, and if the category of that word is known (person name or company name or place name), we can train a bigram language model for that keyword from the category-specific text corpus by only changing the category-indicator word with our keyword.

As an example, we can think of a keyword spotter that tries to spot the company names on the evening economy news, and a new company called “Yenişirket” joins the stock market. Only the acoustic information is available when we use a null-grammar language model, which makes it harder to spot “Yenişirket”. Now let’s use our proposed method. We select a popular company name, for example “Arçelik”, and form a new text

corpus by pulling the sentences containing “Arçelik” from the big text corpus. It is very likely that the new text corpus includes sentences like:

“Arçelik hisseleri bugün yüzde iki değer kazandı.” (Arçelik have appreciated by 2 percent today)
“Arçelik’in halka arzı pazartesi başlayacak.” (Arçelik’s initial public offering will be starting on Monday)

Then we replace all “Arçelik”s with “Yenişirket” in this new corpus and train our language model. As the language model is trained from sentences like “Yenişirket hisseleri bugün yüzde iki değer kazandı.”, it’s more likely for our spotter to spot “Yenişirket” on the evening economy news.

This adaptation algorithm for the newly introduced words will be implemented in the computer program which will be discussed at Appendix A.

# 4. EXPERIMENTAL FRAMEWORK

## 4.1. Introduction

In this chapter, we will give a general overview of the keyword spotting system, and the experimental framework. First, a general overview of the keyword spotting system will be given. Then, acoustic training of the system will be explained, including the selection of the acoustic feature set, and the modeling of the context-dependent HMMs. Training of the language model and the text corpus will be introduced next. Test database, selected keywords and the performance measures will be explained in the remaining sections.

## 4.2. General Overview

Our task is to spot the selected keywords from continuous speech recordings using an LVCSR-based keyword spotting system. Block diagram of the keyword spotter is given in Figure 4.1. We will test and compare systems using different language models. Our base system will use a word-pair language model, and the others will use two different bigram language models.

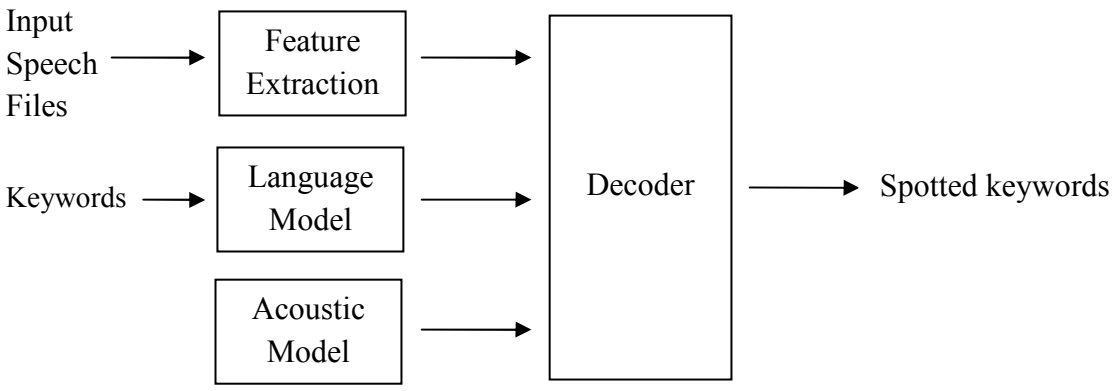


Figure 4.1. Block diagram of the word spotter.

Keyword spotter should be feasible to real working conditions. It must be speaker-independent, and its computation time should be faster than real time with reasonable spotting accuracy. So our task also includes designing a practical keyword spotter that can be used in commercial applications.

All of the acoustic and language modeling, together with decoding is done using Hidden Markov Model Toolkit (HTK) [27].

## 4.3. Acoustic Training

Acoustic training includes selecting the useful feature parameters to parameterize the speech files, and then training the HMMs using the training database. Approximately 49 hours of speech data, collected in Sabanci University, Bogazici University, METU and Sestek Inc is used in the acoustic training. Sampling rate of the files is 16 kHz, and the bit rate is 256 kbps. First the selected feature set will be introduced.

### 4.3.1. Acoustic Feature Set

MFCCs are chosen as the main parameters. MFCCs are the most conventional choice for many speech recognition applications, as they give good discriminations, and as they are easy to compute and manipulate. 12 coefficients are chosen. $0^{th}$ cepstral coefficient is also appended to the feature vector, as it is said to be helpful in improving the KWS performance [4]. This parameter is used instead of the log-energy.

We can easily deal with transmission channel effects on the input speech when using MFCCs. In [5], dealing with linear channel effects is investigated. Transmission channel effects can be seen as multiplying the input speech spectrum by the channel transform function. As MFCCs are calculated in the log-cepstrum domain, this multiplication by the channel transform function becomes a simple addition. So by subtracting the cepstral mean from all input vectors, we can get rid of the transmission

channel effects that can be caused by using different microphones or audio channels. This simple technique is called *Cesptral Mean Normalization*, and it is applied to our feature set.

Although we assume that neighboring speech frames are uncorrelated, they are not. Instead, they are very much correlated. To solve this problem, we can also append differentials of the static coefficients to our feature vectors. In our work only the first order differentials are appended. In [4], it is concluded that adding the derivative would increase the spotting performance. In the end, we have a parameter vector of length 26 for each speech frame.

### 4.3.2. Acoustic Training of the Sub-word Units

Both whole-word models and sub-word models are used in speech recognition applications. Whole-word models are mainly used in small-vocabulary, isolated word recognition applications. As the vocabulary size increases, amount of training data that is needed to accurately model each word also increases, and at some point, using whole-word models becomes impractical. Therefore, instead of whole-word models, sub-word models are widely used in LVCSR applications, which our word spotter is based on.

Most popular sub-word models used in speech recognition are monophones, and their context-dependent forms, triphones. We start our acoustic modeling by training an HMM for each of the monophones. 29 distinct sounds of the Turkish alphabet are chosen as the monophone list. Each monohone is modeled by a left-to-right HMM with 3 emitting states and 10 Gaussian mixtures. A similar left-to-right HMM with 3 emitting states is used to model the silences. This model has extra transitions, from the first emitting state to the third emitting state, and the other way round. A simple 1 emitting state HMM is used to model the short pauses, and its emitting state is tied to the second emitting state of the silence model. Topologies of the Turkish monophones and silence monophones are shown in Figure 4.2.

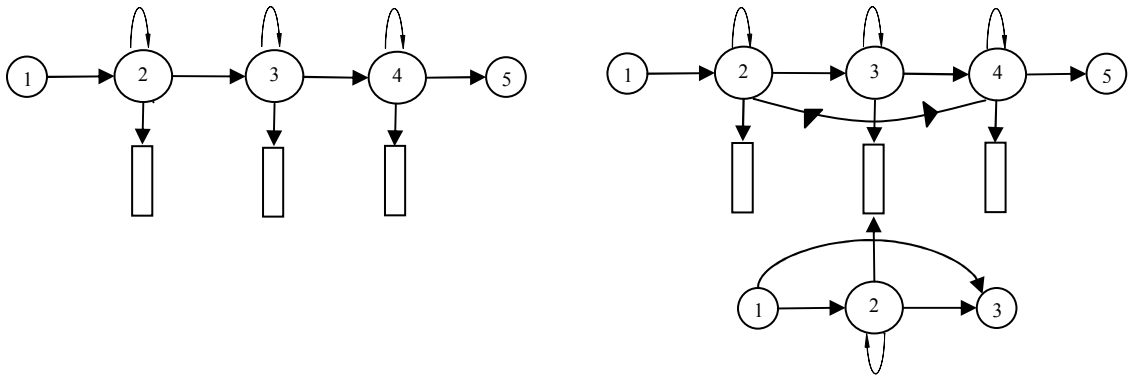


Figure 4.2. Topology of the Turkish, Silence and short pause monophones

First, initial monophone HMMs are trained using labeled training data. Then using these initial models, unlabeled training data is labeled by force-alignment, and used in the training of the final estimates of our monophone models.

Monophones are context-independent. They don't consider their neighbor monophones. However, speech units strongly depend on their neighbors. A phone can cause large variations on the pronunciation of its neighbor phones. For this reason, context-dependent models, called triphones, are widely used in speech recognition. A triphone considers both its left and right neighbors, and every phone has a distinct HMM model for every unique pair of its neighbors. Triphones outperform monophones in recognition accuracies.

To train our triphone models, we first list all the triphones that occur at least one times in the training data. Then for each triphone in the list, corresponding monophone models are copied and re-estimated to get the model.

Practically, there is insufficient data for modeling all the triphones accurately. Some of the triphones occur very few times in the training data, and their estimations are poor. To overcome this problem, after the training of the triphones, similar acoustic states are tied, so that all state distributions can be robustly estimated. Also some of the triphones

that are used in the language may never occur in the training data. In this case, each non-occurred triphone is mapped to a trained triphone according to its acoustic similarity.

### 4.4. Language Model Training

Language modeling is a less focused part in keyword spotting applications. Keyword spotters generally use null-grammar language models, or some very simple ones. Previously proposed Turkish keyword spotting systems use null-grammar language models [10], and it is said that, it is convenient to use null-grammar language models in keyword spotting applications. In this study, we will implement word spotters that use 3 different language models and compare their performances. Implemented and compared language models will be: (1) null-grammar model, (2) general model, and (3) keyword-adapted model.

In the training of the bigram models, text corpus that was collected at Sabanci University is used [14]. This corpus was collected from internet sources in 4 different topics (political news, economy news, sports news, and some general writings) and it has 5.556.449 sentences and 1.170.526 unique words. HTK is used in the calculation of the bigram probabilities with back-off smoothing. Again, different bigram models for lexicon sizes of 100, 500, 2000, and 5000 words are constructed. The most frequent words in the text corpus and the keywords are chosen as the lexicon entries.

### 4.5. Test Database

Our test database is approximately 3 hours 45 minutes in length, and it was collected at Sabanci University and Sestek Inc. There are 18 women (2 hours in length), and 15 men (1 hour 45 minutes in length) speaking 357 different sentences. Duration ratio of the test database to train database is 8 per cent.

## 4.6. Keywords

The keywords are chosen from the most frequent words in the test database. Keywords are selected in varying lengths, and the number of letters they contain vary from 2 to 10. For each number of letters, there are two words. Keyword list is given in Table 4.1.

| # of Letters | Keyword |
|---|---|
| 2 | Bu, En |
| 3 | Göz, Yıl |
| 4 | Gibi, Mavi |
| 5 | Ancak, Pazar |
| 6 | Sağlık, Toprak |
| 7 | Söyledi, Türkiye |
| 8 | Birlikte, Olduğunu |
| 9 | Özellikle, Vurguladı |
| 10 | Belediyesi, Tarafından |

Table 4.1. Keywords

Keywords are chosen as this to normalize the effects of word's linguistic type and length in the overall spotting process. Also, effects of word insertion penalty on keywords of different lengths will be analyzed. Figure 4.3 shows the number of each individual keyword on the test set. Total number of keywords in the west set is 990.

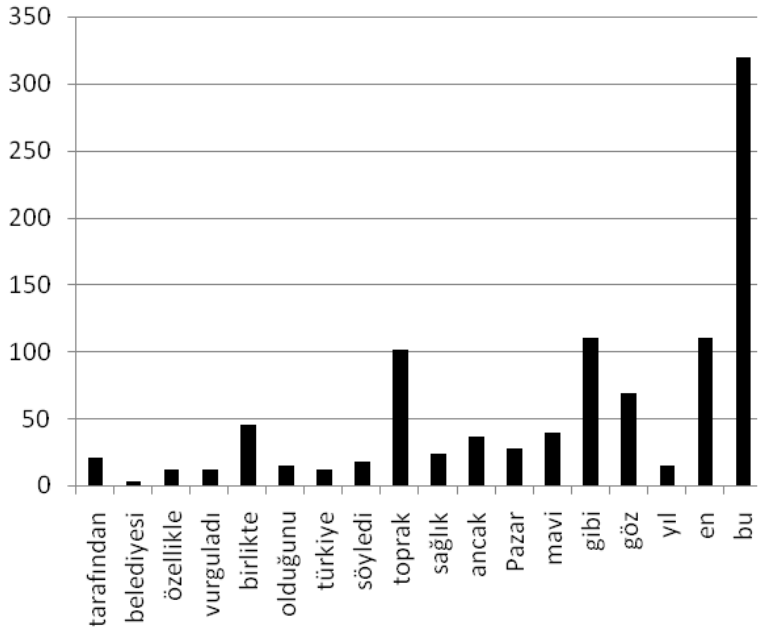


Figure 4.3. Number of Keywords on the Test Set

### 4.7. Performance Evaluation

There are two possibilities when a keyword is spotted. Either it is a true hit, or it is a false alarm. Correct detection of a keyword is a true hit, and spotting a keyword although there isn't any is a false alarm. Performance of a keyword spotter is evaluated by looking at the number of hits and false alarm, but there is no robust or conventional performance evaluation method.

*Receiver Operating Characteristics* (ROC), and *Figure of Merit* (FOM) are often used evaluation measures [10]. In ROC, typically a curve of false alarm rate versus true positive rate is plotted and it can be seen as a trade-off between selectivity and sensitivity. FOM is computed over a specific number of false alarms per keyword and per hour, and it is a more stable measure. Although they are often used, these two measures are hard to understand and interpret, and sometimes they lack in information and can give misleading results. Table 4.2 shows an example ROC and FOM result:

| | Keyword | Hits | False Alarms | ROC (at 10th false alarm per hour) | FOM |
|---|---|---|---|---|---|
| System 1 | bu | 138 (of 165) | 767 | 0.0 | 0.0 |
| System 2 | bu | 100 (of 165) | 767 | 0.0 | 0.0 |

Table 4.2. ROC and FOM Example

In this example correct keyword detection rate of the first system is 0.83 (138/165). But as the number of false alarms is high, both of its ROC and FOM are zero (0), and we can't get any information about its detection rate. System 2 truly hits 100 times, and its ROC and FOM are again zero. But this time spotting rate is 0.60 (100/165). We can't get any idea about which system gives better spotting results by looking at ROC and FOM results.

Instead of ROC and FOM, we will use *Recall* and *Precision* in the evaluation of the spotting performance. These criteria are used very commonly in information retrieval applications. Recall deals only with true hits while precision also considers the false alarms [29]. So these measures supplement each other and give more healthy results. These measures will be explained by using Figure 3.4.

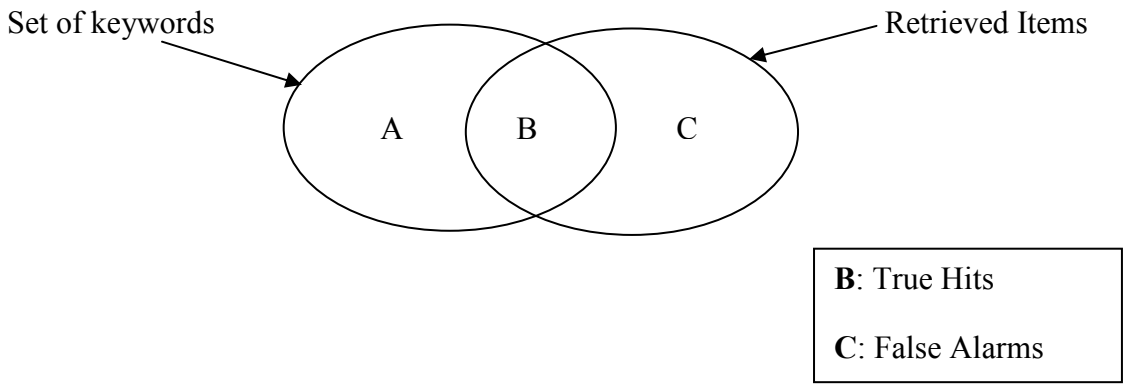


Figure 4.4. Recall and Precision Diagram

Recall is the ratio of relevant documents that are retrieved, to all relevant documents available. In other words it is the true detection rate. From figure 4.4, it can be calculated as:

$$RECALL = \frac{B}{A+B} \tag{4.1}$$

Precision is the ratio of retrieved and relevant documents to all the documents retrieved. In other words it is the ratio of true hits to all detections. From figure 4.4, precision can be calculated as:

$$PRECISION = \frac{B}{B+C} \tag{4.2}$$

Recall and precision results of the previous example is shown in Table 4.3. This time as the recall and precision results of the systems both differ, we can easily compare the spotting performances of the systems.

| | Keyword | Hits | False Alarms | Recall | Precision |
|---|---|---|---|---|---|
| System 1 | bu | 138 (of 165) | 767 | 0.83 | 0.15 |
| System 2 | bu | 100 (of 165) | 767 | 0.60 | 0.11 |

Table 4.3. Recall and Precision Example

Computation time is another performance evaluation criterion. It shows whether or not the spotter is feasible for practical applications. Ideal spotters should work faster than real-time to make it reasonable to replace manual spotting. So ratio of the total spotting time to real time is an important performance criterion for us. Spotting experiments are done on a Pentium 4, 3.2 GHz, 512 MB RAM PC.

