PROSODIC, MORPHOLOGICAL AND LEXICAL FEATURE EXTRACTION OF TURKISH BROADCAST NEWS DATA

İZEL D.REVİDİ

IŞIK UNIVERSITY
2014

PROSODIC, MORPHOLOGICAL AND LEXICAL FEATURE EXTRACTION OF TURKISH BROADCAST NEWS DATA

İZEL D.REVİDİ

B.S., Electrical and Electronics Engineering, Işık University, 2012

Submitted to the Graduate School of Işık University
In partial fulfillment of the requirements for the degree of
Master of Science
In
Electronics Engineering

IŞIK UNIVERSITY
2014

IŞIK UNIVERSITY

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

PROSODIC, MORPHOLOGICAL AND LEXICAL FEATURE EXTRACTION OF TURKISH BROADCAST NEWS DATA

İZEL D.REVİDİ

APPROVED BY:

Assoc. Prof. Ümit Güz Thesis Supervisor	Işık University	_____
Prof. Dr. Sabri Arık	Işık University	_____
Prof. Dr. Serdar Özoğuz	Istanbul Technical University	_____

APPROVAL DATE: 05.06.2014

PROSODIC, MORPHOLOGICAL AND LEXICAL FEATURE EXTRACTION OF TURKISH BROADCAST NEWS DATA

Abstract

Sentence segmentation from speech is part of a process that aims at enriching the unstructured stream of words that are the output of standard speech recognizers. Its role is to find the sentence units in this stream of words. Sentence segmentation is a preliminary step toward speech understanding. Once the sentence boundaries are detected, further syntactic and/or semantic analysis can be performed on these sentences.

Usually, speech recognizer output lacks the textual cues to these entities (such as headers, paragraphs, sentence punctuation, and capitalization). However, speech provides extra non-lexical cues, related to features like pitch, energy, pause and word durations as prosodic features; verb, noun or adjective as a morphological features and also lexical features. These prosodic, morphological and lexical features are provides a complementary information for segmentation of speech into sentences.

Our goal is examine feature the extraction and use of prosodic information which has been done in previous works, in addition to lexical features and morphological for spoken language processing of Turkish with open source tools.

TÜRKÇE HABER VERİSİNDEN BÜRÜNSEL, BİÇİMSEL VE SÖZCÜKSEL ÖZELLİKLERİN ÇIKARIMI

Özet

Cümle bölütlemesi otomatik konuşma tanıma sisteminden çıkan sözcüklerin içeriğini zenginleştirmeyi hedefleyen sürecin bir parçasıdır. Cümle bölütlemesi, gelen kelime akışının bütün bir cümle olarak tanımlanması görevini üstlenir ve konuşma anlamının çıkarılması sürecinin bir önceki aşamasını oluşturur. Cümle sınırlarının bulunması ile birlikte cümle üzerinde sözdizimi ve/veya anlamsal analiz yapılabilmektedir.

Genellikle otomatik konuşma tanıma sisteminden alınan çıktılarda başlık, paragraf, noktalama, büyük/küçük harf gibi bilgileri içeren metin işaretleri yer almamaktadır. Ancak konuşma hali hazırda enerji, duraklama bilgisi, kelimenin geçiş süresi gibi bürünsel özellikleri; kelimenin yüklem, isim veya sıfat olması gibi biçimsel özellikleri ve sözcüksel özellikleri barındırmaktadır. Bu bürünsel, biçimsel ve sözcüksel özellikler cümle bölütlemesinin yapılabilmesi için tamamlayıcı bir bilgi sağlamaktadır.

Yapılan çalışmadaki amacımız daha önceki çalışmalarda yapılmış bürünsel özelliklerin çıkarımı ve kullanımına ek olarak; biçimsel ve sözcüksel özellikler açık kaynak kodlu araçlar ile Türkçe Konuşma Dili üzerinde çıkarımı ve kullanımıdır.

Acknowledgments

There are many people who helped to make my years at the graduate school most valuable. First, I thank my supervisor, Assoc. Prof. Ümit Güz and my co-supervisor, Assoc. Prof. Hakan Gürkan. Having the opportunity to work with them in such project was intellectually rewarding and fulfilling. My special thanks go to Research Asst. Doğan Dalva whose friendship I deeply value.

Many thanks to members of Senkron Security who give me support for my education while I was working at the same time. I also thank my beloved Alara Deşilton for her endless support through this long journey.

The last words of thanks go to my family. I thank my parents, my mother Suzan Revidi and my father İsak Revidi, for their patience and encouragement. Lastly I thank all my friends for their moral support.

To my family

Table of Contents

List of Tables

List of Figures

List of Symbols

ASR	Automatic Speech Recognizer
DSP	Digital Signal Processing
WAV	Waveform Audio File Format
AIFF	Audio Interchange File Format
HMM	Hidden Markov Model
HTK	Hidden Markov Model Toolkit
IPA	International Phonetic Alphabet
ARPA	Advanced Research Project Agency
LM	Language Model
LTM	Lognormal Tied Mixture Model
MFCC	Mel Frequency Cepstral Coefficients
FST	Finite State Transducer
SFST	Stuttgart Finite State Transducer
VOA	Voice of America
STM	Segment Time Marks

Chapter 1

Introduction

Speech, writing and sign are the only three main acts to communicate humans between them. And speech is the easy way for the communication in case two people speak with same language. By the development of technology new communication types are occurred such as e-mail, video calls etc.

First communication start in 3500 BC with the paintings on the walls of the caves and it continued to develop rapidly until the internet-WWW is born in 1994. Invention of phonautograph which is the earliest known device for recording and printing waveform of the sounds in to a paper, could be accepted the first step of speech processing. This device invented by Edouard-Leon Scott de Martinville in 1857. Today speech signals could represent with an electrical signal by using sources such as a microphone. These signals could be processed in any basic computer or electronic device. Also these signals could reconstruct and transmit easily with digital signal processing (DSP) methods.

In this work we concentrate on Turkish Spoken Language and convert ASR system's output which system is used for convert speech signal to a simple text file with recognized words, into a meaningful data. It is must for the human and machine communication. A data without sentence boundary labels doesn't make sense for the human. Using by prosodic, morphological and lexical features; we aimed to label sentence boundaries automatically. Sentence segmentation is need for such as topic segmentation, topic summarization, parsing, machine translation, information extraction, online subtitling and question answering applications.

The goal of the sentence segmentation is made a decision for the each word, is it a sentence boundary or not, with an acceptable error. Prosodic features include timing

and pitch patterns; morphological features include what and how information encoded in to a word; and lexical features include order information of the words. All these information is used for giving decision correctly.

Thesis starts with the information about related works which is done previous works. In Chapter 3, definition of ASR, modified ASR into Turkish Spoken Language, Modeling (Hidden Markov Model, Word Model, Acoustic Model and Language Model) and usage of HTK Toolkit has introduced. In Chapter 4, definition of Prosodic Feature, types of Prosodic Feature, how to extract Prosodic Features and usage of Praat Toolkit has introduced. In Chapter 5, definition of Morphological Features, morphological process, how to extract Morphological Features and usage of TRmorph Toolkit hast introduced. In Chapter 6, definition of Lexical Features, modeling and usage of N-gram models has introduced. In Chapter 7, introduction to sentence segmentation problem, approaches, usage of Icsiboost has introduced. And lastly Chapter 8 includes overview of experiments, experiments and conclusion has introduced.

Chapter 2

Related Works

ASR systems are very trend topics; there are lot of researches and publications with several ways on this subject. Although ASR systems work for conversion of speech in to a text file, in our work we focused on Sentence Segmentation which tries to conversion of a text file into meaningful state, topic from a lot of subject which related with ASR system. Again Sentence Segmentation problem is solved with several ways for several different situations. Different languages, different feature sets and different learning algorithms separate all these researches in their self. Thus we can group these researches into type of language, type of features, feature extraction methods and type of machine learning algorithms.

Sentence boundary detection (and similarly adding punctuation mark) in speech has been studied in an attempt to enrich speech recognition output [1, 2, 3, 4] and in the previous approaches for this task, different classifiers have been evaluated (e.g. hidden Markov model (HMM), maximum entropy), utilizing both textual-prosodic information [5]. In example, different approaches such as HMM, maximum entropy and conditional random fields are applied in same research for both conversational telephone speech and broadcast news speech [4]. In ‘’The ICSI+ multi-lingual sentence segmentation system" reasearch is based on Mandarin and English Spoken Language (Multi-Language System) but also there is applications for the other different languages such as Czech, Chineese etc.

In past, feature extraction systems have depended mostly on lexical information for segmentation (Kubala et al., 1998; Allan et al.1998; Hearst, 1997; Kozima, 1993; Yamron et al., 1998; among others) [1]. In “Prosody-based automatic segmentation of speech into sentences and topics” research against to past

automatic information extraction systems lexical and prosodic features are used together based on English Spoken Language. Sentence segmentation decision is given using decision tree and Hidden Markov Modeling techniques where prosodic cues with word-based approaches are combined. Performance is evaluated on two speeches which are broadcast news and switchboard, result of this it is shown that prosodic model is performed better than the other word-based statistical language models for both two speeches.

Differently in “Automatic Speech Recognition System for Turkish Spoken Language” [9] research only Prosodic Features are used for deteceting sentence boundaries in Turkish broadcast news data. HTK tool is used for application of ASR and feature extaction is done with Praat Toolbox [10]. This research shows that related to tool’s feature outputs; when F_0 features, duration features and energy features are used together, system performs the maximum performance. Again in differently, adaboost algorithm[12] is applied for improve the scores and to get a strong learning algorithm.

In related works as we see mostly lexical and prosodic features are used and researches are mostly based on English Spoken Language. But against to others in ‘’A Freely Available Morphological Analyzer for Turkish” [7] and "A statistical information extraction system for Turkish," [8] researches morphological features are used to detect sentence boundaries and both researches are based on Turkish Spoken Language however in English Spoken Language there is very small number of possible word forms with a given word if it is compared with Turkish Spoken Language. To conclude it is seen that because of Turkish is a agglutinative type of language, the construction of a language model for Turkish Spoken Language can’t be directly adapted from English Spoken Language. At the other side Çağrı Çöltekin’s tool against to Kemal Oflazer’s tool created before, is the first freely available two-level morphological analyzer for Turkish Spoken Language.

Above all of these in our work we try to combine all these different approaches together. As a solution to sentence segmentation problem which is detected to sentence boundaries; prosodic, morphological and lexical features are used together. All features are extracted with mathematically modeled language models by helping related tools. For reach the highest scores and to get a strong learning algorithm, boosting method is used. Lastly our system is based on Turkish Spoken Language and also we try to use open source and easy to modified for all languages tools.

Chapter 3

Automatic Speech Recognition

3.1 Definition

Automatic Speech Recognition (ASR) can be defined as conversion of spoken word in to a text file in basically. This system that allows, machine is identified words which comes out from the speaker in to a source device (microphone etc.) and transform it to a digital text file. Thus provides communication between human and machine and the most important benefit is cost reduction. In present day's technology, machines take place of the human and it obliges machine-human relationship. ASR system requires limited time period of speaker training after that system is captured words from a large vocabulary with high accuracy. ASR system researches has drove for more than 50 years, the goal of ASR is, recognizing doing with hundred percent accuracy, for all words, with independent speakers, with unknown vocabulary size, with noise and with all type of languages. In our work we have focused ASR system for Turkish Spoken Language.

3.2 Introduction

We examine speech recognition in two sides. One of them is human side and the other of them is machine side. 2 sides are summarized in Figure 3.1.

At the human side sound waves are produced by vibration by helping articulation. Than ears conveys this vibrations in to the brain. Last step brain is processed vibration and recognition is completed as shown in Figure 3.2.

