VOICE DRIVEN KEYWORD SPOTTER

by

Onur Çengelci

B.S., Electrical and Electronics Engineering, Boğaziçi University, 2003

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

Graduate Program in Electrical and Electronics Engineering
Boğaziçi University

2006

ACKNOWLEDGEMENTS

I would like to thank Assoc. Prof. Levent M. Arslan and Assist. Prof. Murat Saraçlar for their understanding and support during the difficult study period. I could not complete this thesis without giving up my professional career if they were not such tolerable to me. I would not have the intention to complete this tiring task without their creative ideas and the encouragement they provided.

I would also like to thank Prof. Fikret Gürgen, Assist. Prof. Burak Acar and Assist. Prof. Kerem Harmancı for their gentle attitude and participation in my thesis committee.

I should shout my special thanks to my daddy and mommy for their moral support. I also send my very special thanks to Dear Sister, Ömür, for her limitless understanding and motivation.

But first of all, I would like to thank my teachers and instructors at Boğaziçi University for the qualified education they provided. I could not start this M.S. program and thus could not create this thesis if I did not have the information and background I obtained from them during my B.S. program.

ABSTRACT

VOICE DRIVEN KEYWORD SPOTTER

We designed a voice driven keyword spotter. To improve the success of the system, we made use of synthetically generated voice inputs in addition to natural voice inputs and used approximate string matching instead of exact string matching.

Classical keyword spotters are mostly text driven. However, we have taken the input in the form of voice. Different people may pronounce the same keyword in different ways because effects such as gender, age, nationality, intonation, accent, emotional mood, environment, noise etc. play an important role on pronunciation. Even the samples of a keyword taken from the same person at different times may be different. Therefore, driving the keyword spotter with voice instead of text provides us with a source of variety. This variety increases the probability of spotting the keyword.

Classical keyword spotters are mostly language dependent. In our spotter, many phoneme recognizers trained with different languages may be used in co-operation. We believe that, this ability of our spotter is highly likely to make it language independent. Even if a phoneme recognizer of only one language is used, it will make similar errors for both the input side and the search database side and the system may still have the chance of being language independent to some extent.

As we take the input in voice format, we have the chance of collecting many samples of the keyword and producing their appropriate transformations. This ability of our spotter alleviates speaker dependency.

ÖZET

SES GİRİŞLİ ANAHTAR KELİME TARAYICI

Ses beslemeli bir anahtar kelime tarayıcısı tasarladık. Sistemin başarısını artırmak için, doğal ses girişlerine ek olarak sistem tarafından doğal seslerin başkalaştırılması yoluyla yapay olarak yaratılan ses girişlerini ve birebir karakter dizisi eşleştirmesi yerine benzer karakter dizisi eşleştirmesini kullandık.

Geleneksel anahtar kelime tarayıcıları, daha çok karakter dizisi beslemelidirler. Oysa biz tarayıcıyı ses ile besledik. Farklı insanlar aynı kelimeyi farklı şekillerde söyleyebilirler. Söylemde cinsiyet, yaş, milliyet, vurgulama, aksan, duygusal durum, çevre, gürültü vb. etkenlerin önemli bir etkisi vardır. Hatta, aynı kelimenin aynı insandan farklı zamanlarda alınan örnekleri bile farklılık gösterebilir. O nedenle, anahtar kelime tarayıcıyı, karakter dizisi yerine sesle beslemek bize bir tür çeşitlilik kaynağı sağlar. Bu çeşitlilik anahtar kelimeyi bulma olasılığımızı artırır.

Geleneksel anahtar kelime tarayıcılar çoğunlukla dil bağımlıdırlar. Geliştirdiğimiz tarayıcıda, farklı dillere ait bir çok ses birimi tanıyıcısı bir arada bütünleşik olarak çalışabilmektedir. Sistemimizin bu özelliğinin onu büyük olasılıkla dilden bağımsız yapacağını düşünmekteyiz. Bir tek dile ait ses birimi tanıyıcısı kullanılsa bile, ses birimi tanıyıcısı hem girdi tarafında için hem de tarama yaptığımız tarafta benzer hataları yapacak ve anahtar kelime tarayıcısı yine büyük ölçüde dilden bağımsız çalışabilme şansına sahip olacaktır.

Girdiyi ses formatında aldığımız için, anahtar kelimenin birçok örneğini toplama ve bu örneklerin uygun dönüşümlerini yaratma şansımız vardır. Bu özellik, sistemin konuşmacıya olan bağımlılığını azaltır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS / ABBREVIATIONS

C_{PFS}	Coefficient of phoneme frequency score
C_{EDS}	Coefficient of edit distance score
C_{MS}	Coefficient of matching score
EDS_N	Normalized edit distance score
F	F-Measure
MS_N	Normalized matching score
OS_N	Normalized overall score
PFS_N	Normalized phoneme frequency score
ASR	Automatic speech recognition
FAR	False alarm rate
FL	Frame length
FOM	Figure of merit
HMM	Hidden Markov model
KWS	Keyword spotting
MIP	Minimum intersection per cent
ROC	Receiver operating characteristics
SDR	Spoken document retrieval
SPCM	Statistical phoneme confusion matrix
SPGCM	Statistical phoneme group confusion matrix
WER	Word error rate

1. INTRODUCTION

Speech has begun to be used as one of the newest ways of data transfer between computers and people. It seems that it will be more and more popular day by day, because humans, by nature, prefer verbal communication to any other way of it. In the past, we used to dial the numbers on our telephones to transfer data to a bank's computer system, but now we can just "**say**" the information to computers by means of IVR (Interactive Voice Recognition) systems.

The success of computers in understanding human speech is growing day by day. However, this new technology has brought reliability problems with it. Therefore, there is need to record and store critical speech based communication between people and computers. Again, for security and reliability reasons in some cases, there is need to store speech based communication between people. That need in fact requires a tool to perform searching in those large amounts of stored speech data. Topics such as **Spoken Document Retrieval** (SDR), **Utterance Retrieval** (UR) and **Keyword Spotting** (KWS) focus on that area of speech technology.

1.1. Applications of KWS

KWS has large market potential including security concerns, call centers, IVR systems, advertising market, research market, etc.

The global security market tries to decrease the threat of worldwide terrorism. The main benefit of KWS in security market is that it can be used to search for specific keywords, reminding of terrorism, in suspicious phone and mobile phone conversations, radio and TV broadcasts, etc. **[1]**

In Call Center market, KWS can be used to analyze conversations between call center agents and customers to measure customer satisfaction and to allow the directors to join problematic cases in real-time. By means of KWS system, a call center director can be

informed of a probable argument between an agent and a customer and this argument can be prevented. **[1]**

Many companies using IVR systems to serve their customers also record the conversations between their customers and their IVR system to satisfy ISO standards. This results in a large amount of stored conversation data. KWS could be very helpful when there is a need to search for a specific keyword or sentence in this stored data. This way, abuse of the IVR system could be prevented as well.

KWS systems may be very beneficial for advertising market, too. An advertisement company can use this system to detect for how many minutes an advertisement they produced is broadcast on TV or radio, provided that the advertisement includes speech. Similarly, a research company could use a KWS system to detect how frequently a topic or a keyword is broadcast on TV or radio.

1.2. Literature Review

Especially in the last ten years, a lot of research has been done and many papers have been written on the topic of KWS.

Chang and Lippmann **[2]** proposed to use ***artificially generated data*** to improve KWS performance. They thought that lack of training data is an important problem which limits KWS performance. On the other hand, collecting enough training data is difficult and expensive. Therefore, they proposed that existing training data can be modified using ***voice conversion*** to increase the training data size and KWS performance. However, their system was ***text input*** and they applied voice conversion to the speech recordings in the search database not to the voice input as we propose to do. To evaluate the performance, they used **Figure of Merit (FOM)**. FOM is the average word detection rate over false alarm rates ranging from 0 to 10 false alarms per keyword per hour. They concluded that adding artificial training data increases FOM.

Nishizaki and Nakagawa **[3]** proposed a Japanese SDR system that uses ***voice input queries*** which are transcribed with N-best hypothesis. Before retrieval process, they group together keyword candidates with high similarity based on mutual information between keywords. They made tests with both isolated keywords and spontaneously spoken queries and evaluated the performance with **F-Measure**. F-Measure is the geometric mean of precision and recall. They concluded that spontaneously spoken queries gave better keyword detection rates than isolately spoken keywords.

Kaiser **[4]** proposed a speech input KWS system where he made use of the idea proposed by Chang and Lippmann. He created two new utterances for each stored utterance in the search database by linearly expanding and compressing the short term spectral envelope (**frequency warping**). The recognizer was HMM based and used **keyword models** and **garbage (filler) models**. He used FOM to evaluate the performance and concluded that lower FOM is achieved when frequency warping was applied.

Saraclar and Sproat **[5]** claim that taking the single best output of ASR and performing text retrieval for this output is not reasonable if WERs are high. They propose an indexing procedure for KWS that works on **ASR lattices** instead of single best ASR output. They claim that using ASR lattices makes the system more robust to recognition errors. Their system is not a voice input system but a text input query system. They tested both word and sub-word indices and showed that using a hybrid scheme produces the best results. They used **word error rate (WER)** for evaluating ASR performance and **F-Measure** for evaluating retrieval performance. They concluded that the proposed indexing procedure can improve F-Measure by five points compared to cases where only single best ASR output was used.

Junkawitsch, Neubauer, Höge and Ruske **[6]** proposed a new text input, HMM (with modified Viterbi) based algorithm with **pre-calculated optimal thresholds**. In this algorithm, there is no necessity for filler (garbage) models or language models. However, optimal decision thresholds should be determined for each keyword. Normalized HMM score of each word must be compared with a keyword specific optimal threshold using a modified form of Viterbi search. They concluded that tests with spontaneous speech

databases yield 73.9 per cent FOM when using context-dependent HMMs and 58.5 per cent FOM when using context-independent HMMs.

Zheng, Li, Song and Xu **[7]** proposed a text input, two step KWS method based on ***context-dependent a posteriori probability (CDAPP)*** to eliminate the adverse effects of **keyword weighting**. They used **Extended Template Matching (ETM)**. Traditional **template matching (TM)** achieves good results when input utterances match well with the templates but the performance decreases for mismatched utterances. On the other hand, classical KWS, based on a network of keyword and garbage (filler) models, enable flexible speech input. ETM takes advantages of both TM and classical KWS and at the same time includes online filler models. They concluded that the experimental results are encouraging and can be improved by integrating a language model, dealing with insertion and deletion errors and using a confidence measure.

Duran **[14]**, used Hidden Markov Models for keyword spotting problem. He investigated some models for garbage models for keyword spotting and tried to find some confidence measure for detection of out of vocabulary words in an isolated word recognizer. He concluded that using the monophone models of the words and one-state 16-mixture general garbage models with different bonus values gives the best performance. He evaluated the performance of monophone models as garbage model for isolated word recognition. He concluded that likelihood and phoneme duration are important for obtaining a good confidence measure. He measured the performance of the system by means of Receiver Operating Characteristic (ROC) curves which show the probability of detection versus probability of false alarm.

Knill and Young **[16]** also used HMMs for speaker dependent keyword spotting problem. Their baseline word spotter consists of a parallel network of keyword and background filler models. They used a second pass using the filler models only to re-score the putative keyword hits. They measured word spotting performance using whole word and sub-word keyword models and concluded that sub-word models yield a higher hit rate, especially before the first false alarm occurs. The best hit rate they obtained before the first false alarm occurred was 51.1per cent when word models were used and 59.9 per cent when sub-word models were used.

Yapanel **[15]**, investigated the spotting performance of different garbage modeling techniques for out of vocabulary word modeling problem. He developed a statistical model for Turkish language and determined the most frequent triphones for Turkish language. Instead of using a two pass algorithm commonly used in recognition problems, he tried "*a new one pass algorithm*". He compared the performances of one-state general garbage model, phone class garbage model and monophone garbage model and concluded that monophone garbage model gives the best performance. Probability of detection was 0.29 for one-state general garbage model, 0.38 for phone class garbage model and 0.46 for monophone garbage model at 45 false alarms per keyword per hour.

1.3. Benefits of Our KWS System

In this thesis, we propose a robust method for KWS. We concentrated on the main weaknesses of traditional KWS systems and proposed a new method that compensates for those weaknesses.

The main weakness of traditional KWS systems is that they take text input, which is usually the correct phonetic transcription of the keyword given in the dictionary. They process the records in their search database through the ASR engine, which may or may not give the correct phonetic transcription but then they compare the transcription obtained from the ASR engine with the text input. Therefore, if the ASR engine makes errors and cannot produce the correct phonetic transcription, the comparison stage will be affected from that error and the KWS may eventually fail. Even if the comparison is not an exact comparison but an approximate one, classical keyword spotters do not mostly take into account what kind of errors the ASR engine makes and do not tolerate its errors in the comparison stage. Briefly, traditional KWS systems compare the phonetically correct transcription of the keyword with the ASR transcription of the "**pronounced versions**" of the keyword. This system cannot be named "**reliable**". Therefore, we decided to take voice input. The user is asked to pronounce the keyword many times. In addition, different users may utter the keyword to provide a source of variety. Our system transcribes those input pronunciations through ASR engine. Then, those phonetic transcriptions are searched in the database which consists of the transcriptions produced by processing the stored speech data through the **same** ASR engine. Even if the ASR engine makes some errors, those

errors do not cause dramatic failures because the two compared things, e.g. the input voice transcriptions and the stored speech transcriptions, both pass through the same ASR engine and face similar errors.

However, taking the input in voice format instead of phonetic transcription, as we did, introduces the risk of decreasing the performance of the KWS system, because there may be cases where somehow a badly recorded or pronounced keyword sample may not exactly represent the keyword or may even be misleading. To compensate for that risk, we should try to collect as many voice samples of the keyword as possible at the input side, which is not possible practically. For that reason, we used not only the keyword samples spoken by the users but also synthetically generated transformations of them. To produce such transformations, we used Sestek VoDi, software developed by SesTek. In addition, we gave the user a chance to utter the keyword in full sentences and then to select the portion in these full sentences where the keyword is spoken. Because, word pronunciations may somehow differ according to the context, e.g. the same keyword may give slightly different phoneme sequences when spoken in different contexts. In order to improve performance by preventing failures due to context dependency, we ask the user to pronounce different forms of the keyword, e.g. the keyword with prefixes and suffixes, spontaneously in some sentences. The user is then able to identify the keyword beginning and ending times in the sentences he/she uttered. That way, we have different phoneme transcriptions of the same keyword for different contexts and we believe that this variety is likely to increase the success of the KWS system.

Another weakness of traditional text driven KWS systems is that they do not take gender, intonation, accent, emotional mood, etc. differences, which affect pronunciation, into account. Traditional text driven KWS systems take only the single correct phonetic transcription of the keyword at the input. However, the search is done over a database where verbal recordings are gathered from many different speakers. Gender, intonation, accent, emotional mood, etc. differences between people affect pronunciation and the correct phonetic transcription of the keyword alone cannot represent such variety and the spotter may fail eventually. Therefore, in the KWS we developed, we collect verbal samples of the keyword from many different speakers to represent such kind of variety. In addition, those samples are transformed into synthetic versions such that they would

simulate different versions of the keyword if different speakers would pronounce it. To produce such transformations, we used Sestek VoDi, software developed by SesTek. When the transformation stage finishes, we have not only the original utterances but also many different versions of them at hand. This abundance and variety of keyword utterances at the input side increases the probability of spotting the keyword in the search database.

Another weakness of traditional KWS systems is that they are generally language dependent. However, we believe that the KWS system we have developed is not language dependent, because it is based on phonetic speech units, **phonemes**. Using a Turkish phoneme recognizer, even if the keyword and the recordings in the search database are spoken in another language will probably succeed, because the same ASR engine is used for both sides (input side and the search database side) and it will make similar errors for both sides and those errors will be mostly compensated. Whereas, using more than one ASR engine, each of a different language, in parallel may improve the overall success of the spotter. The KWS system we have developed allows the cooperation of more than one ASR engine at the same time.

2. VOICE DRIVEN KEYWORD SPOTTER

2.1. Overview of Keyword Spotter

Figure 2.1 shows the general layout of the keyword spotter we have implemented.

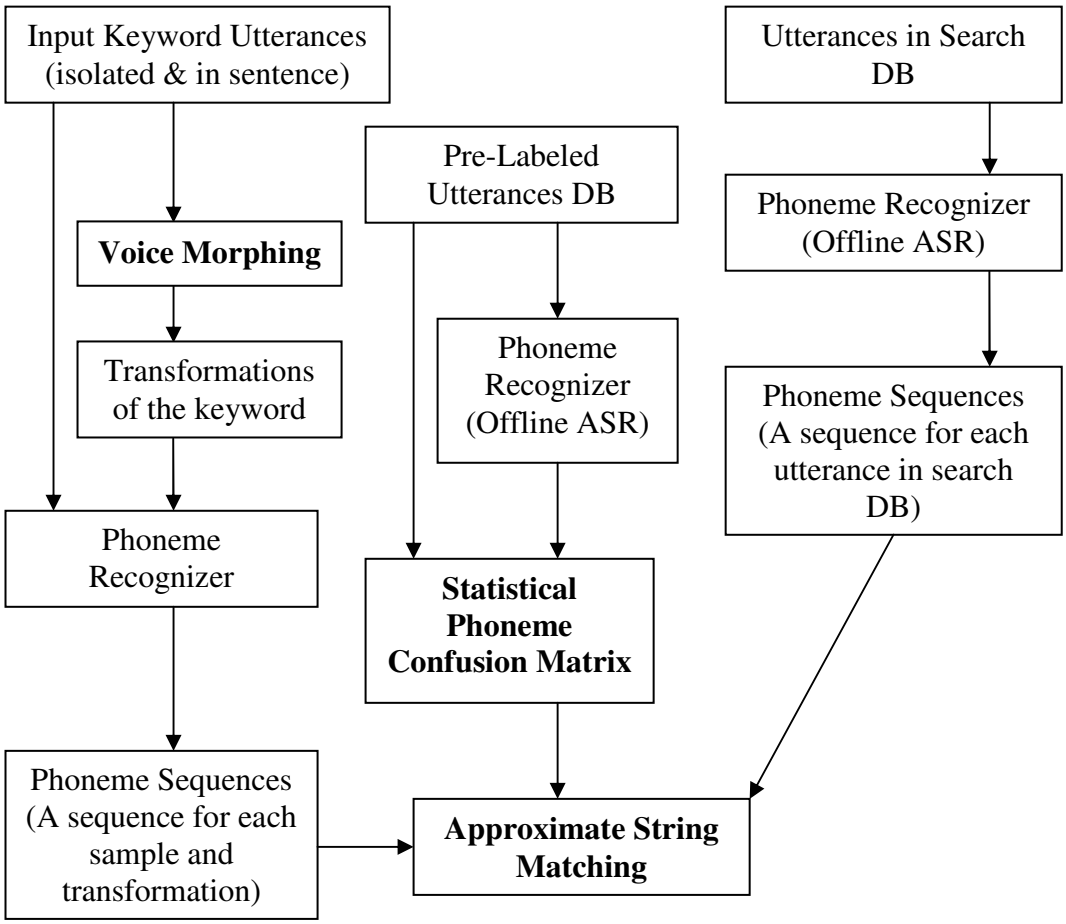


Figure 2.1. Layout of the keyword spotting system

First, previously recorded utterances in the search database are processed offline by the phoneme recognizer. We use the **phonetic aligner** developed by Sestek for this task. This recognizer can identify the 29 phonemes in Turkish language plus silence, thus totally 30 phonemes. At the end of this stage, we obtain phoneme sequences, together with time

intervals of the phonemes, which represent the phonetic transcription of the recorded utterances in the search database.

At the input side, the user utters many **isolated** samples of the keyword to be searched to the microphone and the system records them. In order to increase the input variety, the user may also provide utterances of the same **isolated** keyword spoken by other speakers. On the other hand, word pronunciations may differ somehow according to the context, e.g. the same keyword may give different phoneme sequences when spoken in different contexts. In order to improve performance by preventing failures due to context dependency, we ask the user to pronounce the keyword spontaneously in some sentences so that we can have different phoneme transcriptions of the same keyword for different contexts. The user is able to identify the keyword beginning and ending times in the sentences he/she uttered. For example, the user may wish to spot the keyword "*bakan*", but this keyword may exist as "*bakanlık*", "*bakanı*", "*bakanlıktan*", etc. in the search database. Therefore, our system asks the user to pronounce different forms of the keyword, too. Using the keyword and its different forms together at the input side increases the detection rate of the spotter.

Then, the **voice morphing module** takes place and transforms the input keyword utterances. We used VoDi developed by Sestek for the morphing task. VoDi can transform the input utterance into woman version, man version and child version. After that process, we have the original input utterances and their transformed versions at hand together at the input side.

Next, all of those input keyword utterances (originals and transformations) are processed online by the same phoneme recognizer used in the offline stage. This process produces a phoneme sequence for each of the input utterances, together with time intervals of the phonemes.

Then, the textual approximate string matching module takes part and compares the phoneme sequences of the input utterances produced online with the phoneme sequences of the utterances in the search database produced offline.

The decision of which strings are approximate and which are irrelevant will be given based on the **statistical phoneme confusion matrix**. This confusion matrix is calculated by processing offline a pre-labeled voice database through the same phoneme recognizer used in online and offline stages. The phoneme recognizer is actually an aligner and is able to perform both forced-aligning to existing labels and unconstrained aligning. We run the aligner 2 times. For the first time, it works in unconstrained aligning mode and aligns the utterances in the pre-labeled database to 30 phonemes it can recognize. In the second run, it works in forced-alignment mode and tries to align the utterances in the pre-labeled database to their existing labels. Comparing the outputs of the two runs, we can calculate which phonemes the aligner confuses with which phonemes with what probability and this data becomes statistical if there is enough pre-labeled voice data. This statistical data is used to form the statistical phoneme confusion matrix.

The textual approximate string matching module is based on **Edit Distance** calculation.

2.2. Phoneme Recognizer (Sestek Aligner)

We used Sestek Aligner as the phoneme recognizer. The software takes 3 inputs:

1- full path of the folder containing the .wav files to be processed
2- running mode
    a. unconstrained mode
    b. forced alignment mode
3- language

We did our experiments with Turkish language. Sestek Aligner processes every .wav file in the given folder and produces a text output, containing the alignment results, for each of the .wav files there. Output text files carry the extension **.lab**.