# 5. EXPERIMENTS

## 5.1. KWS Using Null-Grammar Language Model

Keyword spotter that uses null-grammar language model will be our base system. Performance results of the other systems will compared to this system's results in the following sections.

In a null-grammar language model, every word in the vocabulary can be followed by another word with equal probability. Keywords are input to the system at the language modeling phase, where the lexicon is constructed from the most frequent words in the training text corpus plus the keywords. Lexicon sizes of 100, 500, 2000, and 5000 words are used to spot keywords. The reason for using different lexicon sizes is to investigate the effects of lexicon size on the spotting performance.

Spotting results of different lexicon sizes will be given in the following subsections. Then, lexicon sizes will be compared, and a brief conclusion will be made at the end of the section.

### 5.1.1. Lexicon Containing 100 Words

In this experiment, lexicon contains the most frequent 100 words in the training text corpus and the keywords. Spotting results of the individual keywords, and the overall system performance is given in Table 5.1.

| Keyword | Recall | Precision |
|---|---|---|
| Bu | 0.7906 | 0.1825 |
| En | 0.4909 | 0.1129 |

| | | |
|---|---|---|
| Göz | 0.8260 | 0.1177 |
| Yıl | 0.5333 | 0.0202 |
| Gibi | 0.5909 | 0.3240 |
| Mavi | 0.8205 | 0.0969 |
| Ancak | 0.8055 | 0.1686 |
| Pazar | 0.7142 | 0.0769 |
| Sağlık | 0.7083 | 0.0502 |
| Toprak | 0.9009 | 0.2870 |
| Söyledi | 0.8333 | 0.0931 |
| Türkiye | 0.7500 | 0.1323 |
| Birlikte | 0.6666 | 0.2564 |
| Olduğunu | 0.5333 | 0.1212 |
| Özellikle | 0.7500 | 0.1363 |
| Vurguladı | 0.8333 | 0.1694 |
| Belediyesi | 0.6666 | 0.0645 |
| Tarafından | 0.9097 | 0.1570 |
| **Overall** | **0.7474** | **0.1458** |

Table 5.1. Null-Grammar Language Model With 100-word Lexicon Results

Overall recall is 0.74, which means that nearly 3 of the 4 keywords are spotted correctly. “Tarafından” has the highest recall of 0.9097, while “en” has the lowest recall of 0.4909. Overall precision rate is lower than the recall rate because of the high number of

false alarms. We are 14 per cent sure that a spotted word is a true hit. Recall and precision graphs of the individual keywords are given in Figure 5.1.

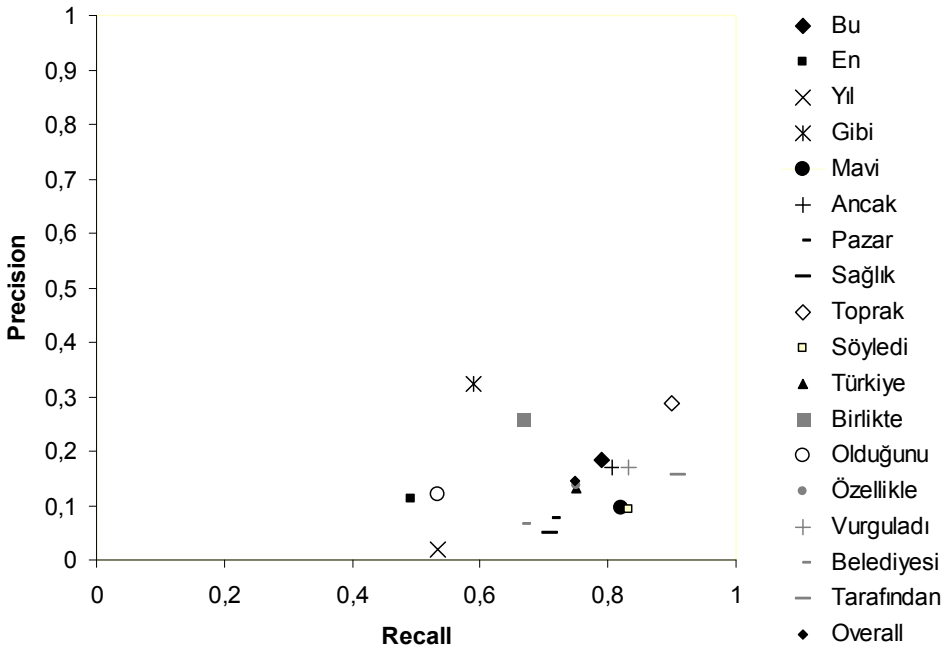


Figure 5.1. Performance Graph for 100-Words Lexicon

Average spotting time is 2293 seconds, which is approximately 6 times faster than real time (total test database is 13662 seconds in length). This means that, system can complete a task in one hour which takes at least 6 hours for human beings.

### 5.1.2. Lexicon Containing 500 Words

In this case, lexicon contains the most frequent 500 words in the training text corpus and the keywords. Spotting results of the individual keywords, and the overall system performance is given in Table 5.2.

| Keyword | Recall | Precision |
|---|---|---|
| Bu | 0.5687 | 0.4155 |
| En | 0.4636 | 0.3072 |
| Göz | 0.8550 | 0.2295 |
| Yıl | 0.8000 | 0.1111 |
| Gibi | 0.6272 | 0.4758 |
| Mavi | 1.0000 | 0.2635 |
| Ancak | 0.9444 | 0.5483 |
| Pazar | 1.0000 | 0.3493 |
| Sağlık | 0.9166 | 0.3606 |
| Toprak | 0.9504 | 0.4137 |
| Söyledi | 1.0000 | 0.5806 |
| Türkiye | 0.9166 | 0.5000 |
| Birlikte | 0.6888 | 0.5740 |
| Olduğunu | 0.9333 | 0.5384 |
| Özellikle | 0.7500 | 0.6000 |
| Vurguladı | 1.0000 | 0.6315 |
| Belediyesi | 0.6666 | 0.2000 |
| Tarafından | 0.9523 | 0.5128 |
| **Overall** | **0.7128** | **0.3705** |

Table 5.2. Null-Grammar Language Model With 500-words Lexicon Results

This time overall recall decreases to 0.7128, and overall precision increases to 0.3705. So detection probability of the system decrease while the probability of the detected keyword to be a true hit, increases. Recall and precision graphs of the individual keywords are given in Figure 5.2.

Average spotting time also increases to 5678 seconds, which is approximately 2.4 times faster than real-time.

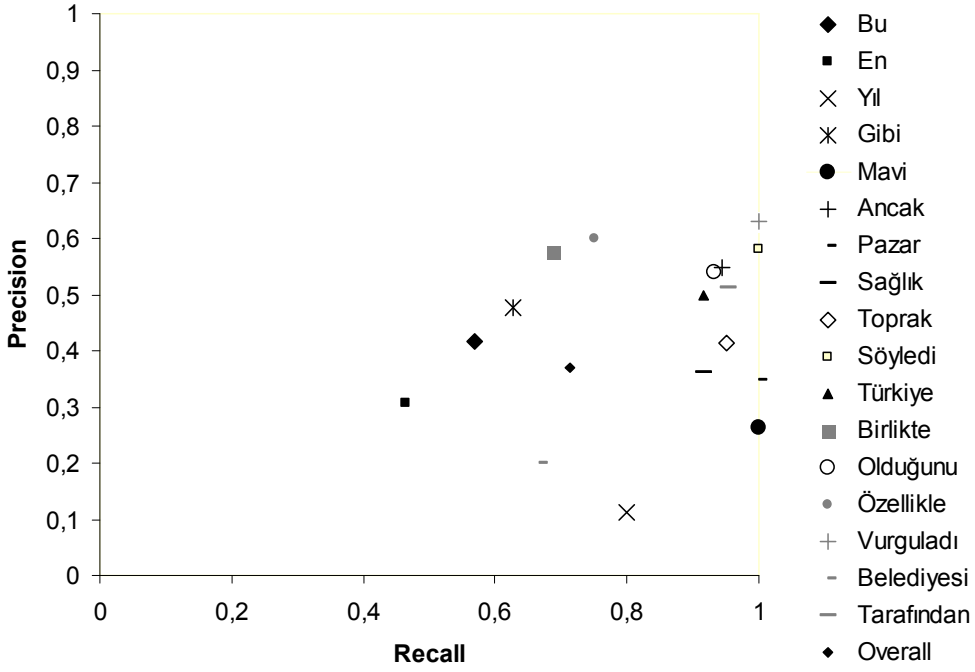


Figure 5.2. Performance Graph for 500-Words Lexicon

### 5.1.3. Lexicon Containing 2000 Words

Now, lexicon size is increased to 2000 so that the lexicon contains the most frequent 2000 words in the training text corpus and the keywords. Spotting results of the individual keywords, and the overall system performance is given in Table 5.3.

| Keyword | Recall | Precision |
|---|---|---|
| Bu | 0.5968 | 0.6608 |
| En | 0.3909 | 0.5810 |
| Göz | 0.7681 | 0.4308 |
| Yıl | 0.5333 | 0.2424 |
| Gibi | 0.5636 | 0.6326 |
| Mavi | 0.8205 | 0.5079 |
| Ancak | 1.0000 | 0.8181 |
| Pazar | 0.9655 | 0.8285 |
| Sağlık | 0.8750 | 0.6363 |
| Toprak | 0.9405 | 0.4589 |
| Söyledi | 1.0000 | 0.7200 |
| Türkiye | 0.9166 | 0.6875 |
| Birlikte | 0.6222 | 0.8235 |
| Olduğunu | 0.9333 | 0.8235 |
| Özellikle | 0.7500 | 0.6428 |
| Vurguladı | 1.0000 | 0.8000 |
| Belediyesi | 1.0000 | 0.5000 |
| Tarafından | 0.9047 | 0.9047 |
| **Overall** | **0.6819** | **0.5963** |

Table 5.3. Null-Grammar Language Model With 2000-words Lexicon Results

Again, overall recall is lower than and overall precision is higher than these of 500-words lexicon. Precision increases to nearly 0.6. Precision of the keyword “tarafından” is 0.9047, which means that we are 90 per cent sure that it is true hit if the keyword is spotted. Recall and precision graphs of the individual keywords are given in Figure 5.3.

Average spotting time also increases to 12663 seconds, which is approximately 1000 seconds faster than real-time. This spotting time can still be considered as convenient for the spotter, but of course it is less favorable than the previous cases.

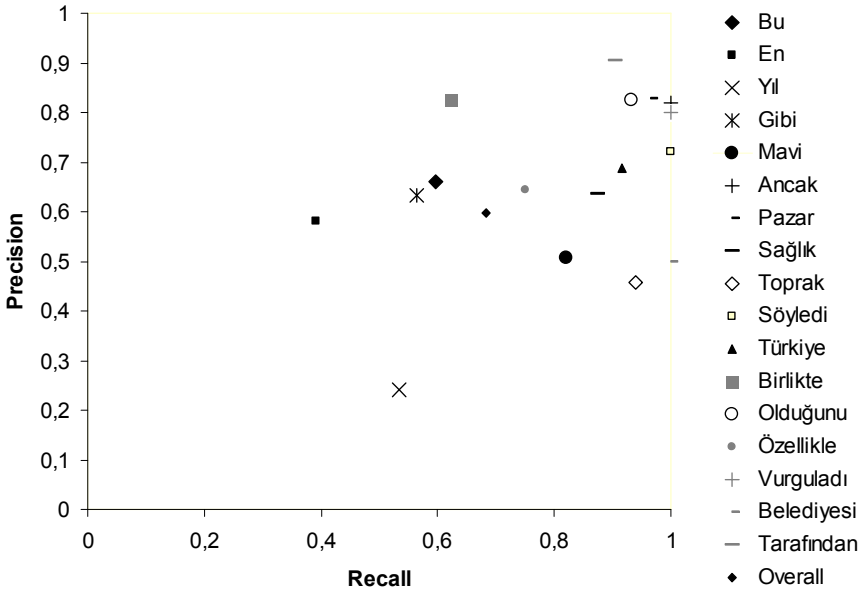


Figure 5.3. Performance Graph for 2000-Words Lexicon

### 5.1.4. Lexicon Containing 5000 Words

Now lexicon size is increased to 5000 words, so that the lexicon contains the most frequent 5000 words in the training text corpus and the keywords. Spotting results of the individual keywords, and the overall system performance is given in Table 5.4.

| Keyword | Recall | Precision |
|---|---|---|
| Bu | 0.5218 | 0.7767 |
| En | 0.2554 | 0.6923 |
| Göz | 0.7826 | 0.7297 |
| Yıl | 0.5333 | 0.3809 |
| Gibi | 0.6545 | 0.7200 |
| Mavi | 0.7948 | 0.6739 |
| Ancak | 1.000 | 0.9230 |
| Pazar | 0. 9642 | 0.8181 |
| Sağlık | 0.8333 | 0.9090 |
| Toprak | 0.8514 | 0.5149 |
| Söyledi | 1.0000 | 0.9473 |
| Türkiye | 0.9166 | 0.6875 |
| Birlikte | 0.5333 | 0.8888 |
| Olduğunu | 0.8000 | 0.8571 |
| Özellikle | 0.7500 | 0.6923 |
| Vurguladı | 1.0000 | 0.9230 |
| Belediyesi | 0.6666 | 0.5000 |
| Tarafından | 0.7619 | 0.8888 |
| **Overall** | **0.6484** | **0.7213** |

Table 5.4. Null-Grammar Language Model With 5000-words Lexicon Results

For the first time, precision exceeds recall rate. Now, probability of spotting a keyword is lower than the probability that a spotted keyword is a true hit. Recall and precision graphs of the individual keywords are given in Figure 5.4.

In this case, average spotting time is 23415 seconds, which is 1.71 times slower than real time. This spotting time is not appropriate for keyword spotting applications as manual spotting can be done much faster. So we do not need to test lexicons having more than 5000 words in future.

Next, we will compare the spotting results of lexicons having different number of words, and we will give a brief conclusion.

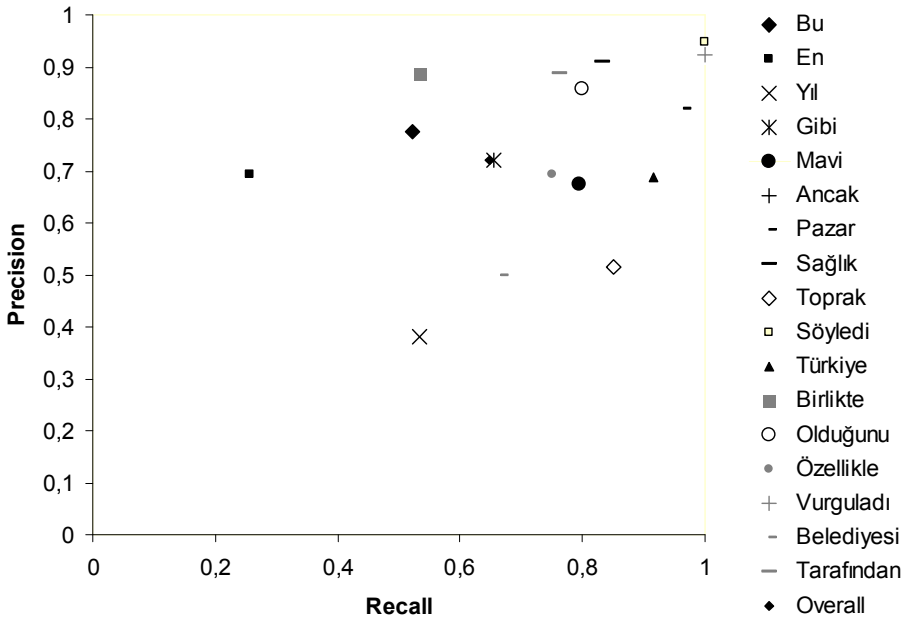


Figure 5.4. Performance Graph for 5000-Words Lexicon

### 5.1.5. Comparison of Results

In this experiment, we implemented a word spotter that uses null-grammar language model, and tested it with different lexicon sizes. Recall and precision results of the individual keywords for different lexicon sizes are given in Figures 5.5 and 5.6.