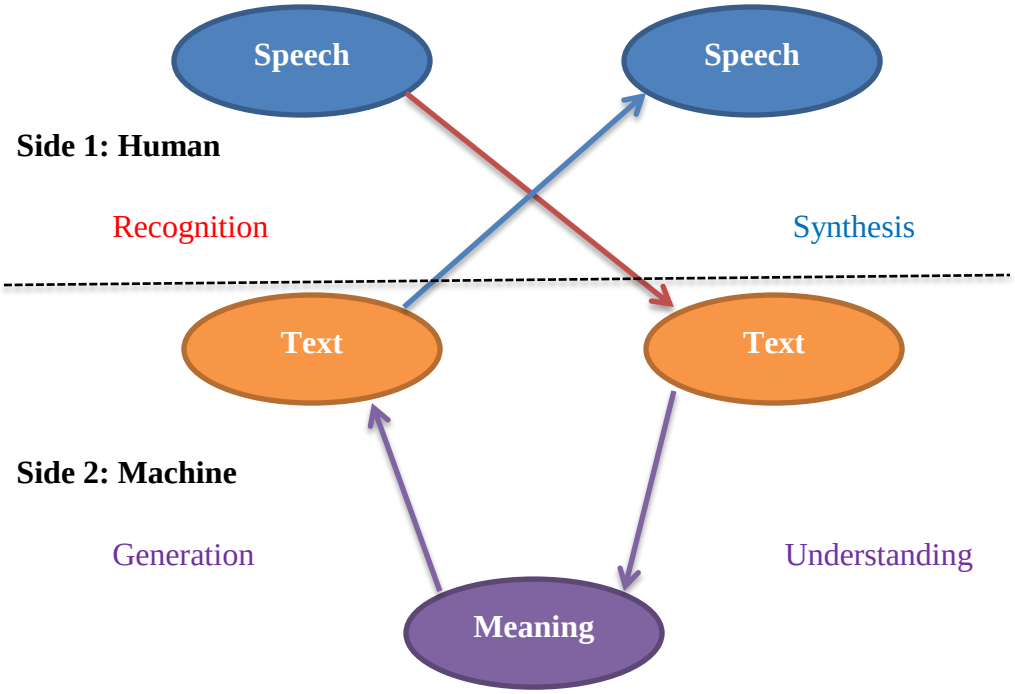


Figure 3.1 Two side communication

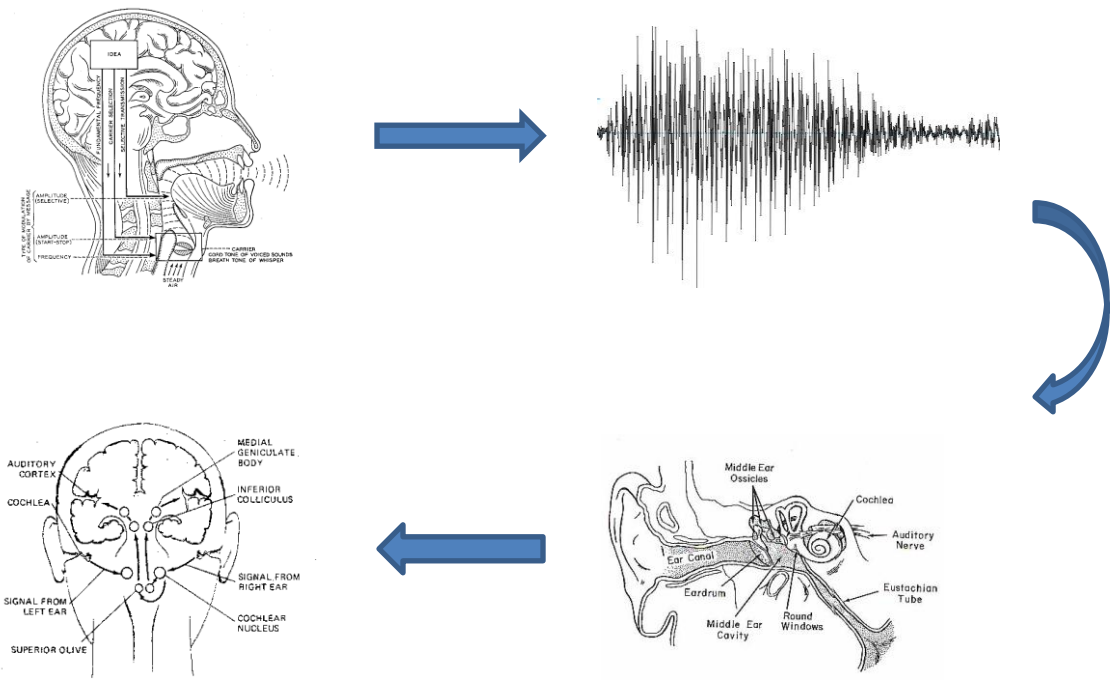


Figure 3.2 Speech Processing (Human)

At the other side machine take the sound waves from outside sources (microphone, wav files etc.) than machine digitize the input files and try to estimate words by linguistic interpretation as shown in Figure 3.3.

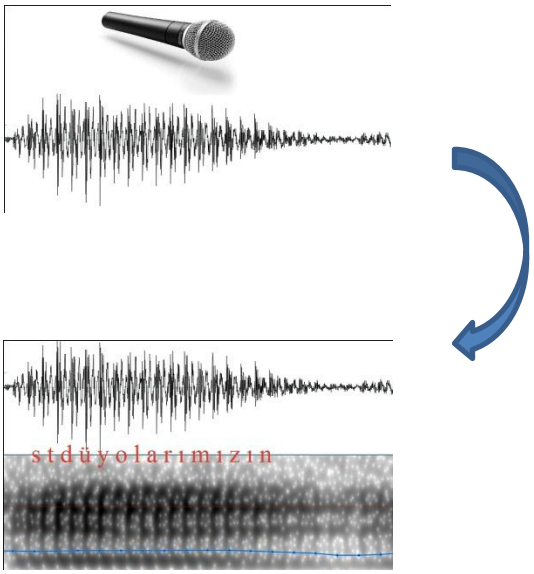


Figure 3.3 Speech Processing (Machine)

3.3 ASR With Turkish Spoken Language

As we mention before type of language is important detail for ASR system. System should be design with characteristic information of language type. All languages has own different alphabet and all different alphabets includes different letters. Same letters could have different phonemes. In example, Turkish Spoken Language; written language and utterance of phonemes orthography is same but in English Spoken Language it is opposite. Shortly Turkish Spoken Language is written as reading. All sounds and phonemes should be known for design ASR system. Turkish Spoken Language Alphabet divides in to two with vowels and consonants as shown below in Table 3.1[12] .

TURKISH ALPHABET	
Vowels	a-e-ı-i-o-ö-u-ü
Consonants	b-c-ç-d-f-g-ğ-h-j-k-l-m-n-p-r-s-ş-t-v-y-z

Table 3.1 Turkish Alphabet

Classification of the vowels and consonants are shown below in Table 3.2 for the Turkish Alphabet [12].

Vowels	Unrounded Vowel		Rounded Vowel	
	Wide	Close	Wide	Close
Back Vowel	a	ı	o	u
Front Vowel	e	i	ö	ü

Table 3.2a Classification of Vowels

Consonants	Fricatives	Stops	Nasals	Semivowels
Voiced	c-j-v-z	b-d-g	m-n	ğ-l-r-y
Unvoiced	ç-f-h-s- ş	t-k-p		

Table 3.2b Classification of Consonants

The list of Turkish Alphabet related to IPA, ARPAbet, HTK (Hidden Markov Toolkit) and Praat, is shown below in Table 3.3 [13].

Alphabet	IPA Phoneme	ARPAbet	HTK	Praat	Alphabet	IPA Phoneme	ARPAbet	HTK	Praat
A	/a/	AA	a	a	M	/m/	M	m	m
B	/b/	B	b	b	N	/n/,/ɲ/	N. NX	n	n
C	/dʒ/	JH	c	c	O	/o/	OW	o	o
Ç	/tʃ/	CH	C1	C	Ö	/œ/		O1	O
D	/d/	D	d	d	P	/p/	P	p	p
E	/e/,/æ/	EH, AE	e	e	R	/r/	R	r	r
F	/f/	F	f	f	S	/s/	S	s	s
G	/g/	G	g	g	Ş	/ʃ/	SH	S1	S
Ğ	/ɯ/		G1	G	T	/t/	T	t	t
H	/h/	H	h	h	U	/u/,/ʊ/	UH	u	u
I	/ ɪ /	IH	I1	I	Ü	/y/	Y	U1	U
İ	/i/	IY	i	i	V	/v/	V	v	v
J	/ʒ/	ZH	j	j	Y	/j/	JH	y	y
K	/k/	K	k	k	Z	/zh/	ZH	z	z
L	/l/,/ɬ/	L	l	l					

Table 3.3 Turkish Alphabet Phonetic Symbol List

3.4 Start-Up

ASR system is performed from two main part which as known as acoustic processor and linguistic decoder. Acoustic processor performs short-term power spectrum which is represented by Mel Frequency Cepstral Coefficients (MFCC). MFCC's are extracted using by HTK HCopy tool. Linguistic decoder is doing pattern classification with acoustic model on HMM which patterns coming from acoustic processor. In second step it tries to estimate spoken words from dictionary by helping N-gram language. At the end of two main part, confidence score is performed for confirm the success of estimation. All these steps are shown below in Figure 3.4.

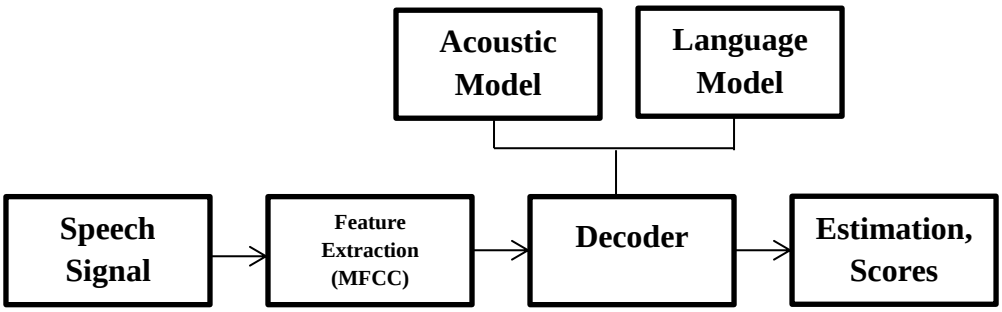


Figure 3.4 Block diagram of ASR system.

3.5 Modeling

3.5.1 Hidden Markov Model

3.5.1.1 Bayes Formulation

The speech vector X is wanted to find in class W with highest probability for recognize the spoken word in a sentence. Posteriori probability P(W|X) is needed for solve this problem. Classification can be defined as,

$$\hat{W} = \arg\max_{W} P(W \mid X) \tag{3.1}$$

By using the Baye's rule equation 3.1 can be written as,

$$P(W \mid X) = \frac{P(X, W)}{P(X)} = \frac{P(X \mid W)P(W)}{P(X)} \tag{3.2}$$

Because of P(X) is independent, it can be eliminated equation can be re-written as,

$$\hat{W} = \arg\max_{w} P_A(X \mid W) P_L(W) \qquad (3.3)$$

Where $P_A(X \mid W)$ represent acoustic modeling and $P_L(W)$ language modeling.

3.5.1.2 Word Modeling

Left-to-right HMM structure is used for word modeling. A vector sequence $\vec{X} = \{\vec{x}_0, \vec{x}_1, \ldots, \vec{x}_{(T_x-1)}\}$ is defined which belongs to word class W. At each time moment $t = 0,1,\ldots,(T_x - 1)$ HMM is equivalent to a state s_{θ_t} and it will generate a vector $\vec{x}_t$ with a probability $p\left(\vec{x}_t \mid s_{\theta_t}\right)$. Than it will make a state transition from state s_{θ_t} to $s_{\theta_{t+1}}$ with a probability $a_{\theta t}$ to $a_{\theta t+1}$. Defined vector $\vec{X}$ can thus be generated by using state of indices $\theta = \{\theta_0, \theta_1, \ldots, \theta_{(T_x-1)}\}$. The initial state probability is given 1 for the first state and probability is given 0 for the other state. Related to this the generation process starts with state s_0 and it continues from left to the right.

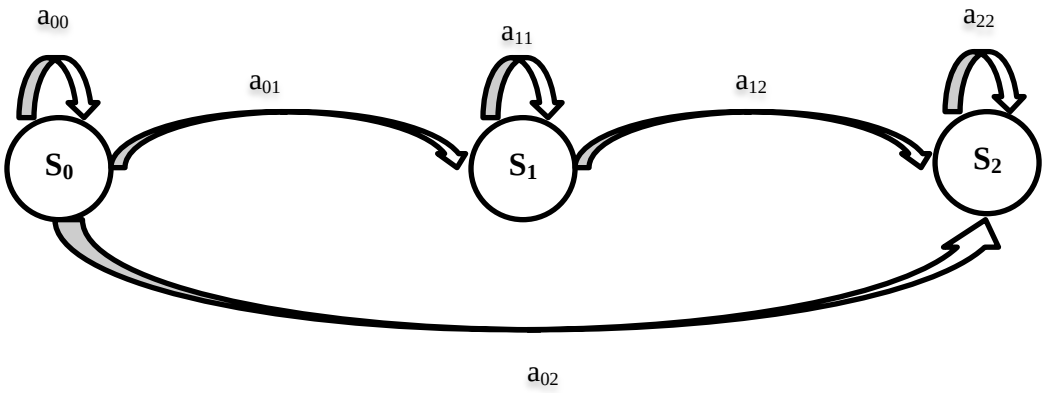


Figure 3.5 Left to right HMM for word recognition.

3.5.1.3 Viterbi Algorithm

Viterbi algorithm is found the best state sequence for maximizes the likelihood of the state sequence in observation sequence that is to say find the shortest way for reach the goal.

Hidden state is defined as,

$$q = (q_1, q_2 .... q_n) \tag{3.4}$$

Observation sequence is defined as,

$$o = (o_1, o_2 .... o_n) \tag{3.5}$$

Emission probability is defined as,

$$b = \{ b_{ik} = b_i(o_k) = P(o_k \mid q_i \} \tag{3.6}$$

Maximal probability of state sequence is represented by $\delta_t(i)$ where t is length and i is state. Equation can be performed as,

$$\delta_t(i) = \max \{ P(q(1), q(2), ..., q(t-1); o(1), o(2), ..., o(t) \mid q(t) = q_i) \} \tag{3.7}$$

Ψ is N by T matrix and it is used to retrieve the optimal state sequence from the previous step.

$$\Psi_1(i) = 0, \ i = 1, ..., N \tag{3.8}$$

$$\delta_1(i) = p_i b_i(o(1)) \tag{3.9}$$

In recursion the most likelihood is found as shown below.

$$\delta_t(j) = \max_i [\delta_{t-1}(i) a_{ij}] b_j(o(t)) \tag{3.10}$$

$$\Psi_t(j) = \arg\max_i [\delta_{t-1}(i) a_{ij}] \tag{3.11}$$

The algorithm is found the most probable states however there could be more than one. In termination most probability is select from probable states ash shown below.

$$p^* = \max_i [\delta_T(i)] \tag{3.12}$$

$$q^*_T = \arg\max_i [\delta_T(i)] \tag{3.13}$$

From final to start path backtracking is expressed as,

$$q^*_T = \Psi_{t+1}(q^*_{t+1}), \text{ where } t = T-1, T-2, \ldots, 1 \tag{3.14}$$

3.5.1.4 Baum-Welch Algorithm

Baum-Welch algorithm is used for set the HMM's parameters. Algorithm's steps are summarized as follows;

- Re-estimation is required for every parameter vector/matrix, storage locations are referred to as accumulators.
- Forward and backward probabilities are calculated for all states (j) and times (t).
- For each states and time, $\gamma_t(i)$ probability is used, and the current observation sequence to update the accumulators for that state.
- Parameter values are performed by using the final accumulator.
- All steps are repeated until finding the maximum probability of $P(O \mid \lambda)$.