If run for Turkish language, Sestek Aligner can align .wav files to the phonemes below:

/a/	/b/	/c/	/ç/	/d/	/e/	/f/	/g/	/ğ/	/h/	/ı/
/i/	/j/	/k/	/l/	/m/	/n/	/o/	/ö/	/p/	/r/	/s/
/ş/	**/sil/**	**/sp/**	/t/	/u/	/ü/	/v/	/y/	/z/	**/Z sil/**	

/sil/: silence
/sp/: short pause
/Z sil/: long pause

However, distinction between silence, short pause and long pause is not important for our task. Therefore, we accept all of them as **SILENCE** and represent silence with **Z**.

/sil/: silence
/sp/: short pause
/Z sil/: long pause

} **/Z/: SILENCE**

Thus, our spotter is able to identify **30 phonemes**, including silence **Z**. In fact, they are the letters in Turkish alphabet; not exactly the phonemes of the language, but the side effects of this assumption can be neglected for Turkish language.

Below is an example output text file, namely **sifir.lab**, produced by Sestek Aligner for **sifir.wav** voice file:

#
0.06 s
0.12 t
0.16 ö
0.31 f
0.48 ö
0.75 r
0.80 Z

The floating number represents the ending time of the phoneme on that line. For the example above, the aligner recognized phoneme /s/ between 0 seconds and 0.06 seconds

and the phoneme /t/ between 0.06 seconds and 0.12 seconds and so on. Sestek Aligner processes the .wav files on the basis of frames of 0.01 seconds.

While processing offline the **pre-labeled database**, we run Sestek Aligner in 2 different modes. In the first mode, the aligner aligns the .wav files in the browsed folder to the phonemes it can recognize. In the second mode, a text file including pre-given labels should be supplied for each of the .wav files. This text file should stay in the same folder where its corresponding .wav file stays and it should have the same file name with its .wav file but with a different file extension, **.txt**. The file should include a space between each pre-given phoneme. Below is an example text file, namely **sifir.txt**, including the **pre-labeled phonemes** for **sifir.wav**:

s ɪ f ɪ r

Below is an example output text file, namely **sifir_fatt.lbl**, produced by Sestek Aligner when it is run in **forced alignment mode** using the pre-given labels above:

#
0.12 s
0.20 ɪ
0.32 f
0.54 ɪ
0.80 r

Comparing the output of the aligner run in unconstrained mode, namely **sifir.lab**, and the output of the aligner run in forced alignment mode, namely **sifir_fatt.lbl**, we calculate the statistical phoneme confusion matrix and the statistical phoneme group confusion matrix.

2.3. Statistical Phoneme Confusion Matrix Calculation

Statistical phoneme confusion matrix (**SPCM**) holds the probabilities of confusing a phoneme with another phoneme. It is an N x N square matrix, where N is the number of phonemes the system is able to identify.

SPCM [i , j] represents the probability of confusing phoneme i with phoneme j that is, the probability of identifying the current phoneme as phoneme j although it was phoneme i actually.

Table 2.3.1 shows the phoneme confusion matrix calculated using Sestek Aligner, which is able to identify **30 phonemes**, over 936 .wav files, taken from 26 different speakers. That is, each of the 26 speakers spoke 36 utterances and 21 of these utterances are isolated words and the remaining 15 are sentences.

Phoneme confusion probability values are calculated by comparing the output of the aligner run in unconstrained mode and the output of the aligner run in forced alignment mode. However, the calculation is done on the basis of not phoneme level but **frame level**. Sestek Aligner processes the .wav files on the basis of frames of 0.01 seconds. Therefore, if we go deep into frame level then the calculated probability values will be more statistical and meaningful.

We begin with an empty matrix whose elements are all zero and then start to update the matrix elements step by step. Let us examine a sample case and assume that the outputs of the aligner run in unconstrained mode and in forced alignment mode are like below:

Forced Alignment Mode	**Unconstrained Mode**
0.12 s	**0.06 s**
0.20 I	**0.12 t**
0.32 f	**0.16 ö**
0.54 I	**0.31 f**
0.80 r	**0.48 ö**
	0.80 r

Table 2.3.1. Statistical phoneme confusion matrix

	a	b	c	ç	d	e	f	g	ğ	h	ı	i	j	k	l	m	n	o	ö	p	r	s	ş	t	u	ü	v	y	z	Z
a	0,44	0,01	0	0	0,03	0,1	0,01	0,01	0	0,02	0,01	0	0	0,01	0,03	0,01	0	0,04	0,11	0,01	0,01	0	0	0,01	0,01	0,01	0,04	0,01	0	0,07
b	0	0,32	0	0	0,06	0,01	0	0,01	0	0,03	0,01	0	0	0,02	0,01	0,05	0,04	0,01	0	0,03	0	0	0	0	0,01	0	0,02	0	0,01	0,36
c	0	0	0,04	0	0,04	0	0	0	0	0,04	0	0	0	0,04	0	0	0,04	0	0	0,13	0	0	0	0,13	0	0	0	0	0	0,54
ç	0	0	0	0,31	0,02	0,01	0,01	0	0	0,02	0,01	0	0	0,12	0	0	0	0	0	0,1	0,05	0,06	0,2	0,04	0	0,01	0	0	0	0,04
d	0	0,05	0	0,01	0,51	0	0	0,03	0	0,01	0,02	0,01	0	0,02	0,04	0,01	0,1	0	0,01	0,02	0	0,03	0	0,03	0	0	0,01	0	0	0,09
e	0,09	0,02	0	0	0,06	0,4	0	0,01	0,02	0	0,01	0,03	0	0,01	0,04	0	0,01	0	0,04	0	0,01	0	0	0,01	0	0,03	0,09	0,04	0	0,08
f	0	0	0	0	0,12	0	0,57	0,05	0	0,04	0	0	0	0	0	0	0,04	0	0	0,04	0,04	0,04	0	0	0,04	0	0	0	0	0,02
g	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
ğ	0,52	0	0	0	0	0	0	0	0,32	0,02	0	0,02	0	0	0	0,02	0	0,05	0	0	0,02	0	0	0	0	0	0,02	0	0	0,01
h	0,42	0	0	0	0	0	0	0	0	0,34	0	0	0	0	0	0	0,09	0	0,04	0	0	0	0	0	0	0	0	0,11	0	0
ı	0,03	0	0,01	0	0,04	0,08	0,01	0,01	0,03	0	0,16	0,02	0	0,01	0,06	0,01	0,05	0,01	0,2	0	0,03	0,05	0	0,02	0,07	0,02	0,02	0,03	0,01	0,02
i	0,02	0,03	0,01	0	0,04	0,16	0	0,05	0,05	0	0,02	0,2	0	0,02	0,02	0,01	0,02	0	0,01	0	0,01	0,01	0,02	0,01	0,01	0,07	0,02	0,11	0	0,08
j	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
k	0,03	0	0	0,01	0,01	0,04	0,02	0	0	0,01	0,01	0,08	0	0,41	0	0	0,01	0,01	0	0,05	0,03	0,01	0,01	0,06	0,01	0,01	0	0	0	0,18
l	0,07	0,01	0,01	0	0,07	0,03	0	0,03	0,02	0,01	0,01	0,03	0	0,03	0,2	0,04	0,07	0,12	0,02	0	0,01	0,02	0	0,01	0,04	0,03	0,06	0,01	0,01	0,04
m	0	0,05	0	0	0	0,01	0,01	0	0,03	0,02	0,05	0,01	0	0,01	0,01	0,49	0,05	0,01	0,01	0,04	0,01	0	0	0	0	0,01	0,12	0,02	0	0,04
n	0,11	0,01	0	0	0,02	0,03	0	0	0,05	0,01	0,04	0,04	0	0	0,03	0,06	0,41	0,03	0,02	0	0,02	0	0	0	0,04	0,01	0,01	0,02	0	0,04
o	0,04	0,03	0	0	0,02	0,01	0,02	0	0	0,01	0	0	0	0	0,01	0	0	0,58	0,01	0,01	0,02	0,01	0	0	0,01	0	0,01	0,03	0,03	0,15
ö	0,19	0,01	0	0	0,05	0,1	0	0,07	0,02	0	0	0	0	0	0,01	0	0	0,02	0,45	0	0,03	0	0	0,01	0	0	0	0,03	0	0,01
p	0	0	0	0	0	0	0,65	0	0	0	0	0	0	0	0	0	0	0	0	0,35	0	0	0	0	0	0	0	0	0	0
r	0,12	0	0	0	0,02	0,16	0,02	0	0,02	0,01	0,03	0,03	0,02	0,01	0,01	0,01	0,02	0,01	0,13	0,01	0,19	0,02	0,03	0,01	0,01	0,05	0,02	0,02	0,01	0,01
s	0	0	0,01	0,01	0	0	0,09	0	0	0	0	0,02	0	0,03	0	0	0	0	0	0,02	0,01	0,64	0,01	0,05	0	0	0	0	0,06	0,05
ş	0,01	0	0	0,01	0	0,03	0,03	0	0	0	0	0,03	0,05	0	0	0	0,01	0	0	0	0,04	0,11	0,58	0,01	0	0,01	0	0	0,05	0,03
t	0,01	0	0	0,01	0,01	0	0,04	0	0	0,01	0,01	0,03	0	0,18	0,01	0	0	0,01	0	0,1	0,06	0,01	0	0,27	0,01	0	0	0	0,01	0,22
u	0,02	0,07	0	0	0,01	0,01	0	0,02	0,03	0,02	0,02	0	0	0,07	0,03	0,01	0	0,12	0,14	0,01	0,01	0,01	0	0	0,35	0,02	0,03	0	0	0
ü	0	0,05	0	0,02	0,03	0,09	0	0,01	0,02	0,01	0,06	0,06	0,01	0,03	0,02	0	0,01	0,01	0,06	0	0,01	0	0	0,02	0,02	0,22	0,02	0,02	0,03	0,17
v	0	0	0	0	0	0,05	0	0	0	0,04	0	0	0	0	0	0	0	0	0	0,04	0	0	0	0	0	0	0	0	0	0,87
y	0,01	0,04	0,01	0	0,02	0,09	0,01	0,04	0,02	0,04	0,01	0,01	0,05	0,05	0,04	0	0	0	0,03	0,01	0	0	0,01	0,01	0	0,07	0,01	0,07	0,04	0,31
z	0	0	0,01	0	0,01	0,01	0,12	0	0,01	0,02	0,03	0,01	0,05	0,02	0,03	0	0	0,01	0,1	0,01	0,03	0,07	0,01	0	0,04	0,09	0	0	0,3	0,02
Z	0,03	0,02	0,01	0,02	0,05	0,04	0,02	0	0,01	0,01	0,03	0,04	0,02	0,06	0,04	0,04	0,08	0,01	0,01	0,01	0,03	0,07	0,03	0,02	0,02	0,02	0,01	0,01	0,06	0,18

Forced alignment mode result contains the correct phonemes that actually exist in the utterance of the processed .wav file and their corresponding time intervals. Unconstrained mode result contains the phonemes identified by the aligner and their corresponding time intervals. According to the results above, there is phoneme /s/ in the processed .wav file between 0.00 seconds and 0.12 seconds, that is the first 12 frames are of phoneme /s/. However, the aligner identified phoneme /s/ in frames 1 to 6 but phoneme /t/ in frames 7 to 12. Therefore we calculate,

$$P\,(/s/, /s/\,) = \frac{6}{12}$$

$$P\,(/s/, /t/) = \frac{6}{12}$$

and add the calculated probabilities to the corresponding elements of the matrix. That is,

$$SPCM\,(/s/,/s/) = SPCM\,(/s/,/s/) + \frac{6}{12}$$

$$SPCM\,(/s/,/t/) = SPCM\,(/s/,/t/) + \frac{6}{12}$$

After all probability calculations are done and all the elements of the matrix are cumulatively updated, we normalize the matrix elements by dividing each element of the matrix by the sum of the elements on the same row. That is,

$$SPCM\,(i,j) = \frac{SPCM\,(i,j)}{\sum_{k=1}^{30} SPCM\,(i,k)} \quad \text{i = 1, ..., 30} \quad \text{j = 1, ..., 30} \qquad \textbf{(2.1)}$$

Hence, the matrix elements now represent the probability of confusing the phoneme in the corresponding row with the phoneme in the corresponding column. That is,

$$\sum_{i=1}^{30} \sum_{j=1}^{30} SPCM\,(i,j) = 1 \qquad \textbf{(2.2)}$$

2.4. Statistical Phoneme Group Confusion Matrix Calculation

We have gathered phonemes into 5 phoneme groups. They are **AFFRICATES** (A), **VOWELS** (V), **SILENCE** (Z), **STOPS** (S) and **OTHERS** (?).

Phonemes that we realized as affricate (A):	**/c/, /ç/, /f/, /j/, /s/, /ş/, /v/, /z/**
Phonemes that we realized as vowel (V):	**/a/, /e/, /ı/, /i/, /o/, /ö/, /u/, /ü/**
Phonemes that we realized as silence (Z):	**/Z/**
Phonemes that we realized as stop (S):	**/b/, /d/, /g/, /k/, /p/, /t/**
Phonemes that we realized as others (?):	**/ğ/, /h/, /l/, /m/, /n/, /r/, /y/**

SPGCM is a 5 x 5 square matrix and **SPGCM [i , j]** represents the probability of confusing a phoneme of group i with a phoneme of group j that is, the probability of identifying the current phoneme as of phoneme group j although it was of phoneme group i actually.

Table 2.4.1 shows the phoneme group confusion matrix calculated using Sestek Aligner, which is able to identify 30 phonemes, over 936 .wav files, taken from 26 different speakers. That is, each of the 26 speakers spoke 36 utterances and 21 of these utterances are isolated keywords and the remaining 15 are sentences.

Phoneme group confusion probability values are calculated by comparing the output of the aligner run in unconstrained mode and the output of the aligner run in forced alignment mode. However, the calculation is done on the basis of not phoneme level but frame level. Sestek Aligner processes the .wav files on the basis of frames of 0.01 seconds. Therefore, if we go deep into frame level then the calculated probability values will be more statistical and meaningful.

Table 2.4.1. Statistical phoneme group confusion matrix

	A	S	V	?	Z
A	0,67	0,09	0,12	0,06	0,06
S	0,05	0,56	0,1	0,1	0,19
V	0,07	0,11	0,61	0,14	0,07
?	0,08	0,08	0,36	0,41	0,07
Z	0,24	0,17	0,19	0,22	0,18

Forced Alignment Mode	**Unconstrained Mode**
0.12 s	**0.06 s**
0.20 I	**0.12 t**
0.32 f	**0.16 ö**
0.54 I	**0.31 f**
0.80 r	**0.48 ö**
	0.75 r
	0.80 Z

Forced alignment mode result contains the correct phonemes that actually exist in the utterance of the processed .wav file and their corresponding time intervals. Unconstrained mode result contains the phonemes identified by the aligner and their corresponding time intervals. According to the results above, there is phoneme /s/ and hence phoneme group "A" in the processed .wav file between 0.00 seconds and 0.12 seconds, that is the first 12 frames are of phoneme group "A". However, the aligner identified phoneme /s/ and hence phoneme group "A" in frames 1 to 6 but phoneme /t/ and hence phoneme group "S" in frames 7 to 12. Therefore we calculate,

$$P("A","A") = \frac{6}{12}$$

$$P("A","S") = \frac{6}{12}$$

and add the calculated probabilities to the corresponding elements of the matrix. That is,

$$SPGCM("A","A") = SPGCM("A","A") + \frac{6}{12}$$

$$SPGCM("A","S") = SPGCM("A","S") + \frac{6}{12}$$

After all probability calculations are done and all the elements of the matrix are cumulatively updated, we normalize the matrix elements by dividing each element of the matrix by the sum of the elements on the same row. That is,

$$SPGCM(i,j) = \frac{SPGCM(i,j)}{\sum_{k=1}^{5} SPGCM(i,k)} \qquad \text{i = 1, ..., 5} \qquad \text{j = 1, ..., 5} \qquad \textbf{(2.3)}$$

Hence, the matrix elements now represent the probability of confusing the phoneme group in the corresponding row with the phoneme group in the corresponding column. That is,

$$\sum_{i=1}^{5} \sum_{j=1}^{5} SPGCM(i,j) = 1 \qquad \textbf{(2.4)}$$

2.5. Voice Morpher (Sestek VoDi)

We used Sestek VoDi, whose GUI can be seen in **Figure 2.5.1**, to morph the natural input keyword utterances to produce synthetically generated extra voices inputs.

We performed the following voice conversions using VoDi:

1- from Man to Woman
2- from Man to Child
3- from Man to Man

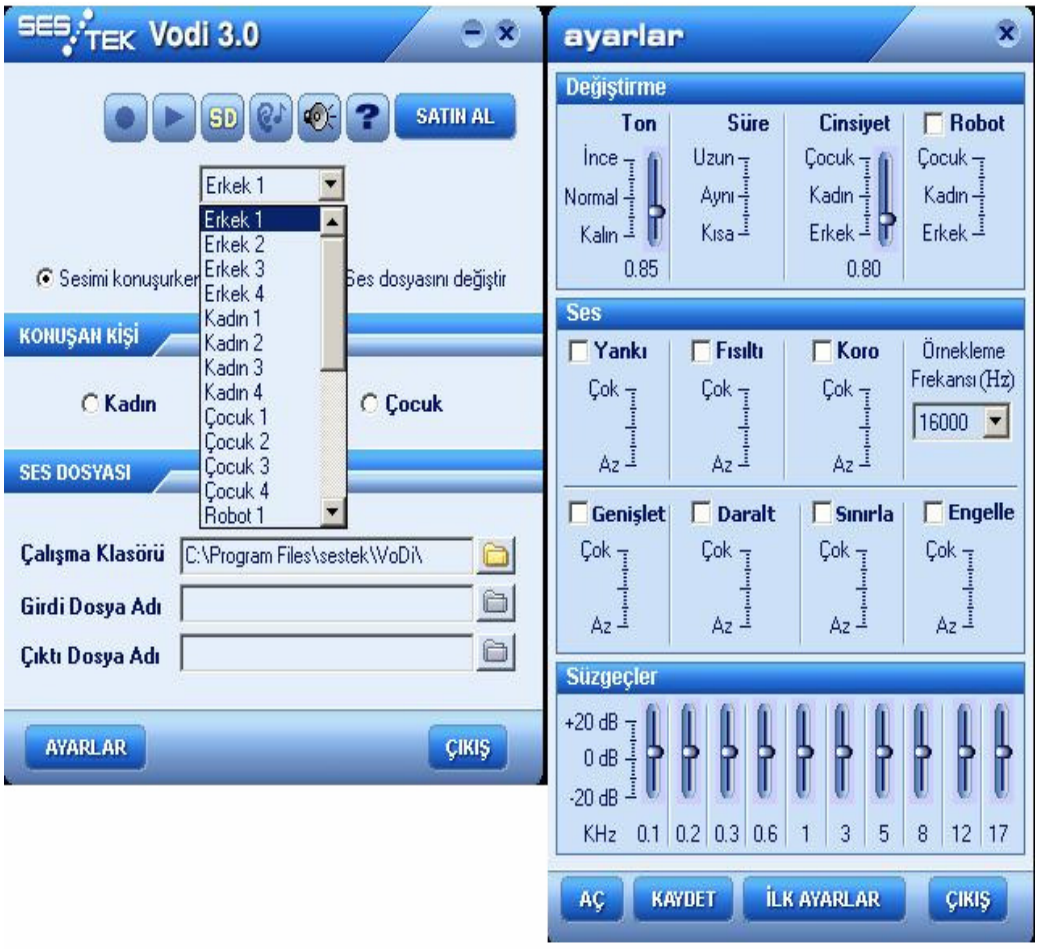


Figure 2.5.1. Graphical user interface of Sestek VoDi

2.6. Voice Driven Keyword Spotter

"Voice Driven Keyword Spotter" application is written in Microsoft Visual Basic .NET. Therefore, Microsoft .NET Framework 1.1 should be installed on the computer to run the application properly. The application uses Sestek Aligner and Sestek VoDi applications internally; therefore those 2 applications should also be installed on the computer.

Two Matlab functions, namely **onur3d_toplu.m** and **onur3d.m**, are written to produce graphical outputs showing the success of the system. The Matlab function utilizes the textual outputs of "Voice Driven Keyword Spotter" application and gives out many graphics, thus we can interpret the success of the system easily.

The source code for "Voice Driven Keyword Spotter" application and the Matlab functions **onur3d_toplu.m** and **onur3d.m** are available in the CD attached.

Figure 2.6.1 shows the graphical user interface of the voice driven keyword spotter we have developed. Below are the definitions of the controls on the GUI.