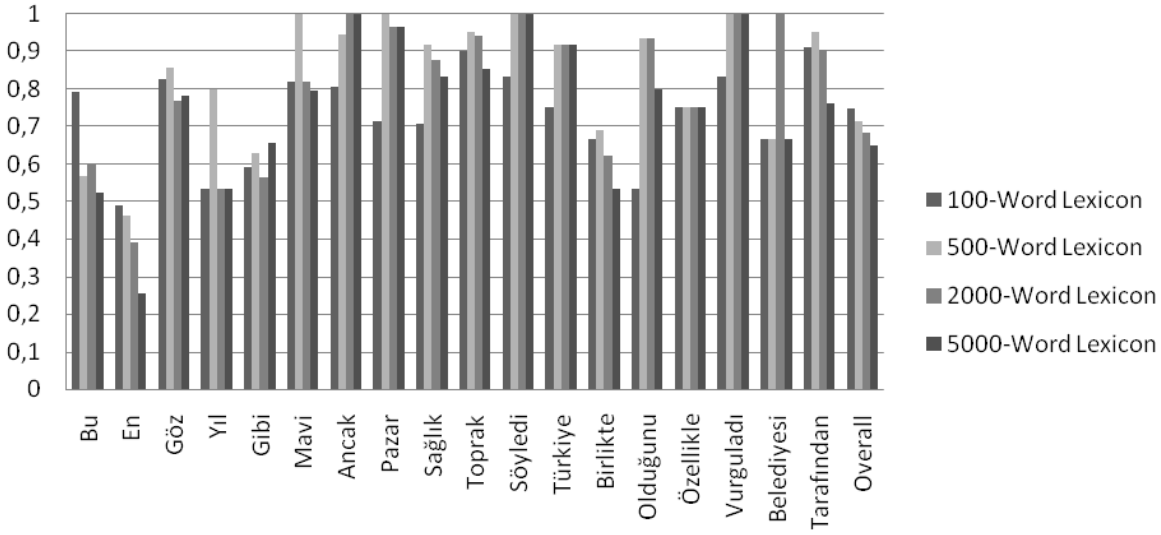


Figure 5.5. Recall Performance of the Individual Keywords

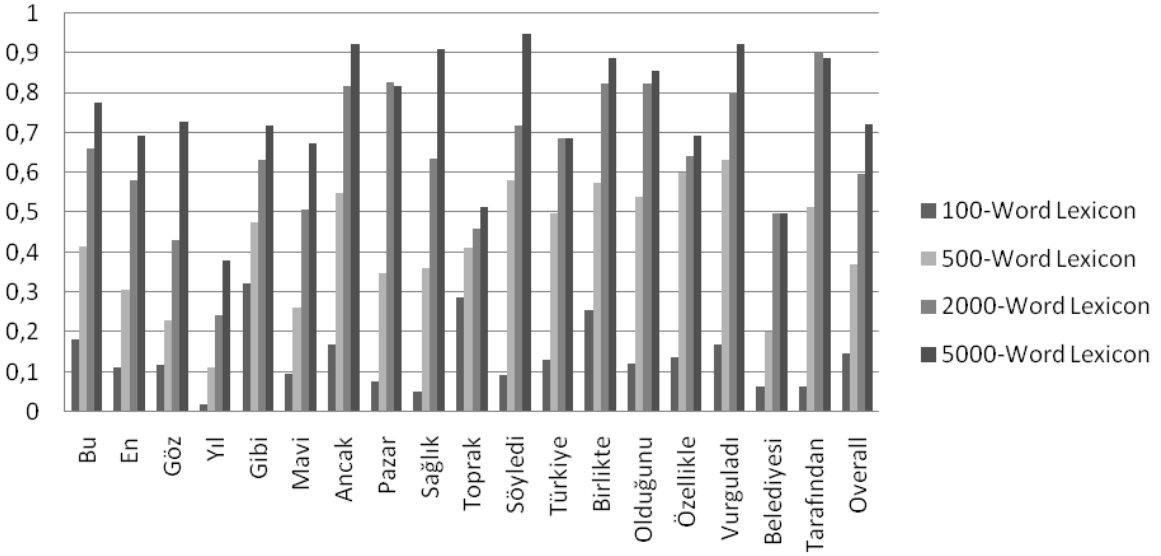


Figure 5.6. Precision Performance of the Individual Keywords

Keyword “en” has the lowest recall for all lexicon sizes while “vurguladı” and “söyledi” has the highest. For lexicon sizes greater than 100 words, “vurguladı” and “söyledi” has recall rate of 1, which means that each of these words in the test set are detected. As shorter words contain less acoustic information, they can be easily inserted in place of other words or vice versa with small acoustic similarities. So it’s expected for the shortest keywords, “en” and “bu” to have low recall rates.

Figure 5.7 shows the overall recall and precision results for different lexicon sizes. Overall recall decreases and overall precision increases as lexicon size is increased. As lexicon size grows, acoustic variability of the system increases. This leads to a increase in the number of words that can deceive the spotter, i.e. more words can take similar likelihood scores to keywords. So, in place of a keyword, spotter can choose a non-keyword that has a slightly higher likelihood score than the keyword’s likelihood score. This can decrease the overall recall rate, as in our case.

Unlike recall, precision increases as lexicon size increases. In small sized lexicons, keywords can be inserted in place of the out-of-vocabulary words as false alarms, which decreases the precision of the system. As lexicon size grows, some of these wrongly detected out-of-vocabulary words are included in the lexicon, so that now they can be recognized correctly leading to a decrease in number of false alarms.

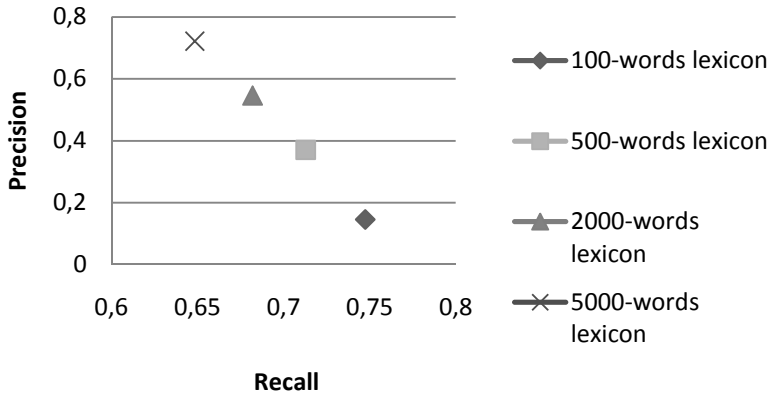


Figure 5.7. Overall Recall and Precision Performance

Spotting times for different lexicon sizes and their ratio to real time are in Table 5.5. and Figure 5.8. Real time ratios are calculated as:

$$RTR = \frac{\text{Spotting time}}{\text{Total Test Set Time}} \tag{5.1}$$

| Lexicon Size | Spotting Time(sec) | Real-Time Ratio |
|---|---|---|
| 100 | 2293 | 0.16 |
| 500 | 5678 | 0.41 |
| 2000 | 12663 | 0.92 |
| 5000 | 23415 | 1.71 |

Table 5.5. Spotting Times

In automatic keyword spotting applications, the aim is to detect the keywords faster than manual keyword spotting. So we want the spotter to operate faster than real-time. In this case, only the 5000-words lexicon exceeds real time, and 2000-words lexicon operates close to real time. We can say that, lexicons having more than 2000 words are inappropriate for automatic keyword spotting applications.

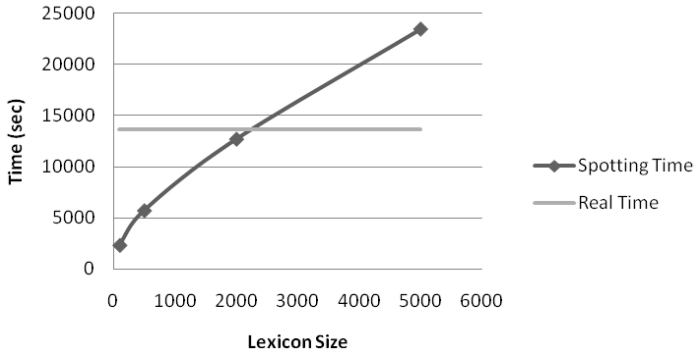


Figure 5.8. Spotting Times of Different Lexicon Sizes

If we compare spotters that use null-grammar language models with different lexicon sizes, we can say that lexicons having 500 and 2000 words give the most suitable

results. Besides its high recall of 0.74, lexicon containing 100 words lack in precision. 5000-words lexicon operates 1.7 times slower than real time, so it is completely inappropriate. If we want a fast spotter, and if we don't care about it's precision too much with acceptable recall, we can choose to use 500 words in our lexicon. This system has a recall of 0.71, precision of 0.37, and operates 2.4 times faster than real time. If we want better precision rate with slower operation time, we can choose to use 2000-words lexicon which has recall of 0.68, precision of 0.59, and operates close to real time.

Spotters that use null-grammar language models can be used in practical applications, but their recall rates can sometimes be insufficient. So we need to increase their recall rates. Using a better language model can increase the recall rates, as we will try in the next section.

## 5.2. KWS Using General Bigram Language Model

In this experiment, we use a general bigram language model in our LVCSR based keyword spotter. By general, we mean that the text-corpus that the model is built from, is task-independent. This corpus was collected from internet sources in 4 different general topics, and it is maybe the biggest Turkish text corpus with 5.556.449 sentences and 1.170.526 unique words.

Bigram models approximate the probability of a word given only the previous word. Bigrams are suitable for keyword spotting applications as they don't need too much processing time. Again, lexicon sizes of 100, 500, 2000, and 5000 words are used to spot keywords and the lexicons are constructed from the most frequent words in the training text corpus plus the keywords. Spotting results of different lexicon sizes will be given in the following section.

### 5.2.1. Lexicon Containing 100 Words

In this experiment, lexicon contains the most frequent 100 words in the training text corpus and the keywords, and the language model is constructed from the bigram probabilities of them. Spotting results of the individual keywords, and the overall system performance is given in Table 5.6.

| Keyword | Recall | Precision |
|---|---|---|
| Bu | 0.8325 | 0.1734 |
| En | 0.6090 | 0.2815 |
| Göz | 0.8985 | 0.1925 |
| Yıl | 0.8666 | 0.0286 |
| Gibi | 0.6000 | 0.3437 |
| Mavi | 0.9230 | 0.1846 |
| Ancak | 1.0000 | 0.2195 |
| Pazar | 1.0000 | 0.1581 |
| Sağlık | 0.8750 | 0.0889 |
| Toprak | 0.9801 | 0.3586 |
| Söyledi | 1.000 | 0.1636 |
| Türkiye | 0.9166 | 0.1294 |
| Birlikte | 0.8000 | 0.4000 |
| Olduğunu | 0.8000 | 0.2033 |
| Özellikle | 1.0000 | 0.1579 |
| Vurguladı | 1.0000 | 0.3870 |

| | | |
|---|---|---|
| Belediyesi | 1.0000 | 0.1875 |
| Tarafından | 1.0000 | 0.2500 |
| **Overall** | **0.8292** | **0.1886** |

Table 5.6. Bigram Language Model With 100-words Lexicon Results

Overall recall is 0.82 which is greater than the overall recall of the spotter that uses null-grammar language model and 100-words lexicon. Precision also increases when bigram language model is used. 7 of the 11 words have recall rates of 1.0, meaning that they are perfectly detected in the test set. Again, shorter words like “en”, and “gibi” have the lowest recall rates. “Yıl” has the lowest precision of 0.02 while “birlikte” has the highest recall of 0.4. Recall and precision graphs of the individual keywords are given in Figure 5.9.

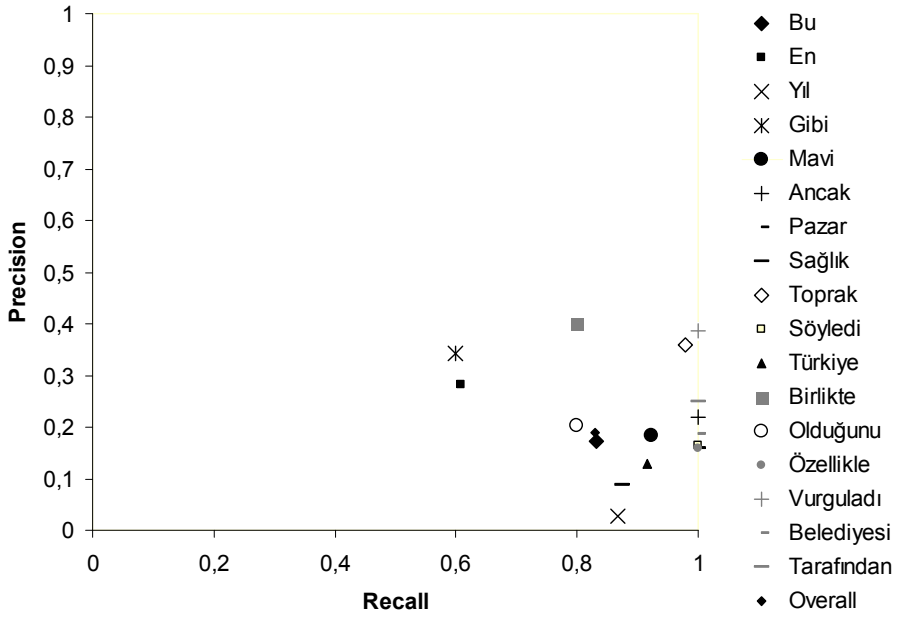


Figure 5.9. Performance Graph for 100-Words Lexicon

Average spotting time is 2014 seconds, which is also shorter than the average spotting time of the spotter that uses null-grammar language model. Bigram language

model that uses 100-wors lexicon outperforms the null-grammar language model in all of the spotting criteria.

### 5.2.2. Lexicon Containing 500 Words

In this case, lexicon contains the most frequent 500 words in the training text corpus and the keywords. Spotting results of the individual keywords, and the overall system performance is given in Table 5.7.

| Keyword | Recall | Precision |
|---|---|---|
| Bu | 0.7812 | 0.3234 |
| En | 0.6000 | 0.3793 |
| Göz | 0.8695 | 0.2955 |
| Yıl | 0.8000 | 0.0983 |
| Gibi | 0.6363 | 0.4666 |
| Mavi | 1.0000 | 0.3679 |
| Ancak | 1.0000 | 0.5217 |
| Pazar | 1.0000 | 0.5384 |
| Sağlık | 0.9166 | 0.3859 |
| Toprak | 0.9603 | 0.4330 |
| Söyledi | 1.0000 | 0.5625 |
| Türkiye | 0.9166 | 0.4583 |
| Birlikte | 0.7111 | 0.5925 |
| Olduğunu | 0.9375 | 0.4285 |

| | | |
|---|---|---|
| Özellikle | 0.6666 | 0.4705 |
| Vurguladı | 1.0000 | 0.7058 |
| Belediyesi | 0.6666 | 0.3333 |
| Tarafından | 0.9523 | 0.5263 |
| **Overall** | **0.8030** | **0.3702** |

Table 5.7. Bigram Language Model With 500-Words Lexicon Results

This time recall decreases to 0.8, and precision increases to 0.37. Precision approximately doubles itself with a 2 per cent decrease in recall, and this is a desirable situation. Recall and precision graphs of the individual keywords are given in Figure 5.10. Average spotting time is 5054 seconds which is 2.7 time faster than real time.