3.5.2 Acoustic Modeling

As we mention before in Bayes Formulation, conclusion performed by acoustic model equation and language model equation. Acoustic model is an algorithm which contains statistical representations of each sound that makes up a word. In English language it is enough for work with 40 acoustic-phonetic models but in Turkish Spoken Language which we are working on, only 29 acoustic-phonetic models are enough because written language and utterance of phonemes orthography is same in Turkish Language. This statistical representation expressed with helping by HMM. Acoustic model equation is expressed in equation 3.3.

$$P_A(X \mid W) = P_A\left(\{X_1, X_2, \ldots, X_T\} \mid \{W_1, W_2, \ldots, W_T\}\right) \tag{3.15}$$

If assumptions are done as below,

- t is lined up with word model i
- HMM model affirm by j
- Independent
- Each X_T response to w_j^i

Equation 3.4 can be re-write as below;

$$P_A(X \mid W) = \prod_{t=1}^{T} P_A\left(X_T \mid w_j^i\right) \tag{3.16}$$

Each phonetic unit modeled with a mixture of Gaussians;

$$b_j\left(X_T\right) = \sum_{k=1}^{K} c_{jk} N\left(X_T \mid \mu_{jk}, U_{jk}\right) \tag{3.17}$$

Where $b_j\left(X_T\right)$ represents to mixture of Gaussian normal densities, k represents to the number of mixture components in the density function , c_{jk} represents to the weight of the mixture component in state j with corresponds to k , N represents to Gaussian density function, μ_{jk} represents to mean of Gaussian density function corresponds to $j\ k$ and U_{jk} represents to covariance of Gaussian density function corresponds to $j\ k$. In dictionary all words can be defined as mono-phones or tri-phones. In example for “amerika” word; mono-phones are expressed “a-m-e-r-i-k-a” and tri-phones are expressed “ame-mer-eri-rik-ika”.

3.5.3 Language Modeling

The second conclude equation of Bayes Formulation is language modeling. Language model is an algorithm which contains word sequences and their probabilities. Word sequence can be expressed as:

$$W = w_1, w_2, ..., w_n \tag{3.18}$$

And language model equation which is expressed in equation 3.3 can be re-write as;

$$P_L(W_n) = P(W_1)P(W_2 \mid W_1)P(W_3 \mid W_1W_2).....\quad (3.19)$$

$$P_L(W_n) = P(W_n \mid W_{n-2}W_{n-1})\quad (3.20)$$

It is not cost effective to doing estimation for large terms. Equation could become effective with using N-gram model. (Eq. 3.21)

$$P(W_n \mid W_1W_2....W_{n-1}) \cong P(W_n \mid W_{k-n+1}...W_{n-1})\quad (3.21)$$

When chain rule is applied into equation 3.10,

$$P_L(W) = \prod_{i=1}^{n} P(W_i \mid W_{i-1}W_{i-2}...W_{i-k+1})\quad (3.22)$$

3.6 Hidden Markov Toolkit

3.6.1 Introduction

Hidden Markov Toolkit [15] which is based on Hidden Markov Model, is used for recognize the speech. In our work, HTK Turkish speech recognition system which is reconfigured in Boğazici University, is used. HTK toolkit is build up on two main base; training tools and recognizer. Firstly training tools are estimated mathematical parameters with respect to HMM and secondly unknown utterances are transcribed with helping recognizer.

HTK toolkit performed with own tools and these tools are build up with 4 main processing steps; data preparation, training, testing and analysis.

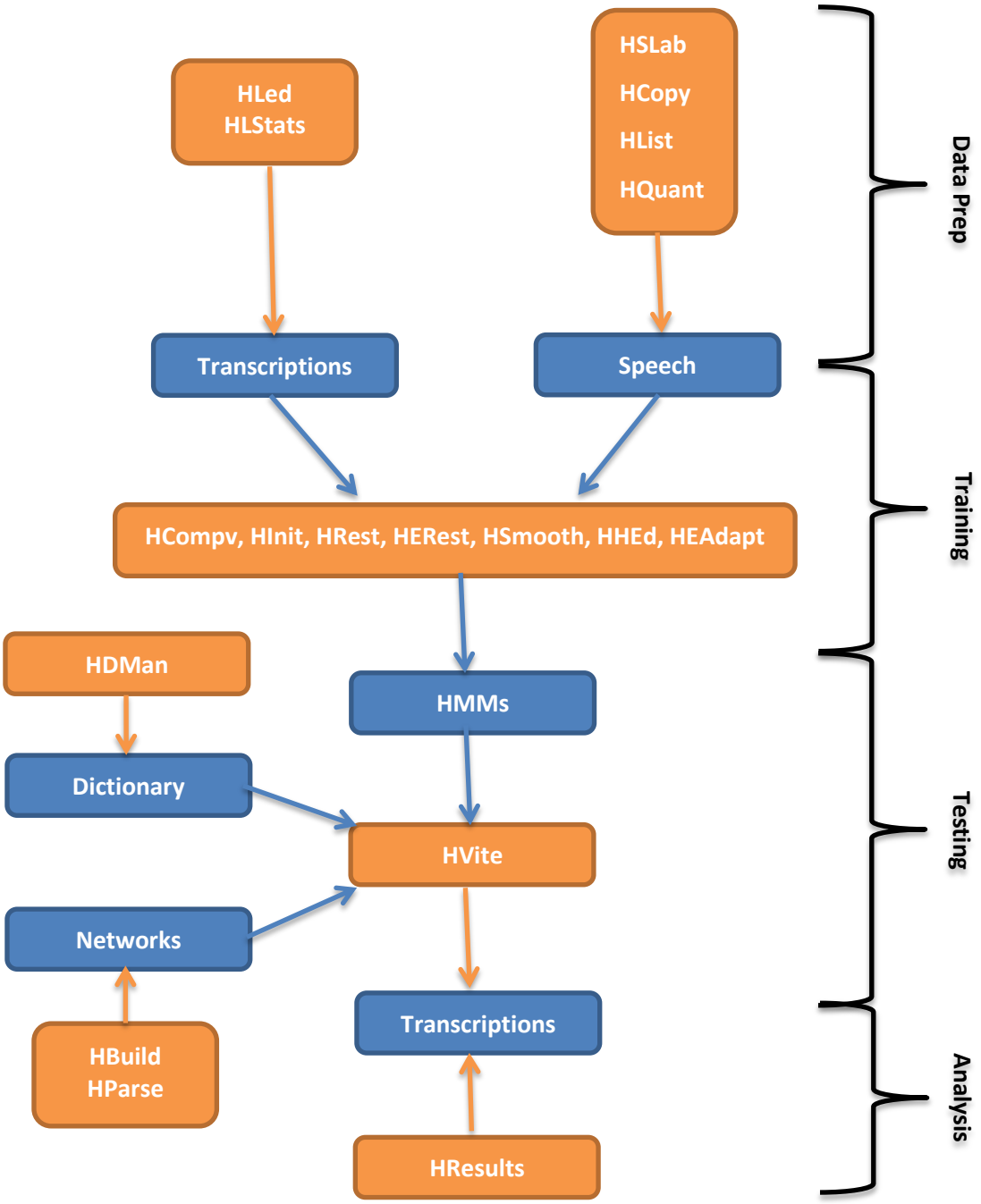


Figure 3.6 Flow diagram of HTK processes.

3.6.2 Data Preparation

Process starts with data preparation. An audio file (speech data) and their transcriptions are needed for the input, if you haven't, you can record it with HSLab. And this speech data is needed to be converted in correct form with phone or word labels. HCopy is a tool for parameterizing the data which is used for extracting MFCC of the speech signal too. This parameterized data could be seen with HList tool and data could be designed to make the required transformations with a label editor tool HLed which could construct the *Master Label Files* too. Last step in data preparation, statistics on label files are displayed with HLStats tool and VQ codebook is built with HQuant tool. Now we are ready for the next step of the process.

3.6.3 Training

HTK works with HMMs for built the desired topology. In this second step of the process, phone models are built. An initial set of models are extracted with HInit tool than further re-estimation doing for isolated words with HRest tool. Utterance's (i.e phone) locations are labeled for used as *bootstrap data.* Initial set of parameter values are computed with *segmental k-means* procedure than mean and variance is computed. In the initial estimation parameters are defined by using Viterbi alignments than in re-estimation stage which is performed by HRest tool, Baum-Welch algorithm is used. When there is no bootstrap data, initialization done by HCompV tool. With initial set of models are created, HERest tool is performed for *embedded training*. Baum-Welch and forward-backward algorithms are used in this tool, in summary this tool is the heart of training process. As we mention before HMMs are used, HHEd tool is an editor for change the HMM parameters of system.

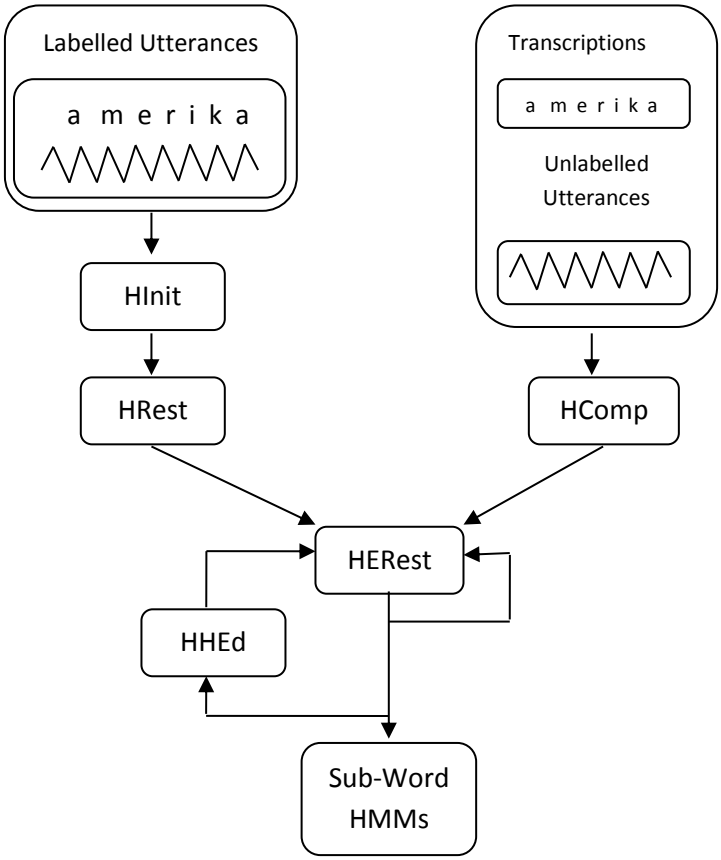


Figure 3.7 Flow diagram of training process.

3.6.4 Testing

HVite tool which is allow recognition by applying language model into speech data. It uses Viterbi algorithm. HVite takes as input word sequence, dictionary with pronunciation and set of HMMs. All this needs are completed in previous steps with helping BUSIM for Turkish Language. At the output tool is obtained phoneme based and word based labeled segment time marks for the spoken data, than this output will be used for prosodic feature extraction which will explain in next chapters.

3.6.5 Analysis

HTK processes are completed with measuring performance of the system. By helping HResults tool, recognized words from the HTK tool are compared with given manually wrote text input file. Success of system is appeared.

Chapter 4

Prosodic Features

4.1 Definition

With the application of ASR system into an audio file, speech is converted into a text file by the way pitch patterns and time scales are lost as mentioned before. These lost patterns are called prosody. It carries structural, semantic and functional information. Output of ASR is a text file and this file includes only words, there isn't any sentences boundary, capitalization, punctuation, headers or paragraphs. Prosodic features along with to solve these problems. Prosodic feature includes pausing, pitch and amplitude change difference, global pitch declination, melody, boundary tone distribution and speaking rate variation. However prosodic features are irresponsive from the word's meanings and ASR; according to this, system get better performance with no additional training data. Thus, performance gains can be evaluated quickly and cheaply, without requiring additional infrastructure [5].

4.2 Features

There are three main types of features such as basic features, statistical features and derived features.

4.2.1 Basic Features

Basic feature includes four types of feature. These features are;

-Base Features

-Duration Features

-F_0 Features

-Energy Features

4.2.1.1 Base Features

Base features are included only the basic information of audio file such as location, gender and identity.

4.2.1.2 Duration Features

Duration features are the basic type of prosodic features. It examines inter-words according to pauses and durations of phones and rhymes. Pause features are passing time between two boundaries word in second as shown in Figure 4.1. Furthermore this features also using for to detect semantic information. The other side phone and rhyme duration features are phone duration which is in previous rhyme of a word. However this features using for to detect semantic information too. Word duration, following word duration, last time rhyme duration and last phoneme duration are given example for the duration features.

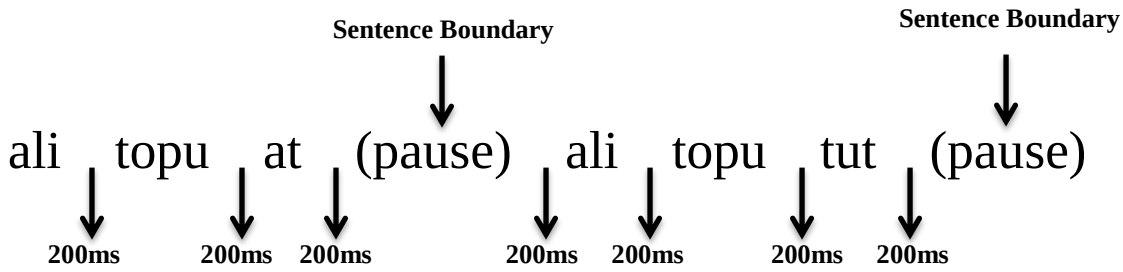


Figure 4.1 Sentence Boundaries

4.2.1.3 F_0 Features

F_0 Features are more difficult to model than the other prosodic features. It is related with pitch information where pitch includes highness or lowness tone. This is largely attributable a variability in the way pitch is used across speakers and speaking contexts, complexity in representing pitch patterns, segmental effects and pitch tracking discontinuities (such as doubling errors and pitch halving, the latter of which is also associated with non-model voicing.). F_0 obtained by using get_f0 and fundamental frequency estimated by using autocorrelation. Output of this operation, two main noises are produced. Probability of halving and doubling are estimated by a lognormal tied mixture model (LTM). Then median filter is applied for smoothing to

the two main noises. At the last step F_0 is stylized using greedy algorithm which detects discontinuities by mean square error method and features are completed by compute the slopes. Minimum style fit F_0, maximum style fit F_0 and mean style fit F_0 are given example for F_0 features. Feature extraction is summarized in Figure 4.2.