Ses Tanıma Uygulamasının (SestekAligner.exe) Yolunu Seçiniz:

Path of the executable of Sestek Aligner, which the spotter uses internally to transcribe .wav files into phoneme sequences

Anahtar Kelime Ses Kayıt Dosyalarının Yolunu Seçiniz:

Full path of the folder of .wav files of the samples of the keyword

Anahtar Kelimenin Aranacağı Ses Dosyasının Yolunu Seçiniz:

Full path of the folder of .wav files in which the keyword will be searched

Eşleşme Eşik Skorunu Giriniz:

The value of the threshold over 100

Toplu Gösterim İçin Minimum Kesişme Yüzdesini Giriniz:

Minimum intersection percent (**MIP**) of two consequent frames to realize them as belonging to the same word

Basit Metod Ağırlığını Giriniz:

The coefficient of Phoneme Frequency Score over 100 ($\mathbf{C_{PFS}}$)

Edit Distance Ağırlığını Giriniz:

The coefficient of Edit Distance Score over 100 ($\mathbf{C_{NEDS}}$)

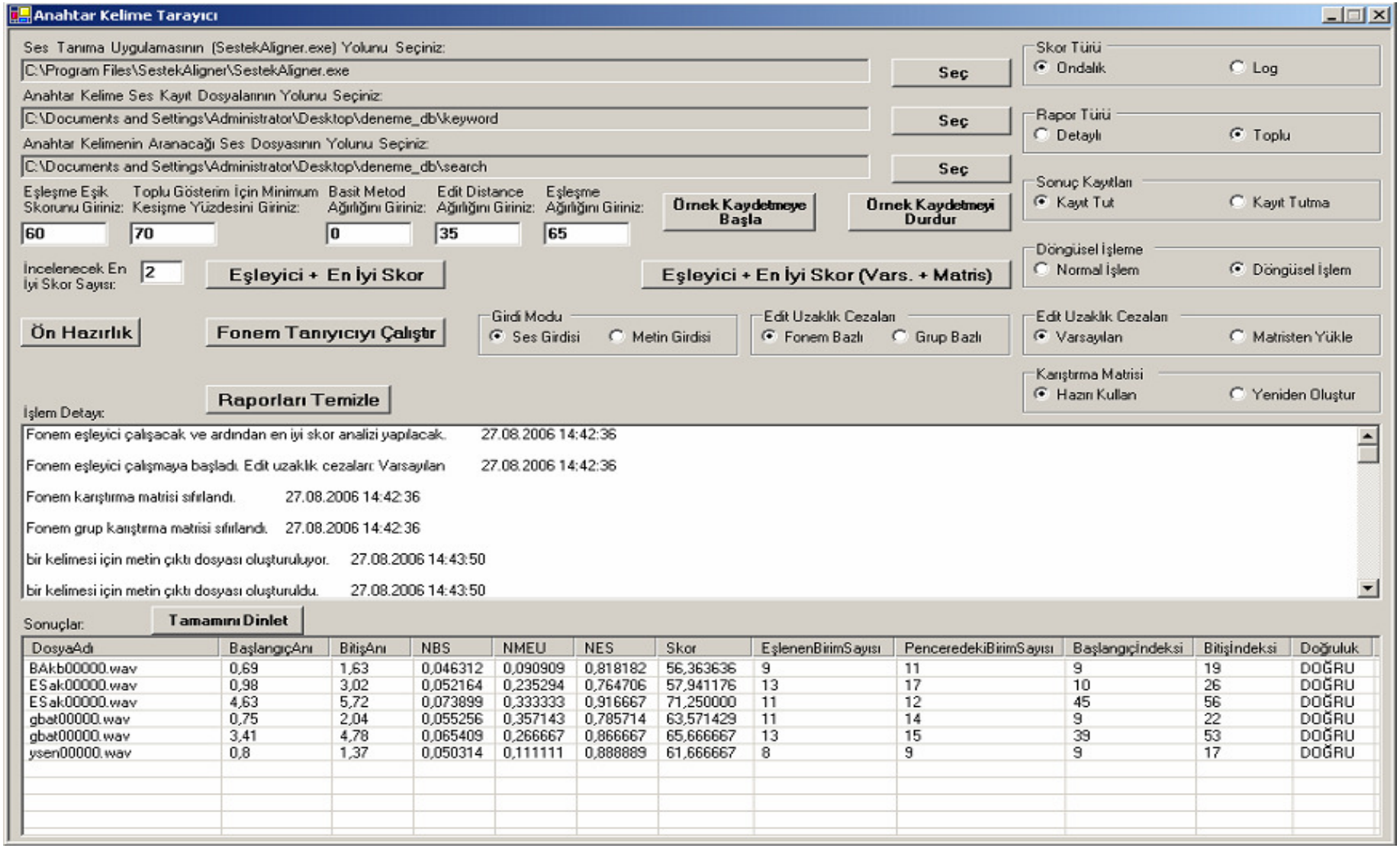
Eşleşme Ağırlığını Giriniz:

The coefficient of Matching Score over 100 ($\mathbf{C_{MS}}$)

$$CPFS + CEDS + CMS = 100 \qquad \textbf{(2.5)}$$

İncelenecek En İyi Skor Sayısı:

Number of best scores to be analyzed in more details



DosyaAdı	BaşlangıçAnı	BitişAnı	NBS	NMEU	NES	Skor	EşlenenBirimSayısı	PenceredekiBirimSayısı	BaşlangıçİndeksI	BitişİndeksI	Doğruluk
BAkb00000.wav	0,69	1,63	0,046312	0,090909	0,818182	56,363636	9	11	9	19	DOĞRU
ESak00000.wav	0,98	3,02	0,052164	0,235294	0,764706	57,941176	13	17	10	26	DOĞRU
ESak00000.wav	4,63	5,72	0,073899	0,333333	0,916667	71,250000	11	12	45	56	DOĞRU
gbat00000.wav	0,75	2,04	0,055256	0,357143	0,785714	63,571429	11	14	9	22	DOĞRU
gbat00000.wav	3,41	4,78	0,065409	0,266667	0,866667	65,666667	13	15	39	53	DOĞRU
ysen00000.wav	0,8	1,37	0,050314	0,111111	0,888889	61,666667	8	9	9	17	DOĞRU

Figure 2.6.1. Graphical user interface of voice driven keyword spotter

Döngüsel İşleme:

Normal İşlem:

Runs only for the given values of the threshold, C_{PFS}, C_{EDS} and C_{MS}

Döngüsel İşlem:

Runs in a loop for all pre-determined values of the threshold, C_{PFS}, C_{EDS} and C_{MS}. Threshold is changed from 20 to 100 with a step of 20. C_{PFS}, C_{EDS} and C_{MS} are changed from 0 to 100 with a step of 20. However, the values of C_{PFS}, C_{EDS} and C_{MS} are adjusted so that **their sum always equals 100**.

Edit Uzaklık Cezaları:

Varsayılan: Neither SPCM nor SPGCM is used when checked

Matristen Yükle: Either SPCM or SPCGM is used when checked

Edit Uzaklık Cezaları:

Fonem Bazlı: SPCM is used

Grup Bazlı: SPGCM is used

Karıştırma Matrisi:

Hazırı Kullan:

SPCM or SPGCM is not calculated again but SPCM pre-given in the text file **phoneme_confusion_matrix.out** or SPGCM pre-given in the text file **phoneme_group_confusion_matrix.out** is used.

Yeniden Oluştur:

SPCM or SPGCM is re-calculated.

Girdi Modu:

Ses Girdisi:

The input to the system is taken in the form of voice. The user should provide either pre-recorded .wav files of the keyword to be spotted or should record wav files using the interface.

Metin Girdisi:

The input to the system is taken in the form of text. The user should only give the text of the keyword to be spotted. This option is included in order to compare the success measure of the voice driven spotter with the success of the text driven spotter.

İşlem Detayı: Lists the task currently performed by the spotter and any helpful outputs

Sonuçlar: Listbox that lists the outputs of the spotter.

Dosya Adı: Name of the .wav file in which the spotter detected a keyword

Başlangıç Anı: The starting time in seconds of the detected keyword

Bitiş Anı: The ending time in seconds of the detected keyword

NBS: Normalized Phoneme Frequency Score (PFS_N) of the detected keyword

NMEU: Normalized Edit Distance Score (EDS_N) of the detected keyword

NES: Normalized Matching Score (MS_N) of the detected keyword

Skor: Normalized Overall Score (OS_N) of the detected keyword

Eşlenen Birim Sayısı: Number of phonemes matched in the detected keyword

Penceredeki Birim Sayısı: Number of phonemes in the detected keyword

Başlangıç İndeksi: The starting index of the detected keyword

Bitiş İndeksi: The ending index of the detected keyword

Doğruluk: If pre-given labels exist, announces DOĞRU if the keyword is detected correctly or YANLIŞ if not. If pre-given labels do not exist, announces BİLİNMİYOR.

Örnek Kaydetmeye Başla:

The user should click this button to start recording an input voice sample of the keyword to be spotted.

Örnek Kaydetmeye Durdur:

The user should click this button to stop recording an input voice sample of the keyword to be spotted and to save the recording in wav format.

Raporları Temizle:

Clears the reports on the screen.

Ön Hazırlık:

Processes the pre-labeled text files to convert them into the format which Sestek Aligner understands.

Fonem Tanıyıcıyı Çalıştır:

Runs Sestek Aligner both in forced-alignment mode and in unconstrained mode to produce phoneme sequences from the wav files.

Eşleyici + En İyi Skor:

Runs the application and then finds the best N outputs (N is specified in the text box named "İncelenecek En İyi Skor Sayısı") and makes a more detailed analysis for those best outputs.

Eşleyici + En İyi Skor (Vars. + Matris):

Triggers 2 consecutive runs:

1- Runs the application and then finds the best N outputs (N is specified in the text box named "İncelenecek En İyi Skor Sayısı") and makes a more detailed analysis for those best outputs, not using either SPCM or SPGCM.

2- Runs the application and then finds the best N outputs (N is specified in the text box named "İncelenecek En İyi Skor Sayısı") and makes a more detailed analysis for those best outputs, using either SPCM or SPGCM.

2.7. Frame Score Calculation

Our keyword spotter searches in the .wav files in the search folder on the basis of frames. A frame is in fact a window of a specific number of phonemes. The frame length (**FL**) depends on the average number of phonemes in samples of the keyword, including the original samples and the transformations.

$$FL = \frac{1}{n} \sum_{i=1}^{n} p(i) \qquad \textbf{(2.6)}$$

FL: frame length (window length)

n: number of samples of the keyword (number of .wav files of keyword samples)

p (i): number of phonemes in the i^{th} sample of the keyword

Frame length is calculated at the beginning of the process. Each .wav file in the search folder is analyzed frame by frame in a "**sliding windows**" manner. That is,

$$Frame(i) = \{Pi.....Pi + FL - 1\} \qquad \text{i = 1, ..., L-FL+1}$$

where L is the length (number of phonemes) of the current .wav file.

A normalized overall score ($\mathbf{OS_N}$) over 100 is calculated for **each frame**. If OS_N of the frame is equal to or greater than the **threshold**, the frame is accepted as a keyword and is listed as a line in the outputs list box of the GUI.

The normalized overall score ($\mathbf{OS_N}$) over 100 of a frame is sum of 3 components:

1- *Normalized Phoneme Frequency Score ($\boldsymbol{PFS_N}$) over 100*
2- *Normalized Edit Distance Score ($\boldsymbol{EDS_N}$) over 100*
3- *Normalized Matching Score ($\boldsymbol{MS_N}$) over 100*

$$OSN = (CPFS \cdot PFSN) + (CEDS \cdot EDSN) + (CMS \cdot MSN) \qquad \textbf{(2.7)}$$
$$CPFS + CEDS + CMS = 100$$

2.7.1. Normalized Phoneme Frequency Score ($\mathbf{PFS_N}$)

Assume there are **N** samples of the keyword (original voice and transformations) and each sample is transcribed into a phoneme array using Sestek Aligner. For each distinct phoneme in those N samples of the keyword, phoneme frequencies are calculated.

$$P = \{ pk \} \qquad Fpk = \frac{\sum_{j=1}^{N} Qj\ (pk)}{\sum_{i=1}^{W} \sum_{j=1}^{N} Qj\ (pi)} \qquad \textbf{(2.8)}$$

P: phoneme set whose elements are the **distinct** phonemes present in the N **keyword** samples

p_k: a phoneme in the phoneme set P

Q_j (p_k): number of occurrence of phoneme p_k in the j^{th} keyword sample.

W: number of distinct phonemes our system can identify (30 in our experiments)

F_{pk}: frequency of phoneme p_k

$\mathbf{F_{pk}}$ is calculated by dividing the sum of number of occurrences of phoneme p_k in each keyword sample by the sum of number of occurrences of each distinct phoneme in the phoneme set P in each keyword sample.

$$PFSN = \frac{1}{a}\sum_{i=1}^{a} Fpi \qquad \textbf{(2.9)}$$

a: number of phonemes in the frame and $\mathbf{p_i}$ is the i^{th} phoneme in the frame

2.7.2. Normalized Edit Distance Score ($\mathbf{EDS_N}$)

Assume there are **N** samples of the keyword and each sample is transcribed into a phoneme array using Sestek Aligner. We take the next frame from the .wav file in the search database and calculate the edit distance between the frame and the keyword. However, if we have more than one sample of the keyword at hand, then we calculate the edit distance between the frame and each of the samples of the keyword and then choose the minimum edit distance.

$$EDSN = 1 - \frac{\min(WEDi)}{FL} \qquad \text{i = 1, 2, 3, ..., N} \qquad \textbf{(2.10)}$$

$\mathbf{WED_i}$: Weighted Edit Distance between the frame and the i^{th} sample of the keyword

FL: number of phonemes in the frame (frame length)

2.7.2.1. Weighted Edit Distance Calculation

Edit Distance between two frames is the minimum total number of edit operations (insertion of a phoneme, deletion of a phoneme and substitution of a phoneme with another phoneme) needed to transform the first frame into the second frame. **[9, 11]**

Another version of Edit Distance, where only insertion and deletion is allowed but substitution is forbidden, is called **Levenstein Distance**. Substitution operation in Edit

Distance is achieved by 2 consecutive operations, a deletion operation plus an insertion operation, in Levenstein Distance. **[10]**

Edit and Levenstein distances are based on assignment of unit costs to edit operations. If we assign variable costs for insertion, deletion and substitution, then edit distance becomes **Weighted Edit Distance** and is equal to the minimum total cost of edit operations needed to transform the one frame into another frame. **[10]** In our experiments, we have accepted the cost of inserting a phoneme and the cost of deleting a phoneme to be 1 but assigned the cost of substituting phoneme /p_1/ with phoneme /p_2/ to SPCM[/p_1/,/p_2/] or SPGCM[/p_1/,/p_2/].

We have used the algorithm below **[9, 11, 12, 13]** to calculate the edit distance between two frames, frame A of length m and frame B of length n:

ED (A [1.....m], B [1.....n])
if m=0 return n
if n=0 return m
for i=1 to m T[i , 0] = i
for j=1 to n T[0 , j] = j
for i = 1 to m
for j = 1 to n
$p_{insertion}$ = *T[i , j-1] + price_of_insertion*
$p_{deletion}$ = *T[i-1 , j] + price_of_deletion*
if A[i] = B[j] K = 0 else K = price_of_substitution
$p_{substitution}$ = *T[i-1 , j-1] + K*
T[i , j] = min ($p_{insertion}$, $p_{deletion}$, $p_{substitution}$)
return T[m,n]

2.7.3. Normalized Matching Score (MS_N)

Assume there are **N** samples of the keyword (original samples and transformations) and each sample is transcribed into a phoneme array using Sestek Aligner. For each

distinct phoneme in those N samples of the keyword, phoneme frequencies are calculated using **Formula (8)**. Then,

$$MSN = \frac{1}{a} \sum_{i=1}^{a} K(Fpi) \qquad \textbf{(2.11)}$$

a: number of phonemes in the frame

$\mathbf{Fp_i}$: frequency of i^{th} phoneme in the frame calculated using **Formula (8)**

$$\mathbf{K(Fp_i)} = \begin{cases} 1 \text{ if } \mathbf{Fp_i} > 0 \\ 0 \text{ otherwise} \end{cases}$$

2.8. Frame Merging

The keyword spotter we have developed analyzes .wav files in the search database on the basis of frames (windows of a specific number of phonemes) and calculates if the frame is a keyword or not. The frames are chosen in a "***sliding windows***" manner.

If two frames are labeled as keywords and if they are overlapping by more than minimum intersection percent (**MIP**), which is parametric, we then merge those two frames (and other frames between them if there are any) into one bigger frame, calculate PFS_N, EDS_N, MS_N and OS_N of the new bigger frame and label that bigger frame as a keyword.

Suppose we have two frames, Frame1 and Frame2, where

$$Frame1(i) = \{Pi..... Pi + WL - 1\}$$
$$Frame2(j) = \{Pj..... Pj + WL - 1\}$$

WL is the window (frame) length.

If $\frac{i + WL - 1 - j}{WL} \geq \frac{MIP}{100}$ then we merge Frame1 and Frame2 (and other frames between them if there are any) into one frame, calculate PFS_N, EDS_N, MS_N and OS_N of that new frame and label it as a keyword.

In our experiments, we used MIP = 70.

2.9. Success Measures

Traditional KWS algorithms use performance measures such as Figure of Merit (**FOM**), Equal Error Rate (**EER**), **ROC** (Receiver Operating Characteristics) curves and **F-Measure**.

ROC curves show percentage of detection versus number of false alarms per keyword per hour. **[17]**

FOM is the average word detection rate over false alarm rates ranging from 0 to 10 false alarms per keyword per hour.

Equal Error Rate is defined to be the False Acceptance Rate (**FAR**) or False Rejection Rate (**FRR**) at the point where FAR = FRR. FRR is the percentage of correctly recognized and rejected utterances. FAR is the percentage of accepted utterances plus misrecognized and accepted utterances. **[8]**

F-Measure is defined as the harmonic mean of **precision** and **recall [5]**:

$$F = \frac{2 \times precision \times recall}{precision + recall} \quad \textbf{(2.12)}$$

$$precision = \frac{correct(q)}{output(q)} \qquad recall = \frac{correct(q)}{reference(q)} \quad \textbf{(2.13)}$$

correct (q): number of times the keyword q is found correctly

output (q): number of outputs (answers) of the system

reference (q): number of times the keyword q exists in the search database

Precision is a measure showing what ratio of the outputs of the system consists of correctly detected keywords, not false alarms. **Recall** is a measure showing what ratio of the keywords in the database is detected correctly.

Let us assume that there are 100 keywords in the search database but the spotter has detected 20 and only 12 of those 20 are correctly detected. The other 8 are false alarms. Then,

$$reference(q) = 100 \qquad output(q) = 20 \qquad correct(q) = 12$$

$$precision = \frac{12}{20} = \frac{3}{5} \qquad recall = \frac{12}{100} = \frac{3}{25} \qquad F = \frac{2 \times \frac{12}{20} \times \frac{12}{100}}{\frac{12}{20} + \frac{12}{100}} = \frac{1}{5}$$

We used F-Measure as the success measure in our experiments.

3. EXPERIMENTAL RESULTS

We used a part of SRPC database, available at Boğaziçi University Electrical and Electronics Engineering Department Signal and Image Processing Laboratory (BUSIM), in our tests.

We used 2 different search databases. The first search database consists of 21 isolated words taken from 26 different speakers, of which 15 are men and 11 are women. The 21 isolated words are numbers listed below:

SIFIR (0)	BİR (1)	İKİ (2)
ÜÇ (3)	DÖRT (4)	BEŞ (5)
ALTI (6)	YEDİ (7)	SEKİZ (8)
DOKUZ (9)	ON (10)	ONBİR (11)
ON İKİ (12)	ON ÜÇ (13)	ON DÖRT (14)
ON BEŞ (15)	ON ALTI (16)	ON YEDİ (17)
ON SEKİZ (18)	ONDOKUZ (19)	YİRMİ (20)
OTUZ (30)	KIRK (40)	ELLİ (50)
ATMIŞ (60)	YETMİŞ (70)	SEKSEN (80)
DOKSAN (90)	YÜZ (100)	BİN (1000)

The second search database consists of the same 21 isolated words, which are the numbers above, plus 15 sentences taken from the same 26 speakers. The 15 sentences are listed below:

İKİ BİN
(two thousand)

BİN DOKUZ YÜZ DOKSAN
(one thousand nine hundred ninety)

BİN DOKUZ YÜZ SEKSEN
(one thousand nine hundred eighty)

BİN DOKUZ YÜZ ATMIŞ

(one thousand nine hundred sixty)

KADINLAR KENDİLERİNE KARŞI

(women are against themselves)

NE KADAR AĞAÇ DİKTİKLERİNİ SÖYLEDİ

(said how many trees they planted)

VE BÜTÜN BUNLARIN ÜZERİNDEN

(and after all of these)

ERKEKLERE ÇALIŞMALARINDAN BİR PAY

(a share to the men for what they did)

CAHİLLİKTEN BAŞKA BİR ŞEY DEĞİLDİR

(nothing different from ignorance)

YÖRELERİNDE BULUNMAKTADIR

(exists in the neighborhood)

BUNDAN SONRA DA KENDİSİNE

(after that to him)

SEKİZ OLARAK BİLDİRİLMEKTEDİR

(announced as eight)

İNSANLARIN ARASINDAYKEN

(when amongst people)

AMAÇLARININ NE OLDUĞUNU

(what their aim was)

BİR İKİ ÜÇ DÖRT BEŞ ALTI YEDİ SEKİZ DOKUZ ON

(one two three four five six seven eight nine ten)

At the input side, we used 3 keywords: “SIFIR (0)”, “DOKUZ (9)” and “YETMİŞ (70)”. We have made 2 different tests. In the first test, we collected one sample for each of the 3 keywords from only one speaker who is a man. In the second test, we collected one sample for each of the 3 keywords from 8 speakers of whom 4 are men and 4 are women.

We have made different experiments to find answers to the following questions:

1- How does collecting samples from more speakers at the input side affect the success of the system?
2- Does using transformations, which are synthetically generated by morphing the input voices, in addition to natural inputs enhance the success of the system?
3- Are the results obtained by using SPCM or SPGCM better or worse than the results obtained by not using them? What is the performance difference between using SPCM and SPGCM?
4- Does having sentences together with isolated words in the search database enhance or degrade the success of the system compared with the case in which only isolated words exist in the search database?
5- How does driving the spotter with voice input instead of text input affect the success of the system?

3.1. Effect of Collecting Samples from More Speakers

We have made 5 different tests to measure the effect of collecting samples from more speakers. The results of these tests can be seen in **Table 3.1.**

As a result of these tests, we conclude that taking input samples from more speakers always improves the success of the system dramatically.

TEST 1

We have used the 3 keywords to spot and took one sample of each keyword from 1 speaker. The search database consists of 21 isolated words taken from 26 different speakers. The best F-Measure we obtained is 0.31. (See **Figure A.1** in Appendix)

We have used the 3 keywords to spot and took one sample of each keyword from 8 speakers. The search database consists of 21 isolated words taken from 26 different speakers. The best F-Measure we obtained is 0.69. (See **Figure A.2** in Appendix)

Table 3.1. Effect of collecting samples from more speakers

TEST CASE				F-Measure	
Search DB	**Morphing**	**SPCM**	**SPGCM**	**1 speaker**	**8 speakers**
Only isolated words	No	No	No	0.31	0.69
Only isolated words	No	No	Yes	0.31	0.70
Only isolated words	No	Yes	No	0.29	0.70
Only isolated words	3 morphs for each natural input	No	No	0.39	0.68
Isolated words + sentences	No	No	No	0.26	0.64

We see that taking input samples from more speakers dramatically increases the system performance from 0.31 to 0.69.

TEST 2

We have used the 3 keywords to spot and took one sample of each keyword from 1 speaker. The search database consists of 21 isolated words taken from 26 different speakers. SPGCM is used. The best F-Measure we obtained is 0.31. (See **Figure A.3** in Appendix)

We have used the 3 keywords to spot and took one sample of each keyword from 8 speakers. The search database consists of 21 isolated words taken from 26 different

speakers. SPGCM is used. The best F-Measure we obtained is 0.70. (See **Figure A.4** in Appendix)

We see that taking input samples from more speakers dramatically increases the system performance from 0.31 to 0.70 even when SPGCM is used.