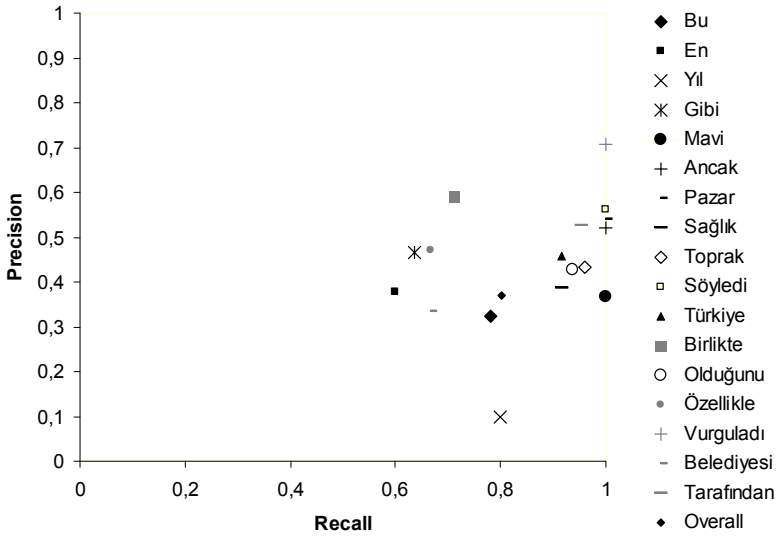


Figure 5.10. Performance Graph for 500-Words Lexicon

### 5.2.3. Lexicon Containing 2000 Words

In this case, lexicon contains the most frequent 2000 words in the training text corpus and the keywords. Spotting results of the individual keywords, and the overall system performance is given in Table 5.8.

| Keyword | Recall | Precision |
|---|---|---|
| Bu | 0.7812 | 0.4416 |
| En | 0.5727 | 0.4666 |
| Göz | 0.7246 | 0.5102 |
| Yıl | 0.6666 | 0.2325 |
| Gibi | 0.6272 | 0.5655 |
| Mavi | 0.7948 | 0.6326 |
| Ancak | 1.0000 | 0.7659 |
| Pazar | 1.0000 | 0.8484 |
| Sağlık | 0.7916 | 0.6551 |
| Toprak | 0.9405 | 0.4656 |
| Söyledi | 1.0000 | 0.7500 |
| Türkiye | 0.9166 | 0.5500 |
| Birlikte | 0.6666 | 0.8333 |
| Olduğunu | 0.9333 | 0.6086 |
| Özellikle | 0.7500 | 0.6000 |
| Vurguladı | 1.0000 | 0.8000 |

| | | |
|---|---|---|
| Belediyesi | 0.6666 | 0.4000 |
| Tarafından | 0.9523 | 0.9090 |
| **Overall** | **0.7747** | **0.5161** |

Table 5.8. Bigram Language Model With 2000-Words Lexicon Results

When increasing the lexicon size from 500 to 2000, recall again decreases, and precision increases. Precision rate is now 0.51, which means that we are approximately 50 per cent sure that a spotted keyword is a true hit. Recall and precision graphs of the individual keywords are given in Figure 5.11.

Average spotting time is 10383 seconds, which is 1.31 times faster than real time.

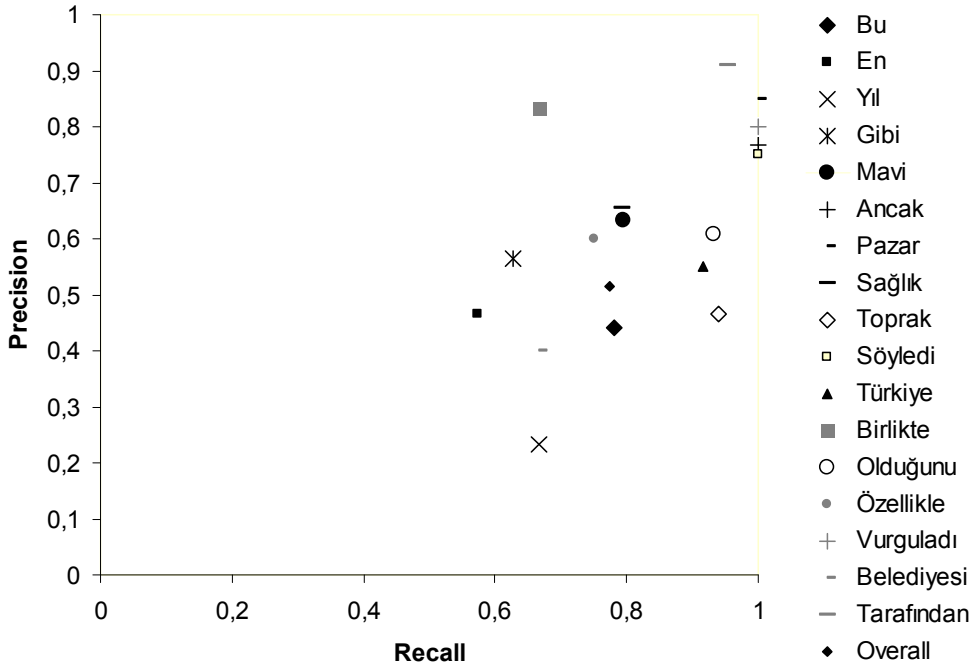


Figure 5.11. Performance Graph for 2000-Words Lexicon

### 5.2.4. Lexicon Containing 5000 Words

This time, lexicon contains the most frequent 5000 words in the training text corpus and the keywords. Spotting results of the individual keywords, and the overall system performance is given in Table 5.9.

| Keyword | Recall | Precision |
|---|---|---|
| Bu | 0.7500 | 0.5183 |
| En | 0.5545 | 0.5495 |
| Göz | 0.7246 | 0.7246 |
| Yıl | 0.6000 | 0.3000 |
| Gibi | 0.7090 | 0.6240 |
| Mavi | 0.9487 | 0.8043 |
| Ancak | 1.0000 | 0.8181 |
| Pazar | 1.0000 | 0.7777 |
| Sağlık | 0.8750 | 0.7500 |
| Toprak | 0.8415 | 0.5029 |
| Söyledi | 1.0000 | 0.9473 |
| Türkiye | 0.9166 | 0.5500 |
| Birlikte | 0.6444 | 0.9062 |
| Olduğunu | 0.9333 | 0.7000 |
| Özellikle | 0.7500 | 0.6428 |

| | | |
|---|---|---|
| Vurguladı | 1.0000 | 1.0000 |
| Belediyesi | 1.0000 | 0.5000 |
| Tarafından | 0.9523 | 0.9090 |
| **Overall** | **0.7686** | **0.6011** |

Table 5.9. Bigram Language Model With 5000-Words Lexicon Results

This time, overall precision increases nearly 10 per cent with a very little decrease in overall recall. This is a desirable situation, but again the slowness of the 5000-words lexicon system makes it inappropriate for keyword spotting. Average spotting time is 20629 seconds which is approximately 1.5 times slower than real-time. This computation time is not suitable for practical applications. Recall and precision graphs of the individual keywords are given in Figure 5.12.

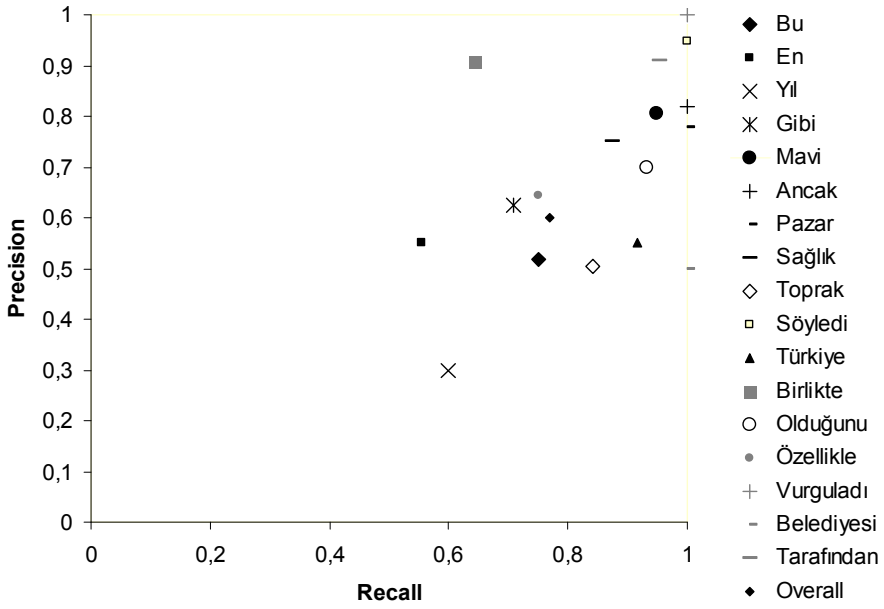


Figure 5.12. Performance Graph for 5000-Words Lexicon

### 5.2.5. Comparison of Results

In this experiment, we implemented a word spotter that uses a general bigram language model, and tested it with different lexicon sizes. Recall and precision results of the individual keywords for different lexicon sizes are given in Figures 5.13 and 5.14.

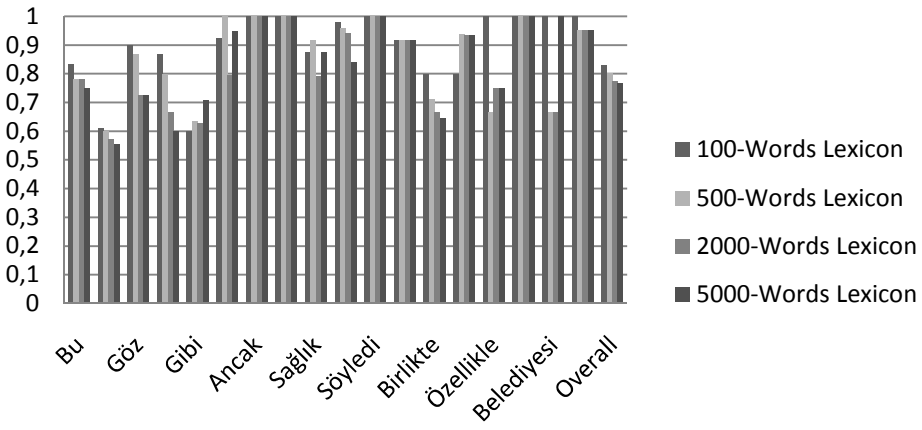


Figure 5.13. Recall Performance of the Individual Keywords

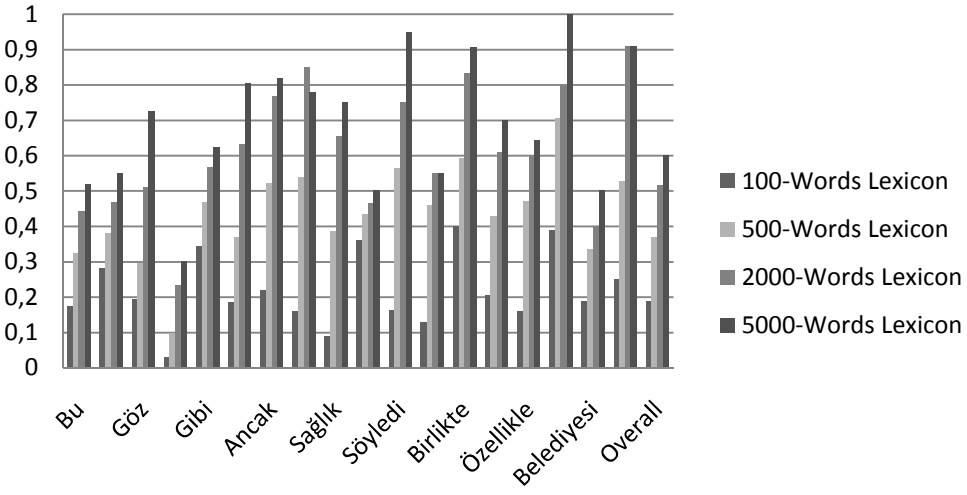


Figure 5.14. Precision Performance of the Individual Keywords

Again, recall decreases, and precision increases as we increase the number of words in the lexicon. The reason for that was explained in the previous chapter. Figure 5.15

shows the overall recall and precision results of different lexicon sizes. For 4 of the keywords, namely “ancak”, “pazar”, “söyledi”, and “vurguladı”, recall is 1.0 for all of the lexicon sizes. Every one of these words in the test set are detected correctly independent of the lexicon size. Again “en” has the lowest recall rate for all of the lexicon sizes because of the insertion problem of the shorter words.

Highest precision is achieved for the keyword “vurguladı” when we use 5000-words lexicon. A this lexicon size, both recall and precision is 1.0 for “vurguladı”, which means that we detect all of the words without any errors. Like in the null-grammar language model case, “yıl” has the lowest precision rates for all of the lexicon size.

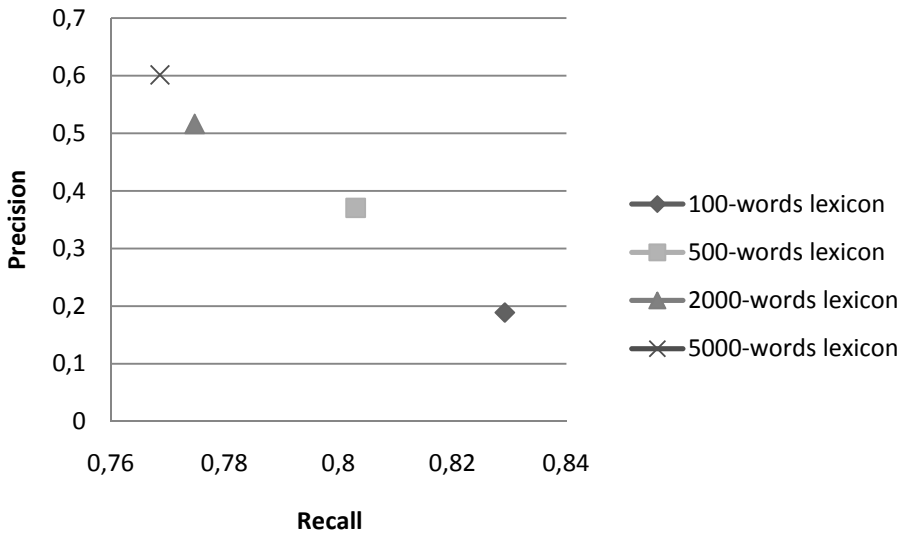


Figure 5.15. Overall Recall and Precision Performance

Spotting times for different lexicon sizes and their ratio to real time are in Table 5.10. Like in the null-grammar language model case, spotting time of the lexicon having 5000 words exceeds real time, so this lexicon size is inappropriate for practical applications.

| Lexicon Size | Spotting Time(sec) | Real-Time Ratio |
|---|---|---|
| 100 | 2014 | 0.14 |
| 500 | 5678 | 0.37 |
| 2000 | 5054 | 0.76 |
| 5000 | 20629 | 1.51 |

Table 5.10. Spotting Times

### 5.2.6. Comparison with the Base Model

As our base system, we implemented a word spotter that uses a null-grammar language model, and the performance results of that system is given in the previous section. Now we can compare the performance of the spotter that uses a general bigram language model with our base system. Figures 5.16, 5.17, and 5.18 shows the comparative results of these systems for the overall recall, precision, and spotting time respectively.

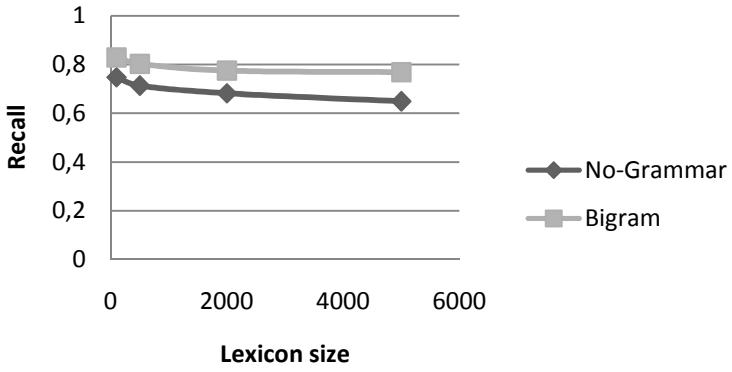


Figure 5.16. Comparative Overall Recall

Spotter that uses a general bigram language model outperforms the spotter that uses a null-grammar language model in the overall recall performance. As a bigram language model models the relations between the words better than a null-grammar language model, this is an expected result as the number of true hits increase.