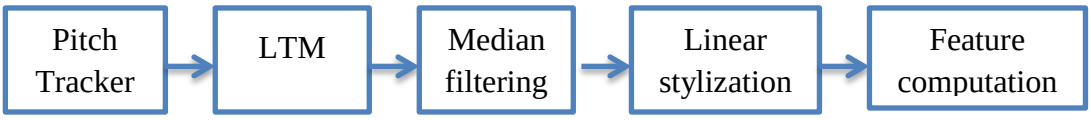


Figure 4.2 Extractions of F_0 Features

4.2.1.4 Energy Features

Energy features are extracted with energy calculations from each word. Minimum energy, maximum energy, minimum next energy and maximum next energy are given example for energy features.

4.2.2 Statistical Features

These features are not output of the tool. But these features are performed by computation of original features like mean and deviation operations.

4.2.3 Derived Features

Derived features are formed by using basic features (duration features, F_0 features and energy features) and statistical features. Derived features can be computed by using two basic features or using computed statistics. Normalized word duration, normalized pause, normalized vowel duration, normalized rhyme duration, F_0 derived features, average phone duration and speaker specific normalization are type of derived feature.

4.3 Prosodic Feature Extraction

4.3.1 Basic Information

The Purdue Prosodic Feature Extraction tool [10] which based on Praat, is used for prosodic feature extraction. The prosodic features are extracted directly from the speech signal given its time alignments to a human generated transcription or to automatic speech recognition (ASR) output. All types of prosodic features are detail explained in Appendix A which features are extracted again in using Praat and also scripts for ‘’Computing Global Statistics & Extraction Prosodic Features’’ expressed in Appendix B. All steps of extraction as shown in Figure 4.3.

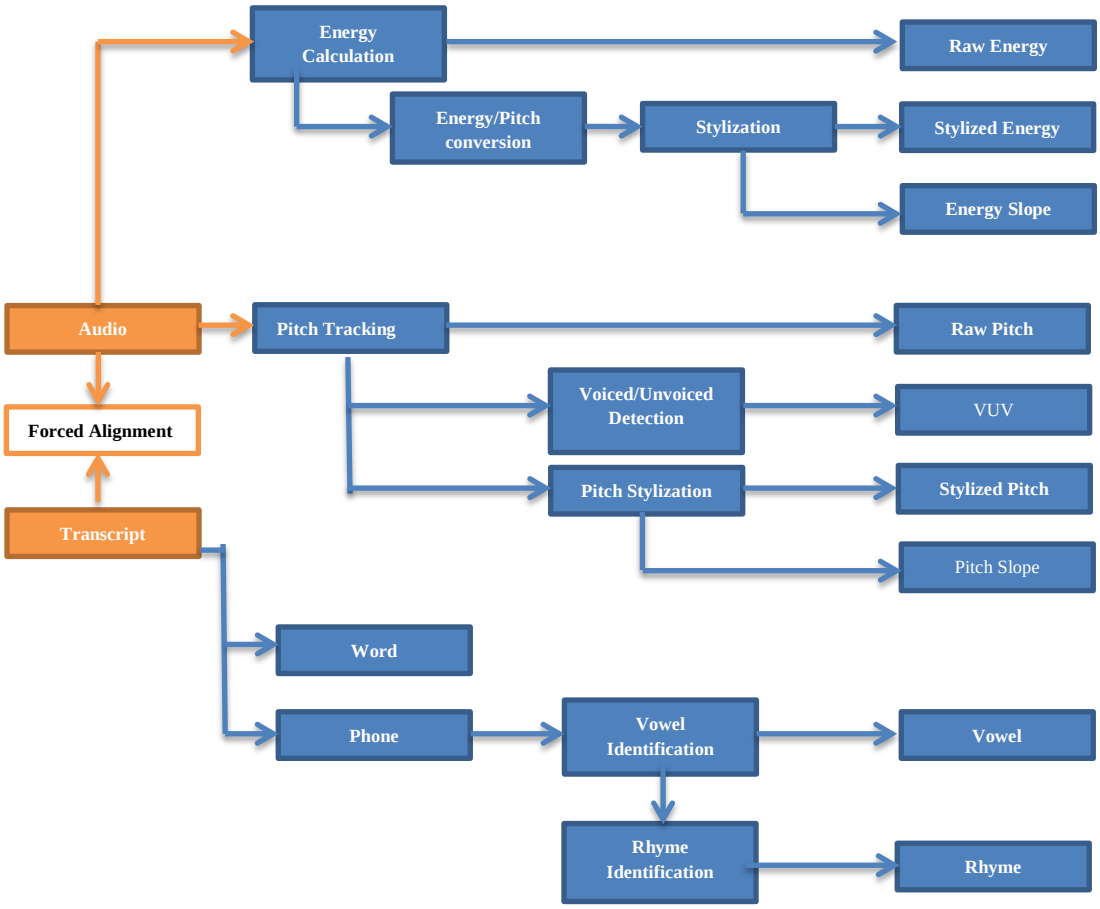


Figure 4.3 All Steps of Prosodic Feature Extraction

Using output of feature extraction which is shown in Figure 4.3 before, types of features can be computed as shown in Table 4.1.

	Duration Features	F_0 Features	Energy Features
Word	+	+	+
Phone	+	-	-
Vowel	+	-	-
Rhyme	+	-	-
VUV	-	+	-
Raw Pitch	-	+	-
Stylized Pitch	-	+	-
Pitch Slope	-	+	-
Raw Energy	-	-	+
Stylized Energy	-	-	+
Energy Slope	-	-	+

Table 4.1 Output of Feature Extraction and Feature Types Relationships.

Tool needs three input files for realizing feature extraction. Such input files are audio file (WAV or AIFF) and phone-word alignments corresponding to audio file. Phone alignments and word alignments should be rearranged for convert in Praat format (.textgrid). Audio waveform corresponding to phone alignments and word alignments are shown in Figure 4.4. ‘’.textgrid’’ praat format for phone and word are shown in Table 4.1.

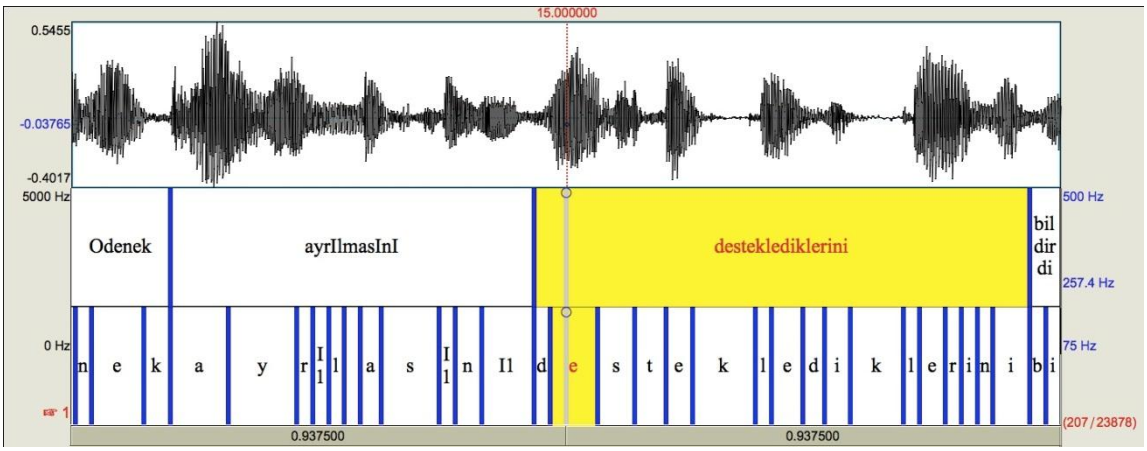


Figure 4.4 Audio waveform corresponding to phone and word alignments.

word.textgrid

```
File type = "ooTextFile"
Object class = "TextGrid"

xmin = 0
xmax = 1781
tiers? <exists>
size = 1
item []:
    item [1]:
        class = "IntervalTier"
        name = ""
        xmin = 0
        xmax = 1781
        intervals: size = 4145

.......

        intervals [34]:
            xmin = 13.87
            xmax = 14.25
            text = "Odenek"
        intervals [35]:
            xmin = 14.25
            xmax = 14.94
            text =
"ayrIlmasInI"
        intervals [36]:
            xmin = 14.94
            xmax = 15.88
            text =
"destekliklerini"
        intervals [37]:
            xmin = 15.88
            xmax = 16.33
            text = "bildirdi"
```

phone.textgrid

```
File type = "ooTextFile"
Object class = "TextGrid"

xmin = 0
xmax = 1781
tiers? <exists>
size = 1
item []:
    item [1]:
        class = "IntervalTier"
        name = ""
        xmin = 0
        xmax = 1781
        intervals: size = 23878

.....

intervals [206]:
            xmin = 14.94
            xmax = 14.97
            text = "d"
        intervals [207]:
            xmin = 14.97
            xmax = 15.06
            text = "e"
        intervals [208]:
            xmin = 15.06
            xmax = 15.13
            text = "s"
        intervals [209]:
            xmin = 15.13
            xmax = 15.19
            text = "t"
        intervals [210]:
            xmin = 15.19
            xmax = 15.24
            text = "e"
        intervals [211]:
            xmin = 15.24
            xmax = 15.36
            text = "k"
        intervals [212]:
            xmin = 15.36
            xmax = 15.39
            text = "l"
        intervals [213]:
            xmin = 15.39
            xmax = 15.45
            text = "e"
        intervals [214]:
            xmin = 15.45
            xmax = 15.49
            text = "d"
        intervals [215]:
            xmin = 15.49
            xmax = 15.54
            text = "i"
        intervals [216]:
            xmin = 15.54
            xmax = 15.64
            text = "k"
        intervals [217]:
            xmin = 15.64
            xmax = 15.67
            text = "l"
        intervals [218]:
            xmin = 15.67
            xmax = 15.72
            text = "e"
        intervals [219]:
            xmin = 15.72
            xmax = 15.75
            text = "r"
        intervals [220]:
            xmin = 15.75
            xmax = 15.78
            text = "i"
        intervals [221]:
            xmin = 15.78
            xmax = 15.81
            text = "n"
        intervals [222]:
            xmin = 15.81
            xmax = 15.88
            text = "i"
```

Table 4.2 ‘’word.textgrid’’ and ‘’phone.textgrid’’ praat format corresponding to figure 4.3 interval.

Additionally wav information list is prepared for labeling speakers which is also using for extraction ‘’Base Features’’. List is shown in Table 4.3.

SESSION	SPEAKER	GENDER	LOCATION
Demo_1	Speaker 1	Male	../demo/data/demo_1.wav
Demo_2	Speaker 2	Female	../demo/data/demo_2.wav
Demo_3	Speaker 3	Female	../demo/data/demo_3.wav
Demo_4	Speaker 4	Male	../demo/data/demo_4.wav

Table 4.3 Wav Information List

4.3.2 Structure Of The Prosodic Feature Extraction Tool

Structure is comprised from two main parts. One of them is called ‘’ Global Statistics Computation’’. This part put efforts for computation basic features (see section 4.2.1) and statistical features (see section 4.2.2). Other part is called ‘’ Feature Extraction’’. This part is extracted prosodic features using basic features and statistical features. (see section 4.2.3). Flow diagram of Praat prosodic feature extraction tool is shown in Figure 4.5.

4.3.3 Software Usage (Praat)

As an input, the wav file and corresponding word and phone aligned files (shown in Figure 4.4) are loaded into the workspace (Example; ‘’demo/work_dir/”). Also for the general information ‘’Wav Information List” table (shown in Table 4.3) should be prepared which respect to input files. For the easy to use, user interface could be used or program could be run from command windows too. Steps to run program correctly could be seen from Table 4.4 both global statistic computation and prosodic feature extraction in Praat.