TEST 3

We have used the 3 keywords to spot and took one sample of each keyword from 1 speaker. The search database consists of 21 isolated words taken from 26 different speakers. SPCM is used. The best F-Measure we obtained is 0.29. (See **Figure A.5** in Appendix)

We have used the 3 keywords to spot and took one sample of each keyword from 8 speakers. The search database consists of 21 isolated words taken from 26 different speakers. SPCM is used. The best F-Measure we obtained is 0.70. (See **Figure A.6** in Appendix)

We see that taking input samples from more speakers dramatically increases the system performance from 0.29 to 0.70 even when SPCM is used.

TEST 4

We have used the 3 keywords to spot and took one sample of each keyword from 1 speaker. In addition, we produced 3 extra transformations for each of the keyword samples. The search database consists of 21 isolated words taken from 26 different speakers. The best F-Measure we obtained is 0.39. (See **Figure A.7** in Appendix)

We have used the 3 keywords to spot and took one sample of each keyword from 8 speakers. In addition, we produced 3 extra transformations for each of the keyword samples. The search database consists of 21 isolated words taken from 26 different speakers. The best F-Measure we obtained is 0.68. (See **Figure A.8** in Appendix)

We see that taking input samples from more speakers dramatically increases the system performance from 0.39 to 0.68 even when morphing is done at the input.

TEST 5

We have used the 3 keywords to spot and took one sample of each keyword from 1 speaker. The search database consists of 21 isolated words plus 15 sentences taken from 26 different speakers. The best F-Measure we obtained is 0.26. (See **Figure A.9** in Appendix)

We have used the 3 keywords to spot and took one sample of each keyword from 8 speakers. The search database consists of 21 isolated words plus 15 sentences taken from 26 different speakers. The best F-Measure we obtained is 0.64. (See **Figure A.10** in Appendix)

We see that taking input samples from more speakers dramatically increases the system performance from 0.26 to 0.64 even when there are sentences in addition to isolated words in the search database.

3.2. Effect of Using Synthetically Generated Samples

We have made 2 different tests to measure the effect of using synthetically generated data. The results of these tests can be seen in **Table 3.2.**

As a result of these tests, we conclude that synthetically generated samples can help to increase F-Measure if there are not enough natural samples at the input. However, synthetically generated samples cannot increase F-Measure more if there are already enough natural samples at the input. The F-Measure may even decrease a little bit in such a case.

Table 3.2. Effect of using synthetically generated samples

TEST CASE			F-Measure				
Search DB	**SPCM**	**SPGCM**	**1 spk.**	**1 spk. 3 morphs**	**1 spk. 7 morphs**	**8 spks.**	**8 spks. 3 morphs**
Only isolated words	No	No	0.31	0.39	0.49	0.69	0.68
Isolated words + sentences	No	No	0.26	0.28	0.33	0.64	0.58

TEST 1

We have used 3 keywords to spot and took one sample of each keyword from 1 speaker. The search database consists of 21 isolated words taken from 26 different speakers. The best F-Measure we obtained is 0.31. (See **Figure A.11** in Appendix)

We have used 3 keywords to spot and took one sample of each keyword from 1 speaker. In addition, we produced 3 extra transformations for each of the keyword samples. The search database consists of 21 isolated words taken from 26 different speakers. The best F-Measure we obtained is 0.39. (See **Figure A.12** in Appendix)

We have used 3 keywords to spot and took one sample of each keyword from 1 speaker. In addition, we produced 7 extra transformations for each of the keyword samples. The search database consists of 21 isolated words taken from 26 different speakers. The best F-Measure we obtained is 0.49. (See **Figure A.13** in Appendix)

We have used 3 keywords to spot and took one sample of each keyword from 8 speakers. The search database consists of 21 isolated words taken from 26 different speakers. The best F-Measure we obtained is 0.69. (See **Figure A.14** in Appendix)

We see that F-Measure of the system, when synthetically generated data is used, is higher than F-Measure of the system when only natural samples are used. Using 1 natural sample plus 3 synthetically generated samples (totally 4 input samples) gives higher F-Measure than using 1 natural sample only. Using 1 natural sample plus 7 synthetically generated samples (totally 8 input samples) gives higher F-Measure than using 1 natural sample plus 3 synthetically generated samples (totally 4 input samples) but lower F-Measure than using 8 natural samples. Therefore, we conclude that synthetically generated samples are not as good as natural samples in representing the keyword but having some synthetic samples in addition to natural samples at the input side is better than having only natural samples.

We have used 3 keywords to spot and took one sample of each keyword from 8 speakers. In addition, we produced 3 extra transformations for each of the keyword samples. The search database consists of 21 isolated words taken from 26 different speakers. The best F-Measure we obtained is 0.68. (See **Figure A.15** in Appendix)

We see that using 8 natural samples gives F-Measure of 0.69 and using 8 natural samples plus 24 synthetically generated samples (3 transformations for each natural sample) gives F-Measure of 0.68. That is, synthetically generated samples can help to increase F-Measure if there are not enough natural samples at the input. However, synthetically generated samples cannot increase F-Measure more if there are already enough natural samples at the input. The F-Measure may even decrease a little bit in such a case.

TEST 2

We have used 3 keywords to spot and took one sample of each keyword from 1 speaker. The search database consists of 21 isolated words plus 15 sentences taken from 26 different speakers. The best F-Measure we obtained is 0.26. (See **Figure A.16** in Appendix)

We have used 3 keywords to spot and took one sample of each keyword from 1 speaker. In addition, we produced 3 extra transformations for each of the keyword samples. The search database consists of 21 isolated words plus 15 sentences taken from 26 different speakers. The best F-Measure we obtained is 0.28. (See **Figure A.17** in Appendix)

We have used 3 keywords to spot and took one sample of each keyword from 1 speaker. In addition, we produced 7 extra transformations for each of the keyword samples. The search database consists of 21 isolated words plus 15 sentences taken from 26 different speakers. The best F-Measure we obtained is 0.33. (See **Figure A.18** in Appendix)

We have used 3 keywords to spot and took one sample of each keyword from 8 speakers. The search database consists of 21 isolated words plus 15 sentences taken from 26 different speakers. The best F-Measure we obtained is 0.64. (See **Figure A.19** in Appendix)

We see that F-Measure of the system, when synthetically generated data is used, is higher than F-Measure of the system when only natural samples are used even when there are sentences in the search database together with isolated words. Using 1 natural sample plus 3 synthetically generated samples (totally 4 input samples) gives higher F-Measure than using 1 natural sample only. Using 1 natural sample plus 7 synthetically generated samples (totally 8 input samples) gives higher F-Measure than using 1 natural sample plus 3 synthetically generated samples (totally 4 input samples) but lower F-Measure than using 8 natural samples. Therefore, we conclude that synthetically generated samples are not as good as natural samples in representing the keyword but having some synthetic samples in addition to natural samples at the input side is better than having only natural samples even when there are sentences in the search database.

We have used 3 keywords to spot and took one sample of each keyword from 8 speakers. In addition, we produced 3 extra transformations for each of the keyword samples. The search database consists of 21 isolated words plus 15 sentences taken from

26 different speakers. The best F-Measure we obtained is 0.58. (See **Figure A.20** in Appendix)

We see that using 8 natural samples gives F-Measure of 0.64 and using 8 natural samples plus 24 synthetically generated samples (3 transformations for each natural sample) gives F-Measure of 0.58. That is, even when there are sentences in the search database, synthetically generated samples can help to increase F-Measure if there are not enough natural samples at the input. However, synthetically generated samples cannot increase F-Measure more if there are already enough natural samples at the input. The F-Measure may even decrease a little bit in such a case.

3.3. Effect of Using SPCM and SPGCM

We have made 2 different tests to measure the effect of using SPCM and SPGCM. The results of these tests can be seen in **Table 3.3.**

As a result of these tests, we conclude that using SPCM or SPGCM can help to increase F-Measure by 1 per cent.

Table 3.3. Effect of using SPCM and SPGCM

TEST CASE			**F-Measure**		
Number of Speakers	**Search DB**	**Morphing**	**No SPCM No SPGCM**	**SPCM**	**SPGCM**
8	Only isolated words	No	0.69	0.70	0.70
8	Isolated words + sentences	No	0.64	0.63	0.64

TEST 1

We have used 3 keywords to spot and took one sample of each keyword from 8 speakers. The search database consists of 21 isolated words taken from 26 different speakers. The best F-Measure we obtained is 0.69. (See **Figure A.21** in Appendix)

We have used 3 keywords to spot and took one sample of each keyword from 8 speakers. The search database consists of 21 isolated words taken from 26 different speakers. SPGCM is used. The best F-Measure we obtained is 0.70. (See **Figure A.22** in Appendix)

We have used 3 keywords to spot and took one sample of each keyword from 8 speakers. The search database consists of 21 isolated words taken from 26 different speakers. SPCM is used. The best F-Measure we obtained is 0.70. (See **Figure A.23** in Appendix)

We see that using statistical phoneme confusion data increases F-Measure of the system by 1 per cent when input samples are collected from 8 speakers and there are only isolated words in the search database. Using SPCM or SPGCM does not produce much difference.

TEST 2

We have used 3 keywords to spot and took one sample of each keyword from 8 speakers. The search database consists of 21 isolated words plus 15 sentences taken from 26 different speakers. The best F-Measure we obtained is 0.64. (See **Figure A.24** in Appendix)

We have used 3 keywords to spot and took one sample of each keyword from 8 speakers. The search database consists of 21 isolated words plus 15 sentences taken from 26 different speakers. SPGCM is used. The best F-Measure we obtained is 0.64. (See **Figure A.25** in Appendix)

We have used 3 keywords to spot and took one sample of each keyword from 8 speakers. The search database consists of 21 isolated words plus 15 sentences taken from 26 different speakers. SPCM is used. The best F-Measure we obtained is 0.63. (See **Figure A.26** in Appendix)

We see that using statistical phoneme confusion data neither increases nor decreases the F-Measure when input samples are collected from 8 speakers and there are sentences in the search database together with isolated words. Using SPCM or SPGCM does not produce much difference.

3.4. Effect of Having Sentences in Search Database

We have made 6 different tests to measure the effect of having sentences in search database. The results of these tests can be seen in **Table 3.4.**

As a result of these tests, we conclude that having sentences in search database may decrease F-Measure by 5 per cent to 11 per cent.

TEST 1

We have used the 3 keywords to spot and took one sample of each keyword from 1 speaker. The search database consists of 21 isolated words taken from 26 different speakers. The best F-Measure we obtained is 0.31. (See **Figure A.27** in Appendix)

We have used the 3 keywords to spot and took one sample of each keyword from 1 speaker. The search database consists of 21 isolated words plus 15 sentences taken from 26 different speakers. The best F-Measure we obtained is 0.26. (See **Figure A.28** in Appendix)

Table 3.4. Effect of having sentences in search database

TEST CASE				F-Measure	
				Search DB	
Number of Speakers	Morphing	SPCM	SPGCM	Only isolated words	Isolated words + sentences
1	No	No	No	0.31	0.26
8	No	No	No	0.69	0.64
1	3 morphs for each natural input	No	No	0.39	0.28
8	3 morphs for each natural input	No	No	0.68	0.58
8	No	No	Yes	0.70	0.64
8	No	Yes	No	0.70	0.63

We see that having sentences in addition to isolated words in the search database decreases F-Measure of the system by 5 per cent when input samples are collected from 1 speaker.

TEST 2

We have used the 3 keywords to spot and took one sample of each keyword from 8 speakers. The search database consists of 21 isolated words taken from 26 different speakers. The best F-Measure we obtained is 0.69. (See **Figure A.29** in Appendix)

We have used the 3 keywords to spot and took one sample of each keyword from 8 speakers. The search database consists of 21 isolated words plus 15 sentences taken from 26 different speakers. The best F-Measure we obtained is 0.64. (See **Figure A.30** in Appendix)

We see that having sentences in addition to isolated words in the search database decreases F-Measure of the system by 5 per cent when input samples are collected from 8 speakers.

TEST 3

We have used the 3 keywords to spot and took one sample of each keyword from 1 speaker. In addition, we produced 3 extra transformations for each of the keyword samples. The search database consists of 21 isolated words taken from 26 different speakers. The best F-Measure we obtained is 0.39. (See **Figure A.31** in Appendix)

We have used 3 keywords to spot and took one sample of each keyword from 1 speaker. In addition, we produced 3 extra transformations for each of the keyword samples. The search database consists of 21 isolated words plus 15 sentences taken from 26 different speakers. The best F-Measure we obtained is 0.28. (See **Figure A.32** in Appendix)

We see that having sentences in addition to isolated words in the search database decreases F-Measure of the system by 11 per cent when input samples are collected from 1 speaker and 3 extra synthetically generated transformations for each of the natural inputs are used at the input side.

TEST 4

We have used 3 keywords to spot and took one sample of each keyword from 8 speakers. In addition, we produced 3 extra transformations for each of the keyword samples. The search database consists of 21 isolated words taken from 26 different speakers. The best F-Measure we obtained is 0.68. (See **Figure A.33** in Appendix)

We have used 3 keywords to spot and took one sample of each keyword from 8 speakers. In addition, we produced 3 extra transformations for each of the keyword samples. The search database consists of 21 isolated words plus 15 sentences taken from 26 different speakers. The best F-Measure we obtained is 0.58. (See **Figure A.34** in Appendix)

We see that having sentences in addition to isolated words in the search database decreases F-Measure of the system by 10 per cent when input samples are collected from 8 speakers and 3 extra synthetically generated transformations for each of the natural inputs are used at the input side.

TEST 5

We have used 3 keywords to spot and took one sample of each keyword from 8 speakers. The search database consists of 21 isolated words taken from 26 different speakers. SPGCM is used. The best F-Measure we obtained is 0.70. (See **Figure A.35** in Appendix)

We have used 3 keywords to spot and took one sample of each keyword from 8 speakers. The search database consists of 21 isolated words plus 15 sentences taken from 26 different speakers. SPGCM is used. The best F-Measure we obtained is 0.64. (See **Figure A.36** in Appendix)

We see that having sentences in addition to isolated words in the search database decreases F-Measure of the system by 6 per cent when input samples are collected from 8 speakers and SPGCM is used.

TEST 6

We have used 3 keywords to spot and took one sample of each keyword from 8 speakers. The search database consists of 21 isolated words taken from 26 different speakers. SPCM is used. The best F-Measure we obtained is 0.70. (See **Figure A.37** in Appendix)

We have used 3 keywords to spot and took one sample of each keyword from 8 speakers. The search database consists of 21 isolated words plus 15 sentences taken from 26 different speakers. SPCM is used. The best F-Measure we obtained is 0.63. (See **Figure A.38** in Appendix)

We see that having sentences in addition to isolated words in the search database decreases F-Measure of the system by 7 per cent when input samples are collected from 8 speakers and SPCM is used.

3.5. Effect of Taking Voice Input Instead of Text Input

We have made 2 different tests to measure the effect of taking the input to the system in the form of voice instead of text. The results of these tests can be seen in **Table 3.5.**

As a result of these tests, we conclude that if the system is driven with sufficient number of voice samples of the keyword, it succeeds better than the case when it is driven with text input.

Table 3.5. Effect of taking voice input instead of text input

TEST CASE	**F-Measure (No Morphing, No SPCM, No SPGCM)**		
	Number of Speakers		
Search DB	**0 (Text Input)**	**1**	**8**
Only isolated words	0.57	0.31	0.69
Isolated words + sentences	0.53	0.26	0.64

TEST 1

We have used the 3 keywords to spot and given text input to the system. The search database consists of 21 isolated words taken from 26 different speakers. The best F-Measure we obtained is 0.57. (See **Figure A.39** in Appendix)

We have used the 3 keywords to spot and took one sample of each keyword from 1 speaker. The search database consists of 21 isolated words taken from 26 different speakers. The best F-Measure we obtained is 0.31. (See **Figure A.40** in Appendix)

We have used the 3 keywords to spot and took one sample of each keyword from 8 speakers. The search database consists of 21 isolated words taken from 26 different speakers. The best F-Measure we obtained is 0.69. (See **Figure A.41** in Appendix)

We see that, if the system is driven with sufficient number of voice samples of the keyword, it succeeds better than the case when it is driven with text input.

TEST 2

We have used the 3 keywords to spot and given text input to the system. The search database consists of 21 isolated words plus 15 sentences taken from 26 different speakers. The best F-Measure we obtained is 0.53. (See **Figure A.42** in Appendix)

We have used the 3 keywords to spot and took one sample of each keyword from 1 speaker. The search database consists of 21 isolated words plus 15 sentences taken from 26 different speakers. The best F-Measure we obtained is 0.26. (See **Figure A.43** in Appendix)

We have used the 3 keywords to spot and took one sample of each keyword from 8 speakers. The search database consists of 21 isolated words plus 15 sentences taken from 26 different speakers. The best F-Measure we obtained is 0.64. (See **Figure A.44** in Appendix)

We see that, if the system is driven with sufficient number of voice samples of the keyword, it succeeds better than the case when it is driven with text input.

3.6. Summary of Experimental Results

1- Taking input samples from more speakers improves the success of the system dramatically.
2- Synthetically generated samples can help to increase F-Measure if there are not enough natural samples at the input. However, they may not increase F-Measure more if there are already enough natural samples at the input. The F-Measure may even decrease a little bit in such a case.
3- Using SPCM or SPGCM does not seem to have a positive or negative effect on F-Measure.
4- Having sentences in search database may decrease F-Measure by 5 per cent to 11 per cent.
5- If the spotter is driven with sufficient number of voice samples of the keyword, it succeeds better than the case in which it is driven with text input.

4. CONCLUSIONS

In this thesis, we have concentrated on 3 important weaknesses of traditional keyword spotters and tried to develop new approaches to minimize their negative effects:

1- Traditional KWS systems are mostly text driven. Therefore, there is lack of variety at the input side. We have developed a voice driven KWS system to overcome this problem.
2- Traditional KWS systems do not take into account the effects of factors like gender, intonation, accent, emotional mood, etc. on pronunciation. We have performed morphing and produced synthetically generated samples of natural voice inputs to simulate effects of such factors.
3- Traditional KWS systems are mostly language dependent. The KWS system we have developed uses the same ASR engine both at the input side and at the search database side and we strongly believe that this will make the system language independent. In addition, the system is able to use multiple ASR engines of different languages in parallel.

We have driven the keyword spotter not with textual phonetic transcription of the keyword but with natural voice samples of the keyword. That way, we have the chance of collecting many different pronunciations of the same keyword from many different speakers and thus increase the variety at the input side. We conclude that if the system is driven with sufficient number of voice samples of the keyword, it succeeds better than the case when it is driven with text input. In addition, taking natural voice samples from more speakers at the input side always improves F-Measure of the system.

Taking the input in voice format has also given us the chance of producing synthetically generated transformations of the natural voice samples and using natural voice samples and synthetically generated voice samples together. These synthetically generated samples can simulate different pronunciations of the keyword if different speakers of different gender, intonation, emotional mood, age, etc. would pronounce it. Factors such as gender, intonation, emotional mood, age, etc. play an important role on

pronunciation. Having so many different samples of the keyword at the input side has compensated negative effects of gender, intonation, emotional mood, age, etc. differences between the speakers from whom we collected the keyword samples and the speakers who spoke the records in the search database. We conclude that synthetically generated samples are not as good as natural samples in representing the keyword but they can help to increase F-Measure of the system for approximately 10 per cent, if there are not enough natural samples at the input. However, synthetically generated samples cannot increase F-Measure more if there are already enough natural samples at the input. F-Measure may even decrease a little bit in such a case.

We have used the **same** ASR engine both at the input side and at the search database side and we believe that it will make the spotter become language independent. Because, even if the ASR engine is of a different language than the language spoken at the search database side or at the input side, it will face similar errors at both sides and those errors will mostly compensate each other. In addition, the system is able to use many ASR engines of different languages in parallel. However, we did not test language independency of the system.

We have also taken context dependency into account and collected natural voice samples of the keyword both when the keyword is uttered isolately and in sentences with suffices and prefixes. The users are then able to identify the keyword beginning and ending times in the sentences they uttered. That way, we have the chance of gathering different phoneme transcriptions of the same keyword for different contexts. We did not measure the effect of this approach to the success of the system but we believe that this variety at the input side will increase the F-Measure.

We calculated statistical phoneme confusion probabilities and fed the edit distance algorithm with those pre-calculated statistical phoneme confusion data. We have observed that using either SPCM or SPGCM does not seem to have a positive or negative effect on F-Measure.

We have made experiments using 2 different search databases where the first search database consists of 21 isolated words taken from 26 different speakers and the second

search database consists of 21 isolated words plus 15 sentences taken from 26 different speakers. We have observed that when there are sentences in the search database, F-Measure of the system may degrade by 5 per cent to 11 per cent.

5. FUTURE WORK

We strongly believe that the KWS system we have developed is language independent. Because, it is capable of using ASR engines of different languages in parallel. In addition, it uses the same ASR engine for both the input side and search database side and we think that even if the ASR engine makes errors, those errors will be similar for both sides and they will mostly tolerate each other. However, we did not test language independency of the system. We have used only one ASR engine, which is a Turkish phoneme aligner. Phonetic aligners of different languages may be used in parallel and language independency tests can be performed as a future work.

We have calculated statistical phoneme and phoneme group confusion probabilities on a database where 21 isolated words and 15 sentences from 26 different speakers are recorded and labeled. Phoneme confusion probabilities may be calculated over a bigger database. Using those new probabilities, which will be more statistical, may contribute to the system performance.

We have grouped phonemes into 5 phoneme groups and selected the phonemes of each phoneme group according to our knowledge of Turkish language and Turkish phonemes. However, phonemes can be regrouped according to the SPCM calculated. Phonemes, which are more probable to confuse with each other according to the SPCM may be put into the same phoneme group and phoneme group confusion probabilities may be recalculated according to this assumption as a future work. This way, using SPGCM may contribute to the system performance.

Currently, Sestek VoDi application, which we have used to generate synthetic transformations of the natural voice inputs, is not integrated into the Speech Driven Keyword Spotter application. Sestek VoDi can be integrated into the main application as a future work.

We have concluded that using voice inputs give better F-Measure than using text input, which is the correct phonetic transcription of the keyword. However, using voice inputs and the correct phonetic transcription of the keyword together at the input side may give the best F-Measure. This condition may also be tested as a future work.