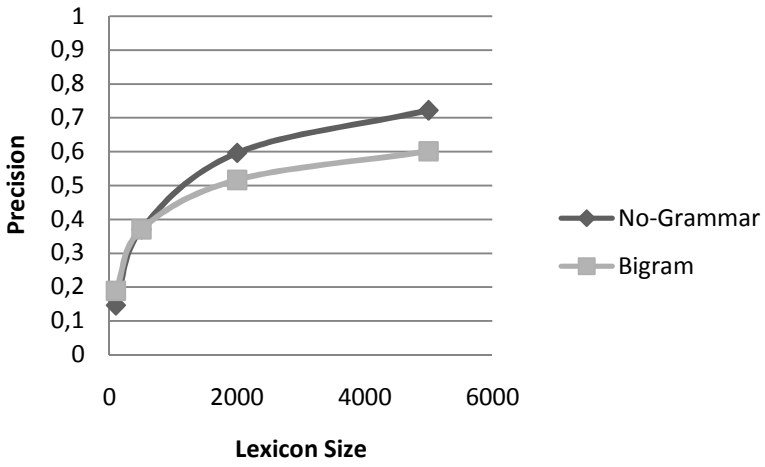


Figure 5.17. Comparative Overall Precision

On the other side, spotter that uses a null-grammar language model has better precision than the spotter that uses a bigram language model for the lexicons containing more than 100 words. This is caused by the high number of out-of-vocabulary (OOV) words, as we use very limited lexicons in our experiments. When we use a null-grammar language model, the words in the lexicon are more evenly distributed over the OOV words, as all of the words have the same language model weights. In the bigram language model case, lexicon words that have higher language model weights are more frequently mapped to the OOV words, and if these highly weighted words are keywords, this leads to a decrease in the precision.

For the 100-words lexicon case, non-keywords in the lexicon are the most frequent 100 words in our training text corpus. Most of the keywords have less language model weights than the non-keywords, so that the OOV words are much more mapped to non-keywords rather than the keywords. In that case, number of keyword insertions decrease because of the non-keyword insertions and precision of the spotter that uses a bigram language model is higher than the spotter that uses a null-grammar language model.

For the lexicons having more than 500 words, keywords start to have higher language model weights than some of the words in the lexicon, as their relative position in

the list of most frequent lexicon words, raise. In that case, keywords are more easily inserted in the place of OOV words rather than other less weighted non-keywords in the lexicon, and precision of the spotter that uses a bigram language model becomes lower than the precision of the spotter that uses a null-grammar language model. We can say that, as the lexicon size increases, the precision difference between these two models will also increase, because of the increase in the number of non-keywords that are less weighted than the keywords.

If we look at the average spotting times, the spotter that uses a bigram language model operates faster then the spotter that uses a null-grammar language model in all of the different lexicon sizes. This is because, in the bigram language model case the decoder has more clues and it can decide faster.

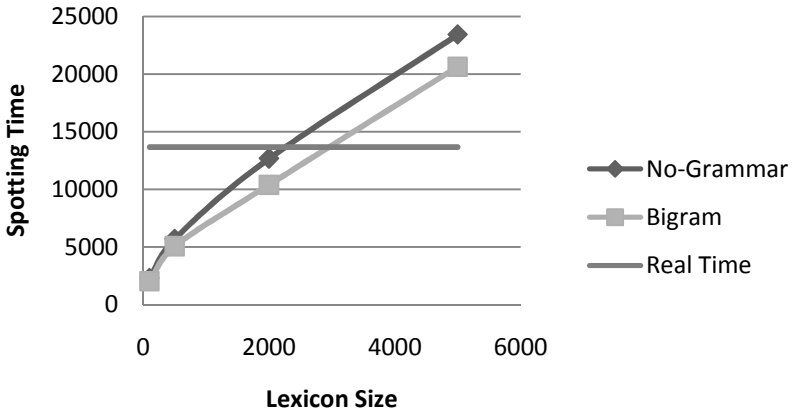


Figure 5.18. Comparative Spotting Time

If we compare the spotter that uses a null-grammar language model, and the spotter that uses a bigram language model, we see the latter outperforms the first one in two of the three criteria. The spotter that uses a bigram language model performs worse only in the precision criteria, which can be solved by some insertion adjustments. As a conclusion, we can say that using a bigram language model increases the overall spotting performance.

## 5.3. KWS Using Keyword Adapted Language Model

Speech recognition systems that use language models are highly sensitive to the topic of the text on which the language model is trained, so the topic or the domain of the training text will also affect the performance of the keyword spotters that are LVCSR-based. In our case, keywords are chosen by the user from a random domain. The spotter has no restrictions on the domain of the keywords, so the keywords are domain-independent which can affect the recognition or spotting performance drastically. Here is what we can do to overcome this domain problem:

As we can't make the keywords domain-specific, why not to make the our domain keyword-specific, or keyword-adapted? This can be done beforehand as we know the keywords before the spotting process starts. We again use the same text corpus that we used to train our general bigram language model, but now we specialize it to the keywords by first detecting the sentences containing the keywords from the big text, and then pulling them out to form a new and keyword specific text. Lexicon is built from the most frequent words in the keyword-specific text, making the bigram language model not only keyword specific, but also somehow task-specific. We call this model "Keyword-Adapted Language Model", and we test it in our LVCSR-based keyword spotter. Table 5.11 compares the adapted text for the keyword "toprak" with the general text as an example. General text corpus is reduced nearly 350 times by size when it's adapted to the keyword "toprak". Number of unique words is also reduced nearly 20 times. Although using reduced text data is expected to decrease the performance of a LVCSR-based speech recognizer, for KWS we expect the adapted text to increase the performance.

| | Big Text Corpus | Keyword- Adapted Corpus |
|---|---|---|
| # of Sentences | 5,556,449 | 15,692 |
| # of Unique Words | 1,170,526 | 60,192 |

Table 5.11. Comparison of the Texts

Again, lexicon sizes of 100, 500, 2000, and 5000 words are used to spot keywords, and spotting results of different lexicon sizes will be given in the following section.

### 5.3.1. Lexicon Containing 100 Words

In this experiment, lexicon contains the most frequent 100 words in the keyword adapted text and the language model is constructed from the bigram probabilities of the most frequent 100 words in that text. Spotting results of the individual keywords, and the overall system performance is given in Table 5.12.

| Keyword | Recall | Precision |
|---|---|---|
| Bu | 0.8718 | 0.1222 |
| En | 0.7818 | 0.0675 |
| Göz | 0.7536 | 0.1176 |
| Yıl | 0.8000 | 0.0205 |
| Gibi | 0.7545 | 0.1650 |
| Mavi | 1.0000 | 0.0709 |
| Ancak | 1.0000 | 0.1180 |
| Pazar | 0.8571 | 0.0866 |
| Sağlık | 1.0000 | 0.0415 |
| Toprak | 0.9405 | 0.3505 |
| Söyledi | 1.0000 | 0.0952 |
| Türkiye | 0.9166 | 0.1774 |
| Birlikte | 0.8666 | 0.2178 |

| | | |
|---|---|---|
| Olduğunu | 1.0000 | 0.1304 |
| Özellikle | 0.8333 | 0.1098 |
| Vurguladı | 1.0000 | 0.1846 |
| Belediyesi | 1.0000 | 0.0769 |
| Tarafından | 1.0000 | 0.1034 |
| **Overall** | **0.8676** | **0.1072** |

Table 5.12. Adapted Language Model With 100-word Lexicon Results

Overall recall is 0.86, and overall precision is 0.10. Overall recall is higher than that of the spotter that uses a general bigram language model, but overall precision is less. 8 of the 18 keywords have recall of 1.0, meaning that they are perfectly detected. Overall recall rate is the highest of all our experiments while the overall precision is the lowest. We are only 10 per cent sure that a detected keywords is a true hit. Overall recall and precision graphs of the individual keywords are given in Figure 5.19.

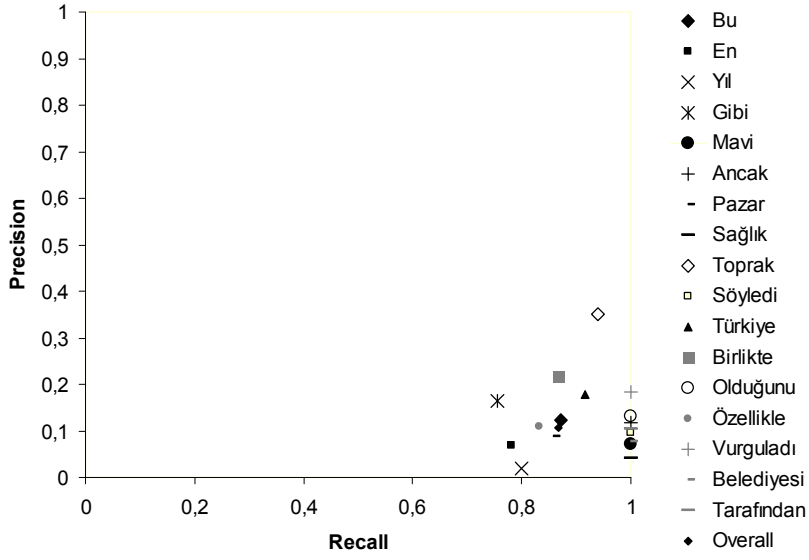


Figure 5.19. Performance Graph for 100-Word Lexicon

Average spotting time is 1998 seconds, which is nearly 6.84 times faster than real-time. This time is shorter than the average spotting time of the spotter that uses bigram language model. This system outperforms the bigram language model in overall recall and average spotting time, but its precision is less.

### 5.3.2. Lexicon Containing 500 Words

In this case, lexicon contains the most frequent 500 words in the keyword adapted text. Spotting results of the individual keywords, and the overall system performance is given in Table 5.13.

| Keyword | Recall | Precision |
|---|---|---|
| Bu | 0.8375 | 0.2126 |
| En | 0.7181 | 0.1567 |
| Göz | 0.7101 | 0.2816 |
| Yıl | 0.80000 | 0.0545 |
| Gibi | 0.7909 | 0.2969 |
| Mavi | 1.0000 | 0.2178 |
| Ancak | 1.0000 | 0.3243 |
| Pazar | 0.8571 | 0.3380 |
| Sağlık | 0.8570 | 0.2692 |
| Toprak | 0.9504 | 0.6233 |
| Söyledi | 1.0000 | 0.3369 |
| Türkiye | 0.9166 | 0.3437 |
| Birlikte | 0.8666 | 0.3451 |

| | | |
|---|---|---|
| Olduğunu | 0.9333 | 0.3111 |
| Özellikle | 0.7500 | 0.4090 |
| Vurguladı | 1.0000 | 0.5714 |
| Belediyesi | 1.0000 | 0.3000 |
| Tarafından | 0.9523 | 0.3846 |
| **Overall** | **0.8454** | **0.2467** |

Table 5.13. Adapted Language Model With 500-Words Lexicon Results

This time recall decreases to 0.84, and precision increases to 0.24. Like in the general bigram language model case, precision more than doubles itself with a 2 per cent decrease in recall, so this can be seen as a desirable situation. Overall recall and precision graphs of the individual keywords are given in Figure 5.20.

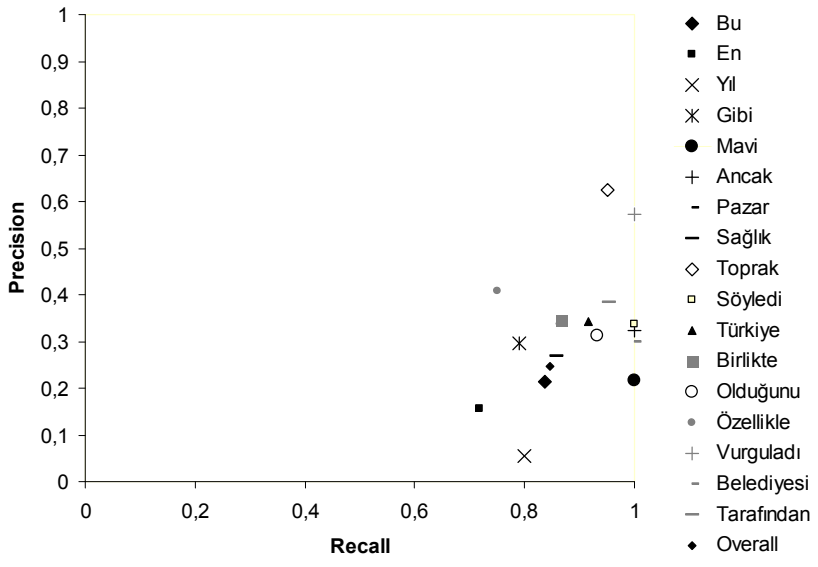


Figure 5.20. Performance Graph for 500-Words Lexicon

Average spotting time is 4867 seconds which is 2.8 time faster than real time.

### 5.3.3. Lexicon Containing 2000 Words

Now, lexicon contains the most frequent 2000 words in the keyword adapted text. Spotting results of the individual keywords, and the overall system performance is given in Table 5.14.

| Keyword | Recall | Precision |
|---|---|---|
| Bu | 0.8468 | 0.3672 |
| En | 0.6272 | 0.2923 |
| Göz | 0.7391 | 0.4678 |
| Yıl | 0.7333 | 0.1028 |
| Gibi | 0.7545 | 0.4462 |
| Mavi | 0.9487 | 0.3245 |
| Ancak | 1.0000 | 0.5217 |
| Pazar | 0.9285 | 0.7027 |
| Sağlık | 0.7916 | 0.5000 |
| Toprak | 0.9405 | 0.7600 |
| Söyledi | 1.0000 | 0.5294 |
| Türkiye | 0.9166 | 0,4400 |
| Birlikte | 0.7555 | 0.6415 |
| Olduğunu | 0.9333 | 0.5000 |
| Özellikle | 0.7500 | 0.5000 |
| Vurguladı | 1.0000 | 0.7058 |

| | | |
|---|---|---|
| Belediyesi | 1.0000 | 0.3750 |
| Tarafından | 0.9523 | 0.7692 |
| **Overall** | **0.8272** | **0.4161** |

Table 5.14. Adapted Language Model With 2000-Words Lexicon Results

This time recall decreases to 0.82, but precision increases to 0.41, which is a huge increase. Average spotting time is 9853 seconds which is 1.38 times faster than real time. This system looks very ideal for practical keyword spotting applications. Overall recall and precision graphs of the individual keywords are given in Figure 5.21.