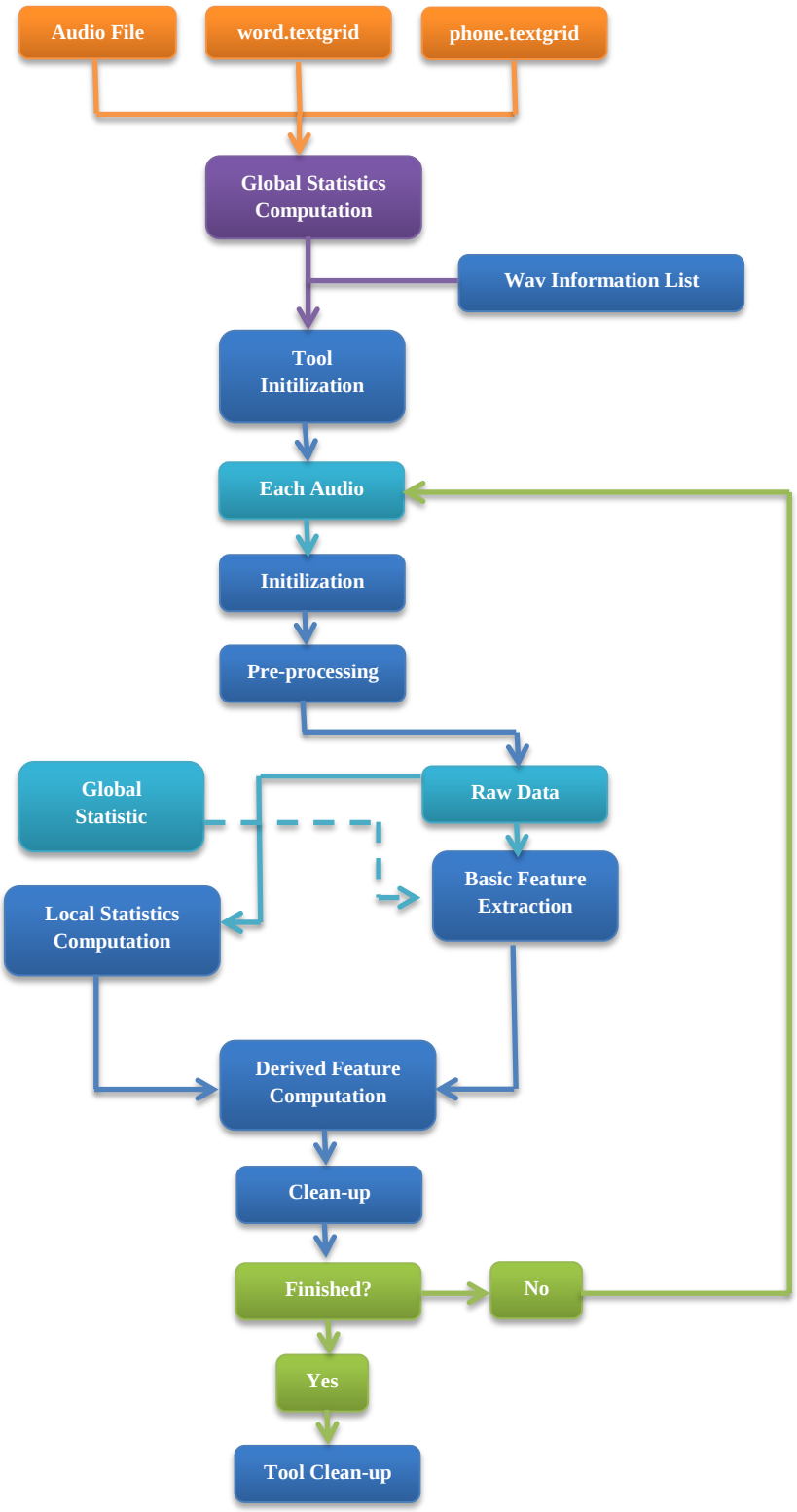


Figure 4.5 Praat Prosodic Feature Extraction Toolbox Flow Diagram

Global Statistics Computation

Enter the code.

```
praatstats_batch.praat ../demo-wavinfo_list.txt ../demo/work_dir yes
```

1. Run *Praat*.
2. *Praat Objects / Read / Read from file / select_stats_batch.praat*
3. Click *Run* on *Script Editor*.
4. Type *../demo-wavinfo_list.txt* and *../demo/work_dir* into the boxes and click *yes* if you want to use existing parameter files, *no* to generate parameter files from the beginning.
5. Click *OK*.
6. Process is displayed in the *Praat Info Window*.

Prosodic Feature Extraction

Enter the code,

```
praatmain_batch.praat ../demo-wavinfo_list.txt
user_pf_name_table.Tab\ ../demo/work_dir/stats_files
../demo/work_dir yes
```

1. Run *Praat*.
2. *Praat Objects / Read / Read from file / select main_batch.praat*
3. Click *Run* on *Script Editor*.
4. Type *../demo-wavinfo_list.txt* and *../demo/work_dir* into the boxes and click *yes* if you want to use existing parameter files, *no* to generate parameter files from the beginning.
5. Click *OK*.
6. Process is displayed in the *Praat Info Window*.

Table 4.4 Usage of Praat[10]

Extracted prosodic features could be found at the following directory. *"workspace/pf_files/"* by using any word processors. Because of the table is performed from rows and columns, Microsoft Office Excel word processor is best for analyzing prosodic features in correct lines for the given word. Table 4.5 shows an example for output of Praat. As we mention before all Prosodic Features and their detail explanation is appear in appendix A.

WORD	**WAV**	**SPKR_ID**	**GEN**	**WORD_START**	**FEATURE NAMES**	**LAST FEATURE**
Word 1	Location	ID	GENDER	FEATURE	FEATURE	FEATURE
Word 2	Location	ID	GENDER	FEATURE	FEATURE	FEATURE
Word 3	Location	ID	GENDER	FEATURE	FEATURE	FEATURE
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 4.5 Output of Praat

Chapter 5

Morphological Features

5.1 Definition

Every language has own information and every language is encode own information by own words also own letters. Morphology studies with how they encode this information by the words, in addition it studies with structure of the words. Words are performed by small units come together. Smallest unit which includes linguistic information called morphemes and they are consisted from phonemes. Morphemes are classified into two groups; free morphemes and bound morphemes. Free morphemes are performed words by themselves. In example; play, stop etc. Bound morphemes are not performed words by themselves but they attached to free morphemes for performed a new word. In example; +ed, +ness etc. Turkish Spoken Language which we are studying on, is get in to Agglutinative type of languages. In agglutinative type of languages, bound morphemes are attached one or more free morphemes for performed a new word. Nouns, pronouns, participles and infinitives are nominal morphological features, these features are effect the new words because of number (one or more) and cases. Again verb marker morphological features effect the new words because of voice (Active or passive), polarity (Negative or positive), tense, aspect, possessor (Singular or plural; 1,2,3) and modality.

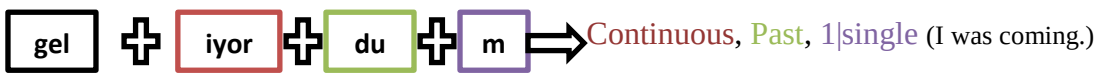


Figure 5.1 Agglutinative Type Of Language Example.

5.2 Morphological Processes

Morphological processes are examined in 3 groups as shown below;

-Inflectional morphology

-Derivational morphology

-Compounding morphology

5.2.1 Inflectional Morphology

Inflection morphology is modified the word because of tense, gender, number, aspect or word contains both free morpheme and bound morpheme.

- Subject-verb agreement, tense, aspect;

Gel-iyor-um.=>Continous-1|Single – I am coming.

Gel-iyor-sun.=>Continous-2|Single - You are coming.

Gel-iyor.=>Continous-3|Single - He/She/It is coming.

Gel-iyor-uz.=>Continous-1|Plural – We are coming.

Gel-iyor-sunuz.=>Continous-2|Plural – They are coming.

Gel-iyor-lar.=>Continous-3|Plural – They are coming.

- Constituent function;

Okula-a gittim.- I went to the school.

Okul-u gördüm. – I saw the school.

Okul-dan nefret ettim. – I hated from the school.

Okul-da kaldım.- I stayed at the school.

- Number, case, possession, gender, noun-class for nouns;

Okul-lar-ımız-dan. – From our schools.

- Gender/Case (Respect to artikel, there is not any example in Turkish) ;

Hamburg ist eine schön-e (die)stadt. – Hamburg is a beautiful city.

5.2.2 Derivational Morphology

Derivation morphology performs a new word with addition phonemes to free morphemes, the new word derived from the existing word. Productive derivational morphology is applied in to almost of vocabulary and unproductive derivational morphology is applied in to only few words of vocabulary.

Büyük (Big, Adj.) => Büyük-lük (Size, Noun)

Git (Go, Verb) => gid+er+ken (While going, Adverb)

5.2.3 Compounding

Compounding morphology performs a new word with addition of two or more free morphemes. New words usually perform with 2 nouns come together. There will be phonemes between two free morphemes in compounding.

Bilgi+sayar (Information; Noun , Counter; Noun) => Computer; Noun

5.3 Combining Morphemes

There's a lot of way for make up a new word when combining morphemes. Bound morphemes could be inserted among of free morpheme, head of free morpheme, end of free morpheme or both head and end of free morpheme when combining morphemes. Effect of this there could be some phoneme changes at boundary. Related with this new word could be performed with duplicate the words or without additional any boundary. Types of combining morphemes are as shown in table 5.1.

Concatenative Combination (Prefixation)	ir+rasyonel
Concatenative Combination (Suffixation)	elçi+lik
Concatenative Combination	şarap=>şarab+ı
Concatenative Combination	burun=>burn+u
Infixation (There is not any example in Turkish)	fikas=>fumikas
Circumfixation (There is not any example in Turkish)	sagen=>gesagt
Reduplication	ma+s+mavi
Zero Morphology	yüz (noun)=>yüz (Verb)

Table 5.1 Types of Combining Morphemes

5.4 Morphological Feature Extraction

5.4.1 Basic Information

According to our aim all type of features are extracted from open source tools such as TRmorph feature extraction tool which is used for morphological feature extraction. TRmorph tool [7] is a fairly complete and accurate two level morphological analyzer for Turkish Spoken Language which is found by Çağrı Çöltekin from University of Groningen. The tool implemented using finite stated transducers (FSTs) like the all other morphological tools, especially Stuttgart finite state transducer tool (SFST) is used. SFST is used because of tool set particularly aimed for implementing morphological analyzers with open source. We used updated version of this tool which is implemented with more popular finites state description languages *lexc* and *xfst* from Xerox (Beesley and Karttunen 2003), using Foma (Hulden 2009) [23] as the main development tool. But it doesn't mean tool could implement with only this compiler (Foma), system could implement with any *lexc* and *xfst* compiler without additional effort. Because of the tool is aimed for usage of developers, every user could customize any settings easily with respect to given instructions on user manual [36]. Tool is as allowed to add or modify the lexical entries and it could use for different subjects such as sentence segmentation, spell checking, toping segmentation or etc. If we want summarized tool in short; system is taken input Turkish spoken word and output of the tool is morphologic analyzed.

5.4.2 Structure Of The Morphological Feature Extraction Tool

Morphological computation could be done in two deals. Computational synthesis produce the word from given a set constituent morphemes or information be encoded. And secondly, computational analysis which we study on, separate and identify the constituent morphemes and mark the information they encode from the given word. Morphological analyzer is took given word as an input and try to find it in memory. The box which we called Data is included the all lexicons for Turkish spoken language which information included is specific for all languages. Lexicons are structured collection of all the morphemes which are root words (free morphemes) and morphemes (bound morphemes). At the second step, the box which we called Engine and it is language independent, takes information from Data. As an end, Engine separates given word (input) in to root words and morphemes. Summarized of the all these steps as shown in figure 5.2.

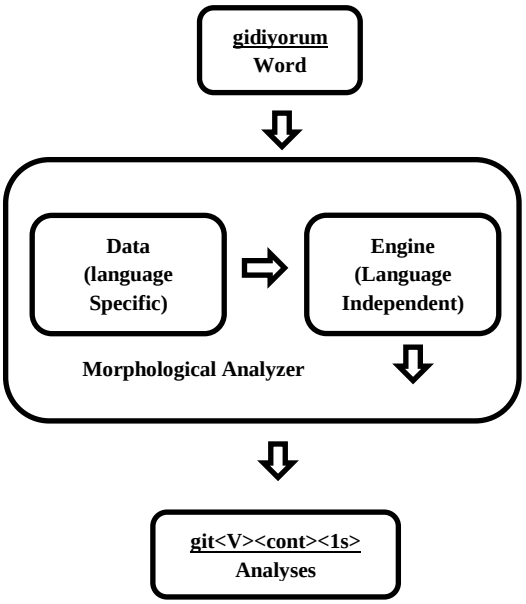


Figure 5.2 Structure of the Morphological Feature Extraction Tool.

5.4.3 Dictionary

As we guess list of all the words in studied language should be given in to dictionary but words format is important for tool works correctly and done analyzing with high accuracy. Root words which are subset of lexicons, should be incorporate with

specific information if available. Specific information is not available in Turkish spoken language but in some languages which are using articles, such as gender, animateness or etc. information could affect the word as we mention previous sections. And also in addition a list of morphemes along with the morphological information and features should be added. These are plural morphemes, verb morphemes and personality morphemes.

All words should be included both lexical form and surface form. Lexical form is underlying representation of morphemes with all morphographemic (morphological model) changes are applied and surface form is actual written form of the word. Consistent representation of the word should be known too because according to specific language rules, same word could be different in lexical form and surface form. However some tags are used in TRmorph analyses which aren't necessarily match with any of the tags used in any grammer books. These tags are allowed to easy access to the points. In the same time these tags could be defined as classification which respect to Turkish spoken language's specific rules. These tags are represented with capital letters and as shown below table 5.2 with lexical-surface form examples.

Tags	**Surface Form**	**Lexical Form**
A={a,e}	Okul+lAr	Okullar
I={ı,i,u,ü}	Okul+lI	Okullu
D={d,t}	Okul+DA	Okulda
P={p,b}	Kitap+PI	Kitabı
K={k,ğ,y}	Bıçak+KI	Bıçağı

Table 5.2 Tags

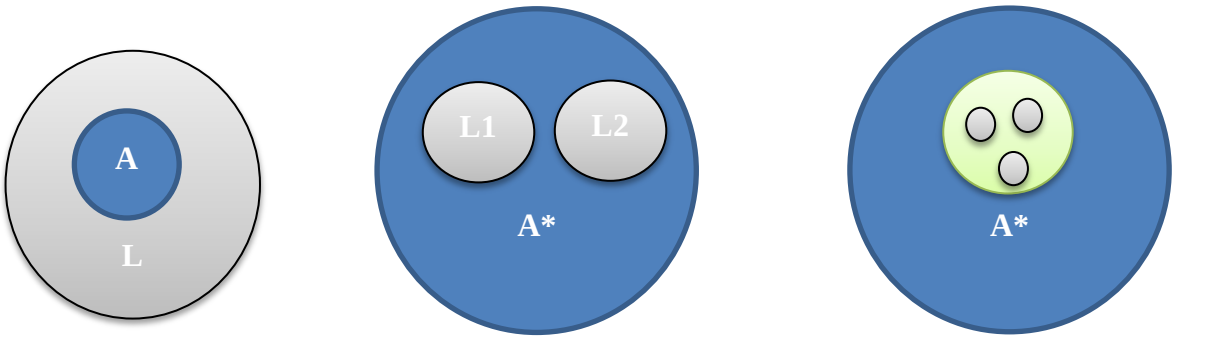
Structure of morphology especially in Turkish spoken language is ambiguous. Same word could be both noun and verb and because of surface form of the words are represented with the tags there could be same written words but without the same meaning. We could generate a lot of situations for ambiguity. Implementation could be done with three approaches; list all word-forms as a database, heuristic/Rule-based affix-stripping and finite state approaches which we'll study on.