APPENDIX A: GRAPHICS OF EXPERIMENTAL OUTPUTS

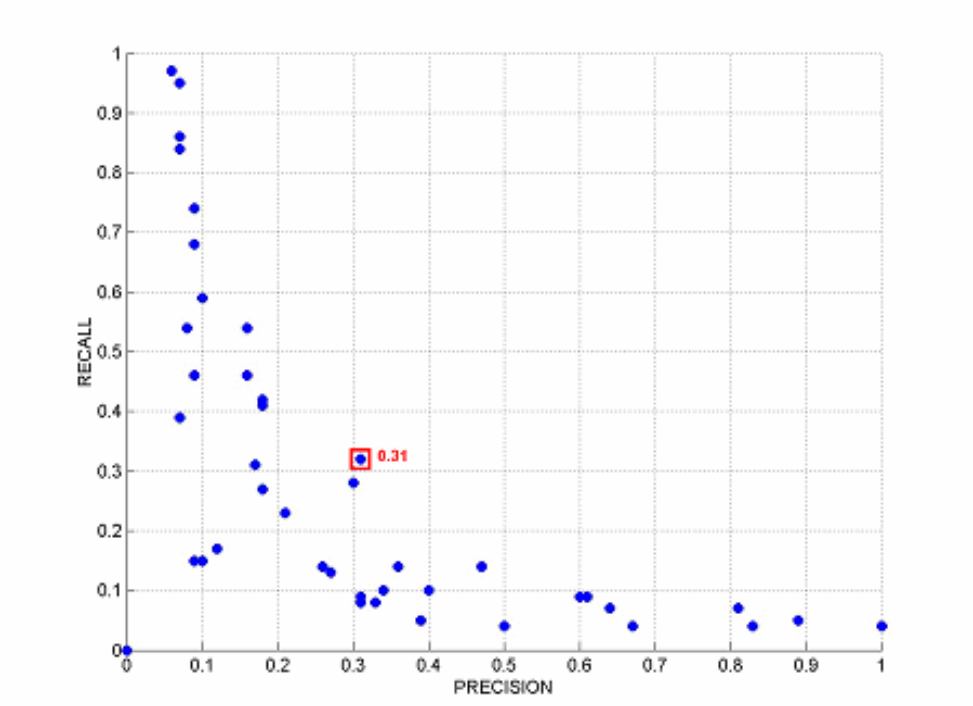


Figure A.1. Precision vs. Recall when input samples are collected from 1 speaker

(Only isolated words in search database, no morphing, no SPCM, no SPGCM)

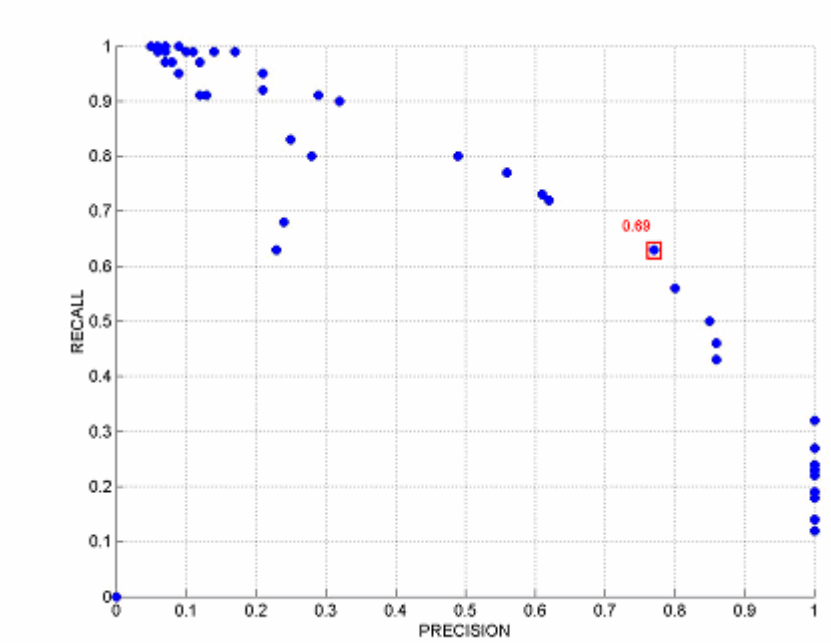


Figure A.2. Precision vs. Recall when input samples are collected from 8 speakers

(Only isolated words in search database, no morphing, no SPCM, no SPGCM)

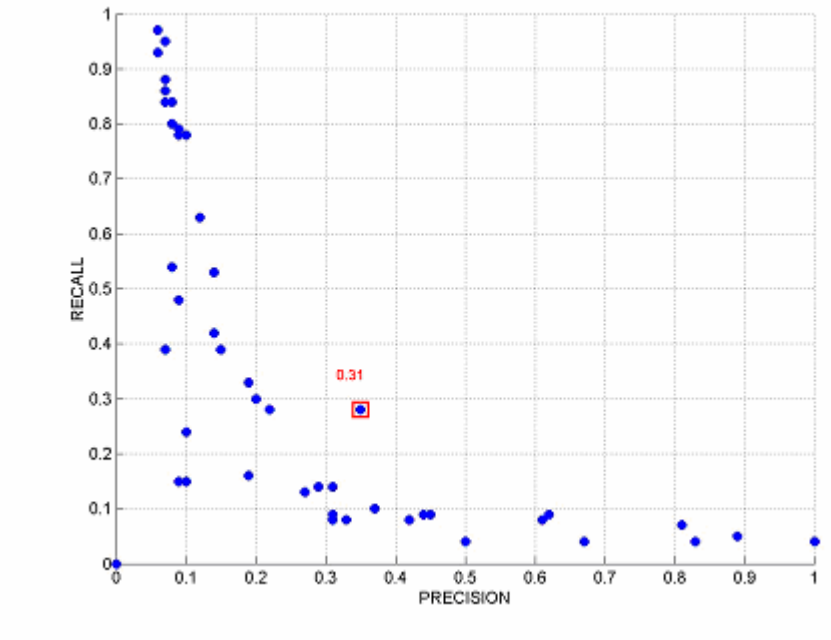


Figure A.3. Precision vs. Recall when input samples are collected from 1 speaker

(Only isolated words in search database, no morphing, no SPCM, **SPGCM is used**)

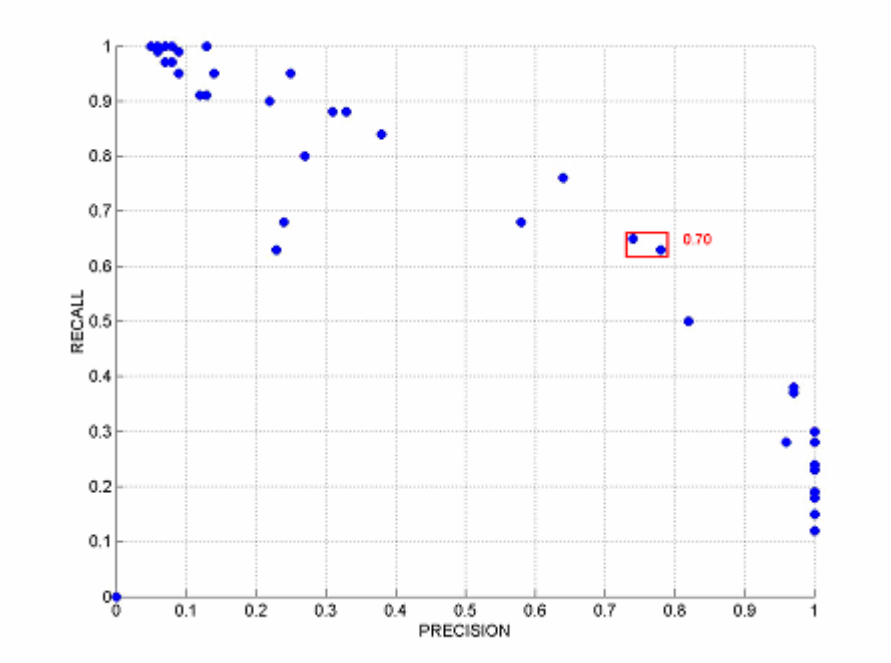


Figure A.4. Precision vs. Recall when input samples are collected from 8 speakers

(Only isolated words in search database, no morphing, no SPCM, **SPGCM is used**)

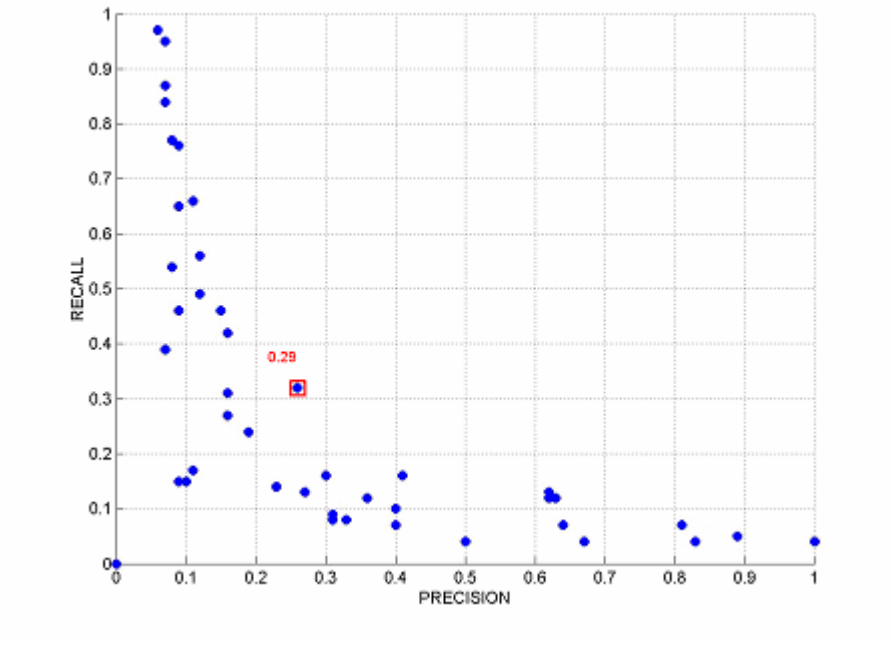


Figure A.5. Precision vs. Recall when input samples are collected from 1 speaker

(Only isolated words in search database, no morphing, **SPCM is used**, no SPGCM)

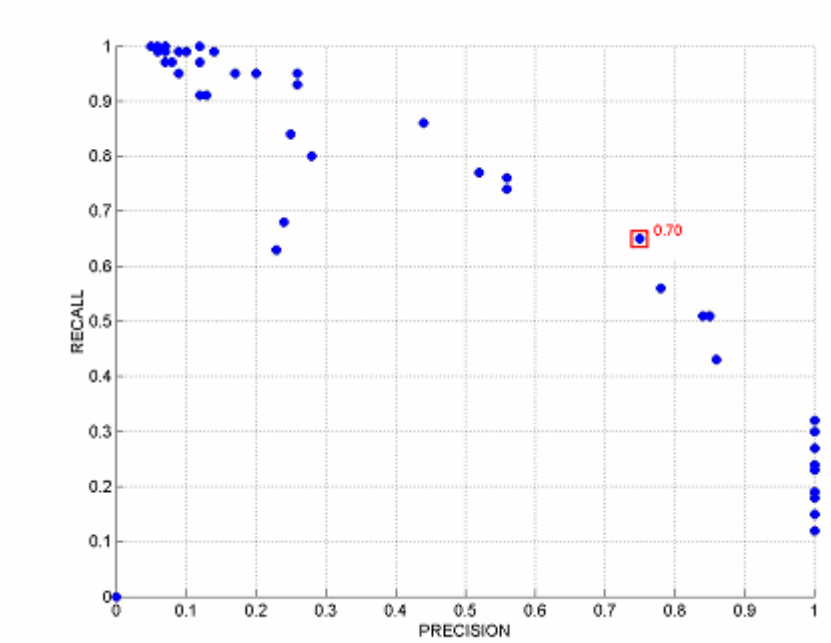


Figure A.6. Precision vs. Recall when input samples are collected from 8 speakers

(Only isolated words in search database, no morphing, **SPCM is used**, no SPGCM)

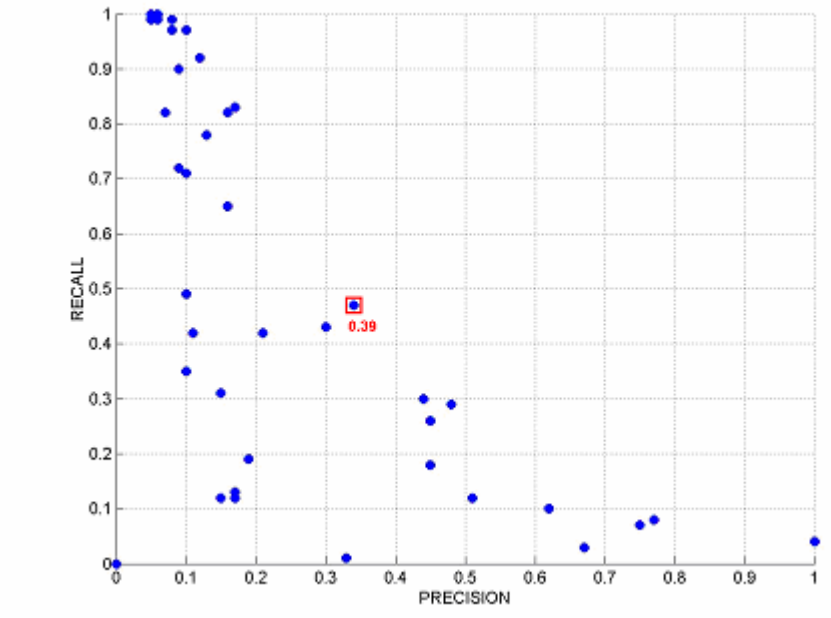


Figure A.7. Precision vs. Recall when input samples are collected from 1 speaker

(Only isolated words in search database, **morphing used (3 extra)**, no SPCM, no SPGCM)

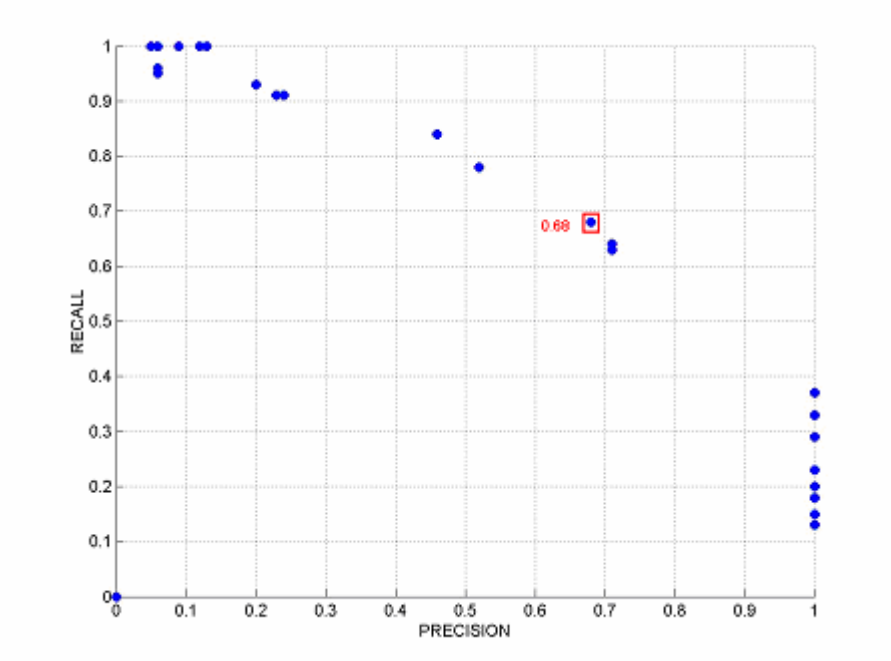


Figure A.8. Precision vs. Recall when input samples are collected from 8 speakers

(Only isolated words in search database, **morphing used (3 extra)**, no SPCM, no SPGCM)

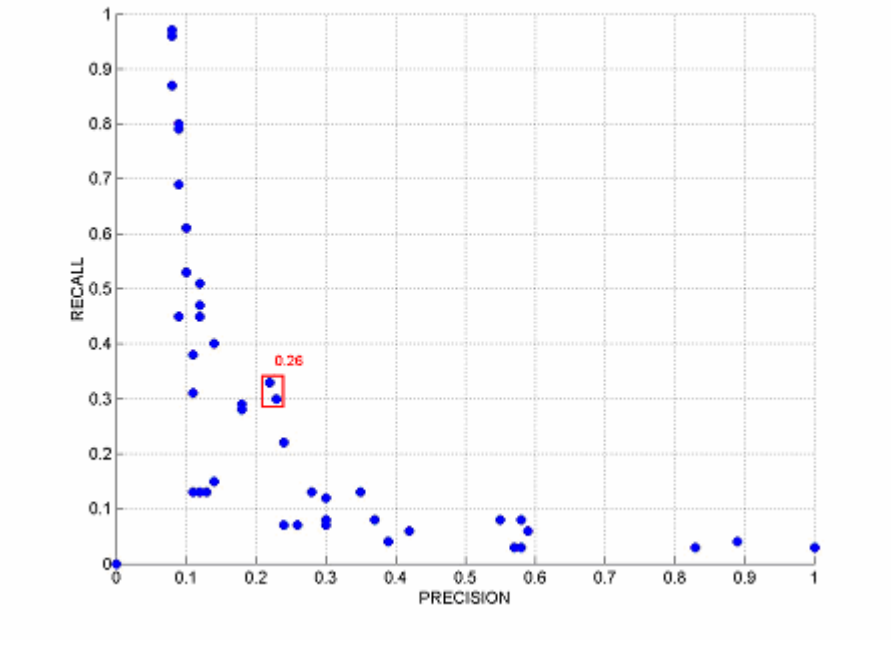


Figure A.9. Precision vs. Recall when input samples are collected from 1 speaker

(Isolated words and **sentences** in search database, no morphing, no SPCM, no SPGCM)

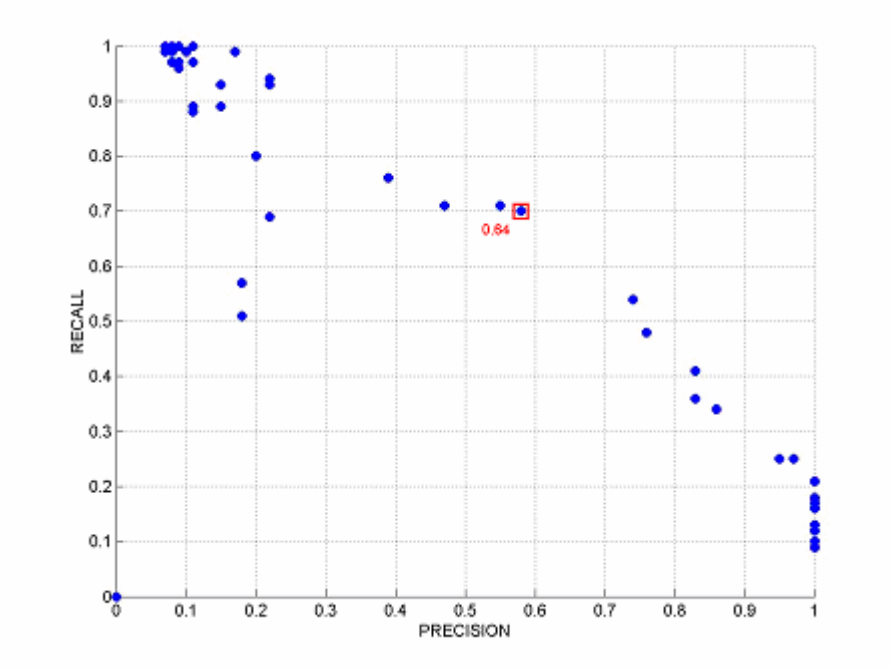


Figure A.10. Precision vs. Recall when input samples are collected from 8 speakers

(Isolated words and **sentences** in search database, no morphing, no SPCM, no SPGCM)

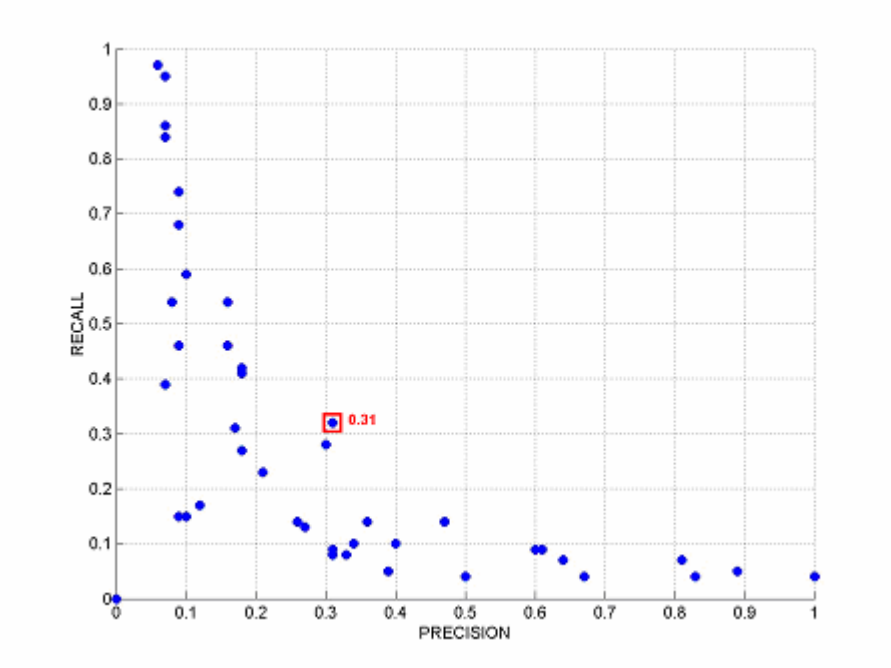


Figure A.11. Precision vs. Recall when input samples are collected from 1 speaker

(Only isolated words in search database, no morphing, no SPCM, no SPGCM)