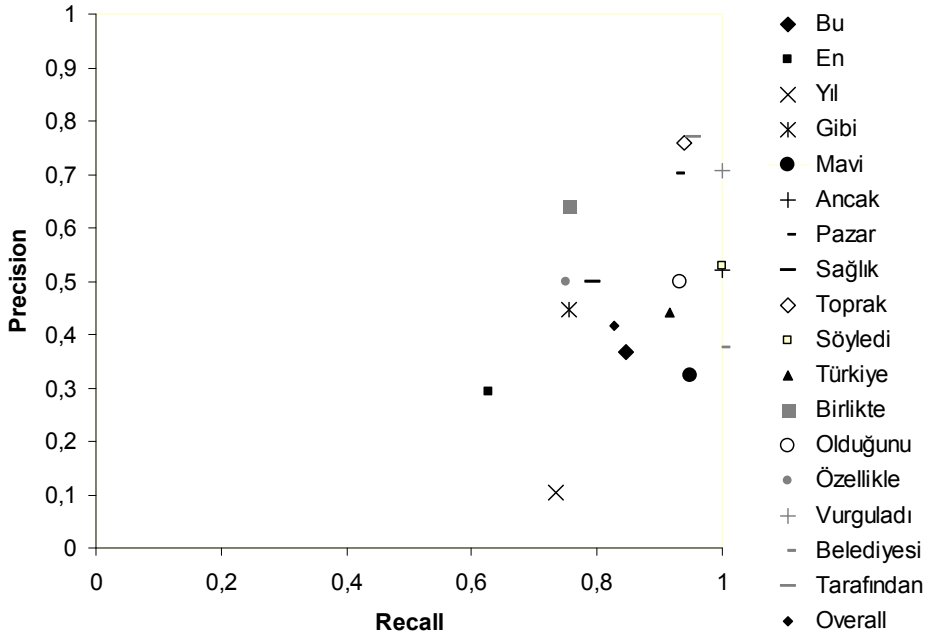


Figure 5.21. Performance Graph for 2000-Words Lexicon

### 5.3.4. Lexicon Containing 5000 Words

This time, lexicon contains the most frequent 5000 words in the keyword adapted text. Spotting results of the individual keywords, and the overall system performance is given in Table 5.15.

| **Keyword** | **Recall** | **Precision** |
|---|---|---|
| Bu | 0.8156 | 0.4780 |
| En | 0.6636 | 0.3882 |
| Göz | 0.7536 | 0.6341 |
| Yıl | 0.7333 | 0.1447 |
| Gibi | 0.8090 | 0.5493 |
| Mavi | 0.9487 | 0.3775 |
| Ancak | 1.0000 | 0.5901 |
| Pazar | 0.9285 | 0.7428 |
| Sağlık | 0.7500 | 0.5625 |
| Toprak | 0.9306 | 0.7642 |
| Söyledi | 0.5555 | 0.6428 |
| Türkiye | 0.9166 | 0.6111 |
| Birlikte | 0.7777 | 0.6730 |
| Olduğunu | 0.9333 | 0.4516 |
| Özellikle | 0.7500 | 0.6000 |
| Vurguladı | 1.0000 | 0.7500 |
| Belediyesi | 1.0000 | 0.5000 |
| Tarafından | 0.9523 | 0.9090 |

| | | |
|---|---|---|
| **Overall** | **0.8062** | **0.5147** |

Table 5.15. Adapted Language Model With 5000-Words Lexicon Results

As lexicon size is increased to 5000-words, decrease in overall recall, and increase in overall precision continues. Precision becomes 0.51 which means we are half a sure that a detected keyword is a true hit. Overall recall and precision graphs of the individual keywords are given in Figure 5.22.

Average spotting time is 20288 seconds, which is 1.47 times slower than real time, so this setup is inappropriate for practical applications.

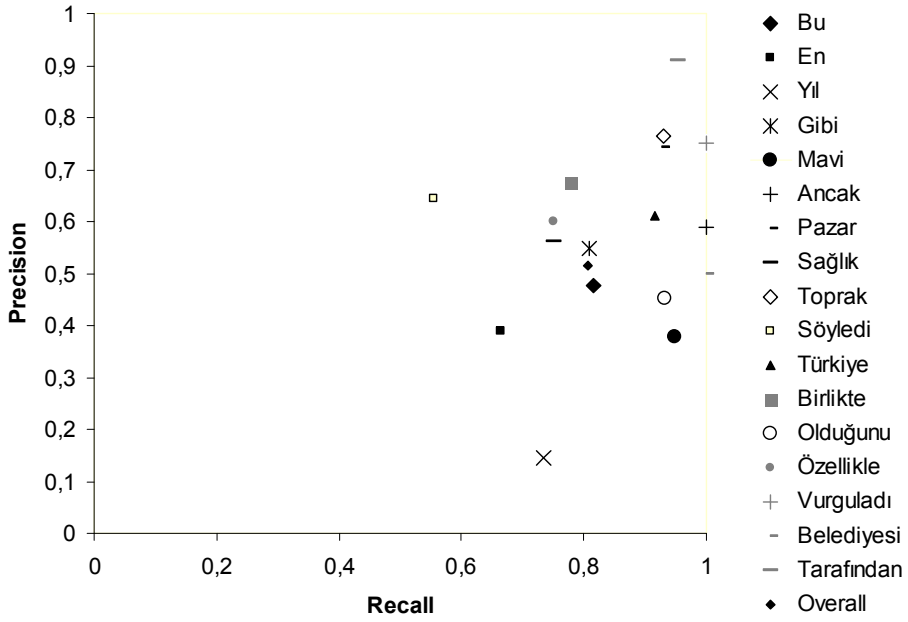


Figure 5.22. Performance Graph for 5000-Words Lexicon

### 5.3.5. Comparison of Results

In this experiment, we implemented a word spotter that uses a keyword adapted bigram language model, and tested it with different lexicon sizes. Recall and precision results of the individual keywords for different lexicon sizes are given in Figures 5.23 and 5.24.

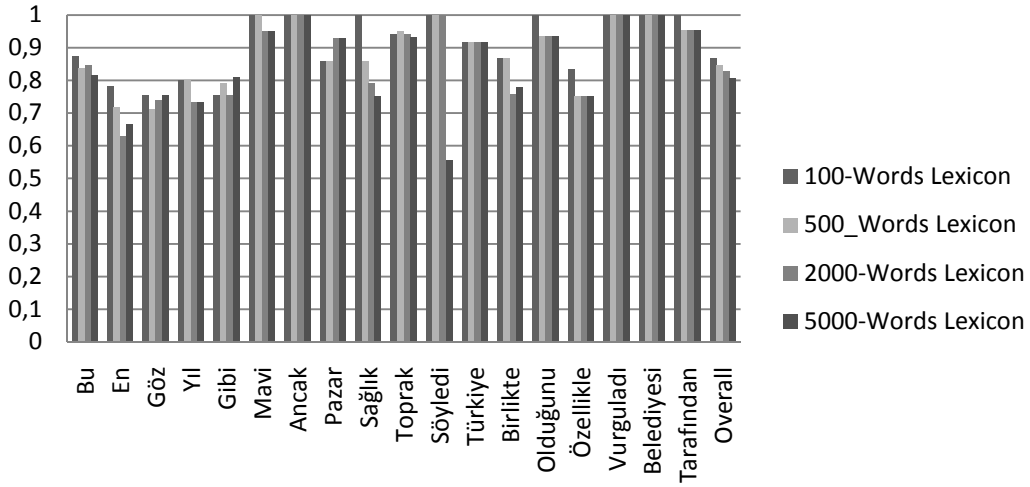


Figure 5.23. Recall Performance of the Individual Keywords

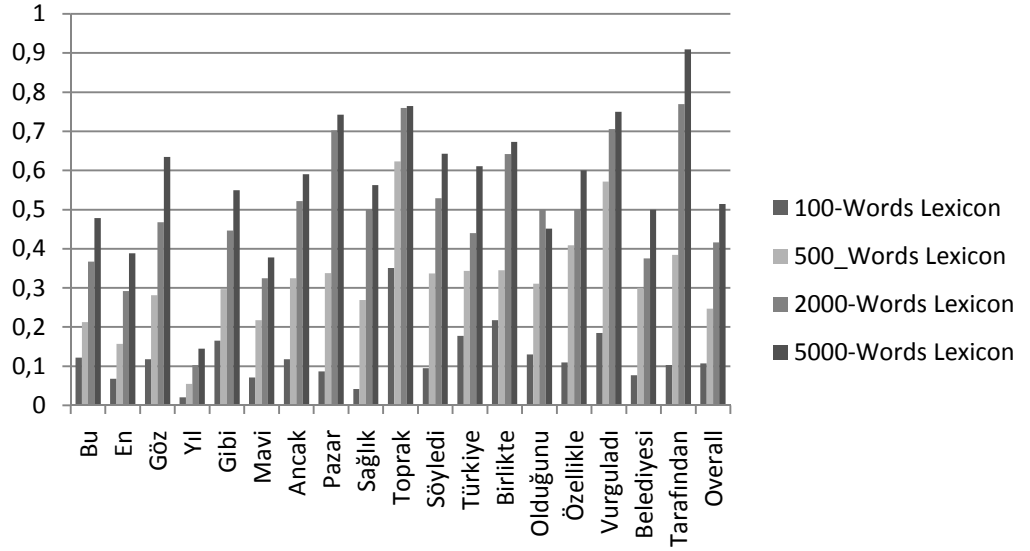


Figure 5.24. Precision Performance of the Individual Keywords

Recall and precision of the spotter that uses a keyword-adapted language model follow the same pattern as the previous spotters. Again, recall decreases, and precision

increases as we increase the number of words in the lexicon. “Ancak”, “vurguladı”, and “belediyesi” have recall of 1.0 for all the lexicon sizes. “Tarafından” has the highest precision of 0.9 for the 5000-words lexicon, and like in the previous experiments, “yıl” has the lowest precision for all of the lexicon sizes. Spotting times for different lexicon sizes and their ratio to real time are in Table 5.16.

| **Lexicon Size** | **Spotting Time(sec)** | **Real-Time Ratio** |
|---|---|---|
| 100 | 1998 | 0.14 |
| 500 | 4867 | 0.35 |
| 2000 | 9853 | 0.72 |
| 5000 | 20188 | 1.47 |

Table 5.16. Spotting Times

Like in the previous cases, spotting time of the lexicon having 5000 words exceeds real time, so this lexicon size is inappropriate for practical applications. KWS using keyword-adapted language model with 2000 words lexicon can be an optimal case because of good recall and precision rates, and faster computation time than real time.

### 5.3.6. Comparison with the Previous Models

If we compare the spotter that uses a keyword-adapted language model with the previous spotters, we see that it has the highest overall recall, and the lowest overall precision for all of the lexicon sizes, because of the heavily weighted keywords in the keyword adapted language model. Figures 5.25, and 5.26, shows the comparative results of these systems for the overall recall, precision respectively. Figure 5.27 shows the overall performance for 2000-words lexicon.

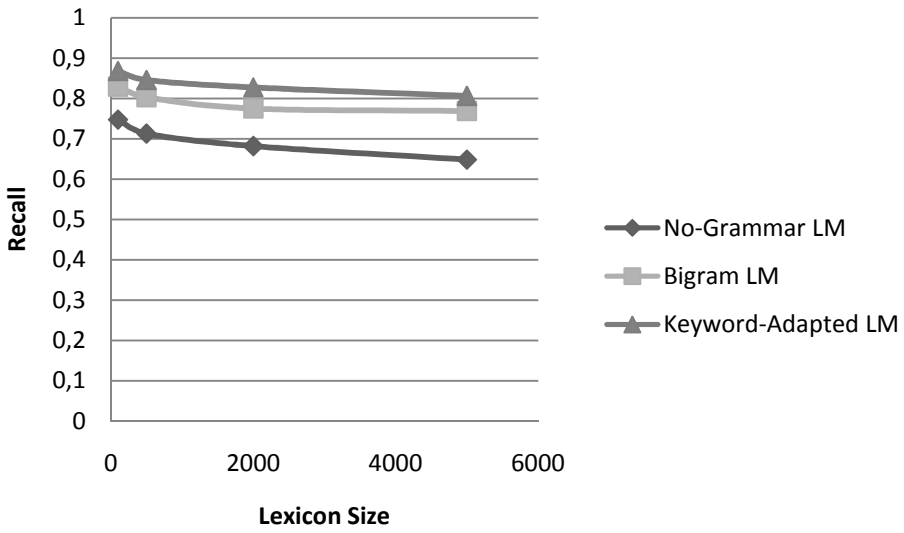


Figure 5.25. Comparative Overall Recall

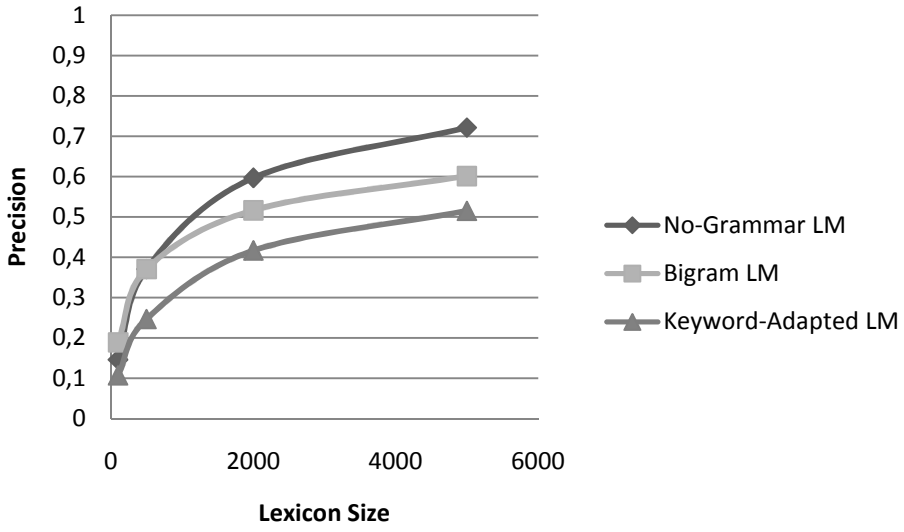


Figure 5.26. Comparative Overall Precision

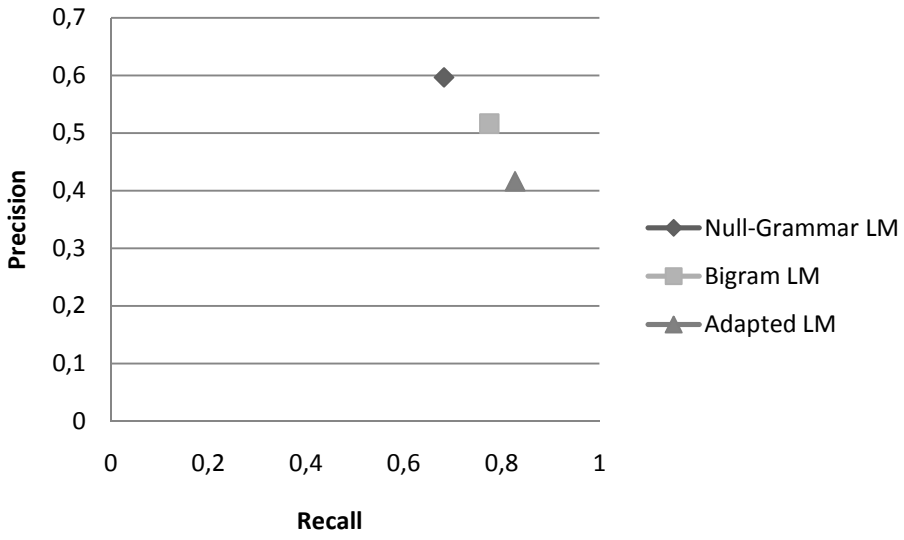


Figure 5.27. Comparative Overall Performance for 2000-words Lexicon

Overall recall of the spotter that uses a keyword-adapted language model is higher than the previous spotters. In this spotter, beside the language model, the recognition vocabulary is also keyword adapted, so it is a highly keyword-domain-specific system. As the domain of the language model has a very big impact on the recognition performance, increase in the number of true keyword hits is an expected result for this system.

Spotter that uses a keyword-adapted language model has the lowest precision performance among the spotters that we've implemented. Like in the general bigram case, the reason for that is the heavily weighted keywords in the adapted language model. As the language model is trained from a keyword-specific text in which the keywords are naturally amongst the most frequent words (usually they are the most frequent), keywords are very heavily weighted in the language model. The gap between the keyword and non-keyword weights are bigger in the keyword-adapted model case. So rather than the less weighted non-keywords in the lexicon, keywords are easily inserted in the place of out-of-vocabulary words, and the number of false alarms exceeds the other spotters' number of false alarms.

Figure 5.28. shows the comperative average spotting times of the spotters that use different language models. Spotter that uses keyword-adapted language model is faster than the other spotters.

If we compare the performance of the spotters that uses different language models, we see that the spotter that uses a keyword-adapted language model outperforms the other two spotters in both recall, and average spotting time., but its precision is less than both of the other spotters' precision. Precision can be increased by some language model processing techniques like smoothing, interpolation or adjusting the word insertion penalty as we will discuss in the next sections. If we want a faster spotter with high detection probability, and if we can eliminate the false alarms quickly, the spotter that uses a keyword-adapted language model would be our best choice.