5.4.4 Approach

Along with the system could be generated in mathematical model, computation done with high accuracy and provide representation towards to system needs; Finite state approach [21] is used in morphological extraction tool.

5.4.4.1 Finite State Approach

Alphabet (A) of the language is a finite set and alphabet could be represent as a subset of all words and all sentence in language (L) which could have meaning in this type of language. But if we generate a set from Alphabet with all combination of the characters (A*) that include in alphabet with meaningless, now this set become superset of set L. A* is an infinite set and that the aim is limit the set for generate a finite set. Because you couldn't generate an infinite dictionary set thereby you couldn't match any word in dictionary set.



Alphabet

All possible string (Infinite)

All possible string that include in chosen type of language (Finite)

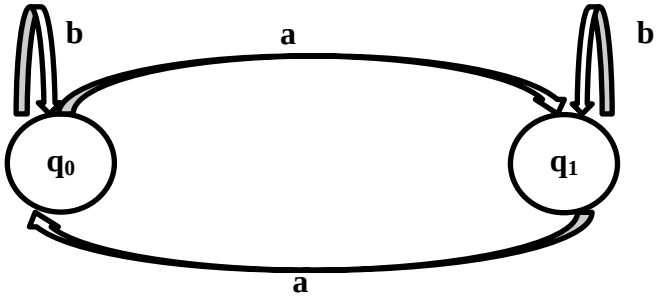
Figure 5.3 Limiting dictionary set.

Now our language is become a regular form so we could continue with Finite State Recognizer. M represent abstract machine for regular languages, it is also accepted as L is subset of A*. System starting in state q0, M proceeds by looking at each symbol in *w* and it is end up in one of the final states when the string *w* is exhausted.

$$M = \{Q, A, q_0, Next, Final\} \tag{5.1}$$

Where,

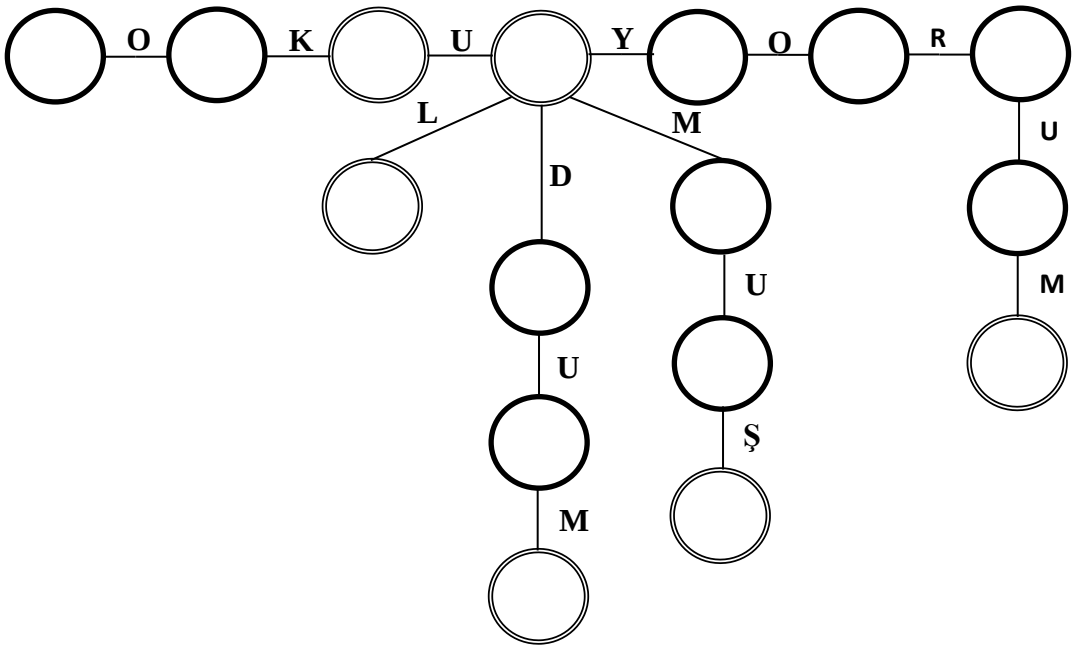
$Q : Set\ of\ states$
$A : Alphabet$
$q_0 : Initial\ State$
$Next : Next\ state\ function\ Q \times A$
$Final : Final\ State$



$A = \{a, b\}$
$Q = \{q_0, q_1\}$
$Next = \{((q_0, b), q_0), ((q_0, a), q_1), ((q_1, b), q_1), ((q_1, a), q_0)\}$
$Final = \{q_0\}$

Figure 5.4 Finite State Recognizer Approach

There are only three conditions for applying finite state recognizer approach. These are finite states, finite alphabet (letters) and finite transition. All the information should be included in recognizer in finite condition. An example is as shown below.



Ok, Okul, Okuyorum, Okudum, Okumuş…..

Figure 5.5 Finite State Recognizer Approach Example

5.4.4.2 Two-Level Description

Kimmo Koskenniemi thought multiple rules could be applied in parallel too. In those days multiple rules are applied with cascade in sequence. He called this new way to describe phonological alternations in finite state terms "*Two-Level Morphology*". Two-level morphology is based on three ideas. Firstly rules are applied in parallel, secondly constraint could refer to lexical form and surface form both at the same time and thirdly lexical lookup and morphological analysis are performed in tandem. Towards all of this information, system should include specific rule which type of language is chosen. The set of possible lexical and surface symbol correspondences with respect to rule is represented by feasible pairs as shown below.

Lexical Form	Kitab0ı
Surface Form	Kitap+ı
Feasible Pair	{k:k, i:i, t:t, a:a, b:p, 0:+, ı: ı }

Table 5.3 Feasible Pairs

The phonotics rules of contemporary Turkish have been encoded using 22 two-level rules while the morphotatics of the agglutinative word structures has been encoded as finite-state machines for verbal, nominal paradigms [18]. Turkish lexicons are performed for nouns, adjectives, verbs, compound nouns, proper nouns, pronouns, adverbs, connectives, exclamations, postpositions, acronyms, technical words, and special cases. In Turkish there are 18500 nominal roots which is include adjectives and nouns, 2450 verbal roots and 100 lexicons for suffixes.

Each rule helps to word for convert lexical form to surface form. Every word gets through from all rules which are operated in parallel and checks independently. Rules and feasible pairs are support themselves for create the output in surface form. Rules are applied letter by letter in each recognizer, sees that same pair of letter. Two-Level Morphology architecture for Turkish Spoken Language is as shown below.

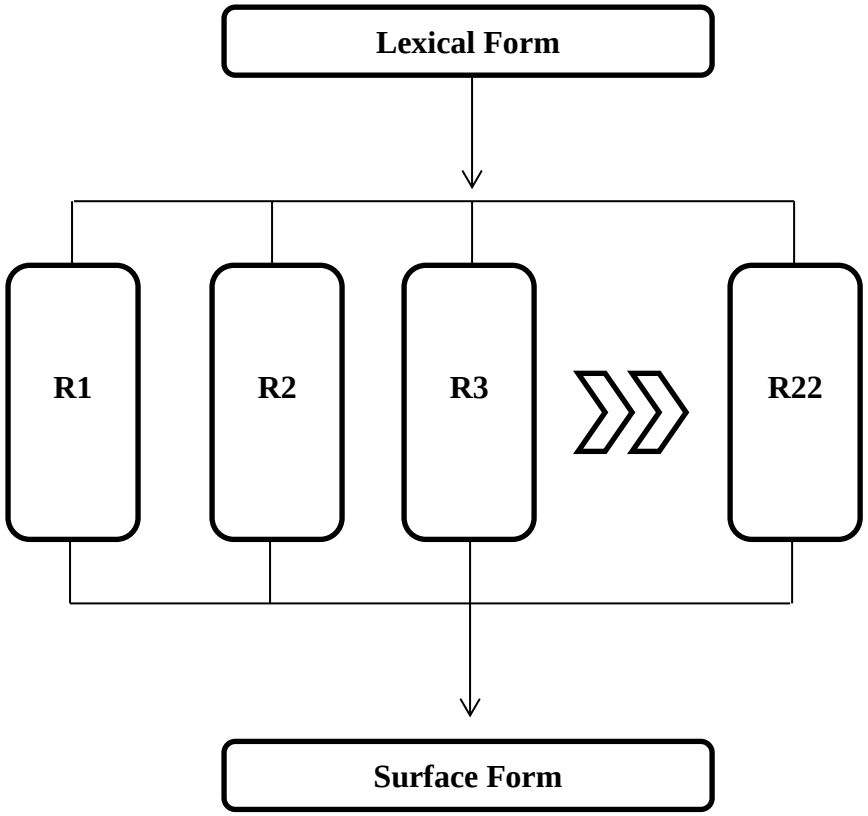


Figure 5.6 Two-Level Morphology Architecture for Turkish Spoken Language

5.4.5 Software Usage (TRmorph)

TRmorph which is an open source tool, could be download from google.code.com freely. To compile TRmorph from the source, a lexc and xfst compiler is needed. We used foma compiler which is an open-source tool too, and again it could be download from google.code.com. It has similar interface with Xerox xfst and the reason for choosing foma compiler is support most of the commands and the regular expression syntax in Xerox xfst. Libreadline-dev and zliblg-dev packages are needed for foma compiler and that could be downloaded with *apt-get* command in Linux.

After all applications towards necessary needs are installed, system is ready with default options. All these options could be customized from options.h file which is included in system files. System is started with *foma* command and then trmorph.fst main file which could customize too, is import. Than morphologic feature extraction could be done with *up* command. An example of TRmorph with morphological feature extraction is shown below.

```
>>cd TRmorph-master/
>>foma
>>foma[0]:regex @”trmorph.fst”;
>>foma[1]:up okuyorum
oku<V><cont><1s>
```

Table 5.4 Usage of TRmoprh

Chapter 6

Lexical Features

6.1 Definition

Lexical features are corresponding with word of a language. N-gram method is applied for extraction of lexical features. N-grams are an idea of word prediction with respect to probabilistic mathematically models which could predict the next word from the previous words. Such of these statistical models related to word sequences are also called language models (LMs). So if we summarized, we are used N-gram method which is a probabilistic language model for predicting the next label. This probabilistic model is based on Markov model like all the other probabilistic models. Our goal is to compute probability of a word with given previous word information. Also that model has applications on probability, communication theory, computation linguistic or biology and data compression.

6.2 Modeling

The main problem is predicted the next word label with given history. We could write probabilistic equation such as;

$$p(word \mid history) \tag{6.1}$$

Words are a sequence such as $w_1, w_2, ..., w_n$ and history is symbolized with w_{n-1}.

Using chain rule of probability equation could be re-written;

$$P(w_1^n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1^2)...P(w_n \mid w_1^{n-1}) = \prod_{k=1}^{n} P(w_k \mid w_1^{k-1}) \tag{6.2}$$

The chain rule is gave the connection between computing the joint probability of current word and conditional probability of given previous word. Languages which are covered the word sequence, are very long sets and computation of equation 6.2 becomes impossible.

Especially, we are used not all of the N-grams which are goes to infinite. So Markov assumption which is the probability of a word depends only on the probability of a limited history (finite), is done for compute this equation. Now we could generalize the bigram which works with only one word into past to trigram which works with two words into the past such as;

$$P\left(w_1^n\right) \approx \prod_{k=1}^{n} P\left(w_k \mid w_{k-1}\right) \tag{6.3}$$

1-gram, Unigram	$p\left(x_n\right)$
2-gram, Bigram	$p\left(x_n \mid x_{n-1}\right)$
3-gram, Trigram	$p\left(x_n \mid x_{n-2}, x_{n-1}\right)$

Table 6.1 N-grams

6.3 N-gram Usage

As we mention in previous section, N-gram is a sequence of terms, with the length of N and it goes to infinite. But the system should be finite and until the trigram is enough for get the correct lexical features. 1-gram sequences are called unigrams, 2-gram sequences are called bigrams, 3-gram sequences are called trigram etc. Word boundary of interest as called ‘’current’’ and the following word boundaries called ‘’previous’’ and ‘’next’’. Six lexical features are expressed when these 3 n-gram types using together;

In other words we could re-write table according to N-gram rule with basically for easy to understand;

- Unigrams:{previous}, {current}, {next}
- Bigrams:{current-next}, {previous-current}
- Trigrams:{previous-current-next}

Model could be applied for both word with respect to letters or sentences with respect to words. In example for the word ‘’Alfabe’’ would be composed of following N-grams;

1-gram sequence	2-gram sequence	3-gram sequence
Unigram	Bigram	Trigram
space ,A,L	A-L, space-A	space-A-L

Table 6.2a N-gram Example (Word)

In example for the sentence ‘’Ali topu at’’ would be composed of following N-grams;

1-gram sequence	2-gram sequence	3-gram sequence
Unigram	Bigram	Trigram
space,Ali,topu	Ali-topu, space-Ali	space-Ali-topu

Table 6.2b N-gram Example (Sentence)

When three n-gram sequences applied in to the word together,

{space} , {A} , {L} , {A-L} , {space-A} , {space-A-L}

When three n-gram sequences applied in to the sentence together,

{space} , {Ali} , {topu} , {Ali-topu} , {space-Ali} , {space-Ali-topu}

As it seen 6 features are expressed with unigram, bigram and trigram using together.