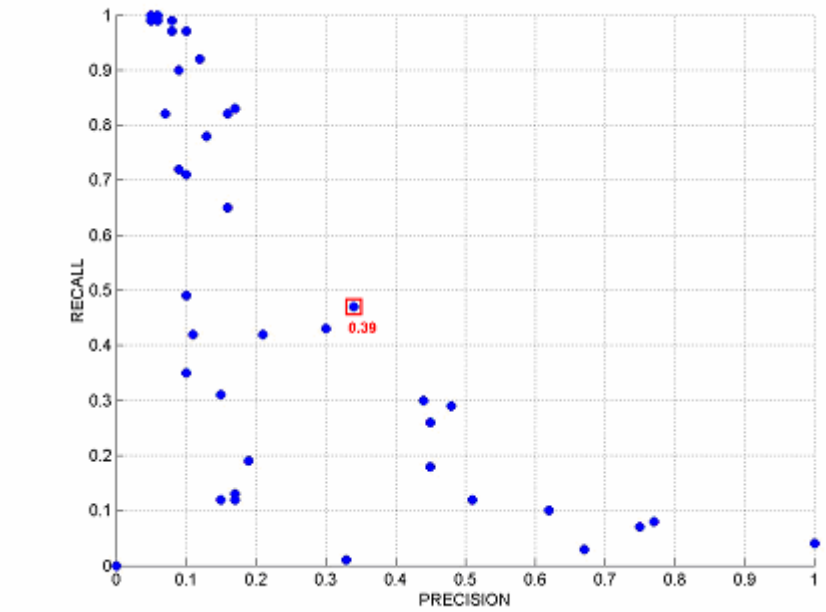


Figure A.12. Precision vs. Recall when input samples are collected from 1 speaker

(Only isolated words in search database, **morphing used (3 extra)**, no SPCM, no SPGCM)

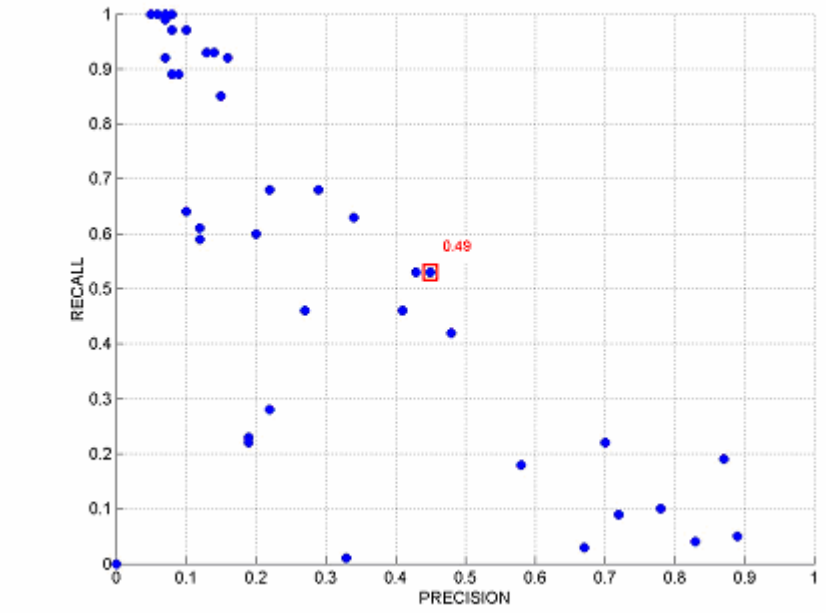


Figure A.13. Precision vs. Recall when input samples are collected from 1 speaker

(Only isolated words in search database, **morphing used (7 extra)**, no SPCM, no SPGCM)

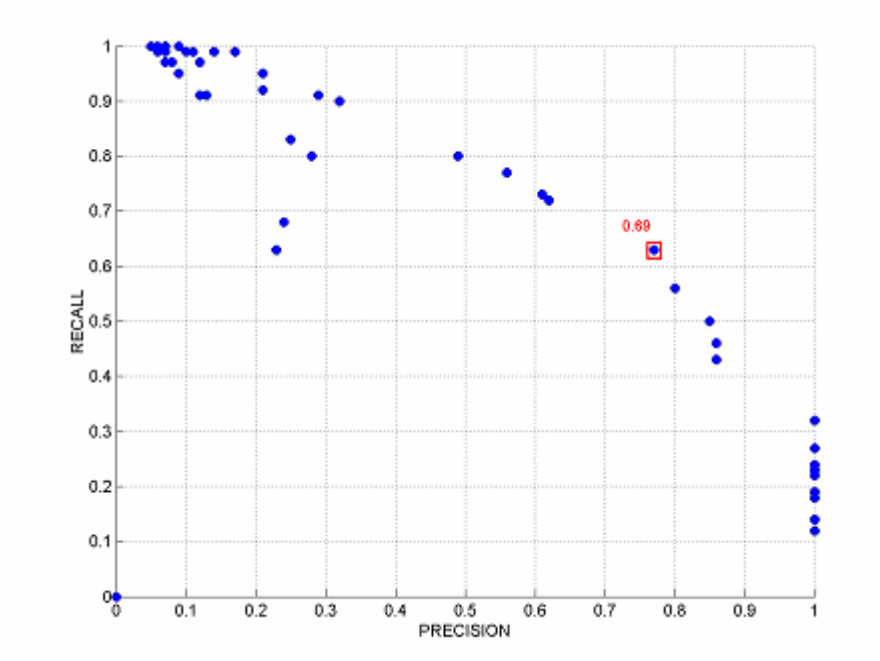


Figure A.14. Precision vs. Recall when input samples are collected from 8 speakers

(Only isolated words in search database, no morphing, no SPCM, no SPGCM)

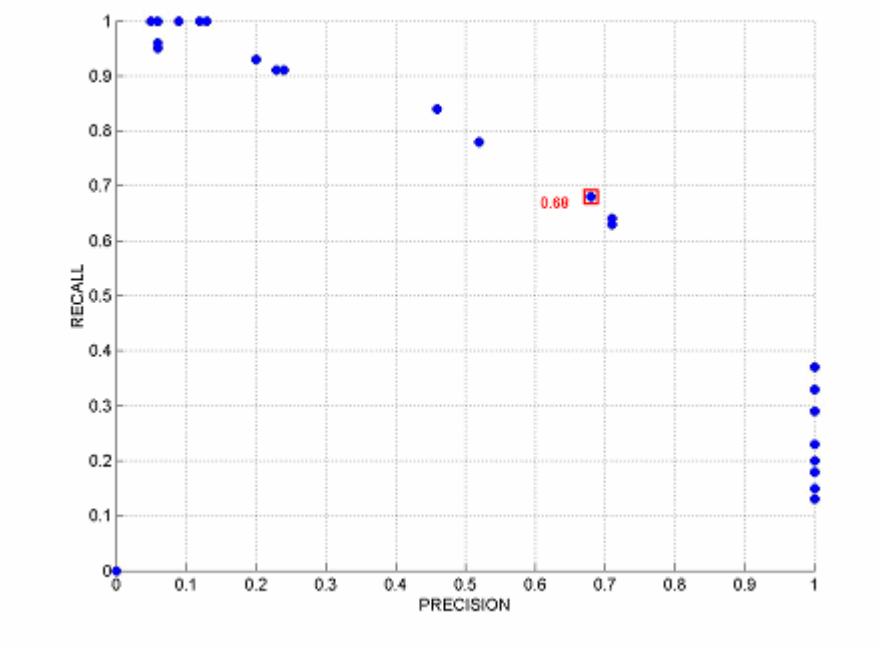


Figure A.15. Precision vs. Recall when input samples are collected from 8 speakers

(Only isolated words in search database, **morphing used (3 extra)**, no SPCM, no SPGCM)

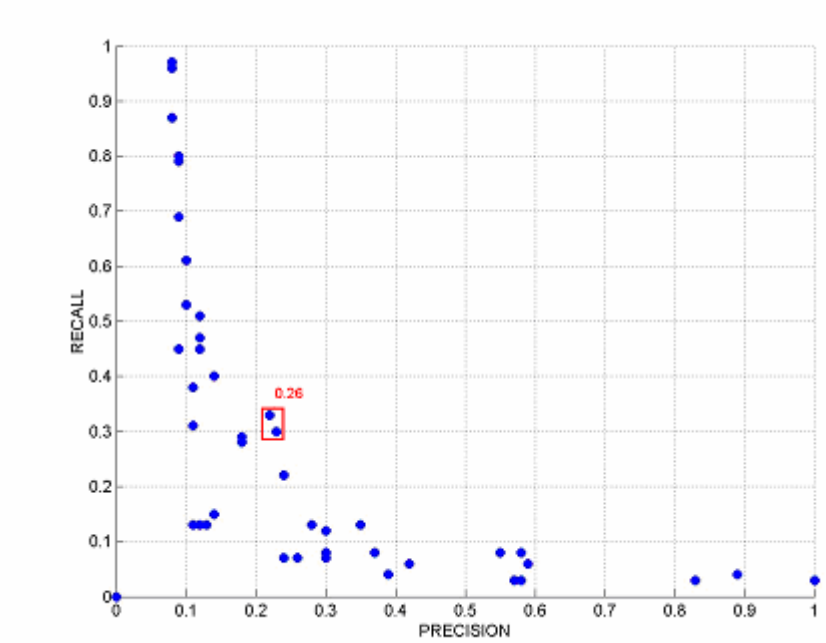


Figure A.16. Precision vs. Recall when input samples are collected from 1 speaker

(Isolated words and **sentences** in search database, no morphing, no SPCM, no SPGCM)

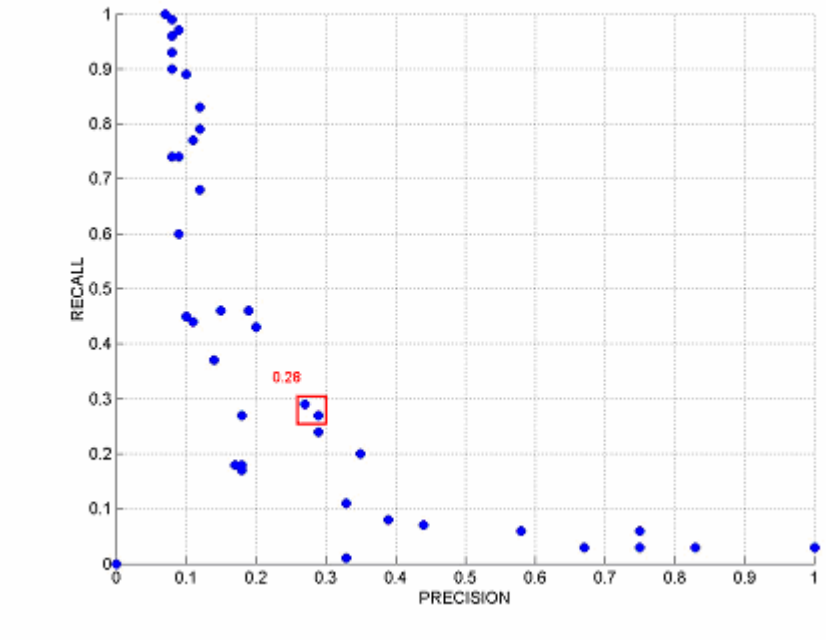


Figure A.17. Precision vs. Recall when input samples are collected from 1 speaker

(Isolated words and **sentences** in search database, **morphing used (3 extra)**, no SPCM, no SPGCM)

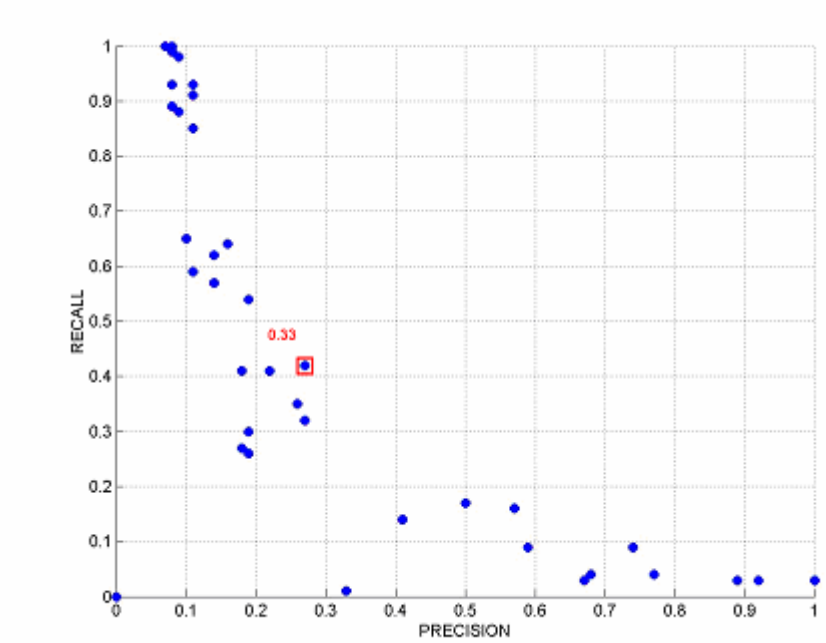


Figure A.18. Precision vs. Recall when input samples are collected from 1 speaker

(Isolated words and **sentences** in search database, **morphing used (7 extra)**, no SPCM, no SPGCM)

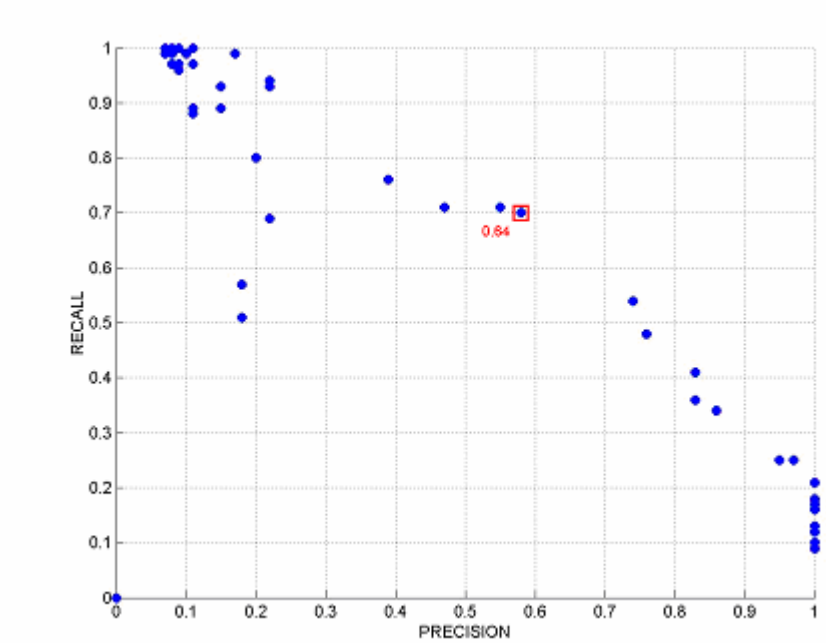


Figure A.19. Precision vs. Recall when input samples are collected from 8 speakers

(Isolated words and **sentences** in search database, no morphing, no SPCM, no SPGCM)

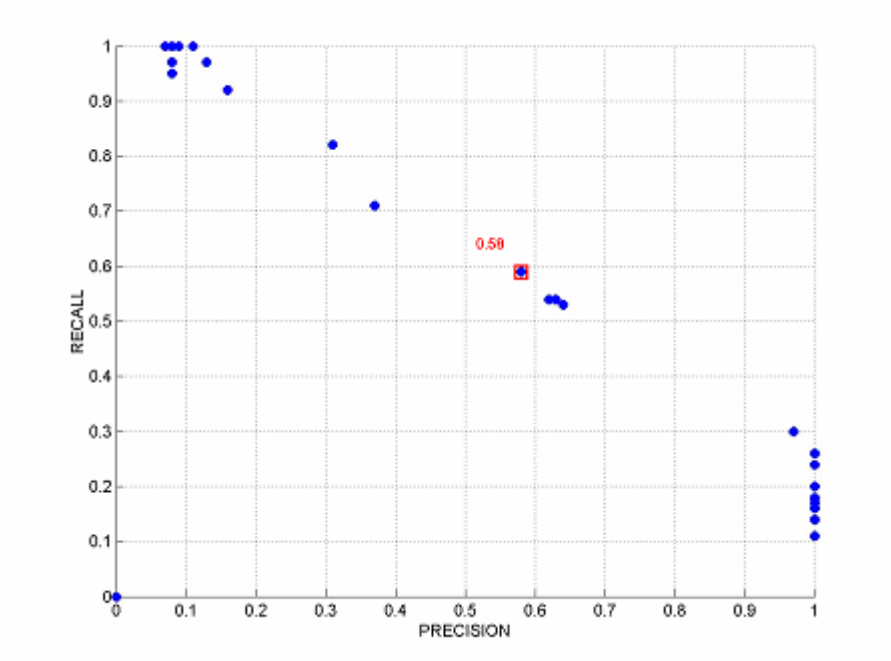


Figure A.20. Precision vs. Recall when input samples are collected from 8 speakers

(Isolated words and **sentences** in search database, **morphing used (3 extra)**, no SPCM, no SPGCM)

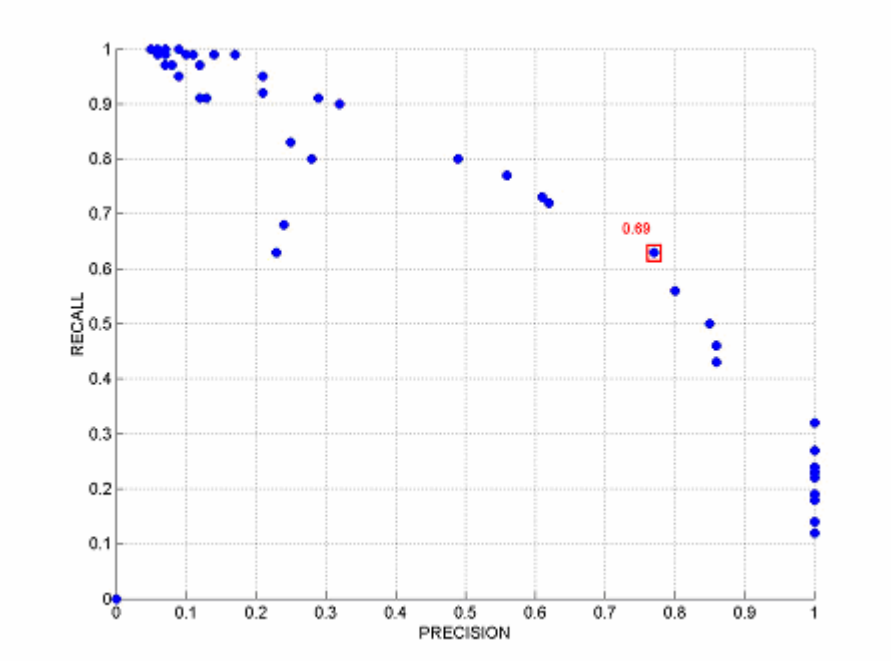


Figure A.21. Precision vs. Recall when input samples are collected from 8 speakers

(Only isolated words in search database, no morphing, no SPCM, no SPGCM)

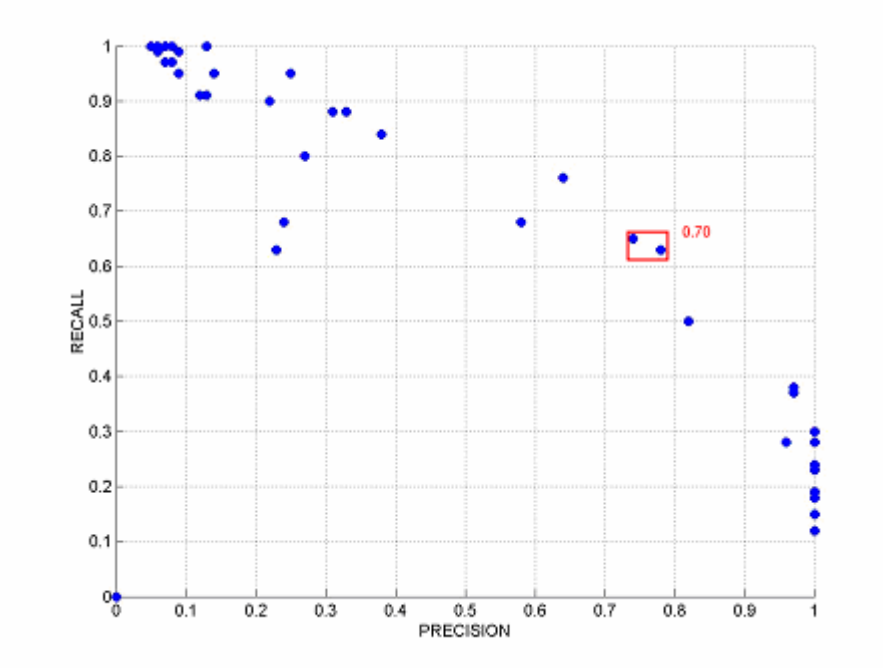


Figure A.22. Precision vs. Recall when input samples are collected from 8 speakers

(Only isolated words in search database, no morphing, no SPCM, **SPGCM is used**)

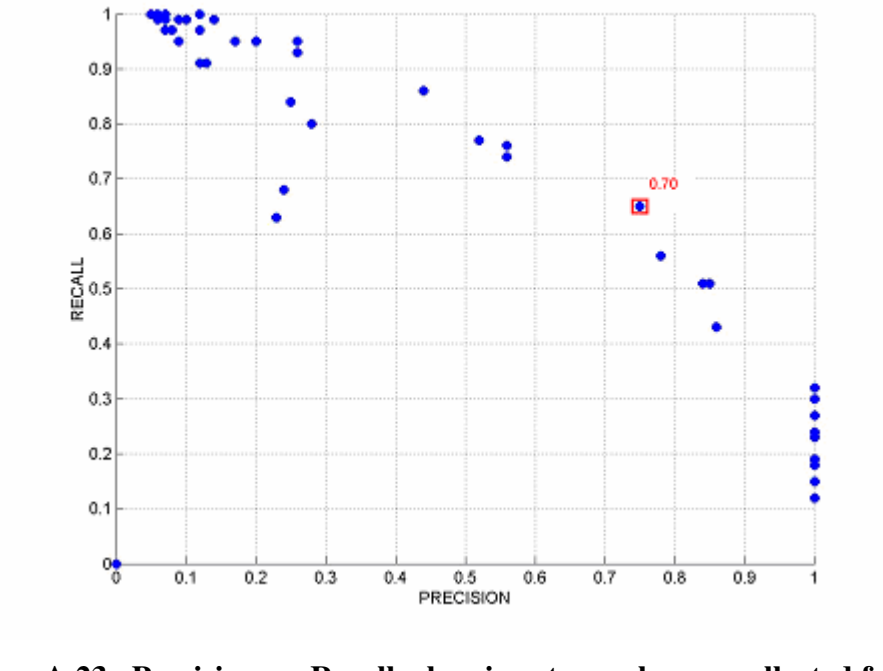


Figure A.23. Precision vs. Recall when input samples are collected from 8 speakers

(Only isolated words in search database, no morphing, **SPCM is used**, no SPGCM)