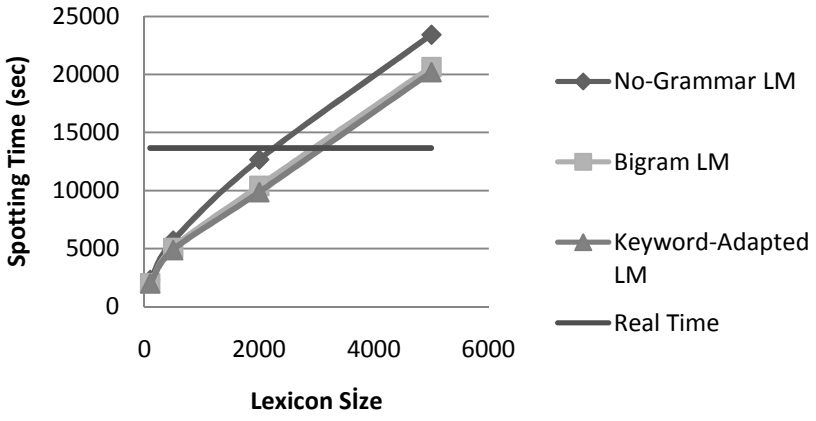


Figure 5.28. Comparative Average Spotting Times

## 5.4. Interpolating the Language Models

In the spotter that uses a keyword-adapted language model precision is very low because of the high number of keyword insertions. Heavily weighted keywords result in false alarms as they can be easily inserted in the places of out-of-vocabulary words. For 500-words lexicon, there is a 13 per cent decrease in precision of the spotter that uses a keyword-adapted language model with respect to the precision of the spotter that uses a

null-grammar language model, in which the keywords are equally weighted with the other lexicon words.

Different methods can be tried to solve the low precision problem of the spotter that uses keyword-adapted language model. One method is "smoothing" the language model. Smoothing tends to make distributions flatter. Low probabilities are raised, and high probabilities such as the keyword probabilities are lowered. A simple smoothing can be done by interpolating the general bigram language model with the keyword-adapted language model. This approach is similar to the one in [8], where a task-specific language model is smoothed with a task-independent language model.

Interpolation of the language models is done according to the following equation:

$$C(w1,w2)_{\text{int}} = C(w1,w2)_{keyword-adapted} + \alpha.C(w1,w2)_{general} \qquad (5.2)$$

where *C(.)* represents the bigram counts from the different text databases which are denoted by the subscripts. α is the weight that reduces the effective size of the general database. It prevents the general database counts to overwhelm the keyword-adapted database counts.

### 5.4.1. Experiment on 100-words Lexicon

An interpolated bigram language model is created for the lexicon containing 100 word. α in (5.2) is taken as 0.5. Results for the individual keywords are shown in Table 5.17.

| Keyword | Recall | Precision |
|---|---|---|
| Bu | 0.8486 | 0.1250 |
| En | 0.6636 | 0.0867 |
| Göz | 0.7536 | 0.1610 |

| | | |
|---|---|---|
| Yıl | 0.7333 | 0.0225 |
| Gibi | 0.8090 | 0.2916 |
| Mavi | 0.9487 | 0.0815 |
| Ancak | 1.0000 | 0.1417 |
| Pazar | 0.9285 | 0.1038 |
| Sağlık | 0.7500 | 0.0486 |
| Toprak | 0.9306 | 0.3798 |
| Söyledi | 0.5555 | 0.1384 |
| Türkiye | 0.9166 | 0.1932 |
| Birlikte | 0.7777 | 0.3135 |
| Olduğunu | 0.9333 | 0.1463 |
| Özellikle | 0.7500 | 0.1369 |
| Vurguladı | 1.0000 | 0.3000 |
| Belediyesi | 1.0000 | 0.6471 |
| Tarafından | 0.9523 | 0.1200 |
| **Overall** | **0.8272** | **0.1267** |

Table 5.17. Interpolated Language Model With 100-Words Lexicon Results

Figure 5.29 shows the overall recall and precision results of three systems. Although the precision increases from 0.10 to 0.12 with the interpolated language model, recall decreases from 0.86 to 0.82. This system neither has recall higher than that of the system that uses adapted language model, nor has precision higher than that of the system that uses general language model. It's recall is very near to the recall of the system using

general language model. So, using an interpolated language model is not beneficial in KWS if we want good recall rates.

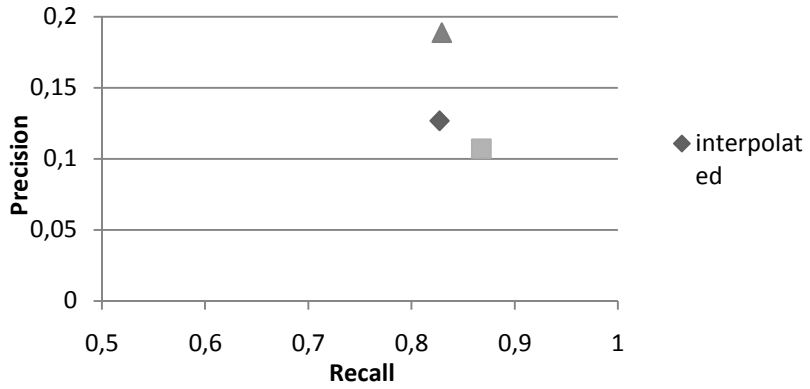


Figure 5.29. Comparative Results for Different Language Models

## 5.5. Using Word Insertion Penalty In KWS

Although smoothing will decreases the number of keyword insertions, it also increase the number of keyword deletions, so recall decreases. It is very important to control the trade-off between insertion and deletion of the keywords. In this section, we will try a simpler solution, in which this trade-off can be controlled more easily. We will use the "word insertion penalty" (WIP).

Keyword insertion is also related to the length of the keyword. Shorter keywords can be inserted more easily with small acoustic similarities. Word insertion penalty can be adjusted to different levels for keywords of different lengths to get the best precision performance. So apart from its easiness, using word insertion penalty to increase the precision of the spotter has other advantages over the other methods like smoothing. In the next section we will briefly describe word insertion penalty. Then experimental results using different insertion penalty levels will be given in the following sections.

### 5.5.1. Word Insertion Penalty

The basic formula of speech recognition is a simple multiplication of the probability terms, one standing for acoustic model, and the other standing for language model. This formula can be given as:

(5.3)

$$P(W \mid A) = P(A \mid W)P(W)$$

Taking the logarithms of the both sides, (5.3) becomes:

(5.4)

$$\log P(W \mid A) = \log P(A \mid W) + \log P(W)$$

As both probabilities are approximations, (5.4) should be modified to balance the absolute values of the probability terms, or the acoustic and language models [28]. A simple and common modification is given in (5.5):

$$\log P(W \mid A) = \log P(A \mid W) + \alpha \log P(W) - \rho \quad (5.5)$$

where α is called the language model weight and ρ is called the word insertion penalty. Language model weight and the word insertion penalty constitute the balance between the acoustic and linguistic models. This means that every language model probability $P(W)$ will be converted to with $(\alpha P(W) + \rho)$ before they are added to the acoustic probabilities of the tokens emitted from the word-end nodes.

Both language model weight and word insertion penalty are heuristic values, i.e. their optimal values can be determined experimentally. In the next section, we will carry on with spotting experiments using different word insertion penalty values.

### 5.5.2. WIP Experiments on Selected Keywords

In our previous experiments we've fixed the language model weight and we haven't used any word insertion penalty (-p option of the HTK function HVite was 0). Now, we again fix the language model probability, but we vary the word insertion penalty. So word

insertion penalty is left as the only variable that controls the trade-off between the insertion and deletions. Increasing the penalty results in more fewer insertions but also in more deletions. As we want fewer keyword insertions in our spotter, we should increase the word insertion penalty without too much keyword deletions.

In HTK [27], word insertion penalty is given in the decoding Hvite function using the "-p" option. Positive values after the "-p" options means that the insertion penalty is low and negative values means that the penalty is high. So in HTK, the value after the "-p" option and the word insertion penalty are inversely proportional. To make our results more apprehensible, we take the inverse approach: higher insertion values mean higher penalties. We expect more insertions when the penalty is -25, and less insertions when the penalty is 25. This is also consistent with (5.5).

Insertion is very highly dependent on keyword length as shorter words are inserted more easily. In the following subsections, we will try different insertion penalties for the selected keywords of 2, 3, 9 and 10 letters in length. Keywords are chosen as this to investigate the effects of WIP on extreme cases. Word insertion penalty values of -25, -20, -15, -10, 0, 5, 10, 20, and 25 will be tested, where 0 means that we are not using any word insertion penalty as in the previous experiments. Keyword adapted language model with a lexicon containing 100 words is used in the experiments. Figures from 5.30 to 5.37 show the spotting performance of the selected keywords (“bu”, “en”, “göz”, “yıl”, “özellikle”, “vurguladı”, “tarafından”, “belediyedi”, respectively) with respect to different WIP levels.

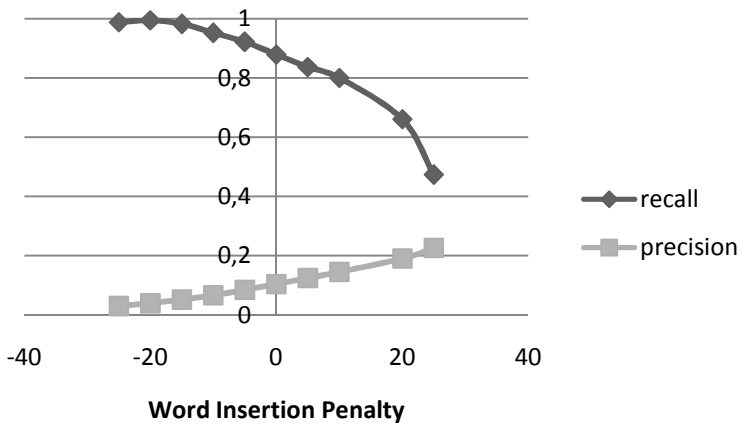


Figure 5.30. Word Insertion Penalty Results For “bu”

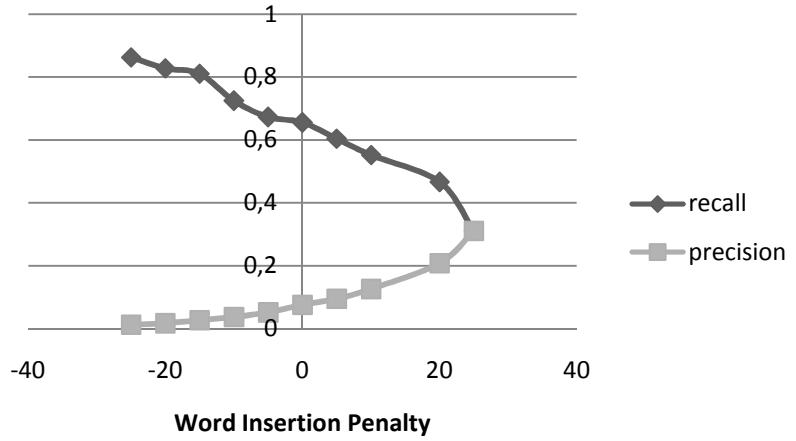


Figure 5.31. Word Insertion Penalty Results For “en”

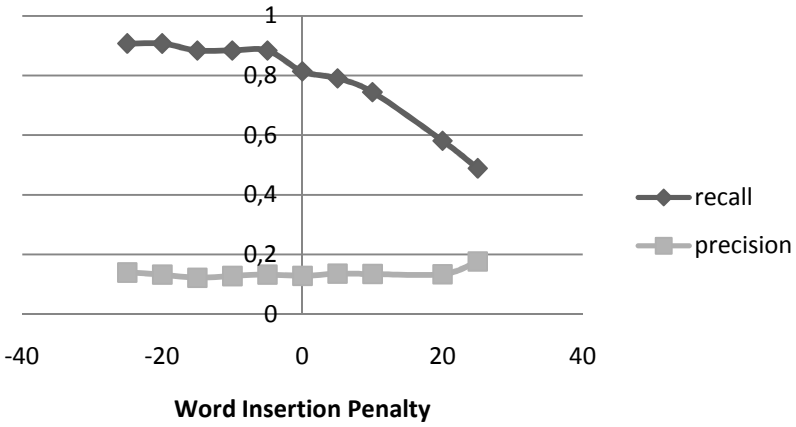


Figure 5.32. Word Insertion Penalty Results For “göz”

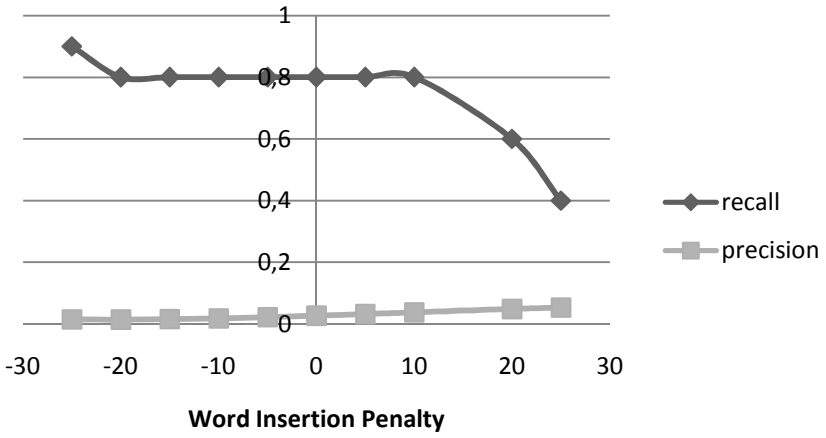


Figure 5.33. Word Insertion Penalty Results For “yıl”

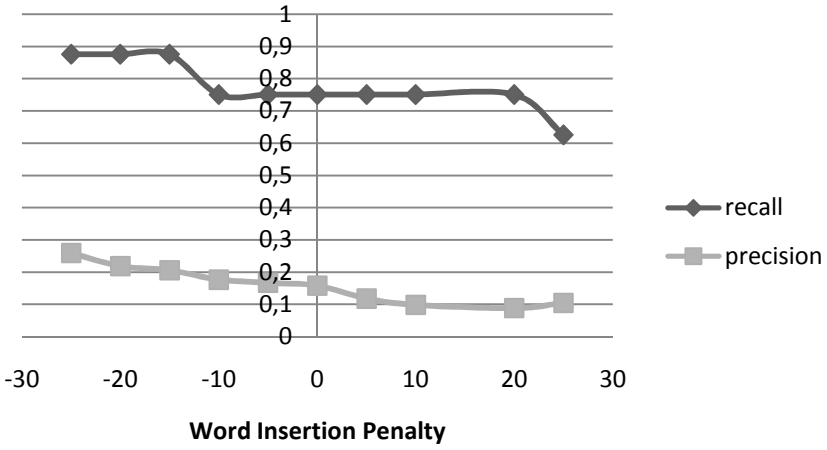


Figure 5.34. Word Insertion Penalty Results For “özellikle”

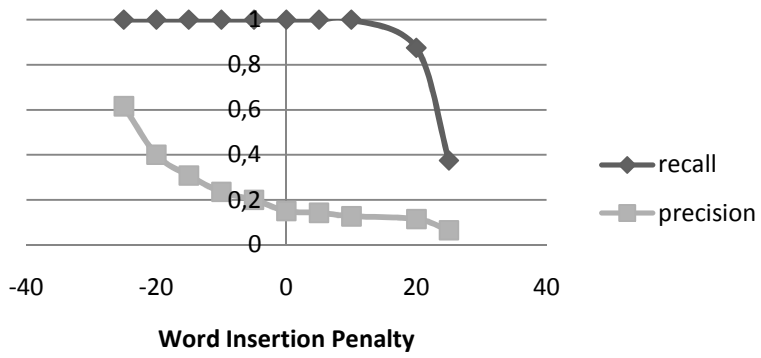


Figure 5.35. Word Insertion Penalty Results For “vurguladı”

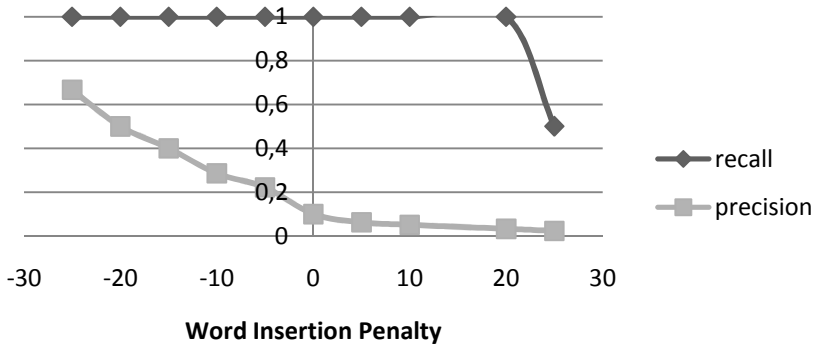


Figure 5.36. Word Insertion Penalty Results For “belediyesi”

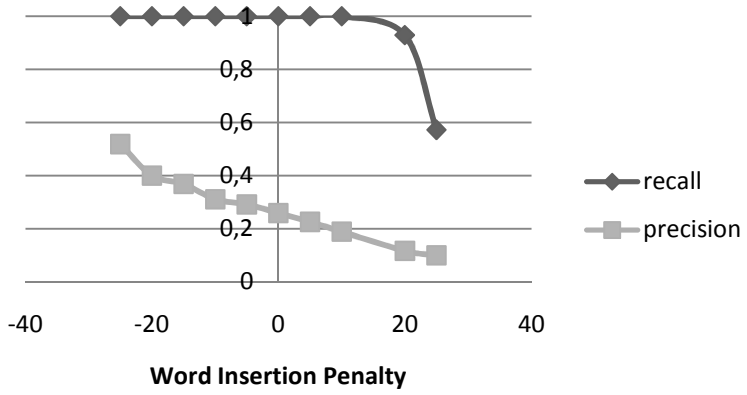


Figure 5.37. Word Insertion Penalty Results For “tarafından”

Shorter and longer keywords follow different spotting result patterns with respect to WIP. Shorter keywords are more potential for insertion errors, as they contain less acoustic information and they can be inserted easily with any small similarity. By looking at the figures from 5.30 to 5.33, we see that increasing the WIP results in a decrease in recall and increase in precision for the shorter words. This pattern is more visible for the shortest keywords, “bu” and “en”. When we increase the WIP, precision increases as the number of insertions decrease as expected, but this also results in decrease in recall because of the deletion of some of the correctly spotted keywords. Trade-off between insertion and deletion should be adjusted WIP carefully according to the needs of the KWS application.