Chapter 7

Sentence Segmentation

7.1 Introduction

As we mention before output of the speech recognition system (ASR) doesn't give any information about the sentence boundary, the output of the system is just utterance of the words. If there isn't information about sentence boundary, text file become nonsense and it's hard to understand and reading emphasis for long messages or sentences. In addition punctuations helps to spread emotions, give information about structure of sentence and intonation. However according to language type, some word's meaning or sentence's meaning could be change with respect to punctuations. For instance two examples below include and not include punctuations (sentence boundary information) for Turkish spoken language.

Sentence	Meaning
Ihtiyar (s) adamı eve götürdü.	Old man took him/her to the home
Ihtiyar adamı eve götürdü.	He/She took old man to the home.

Table 7.1 Sentence Boundary

Sentence segmentation is a subject for automatically divide input text into meaningful grammatical sentences. The way for this is labeling the boundaries in text file. In the previous chapters we mention about prosodic features, lexical features and morphologic features. These features help us to give a decision about the boundaries in text file. In sentence segmentation, the main problem is classified boundaries into two classes;

- Sentence Boundaries
- Non-sentence Boundaries

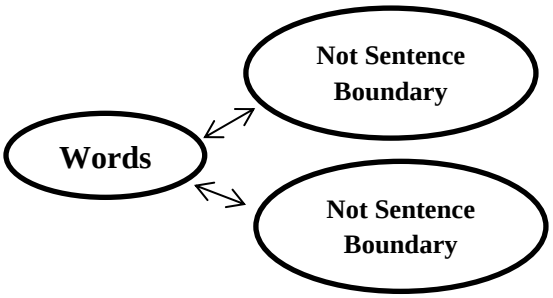


Figure 7.1 Classifications

7.2 Approach

The main problem could reduce such as a boundary classification problem. Suppose that we have N times words, boundaries and features. The goal is try to estimate of classification word between W_n and W_{n+1}. A posterior probability is needed for find the boundary segmentation with highest probability given the information which is word.

$$\arg\max_{B} P(B \mid W) \tag{7.1}$$

Where,

w_i $i = 1, \ldots, N$ Represents word sequence.

f_{ij} $i = 1, \ldots, N$ and $j = 1, \ldots, M$ Represents feature sets.

b_i $i = 1, \ldots, N$ Represents word boundaries.

y_i $i = 1, \ldots, N$ and $\in +1, -1$ Represent logaritmic operation. If $\in +1$, it is a sentence boundary (s), if else $\in -1$, it is not a sentence boundary (n).

For each word, posterior probability is again computed and compared either it is a sentence boundary or not. Probability equation can be re-written in sequence format which formula can be apply all word sequence inside;

$$P\left(b_i = y_i \mid f_i\right) \tag{7.2}$$

$$b_i = s \ (Sentence\ boundary), \quad if \quad \mathrm{P}\left(b_i = +1 \mid f_i\right) > \mathrm{P}\left(b_i = -1 \mid f_i\right) \tag{7.3}$$

$$b_i = n \ (Not\ Sentence\ boundary), \ if \quad \mathrm{P}\left(b_i = -1 \mid f_i\right) > \mathrm{P}\left(b_i = +1 \mid f_i\right) \tag{7.4}$$

7.2.1 Learning Algorithms

We mention about some approach in previous section, we have a data without information about sentence boundaries and our aim is to automatically label all this data for the each word; is it sentence boundary or not. Initially breaks should be found in sentences which we could decide all words as a break and secondly it should be decided for *s* or *n*. Hence it becomes a classification problem and helping by feature information, this classification should be done with high accuracy.

There are three different learning algorithms such as supervised, unsupervised and semi-supervised learning algorithms. Supervised learning algorithm needs as a whole labeled data for training a model and unsupervised learning algorithm is thoroughly opposite with supervised learning algorithm which doesn't need any labeled data for training a model. Because of our aim is to obtain high accuracy with cost effectivity, semi-supervised learning algorithm is chosen which, needs little labeled data in a large amount of unlabeled data set for training a model.

7.2.1.1 Data Subsets

Whole data set is divided into three sets for generate; training, development and test sets. These sets are need for training a model and which sets are optimized - computing the scores for measure the success of trained model.

All of these 3 set are prepared for different purposes. Training set is prepared for to optimize the parameters, development set is prepared for the finding the optimum iteration number which is used to perform optimized model and test set is used to measure final score.

According to our goal, classification should be done respect to speakers, words and features. We have P times different speaker, N times different words and M times different features. Initially we started to dividing data set with divide data sets into

subset with respect to P speakers, then for each subset we performed training, development and test set pairs. Training, development and test sets must be performed identically and without change order of the words. For improve the system K-Fold Cross-Validation method which is one method of Cross-Validation technique, could be used. In this method all data set divide into K subsets. X_i is selected as development set D_i where $X_i = \{ f_i, y_i \}$ and $i = 1, \ldots, K$ also the rest of sets are used for training set T_i.

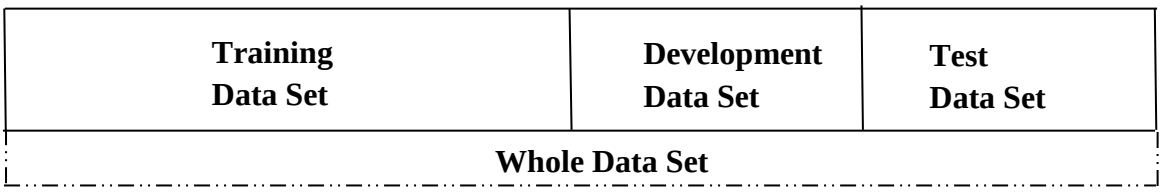
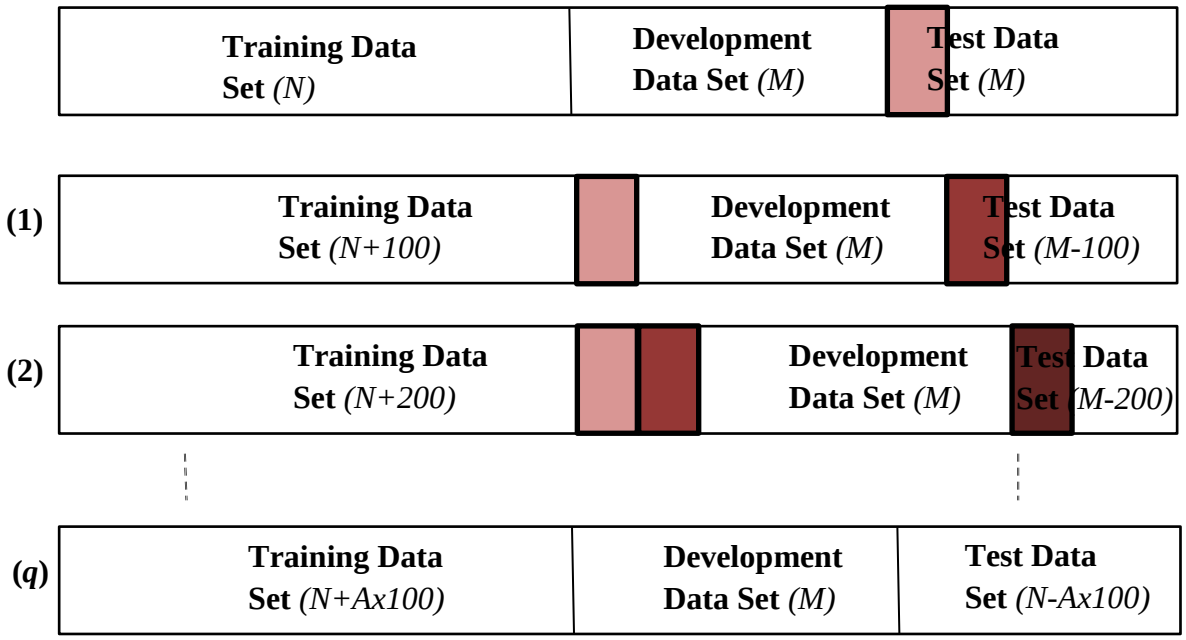


Figure 7.2 Data Sets

7.2.1.2 Self-Training and Baseline

There are several ways that aim to improve the performance of semi-supervised learning algorithm by incorporating large amounts of unlabeled data into the training data set. Self-training method is one of from this methods, it is defined as a single-view weakly supervised algorithm [35]. It is considered us a single view and also it is provided us to only one single model. In this method again optimum iteration number is found by the development data set and test data set is measured the score, after that system is retrained but this time some particular hypothesis data pick up from test data and it is add in to training set. Now training set is included mixture of manually labeled and hypothesis labeled data thus training set is became more stable. This circle or we can say loop, is continued until the reach threshold value which is shown in equation 7.5.

By the way baseline is represented the basic semi-supervised trained model and we'll use it after to compare results between self-trained model. In this model according to we mention in previous section, development data set is used for find the optimum iteration number, test data set is measured the scores and model is trained only one time.



$$\text{Stop Point (q)}: N - A \times 100 \geq \alpha \qquad (7.5)$$

Figure 7.3 Self-Trained Model

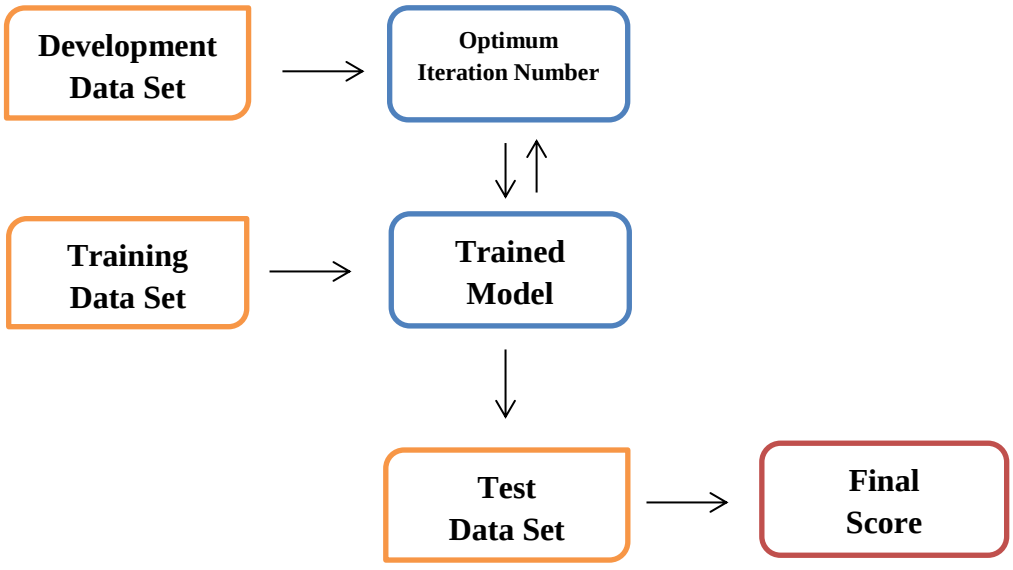


Figure 7.4 Baseline Model

7.2.2 Adaboost Algorithm

Boosting is an approach to machine learning based on the idea of creating a highly accurate prediction rule by combining many relatively weak and inaccurate rules and the AdaBoost algorithm of Freund-Schapire was the first practical boosting algorithm, and remains one of the most widely used and studied, with applications in numerous fields [11]. Main idea is improve performance of learning algorithms, it works combine with output of other weak learning algorithms and also it is compatible with C, python and java against the other boosting algorithms.

Weak learner algorithms for any distribution they can find a classifier with generalization error better than random guessing. According to researches weak learners are classified the data correctly at better than fifty percent. But if boosting algorithm is used, training data could be classified with nearly hundred percent accurate. Adaboost algorithm is focused on difficult data points which have been misclassified most by the previous weak classifier and it combine all these weak classifiers with use an optimally weighted majority vote of weak classifier. Algorithm shown as below;

$Given$ training data $(x_1, y_1),...,(x_m, y_m)$

$where\ x_i \in X$ and $y_i \in \{-1, +1\}$

Distribution initialize with $D_1(i) = \frac{1}{m}$

$For\ t = 1,...,T$:

$Train\ weak\ learner\ using\ distribution\ D_t$.

$Get\ weak\ hypothesis\ h_t : X \to \{-1, +1\}$.

$Aim : select\ h_t\ with\ low\ weighted\ eror$:

$$\varepsilon_t = \Pr_{i \sim D_t}\left[h_t(x_i) \neq y_i\right]$$

Choose $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$

Update, for $i = 1,\ldots,m$:

$$D_{t+1(i)} = \frac{D_t(i)\exp\left(-\alpha_t y_i h_t(x_i)\right)}{Z_t}$$

where Z_t *is a normalization factor* (*chosen so that* D_{t+1} *will be a distribution*).

Finally strong classifier could be defined as,

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \qquad (7.6)$$

Table 7.2 Adaboost Algorithm

Result of the finally strong classifier as shown in equation 7.6 two types of errors are occurred such as training error and generalization error. In the experiments while iteration number is increasing, train error is decreased until the reach optimum iteration. When optimum iteration is found train error becomes always zero. The training error of final classifier defined as;

$$\prod_t Z_t = \prod_t \left[2\sqrt{\varepsilon_t(1-\varepsilon_t)}\right] = \prod_t \sqrt{1-4\gamma_t^2} \le \exp\left(-2\sum_t \gamma_t^2\right) \qquad (7.7)$$

Again in the experiments while iterations number is increasing, same numbers of weak classifier are combined. After optimum iteration is found, some weak classifiers are combined more than the others and system is became complex. Hence the test error in other words generalization error stars to increase. In other words generalization error is the expected test error which could be defined as in terms of training error where *m* is the size of the sample, the VC-dimension *d* of the base classifier space and the number of rounds *T* of boosting with high probability, is at most as shown in equation 7.8.

$$\hat{\Pr}\left[H(x) \neq y\right] + \tilde{O}\left(\sqrt{\frac{Td}{m}}\right) \qquad (7.8)$$

7.3 Software Usage (Icsiboost)

Icsiboost which is an open-source tool is used for applying adaboost algorithm and it can be reached from google.code.com freely [32]. Related to icsiboost there is four input file is needed for the boosting. These are three sets which are dividing before such as training set, development set, test set, and feature (names) file which includes information type of prosodic features, morphological features and lexical features.