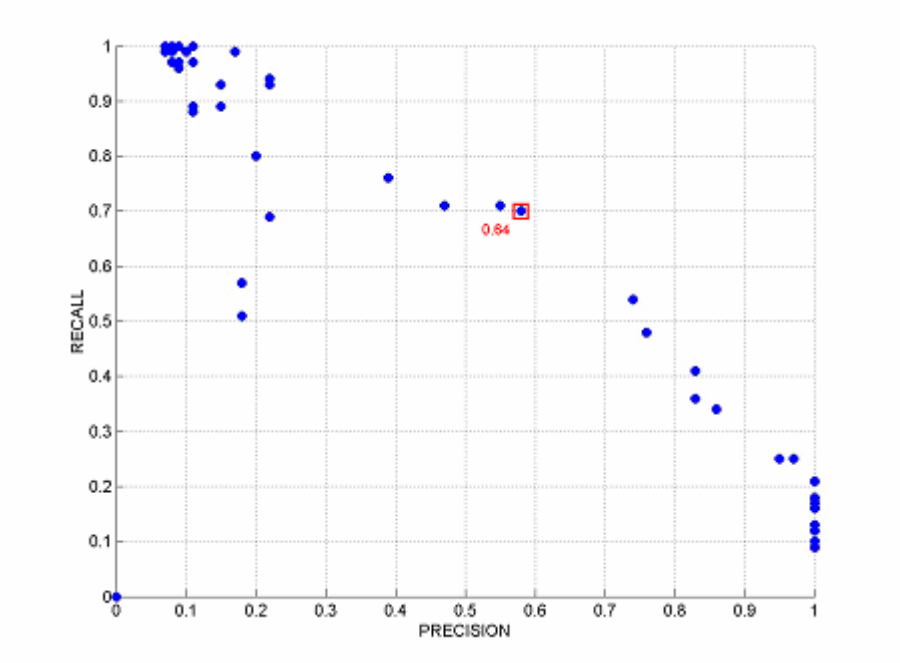


Figure A.24. Precision vs. Recall when input samples are collected from 8 speakers

(Isolated words and **sentences** in search database, no morphing, no SPCM, no SPGCM)

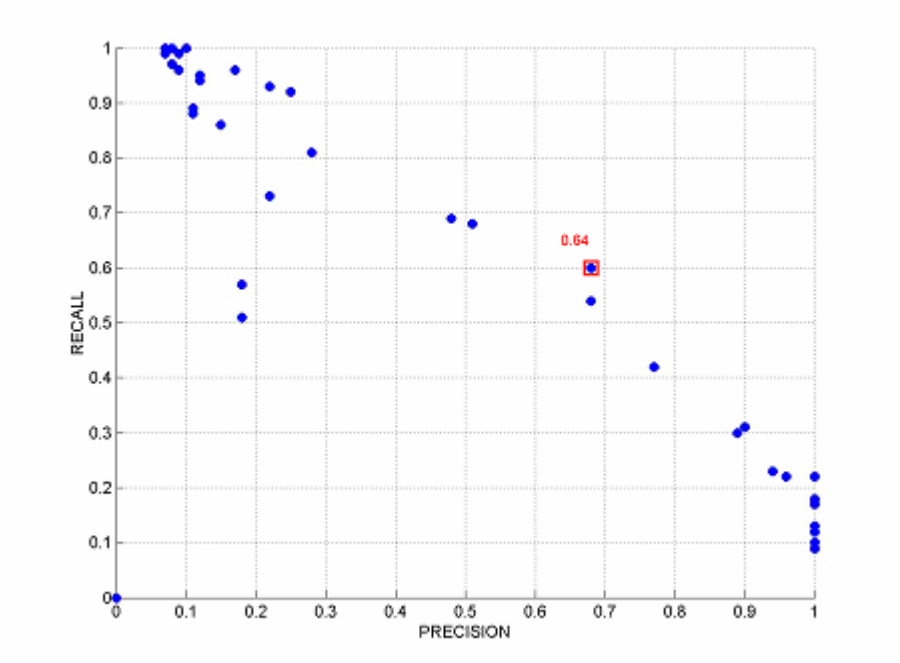


Figure A.25. Precision vs. Recall when input samples are collected from 8 speakers

(Isolated words and **sentences** in search database, no morphing, no SPCM, **SPGCM is used**)

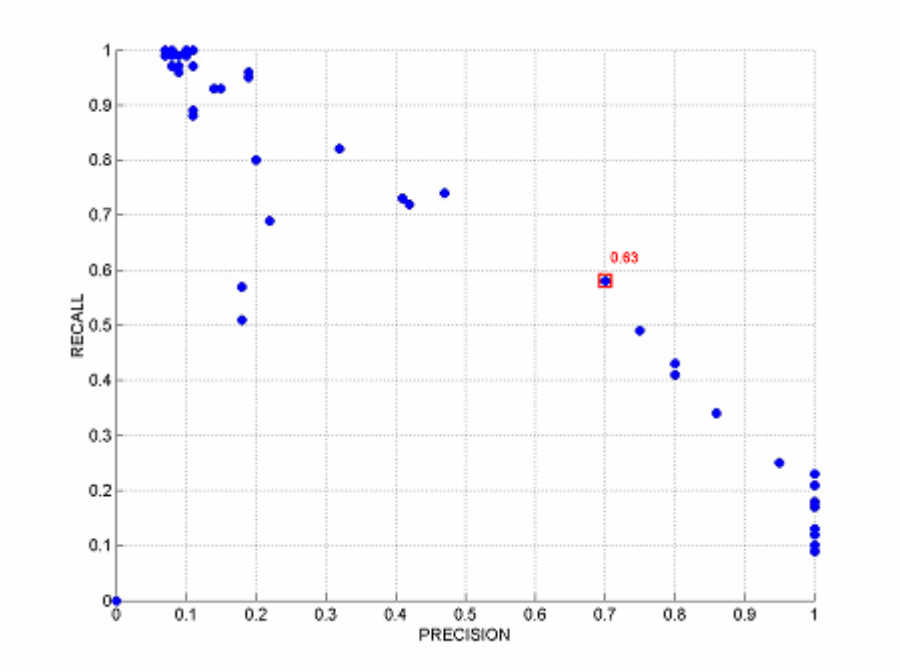


Figure A.26. Precision vs. Recall when input samples are collected from 8 speakers

(Isolated words and **sentences** in search database, no morphing, **SPCM is used**, no SPGCM)

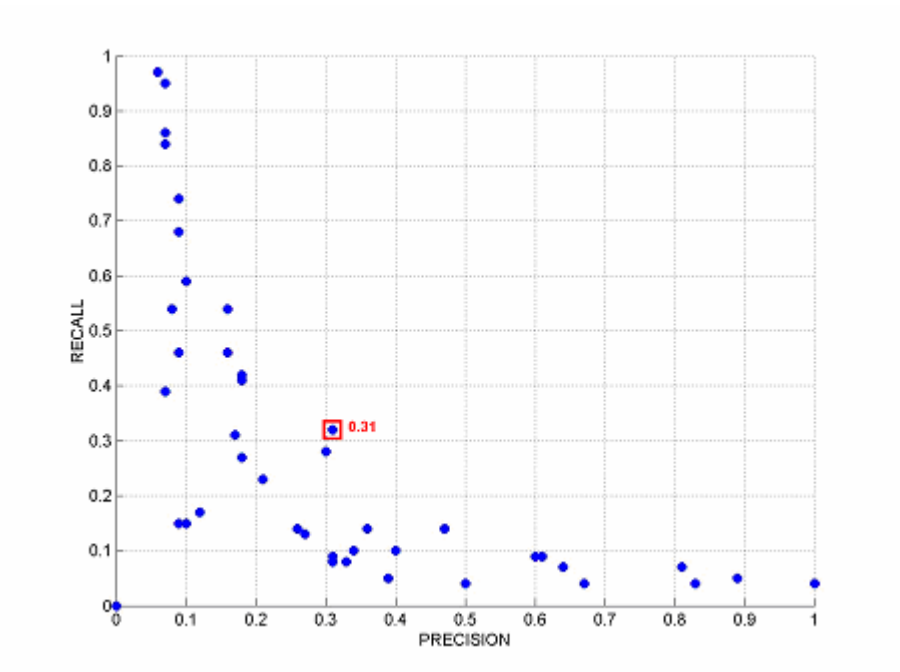


Figure A.27. Precision vs. Recall when input samples are collected from 1 speaker

(Only isolated words in search database, no morphing, no SPCM, no SPGCM)

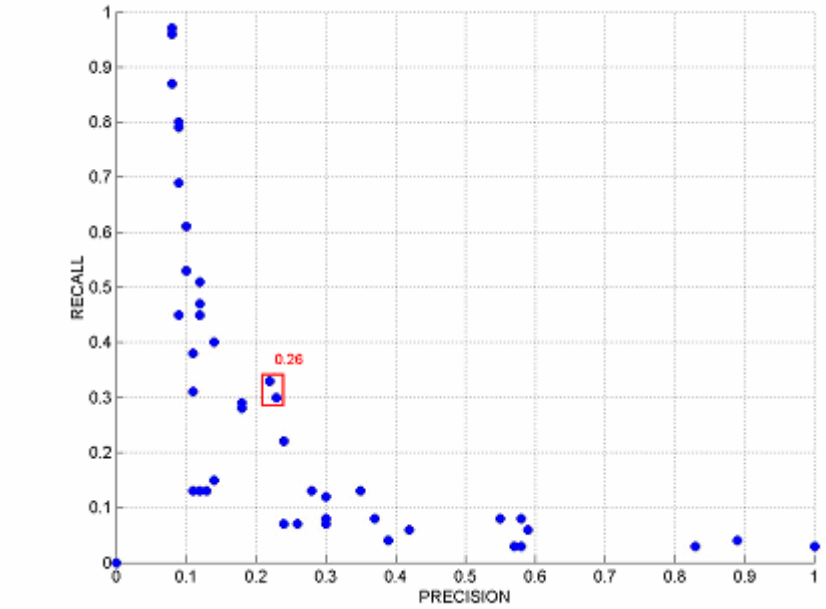


Figure A.28. Precision vs. Recall when input samples are collected from 1 speaker

(Isolated words and **sentences** in search database, no morphing, no SPCM, no SPGCM)

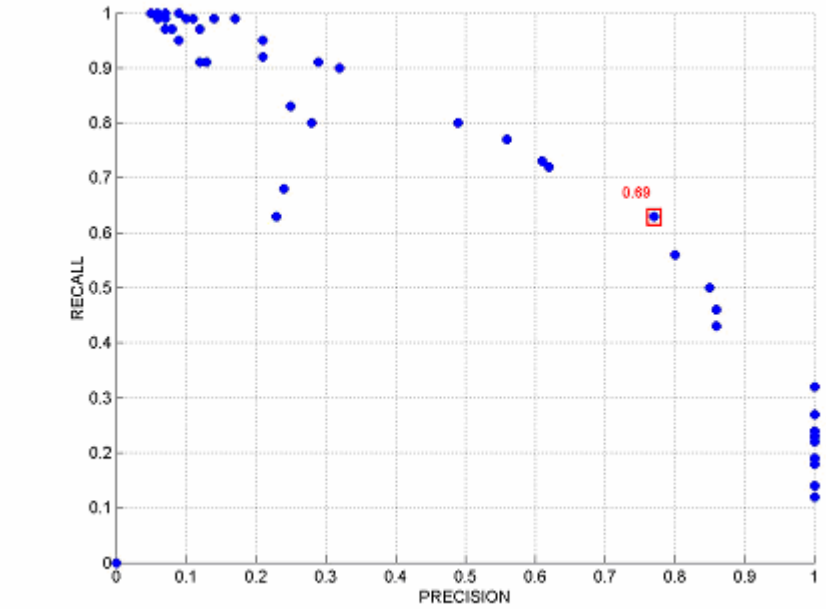


Figure A.29. Precision vs. Recall when input samples are collected from 8 speakers

(Only isolated words in search database, no morphing, no SPCM, no SPGCM)

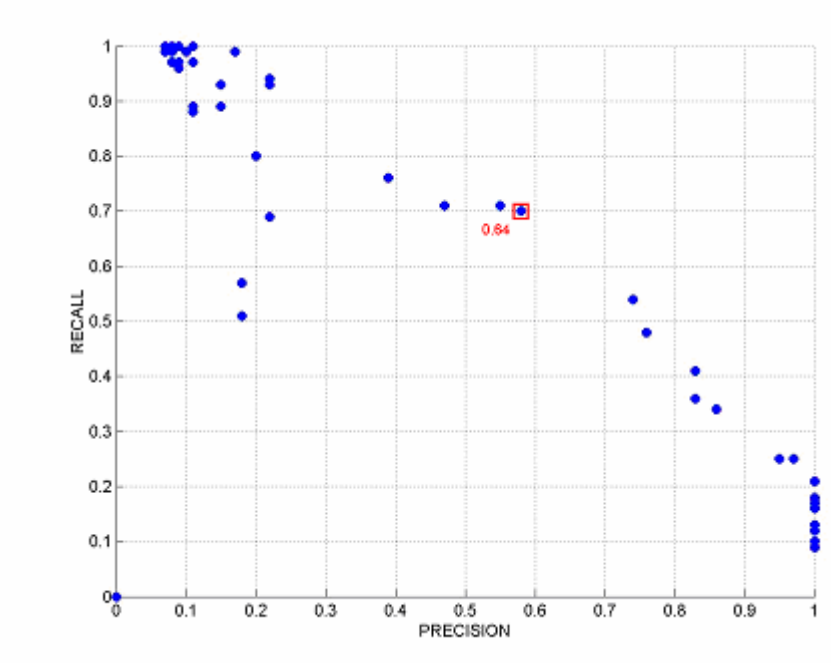


Figure A.30. Precision vs. Recall when input samples are collected from 8 speakers

(Isolated words and **sentences** in search database, no morphing, no SPCM, no SPGCM)

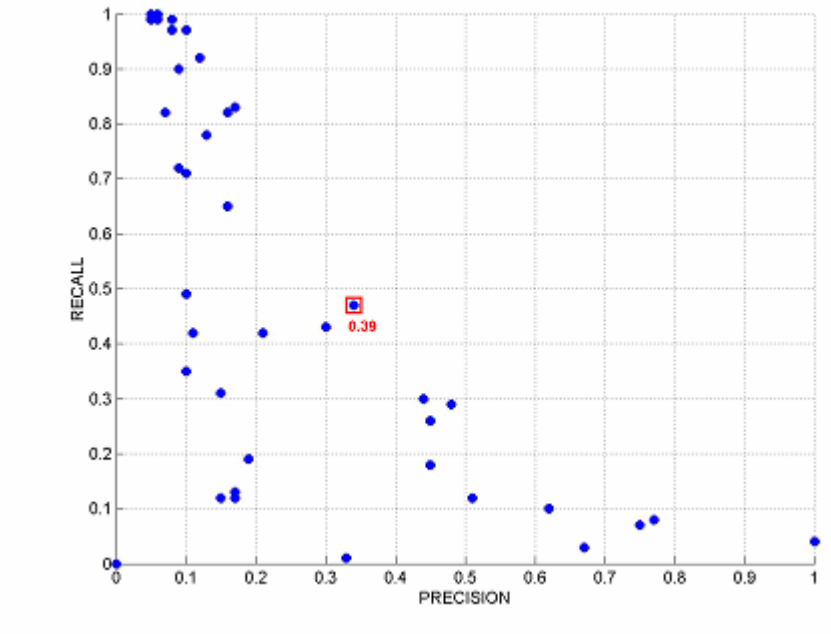


Figure A.31. Precision vs. Recall when input samples are collected from 1 speaker

(Only isolated words in search database, **morphing used (3 extra)**, no SPCM, no SPGCM)

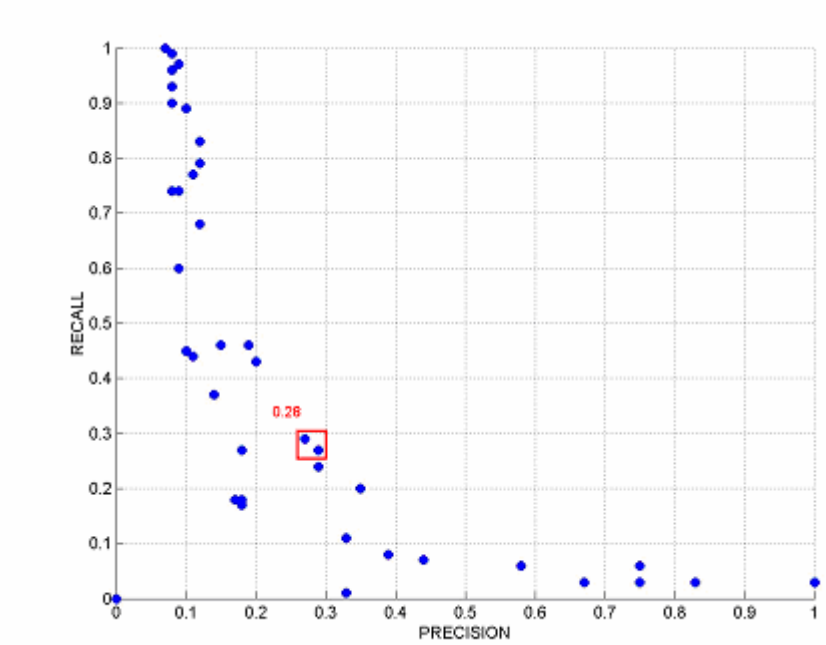


Figure A.32. Precision vs. Recall when input samples are collected from 1 speaker

(Isolated words and **sentences** in search database, **morphing used (3 extra)**, no SPCM, no SPGCM)

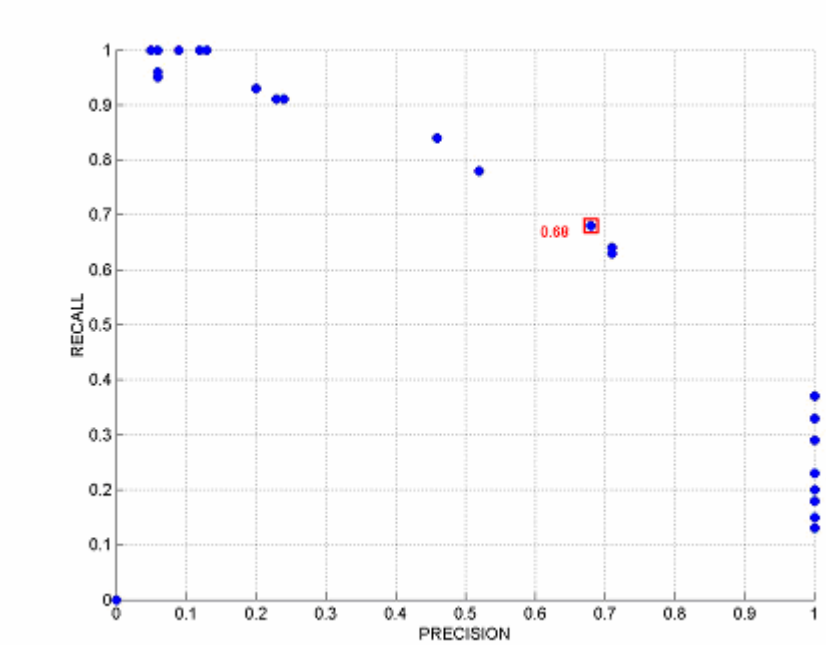


Figure A.33. Precision vs. Recall when input samples are collected from 8 speakers

(Only isolated words in search database, **morphing used (3 extra)**, no SPCM, no SPGCM)

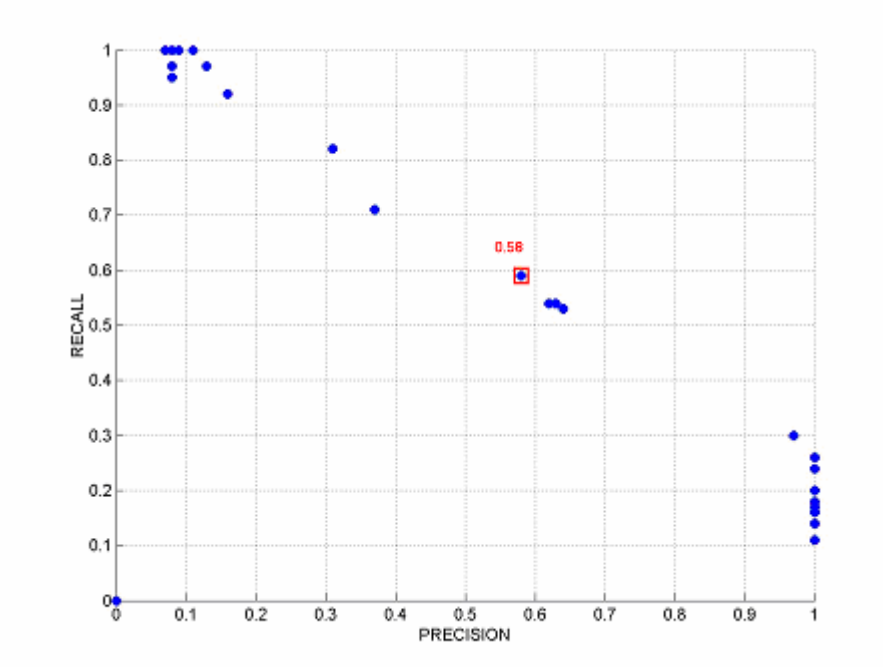


Figure A.34. Precision vs. Recall when input samples are collected from 8 speakers

(Isolated words and **sentences** in search database, **morphing used (3 extra)**, no SPCM, no SPGCM)