We get more interesting results for longer keywords. On the contrary to shorter keywords, this time precision decreases as WIP increases. Increasing the WIP results in less insertions of the shorter words, and this may result in insertion of the longer words in place of the non-inserted shorter words. Recall performance for the longer keywords is less sensitive to WIP, as recall starts to decrease only for the high WIP values. Deletion of the longer words is harder, and it can occur only when using high WIP. So, it is better to use less WIP for longer keywords.

# 6. CONCLUSION AND FUTURE WORK

In this thesis, we implement a LVCSR based KWS system, and test three different language models, one of which is proposed by us. Testes language models are null-grammar language model, general bigram language model, and the proposed, keyword-adapted language model.

For each of the language models, four different lexicon sizes are tested. It is observed that precision increases with increasing lexicon size, but recall decreases. Decrease in the recall is caused by the increase in the acoustic variability of the system. Precision increases as some of the keyword insertions are directed into correct words as these words are now included in the extended lexicon. It is also observed that using lexicons having more than 2000 words is inappropriate for practical KWS applications because of their long spotting times. Spotting times are nearly 6 times faster than real time for small sized lexicons as 100 words. Lexicon sizes between 500 and 2000 words are optimal for LVCSR based KWS.

If we compare the spotting performances of the used language models, proposed keyword-adapted model gives the best recall and spotting time results, but it lacks in precision. Recall absolutely increases nearly 13 per cent from the base model. This result is expected as the model is adapted for the keywords. Highest recall obtained is 0.86 of the keyword adapted spotter that uses 100-words lexicon.

Low precision is caused by the high language weights of the keywords in the adapted model. Best precision is obtained when the null-grammar model is used. Highest precision obtained is 0.72 of the null-grammar spotter that uses 5000-words lexicon. We can say that keyword-adapted spotter with 2000-words lexicon offers the optimal KWS solution with precision of 0.41 and recall of 0.82. It is also 1.38 times faster than real time.

To improve the precision of the proposed model, two simple techniques are tried. First technique is to use an interpolated language model. High keyword probabilities in the

adapted model are smoothed when the model is interpolated with the general bigram model. Although precision is increased with the interpolated model, recall decreases so that this technique doesn't seem beneficial. The other technique to improve precision is adjusting the word insertion penalty. It is observed that decreasing the word insertion penalty results in increase in the precision for the longer keywords. For shorter keywords, increasing the word insertion penalty increases the precision, but it also decreases the recall. So adjusting the word insertion penalty is more guaranteed to improve the system performance for the longer keywords.

LVCSR based KWS can be improved by using more sophisticated language models. Even the usage of higher detailed N-grams such as trigrams can improve the spotting performance. Also a confidence measure related with the acoustic keyword probabilities may be added to decrease the number of false alarms.

Although they've been around for many years, KWS systems are recently started to use in practical applications as their performances and costs become just suitable. KWS is mainly used in call centers and security applications. In the future, KWS can be combined with some other speech applications to offer more sophisticated solutions. One possible application is costumer satisfaction pursuit in call centers, where KWS can be combined with emotion recognition to decide if the calling costumer is happy or angry.

# APPENDIX A: PROGRAM GUI

Screenshots of the implemented GUI-based computer program is shown in Figures A.1 through A. Figure A.1 shows the screenshot of the main interface. Keywords entry window is on the top, and the sound file entry window is in the middle. Bottom window is to show the program status. Combo box at the keyword entry window is to input the type of the keyword that is newly introduced as in explained in Section 3.3.1. A synthetic text is built according to the selected type for the language modeling.

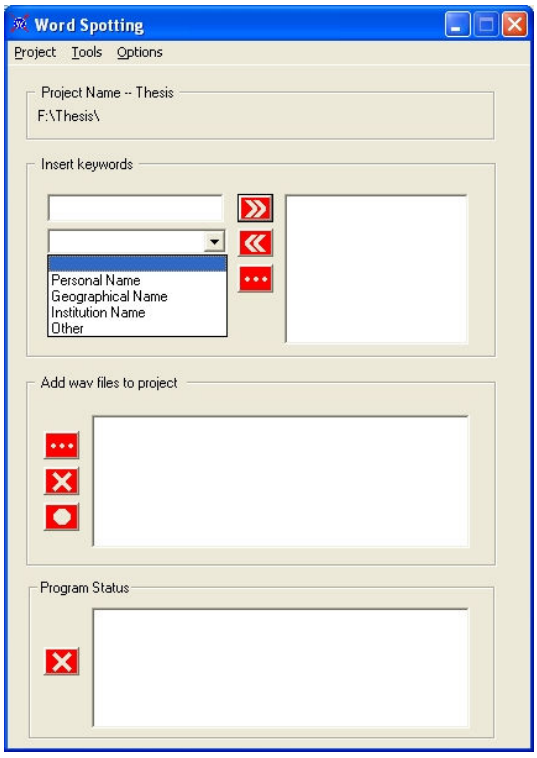


Figure A.1. GUI of the Program

Figure A.2 shows the options window of the program. There are 3 options, each one with a slider control. Options are the “Lexicon Size”, “Pruning Threshold” and “Language Model Multiplier”. Pruning threshold controls the “–t” option of the HTK tool HVITE, and language model multiplier controls the “-s” option. Lexicon size varies from 100 to 5000 as in the experiments.

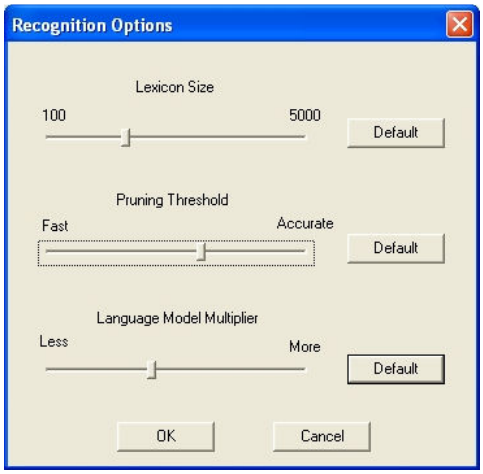


Figure A.2. Program Options

Figure A.3. shows the result screen after the spotting. In this screen, spotted keyword, and the wav file are shown together with the display of the sound waveform. Place of the spotted keyword is highlighted in the waveform. In this screen, the user can play the shown part of the sound file to check if the spotted keyword is a true hit. By using the "extend" edit box, the user can extend the part of the sound file that is shown.

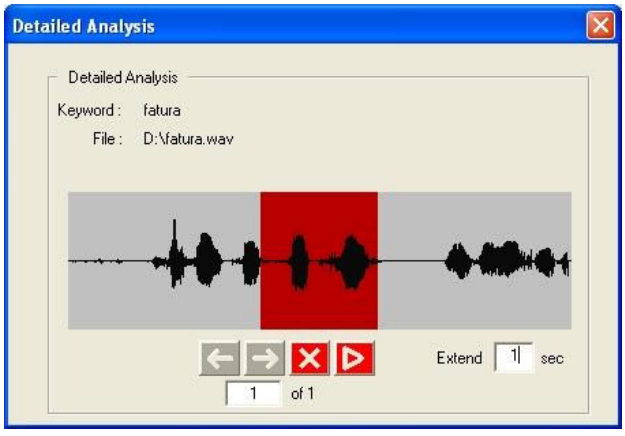


Figure A.3. Spotting Results Screen

Final reporting of the program is as follows:

*Keyword search has been done in 1 sound files:*
*1. D:\fatura.wav*

*Word Spotting Report:*
*"fatura" is found 1 times*

*1. At the file: D:\fatura.wav*
*Start Time: 1.12 sec End Time: 1.73 sec Verification Status: TRUE*

# REFERENCES


1. Bridle J. S., "An Efficient Elastic-Template Method For Detecting Given Words In Running Speech", *Brit. Acoust. Soc. Meeting,* Pg. 1-4*,* 1973

2. Bridle J., M. Brown, and R. Chamberlain, "An Algorithm For Connected Word Recognition", *Acoustics, Speech, and Signal Processing, IEEE InternationalaConference on ICASSP'82.,* Volume 7, May 1982 Pg. 899 – 902

3. Higgins A. L. and R.E. Wohlford, "Keyword Recognition Using Template Concenation", *Proc. Int. Conf. on Acoust., Speech, and Sig. Processing,* May 1989

4. Rohlicek J. R., W. Russel, S. Roucos, and H. Gish, "Continuous Hidden Markov Modeling For Speaker-indepented Word Spotting", *Proc. Int. Conf. on Acoust., Speech, and Sig. Processing,* May 1989

5. Rose R. C. and D. B. Paul, "A Hidden Markov Model Based Keyword Recognition System", *Proceedings of the 1990 International Conference on Acoustics, Speech and Signal Processing*, Vol. 2, pg. 129-132, April 1990.

6. Wilpon J. G., L. R. Rabiner, C. H. Lee, and E. R. Goldman, "Automatic Recognition of Keywords in Unconstrained Speech Using Hidden Markov Models", *1990 IEEE Trans. ASSP,* Vol 38. No. 11, pg. 1870-1878

7. Wilpon J. G., L. G. Miller, and P. Modi, "Improvements and Applications for Key Word Recognition Using Hidden Markov Model Techniques", *1991 IEEE Trans. ICASSP,* pg. 309-312

8. Weintraub, M., "Keyword-spotting Using SRI's DECIPHER Large-Vocabulary Speech-Recognition System"**,** *ICASSP-93*

9. Duran, Ş., *Keyword Spotting Using Hidden Markov Model's*, M.S. Thesis, Boğaziçi University, 2001.

10. Yapanel, U., *Garbage Modeling Techniques For a Turkish Keyword Spotting System*, M.S. Thesis, Boğaziçi Üniversity, 2000.

11. Çarki K., P. Geutner, and T. Schultz, "Turkish LVCSR: Towards better Speech Recognition for Agglutinative Languages", *ICASSP*, 2000.

12. Dutağacı H., and L. M. Arslan, "A Comparison of Four Language Models For Large Vocabulary Turkish Speech Recognition", *ICSLP,* 2002.

13. Arısoy E., and L. M. Arslan, "Turkish Dictation System for Broadcast News Applications", *Signal Processing and Communications Applications Conference, 2005. Proceedings of the IEEE 13th,* 16-18 May 2005, Pg. 629 – 632.

14. Erdogan, H., O. Buyuk, and K. Oflazer, "Using Hybrid Lexicon Units and Incorporating Language Constraints in Speech Recognition", *Signal Processing and Communications Applications Conference, 2005. Proceedings of the IEEE 13th,* 16-18 May 2005, Pg. 111 – 114.

15. Yamashita Y., "Topic Recognition For News Speech Based On Keyword Spotting", *Proc. of 5th International Conference on Spoken Language Processing (ICSLP '98),* Vol 3, pp. 839-842.

16. McDonough J., and H. Gish, "Issues In Topic Identificationon the Switchboard Corpus", *Proc. Of ICSLP '94,* pp. 2163-2166.

17. Godfrey J., E. Holliman, and J. McDaniel, "SWITCHBOARD: Telephone Speech Corpus for Research Development", *ICASSP 1992,* Volume I, pp 517-520.

18. Ariki Y., and Y. Sugiyama, "A TV Retrieval System With Interactive Query Function", *Cooperative Information Systems, 1997. COOPIS '97., Proceedings of the Second IFCIS*, 24-27 June 1997 Pp. 184 – 192.

19. Gelin P., and C. J. Wellekens, "Keyword Spotting For Video Soundtrack Indexing", *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings.,* Volume 1, 7-10 May 1996 Pp. 299 – 302.

20. Logan B., J. M. van Thong, and P. J. Moreno, "Approaches to Reduce the Effects of OOV Queries on Indexed Spoken Audio", *IEEE Transactions on Multimedia,* Vol 7. No5, October 2005.

21. Alon G., "Key-Word Spotting – The Base Technology for Speech Analytics", *White Paper, NSC*, 2005.

22. Young, S., "A Review of Large-vocabulary Continuous Speech Recognition". *IEEE Signal Process. Magazine 13 (5),* Pp 45-57.

23. Huang A., and Hon, *Spoken Language Processing,* Prentice Hall.

24. Rabiner L. R., and B. H. Juang, "An Introduction to Hidden Markov Models", *IEEE ASSP Magazine*, January 1986.

25. Rosenfeld R., "Two Decades of Statistical Language Modeling: Where Do We Go From Here?", *Proceedings of the IEEE,* Vol. 88, No 8. 2000.

26. Buyuk O, A. Haznedaroglu, and L. M. Arslan," Turkish Speech Recognition Software with Adaptable Language Model", *IEEE 15th Signal Processing and Communication Applications Conference*, 2007.

27. S. Young, D. Ollason, V. Valtchev and P. Woodland, *The HTK Book (HTK Version 3.2),* Entropic Cambridge Research Laboratory, 2002.

28. Ogawa A., K. Takeda, and F. Itakura, "Balancing Acoustic and Linguistic Probabilities", *Proc. ICASSP 98*, pp. 181–184

29. vanRijsberngen, C. J. 1975. *Information Retrieval*. London:Butterworths.

30. Jelinek F., B. Merialdo, S. Roukos, M. Strauss, "A Dynamic Language Model for Speech Recognition", *Proceedings of the Speech and Natural Language DARPA Workshop,* February 1991

31. Pietra D. S., R. Mercer, S. Roukos, "Adaptive Language Modeling Using Minimum Discriminant Estimation", *Proceedings of ICASSP '92*, pp 633, 636

32. Lau R., R. Rosenfeld, and S. Roukos, "Adaptive Language Modelling Using the Maximum Entropy Approach", *ARPA Human Language Technologies Workshop,* 1993

33. Chu, S. M., H. Kuo, Y. Y. Liu, Y. Qin, Q. Shi, G. Zweig, "The IBM Mandarin Broadcast Speech Transcription System", *ICASSP 2007*, Volume 2, 15-20 April 2007 pp II-345 - II-348