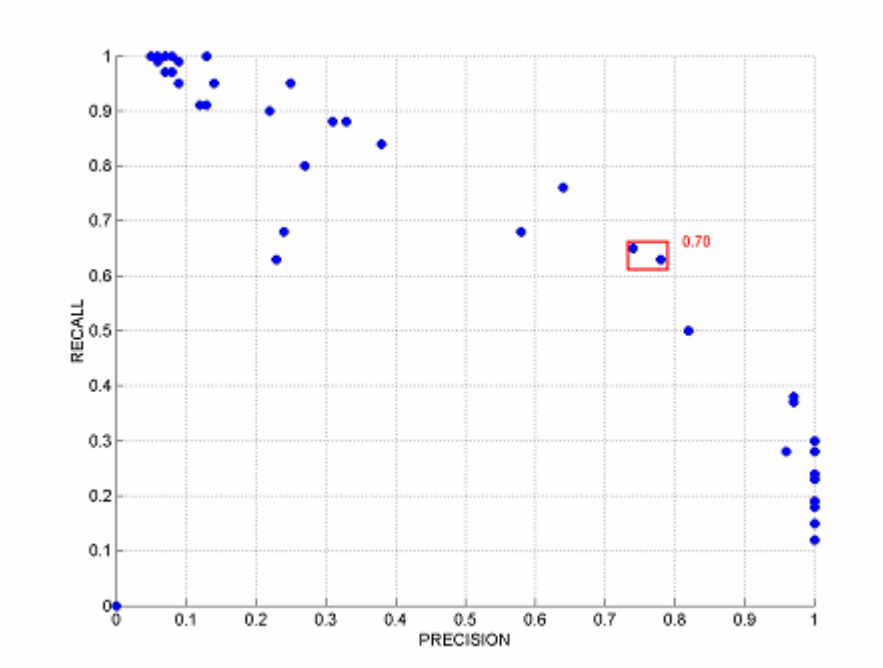


Figure A.35. Precision vs. Recall when input samples are collected from 8 speakers

(Only isolated words in search database, no morphing, no SPCM, **SPGCM is used**)

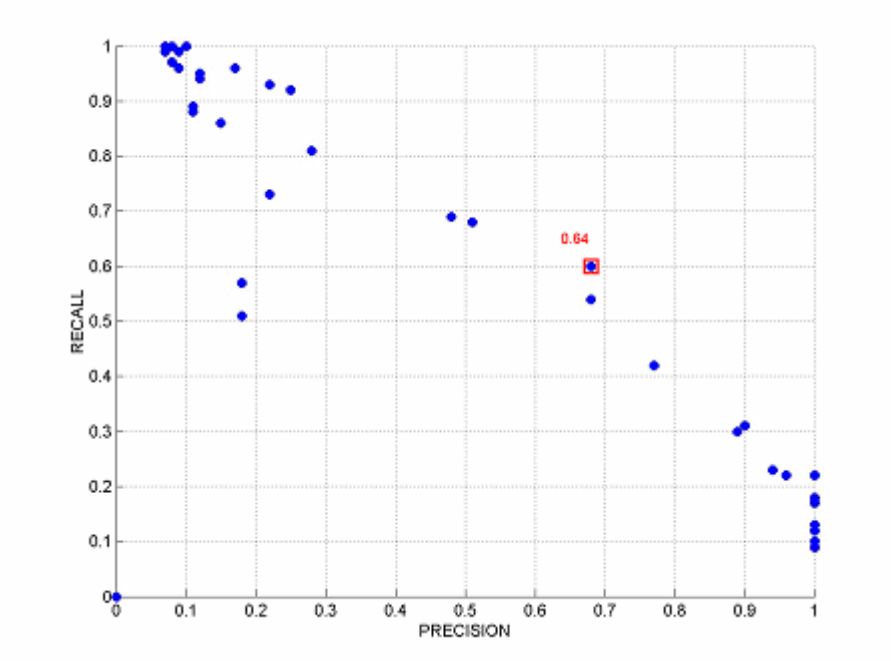


Figure A.36. Precision vs. Recall when input samples are collected from 8 speakers

(Isolated words and **sentences** in search database, no morphing, no SPCM, **SPGCM is used**)

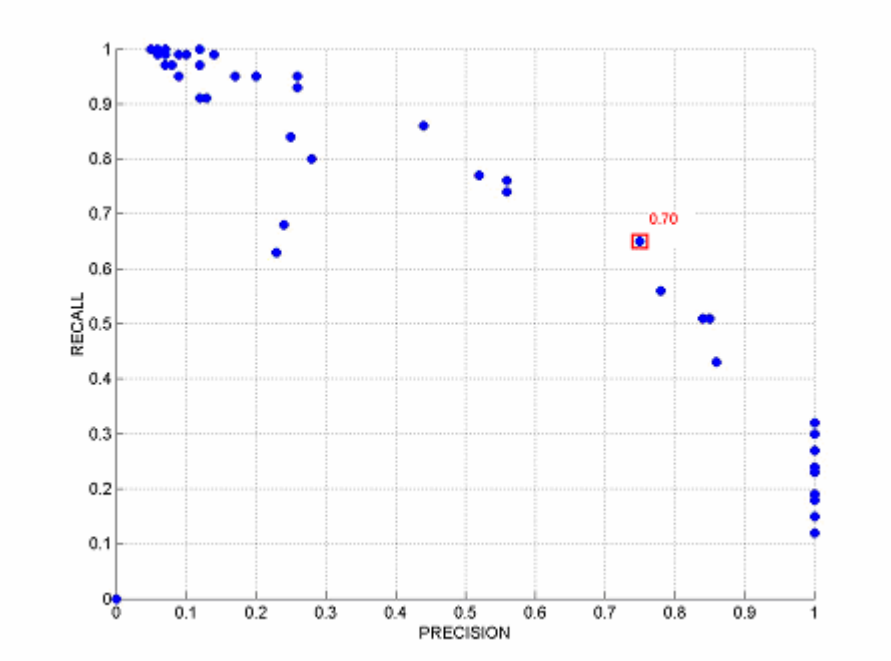


Figure A.37. Precision vs. Recall when input samples are collected from 8 speakers

(Only isolated words in search database, no morphing, **SPCM is used**, no SPGCM)

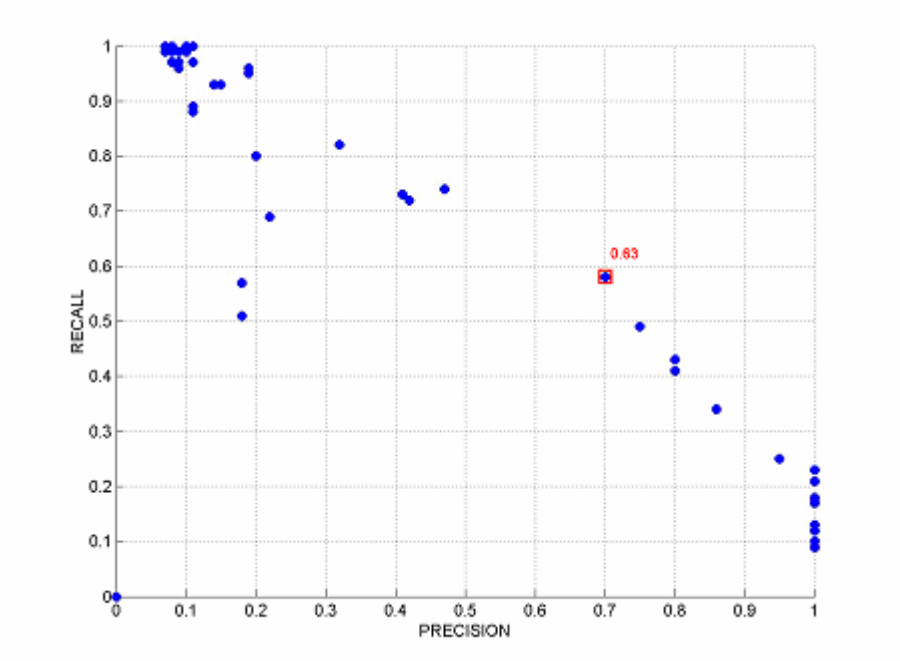


Figure A.38. Precision vs. Recall when input samples are collected from 8 speakers

(Isolated words and **sentences** in search database, no morphing, **SPCM is used**, no SPGCM)

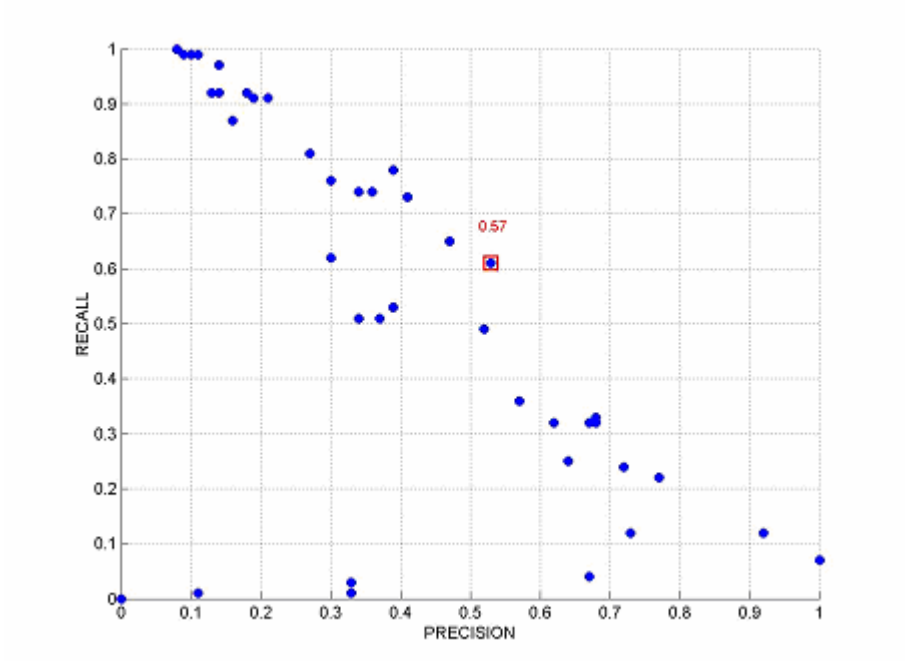


Figure A.39. Precision vs. Recall when input is text

(Only isolated words in search database, no morphing, no SPCM, no SPGCM)

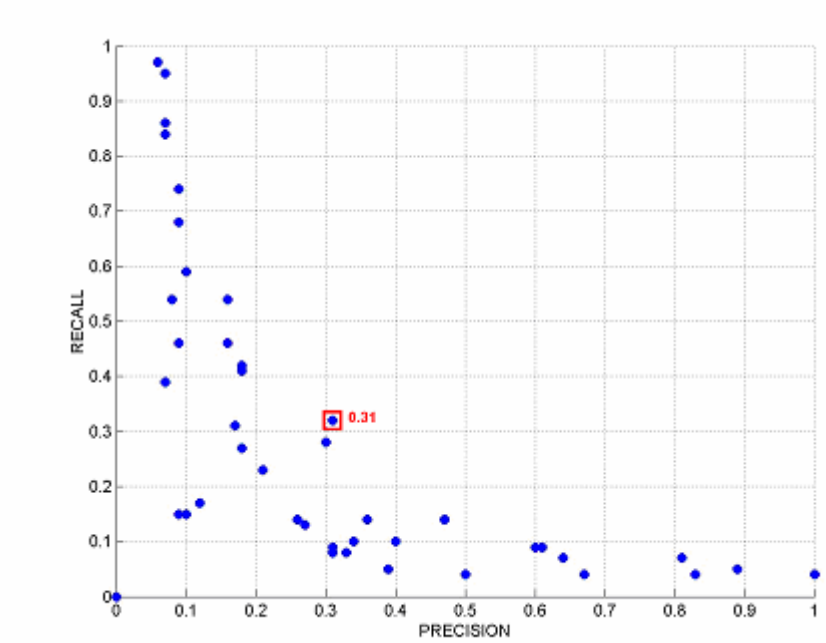


Figure A.40. Precision vs. Recall when input samples are collected from 1 speaker

(Only isolated words in search database, no morphing, no SPCM, no SPGCM)

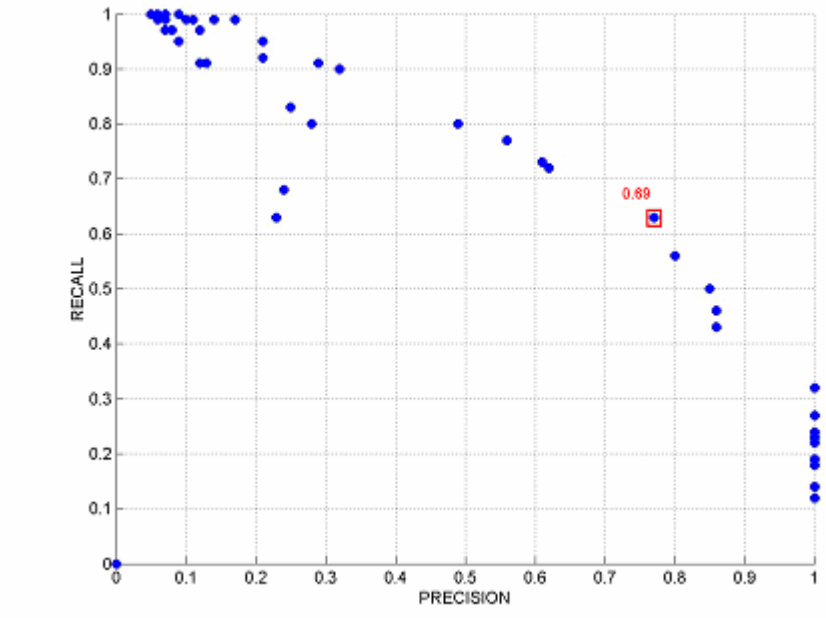


Figure A.41. Precision vs. Recall when input samples are collected from 8 speakers

(Only isolated words in search database, no morphing, no SPCM, no SPGCM)

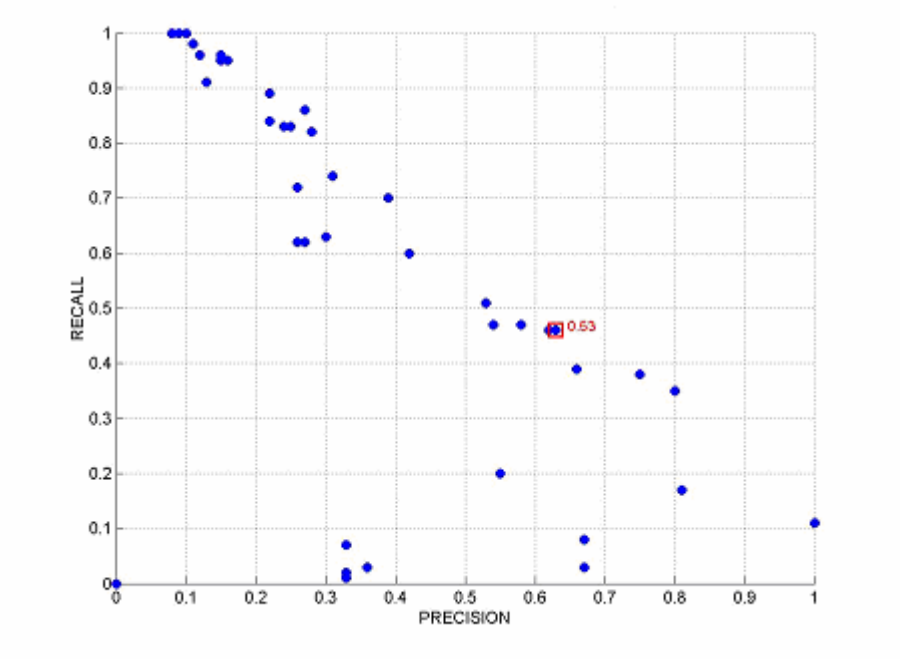


Figure A.42. Precision vs. Recall when input is text

(Isolated words and **sentences** in search database, no morphing, no SPCM, no SPGCM)

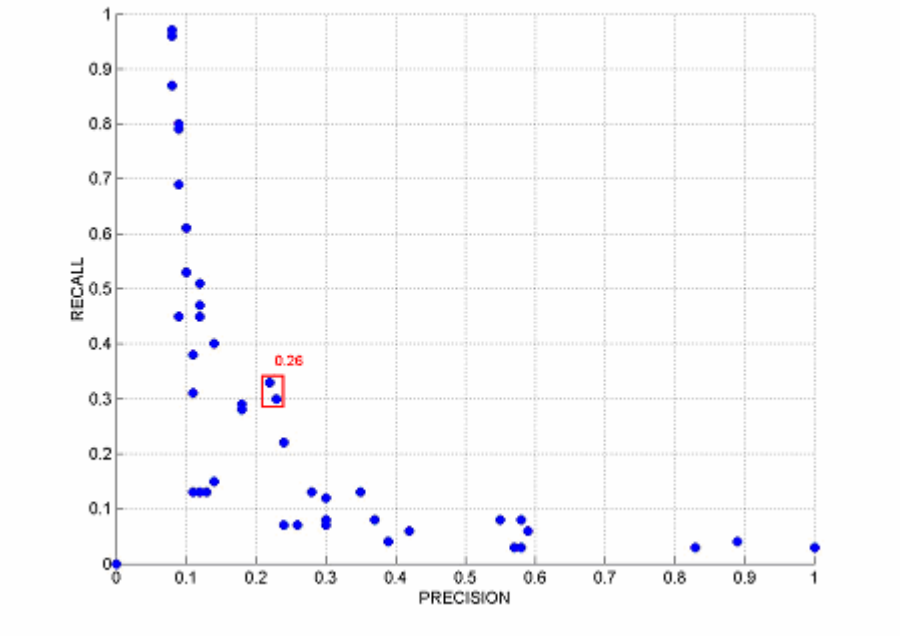


Figure A.43. Precision vs. Recall when input samples are collected from 1 speaker

(Isolated words and **sentences** in search database, no morphing, no SPCM, no SPGCM)

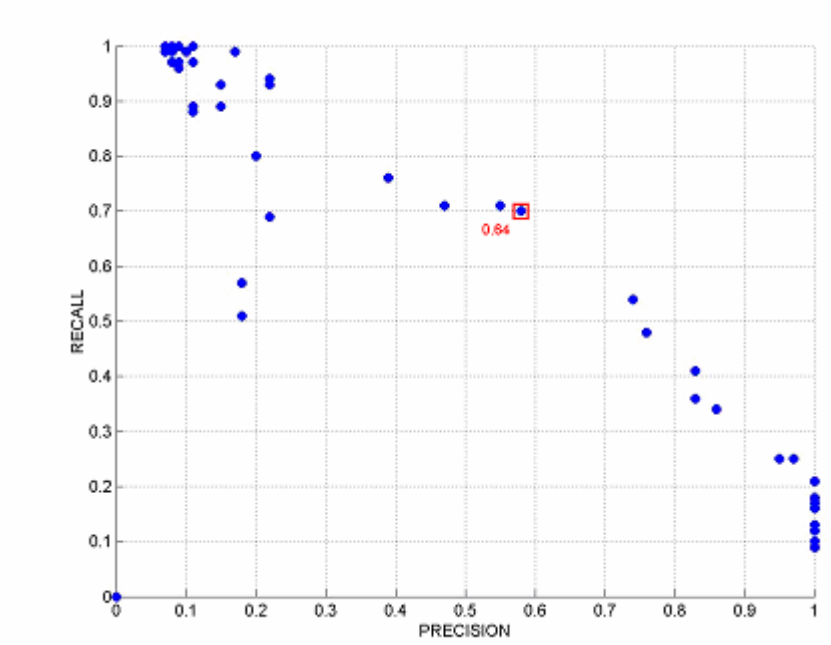


Figure A.44. Precision vs. Recall when input samples are collected from 8 speakers

(Isolated words and **sentences** in search database, no morphing, no SPCM, no SPGCM)

APPENDIX B: CD CONTAINING COMPUTER SOFTWARE, SAMPLE INPUTS AND OUTPUTS

A compact disk, containing the source code for Voice Driven Keyword Spotter application and sample inputs and outputs of the application, is included.

REFERENCES

1. Alon G., Key-Word Spotting – The Base Technology for Speech Analytics, NSC Natural Speech Communication Ltd., 2005, http://www.crmxchange.com/whitepapers/pdf/NSC-kws.pdf

2. Chang E. I., Lippmann R. P., Improving Word Spotting Performance with Artificially Generated Data, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) Proceedings, 1996, v. 1, pp. 526-529

3. Nishizaki H., Nakagawa S., Comparing Isolately Spoken Keywords with Spontaneously Spoken Queries for Japanese Spoken Document Retrieval, International Conference on Spoken Language Processing (ICSLP) Proceedings, 2002, pp. 1505-1508

4. Kaiser J., Word Spotting in the Telephone Dialogue Systems, AST 97, http://wwwbox.uni-mb.si/Dsplab/Clanki/janez/ast97_1.pdf

5. Saraclar M., Sproat R., Lattice Based Search for Spoken Utterance Retrieval, Proceedings of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics Annual Meeting (HLT-NAACL), Boston, MA, USA, 2004, pp. 129-136

6. Junkawitsch J., Neubauer L., Höge H., Ruske G., A New Keyword Spotting Algorithm With Pre-Calculated Optimal Thresholds, International Conference on Spoken Language Processing (ICSLP) Proceedings, v. 4, 1996, pp. 2067-2070

7. Zheng T.F., Li J., Song Z., Xu M., A Two Step Keyword Spotting Method Based On Context-Dependent A Posteriori Probability, International Symposium on Chinese Spoken Language Processing Proceedings, 2004, pp. 281-284

8. Tsiporkova E., Vanpoucke F., Van Hamme H., Evaluation of Various Confidence Based Strategies for Isolated Word Rejection, International Conference on Acoustics, Speech and Signal Processing (ICASSP) Proceedings, Istanbul, Turkey, 2000, pp. 1819-1822

9. Christen P., Single Error Analysis of String Comparison Methods, http://150.203.33.60/publications/2005/pakdd2005-stringcomp.pdf

10. Ying Z., Huang Q., Li H., String Distance, 2003

11. Zobel J., Dart P., Finding Approximate Matches in Large Lexicons, Software - Practice and Experience, v. 25, n. 3, 1995, pp. 331-345

12. Giegerich R., Sequence Similarity and Dynamic Programming, Lecture Notes, Bioinformatics Summerschool, Bad Urach, 2002

13. Zobel J., Dart P., Phonetic String Matching: Lessons from Information Retrieval, Proceedings of the 19th annual international ACM SIGIR conference on Research and Development in Information Retrieval, Zurich, Switzerland, 1996, pp. 166-172

14. Duran S., Keyword Spotting Using Hidden Markov Models, MSc. Thesis, Boğaziçi University, 2001

15. Yapanel Ü., Garbage Modeling Techniques for a Turkish Keyword Spotting System, MSc. Thesis, Boğaziçi University, 2000

16. Knill K.M., Young S.J., Speaker Dependent Keyword Spotting for Accessing Stored Speech. Technical Report CUED/F-INFENG/TR 193, Cambridge, U.K. Cambridge University, 1994, http://citeseer.ist.psu.edu/knill94speaker.html

17. Silaghi M., Boulard H., A New Keyword Spotting Approach Based On Iterative Dynamic Programming, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) Proceedings, Istanbul, 2000, v. 3, pp. 1831-1834