7.3.1 Labeling

The main idea in the system is automatically labeling the data and this process is done by helping relationship of features between sentence segmentation information. Hence obtain the data sets, words boundaries should be labeled in feature output table which is include all type of features. A new column is added into table as shown in Table 7.3 for sentence boundary information. *n* 's are added for not a sentence boundary and *s* 's are added for sentence boundary manually with respect listening audio file.

Words	**Feature Names (f_{ij})**	**Labels(b_i)**
Word *1*	Features f_{ij} where $j=1,...,M$	*n*
Word *2*	Features f_{ij} where $j=1,...,M$	*s*
⋮	⋮	⋮
Word *n*	Features f_{ij} where $j=1,...,M$	*s*

Table 7.3 Labeled Features Table

After the labeled feature table is performed, first column which includes the word's information is removed and also only interested feature's rows are selected for preparing the next step.

Than this decomposed new data is divided into three such as; training set, development set and test set finally sets are divided into subset which is mentioned

before. Training set named as “trial.data”, development set named as “trial.test” and test set named as ‘’tiral.test’’ which shown below.

Selected Features (f_{ij}) **Columns 1 to M**	**Labels(b_i)**	**Data Sets**
Features f_{ij} *Where j=1,…,M and i=1,…,q*	*s* or *n*	Training Set Size=(q,M+1) (trial.data)
Features f_{ij} *Where j=1,…,M and i=1,…,q*+p	*s* or *n*	Development Set Size=(p,M+1) (trial.dev)
Features Where j=1,…,M and i=1,…,q+p+r	*s* or *n*	Test Set Size=(r,M+1) (trial.test)

Table 7.4 Contents

7.3.2 Feature Addressing

As we mention at begin, there is four input files for start the training. Three of them is now ready which of these are training set, development set and test set. Names file which shows the system, addresses and names of the feature still needs for starting to train. There are three types of feature which we are extracted, we worked with two of them such as continues valued features, label valued features. Continues valued features are gave exactly real numbers and label valued features are gave options i.e. is it verb, adjective or noun? This information is included in names file which is named as “trial.names”. At the first line of the file shows the label information (b_i) and it defines to trainer which label is trained i.e. for sentence boundary (*s*) and not a sentence boundary (*n*). Following lines are for listing the features in order with respect to data sets. Continues valued features are defined as ‘’continues’’ and label valued features are defined with all possible outcomes.

s,n.

Feature1: ,continuous.

Feature2: ,continuous.

Feature3: f,r.

⋮

Table 7.5 Names File

7.3.3 Training The Model

Four input files are ready and now we can start to train the model with icsiboost. Before the start we should be sure of all input files are in the same directory. Model training starts with the code;

```
icsiboost –S trial –n 100 > parameters.txt
```

Where “trial” is the input file, “100” is the iteration number and “parameters.txt” is the output file. Output file includes five types of parameters. First column denotes weighted error, second column denotes theoretical error, third column denotes development error, fourth column denotes test error and fifth column denotes training error for each round according chosen iteration number. When model is trained with high iteration number, weight error will increase and either theoretical error or training error will decrease but in oppositely after a certain number of iteration, model become too complex and development-test error decrease until the reach optimum number of operation. After that errors will start to increase.

```
rnd 1: wh-err= 0.274495 th-err= 0.274495 dev= 0.032311 test= 0.041042 train= 0.034448
rnd 2: wh-err= 0.767378 th-err= 0.210641 dev= 0.031609 test= 0.030899 train= 0.024963
rnd 3: wh-err= 0.820982 th-err= 0.172933 dev= 0.031609 test= 0.030899 train= 0.024963
rnd 4: wh-err= 0.883527 th-err= 0.152791 dev= 0.041676 test= 0.033084 train= 0.024963
rnd 5: wh-err= 0.923448 th-err= 0.141094 dev= 0.030906 test= 0.029806 train= 0.020469
rnd 6: wh-err= 0.930887 th-err= 0.131343 dev= 0.030204 test= 0.029338 train= 0.020469
rnd 7: wh-err= 0.947551 th-err= 0.124454 dev= 0.030204 test= 0.029338 train= 0.020469
rnd 8: wh-err= 0.942822 th-err= 0.117338 dev= 0.029267 test= 0.029806 train= 0.020469
rnd 9: wh-err= 0.959168 th-err= 0.112547 dev= 0.033013 test= 0.029806 train= 0.018472
rnd 10: wh-err= 0.960902 th-err= 0.108146 dev= 0.033013 test= 0.029963 train=
0.017973
⋮
```

Table 7.6 Training Model and Errors

7.3.4 Testing Performance

After the model is trained, results of development set and test set appear with the command such as;

```
icsiboost -S trial -C < trial.dev > resultsdev.txt

icsiboost -S trial -C < trial.test > resulttest.txt
```

The first code generates ‘’resultsdev.txt’’ file which includes results for development set and the second code generates ‘’resulttest.txt’’ file which includes results for test set. There are four columns in this file. First and second columns are specified manually labels (b_i) such as sentence boundary (*s*) and not a sentence boundary (*n*). The sequence {0 1} represents not a sentence boundary (*n*) and the sequence {1 0} represents sentence boundary (*s*). Icsiboost is preferred represent labels with [0, 1] instead of [-1,1]. Third and fourth columns are specified results of the binary classifier. Signs are importance to make decision between the class (*s* or *n*) and magnitude of the parameters specified confidence measure scores. The sequence {-,+} represents not a sentence boundary (*n*) and the sequence {+,-} represents sentence boundary (*s*).

```
0 1 -0.601237747037 0.601237747037
0 1 -0.505622123388 0.505622123388
0 1 -0.505622123388 0.505622123388
0 1 -0.283534787175 0.283534787175
0 1 -0.899572923912 0.899572923912
0 1 -0.280858442462 0.280858442462
0 1 -0.505622123388 0.505622123388
0 1 -0.516929451725 0.516929451725
0 1 -0.601237747037 0.601237747037
0 1 -0.505622123388 0.505622123388
⋮
```

Table 7.7 Results

7.3.5 Evaluation Matrix and Methods

All of these four columns which inside of "resultdev.txt / resulttest.txt", should compared for see the agreement of results (trained model) and manually labels. Either trained model and manually labels could agree that the word is sentence boundary and not a sentence boundary or disagree. All of the four possibilities represented such as True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) which four possibility explained detail in table 7.8.

True Positive (TP): Correctly labeled sentence boundary.

True Negative (TN): Correctly labeled not a sentence boundary.

False Positive (FP): Incorrectly labeled sentence boundary, not a sentence boundary in fact.

False Negative (FN): Incorrectly labeled not a sentence boundary, sentence boundary fact.

Decision Table	{1 0}	{0 1}
{+,-}	TP	FP
{-,+}	FN	TN

Table 7.8 Decision Table

Also performance analyzing is done by F-Measure score and Nist Error rate. These two scores are measured with respect count number of TN, TP, FN and FP. In addition Precision and Recall should be evaluated too for performed the F-Measure score and Nist Error rate.

7.3.5.1 Precision

Precision which implies repeatability of system, is the ratio between correctly labeled sentence boundary (TP) and all sentence boundary decisions (TP+FP). Precision is zero when there is no correct decision and it gets the maximum value when there isn't any incorrect decision.

Precision measured as;

$$\mathrm{Pr}ecision = \frac{TP}{TP + FP} \tag{7.9}$$

7.3.5.2 Recall

Recall is the ratio between correctly labeled sentence boundary (TP) and all sentence boundaries in fact (TP+FN). Recall is zero in the case absence of correctly labeled sentence boundary and it gets one when there isn't any incorrect sentence boundary. Recall measured as;

$$\mathrm{Re}call = \frac{TP}{TP + FN} \tag{7.10}$$

7.3.5.3 True Negative Rate

True negative rate measured the performance of detect not a sentence boundary (TN). True negative rate is zero in the case absence of correctly labeled not a sentence boundary and it gets one when there isn't any incorrect labeled sentence boundary. True negative rate measured as;

$$True\ Negative\ Rate = \frac{TN}{TN + FP} \tag{7.11}$$

7.3.5.4 Accuracy

Accuracy is the ratio between all correct labeled decisions (TP+TN) over all decisions (TN+TP+FN+FP). Accuracy is zero when there isn't any correctly labeled decision and accuracy is one when all decisions are correctly labeled. Accuracy measured as;

$$Accuracy = \frac{TP + TN}{TN + TP + FN + FP} \tag{7.12}$$

7.3.5.5 F-measure Score

F-measure is the one of score which we are used for analyzing performance on the graphics. F-measure score tests accuracy in terms of harmonic mean of precision and recall. F-measure score goes to zero when the limit of sum of precision and recall goes to zero from to positive. F-measure score is one when with precision or recall is one too. F-measure score measured as;

$$F\text{-}measure\ Score = \frac{2 \times \mathrm{Pr}ecision \times \mathrm{Re}call}{\mathrm{Pr}ecision + \mathrm{Re}call} \tag{7.13}$$

7.3.5.6 Nist Error Rate

Nist error rate is the other score which we are used for analyzing performance on the graphics. Nist error rate known as National Institute of Standards and Technology error rate and it aims to get minimize score for take a meaning with good performance. Nist error rate is a ratio between all wrong decisions (FN+FP) over all of sentence boundary in fact (TP+FN). Nist error rate is zero when there is no incorrect decision and nist error rate gets maximum value when all manually labels are wrong.

$$Nist\ Error = \frac{FN + FP}{TP + FN} \tag{7.14}$$

Chapter 8

Experiments and Conclusion

8.1 Overview

We work with fifteen different experiment groups for getting the best result. Prosodic features which extracted from Praat, morphological features which extracted from TRmorph and Kemal Oflazer's tool, lexical features which extracted with perl scripts and M1 features [29] which includes in prosodic feature set is used in the experiments. Performance analyzing is done considering based on F-measure score and Nist error rate which values are described detail in previous chapter. Experiments are applied based on both multi-speaker and single-speaker. Both 10-fold cross validation and 5-fold cross validation methods are used.

Voice of America Turkish (VOA) [33] broadcast news records are chosen an input audio file for the experiments. Segment time marks (STM) files are used which extract in previous works [9] from BUSIM [34] speech group in Bogazici University. Original STM files are contained punctuation which used as a reference too. Each broadcast news record contains multi-speaker, takes duration 30 minutes, 16 kHz and 16 bit sampled ''wav'' audio file. Table 1 and Table 2 show us the detail of input whole broadcast news data together.

Type Of Word	Foreign Nouns	Foreign Proper Name	Turkish Proper Name	All types of Word
Quantity	49	762	607	11572

Table 8.1 Data Analysis (Types)

Speakers	Gender	Quantity of Words	Quantity of Sentences
Speaker 1	Male	20K	1361
Speaker 2	Female	20K	1448
Speaker 3	Male	1365	91
Speaker 4	Male	4561	248
Speaker 5	Male	1321	76
Speaker 6	Female	2482	183
All Speakers	Male + Female	49729	3407

Table 8.2 Data Analysis (Quantities)

The other important point which we care about is using open-source tools and software. Ubuntu is used as operation system which based on UNIX and whole tools are compatible with this operation system.

System builds start with re-organizing STM files with several Perl scripts and dividing data with respect to each speaker. Secondly HCopy tool which is inside in HTK, is used for extract MFCC vectors. With using MFCC vectors, the word and phoneme based CTM file are obtained by helping HVite tool which is inside in HTK tool too. Then prosodic features are extracted using Mary Harper's plugin based on Praat tool afterwards sentence boundaries are labeled manually.

Like all this steps, morphological features and lexical features are extracted. Table 3 shows us the all types of feature details and their quantities.

Feature Set	Type	Quantity
LEX	Lexical Feature Set	6
DUR+M1	Duration+M1 Prosodic Feature Set	94
DUR+F0	Duration+Pitch Prosodic Feature Set	142
M1	M1 Prosodic Feature Set	33
MORP	Morphological Feature Set	10

Table 8.3 Feature Sets, Contents and Quantities

Now we are ready to second main step of the system which mostly progressed on icsiboost. The data is divided into three sets such as training set, development set and test. Training is done by icsiboost based on self-training method and performance evaluation is measured with respect to f-measure score and nist error rate.

Speakers are grouped into three such as Speaker 1 only, Speaker 2 only and all speakers (Speaker1+Speker2+Speaker3+Speaker4+Speaker5+Speaker 6). Data size is restricted with 6000 because minimum size of a group is 20K and data should divide into three identically.

Performance of the system is criticized with table view and graphical view. Both of the view is based on f-measure score and nist error rate. Graphics shows us f-measure and nist error rate with percentage for data size 1000 words, data size 3000 words and data size 6000 words. Self-training and baseline plots are showed in same graphic and all this information extracted for both respect to maximum f-measure score and minimum nist error rate. Self-training's f-measure scores are greater than the baseline is expected and also baseline's nist error rates are greater than the baseline is expected too. Again table shows us f-measure and nist error rate values with percentage for data size 1000 words, data size 3000 words and data size 6000 words. All this information extracted for both respect to maximum f-measure score and minimum nist error rate. Self-training's f-measure scores are greater than the baseline is expected and also baseline's nist error rates are greater than the baseline is expected too. Self-labeled (words) row is explained us to how many sample is added from test data to training data with decided labels from previous model to reach maximized model. I.e if we have 1000 word data size and if we reach the maximized model in 3500 word, initially there is 1000 word in training set and 2500 word is adding which is decided in previous model.

8.2 Experiments

15 different experiments are done by several ways, as shown below.

8.2.1 Experimental Group 1

Speaker 1 data, lexical feature set and 10-fold cross validation method is used. In self-training, first 100 samples which gets the maximum confidence score is adding into training data set with decided labels from previous model, for each iteration and this samples are took out from the test set.

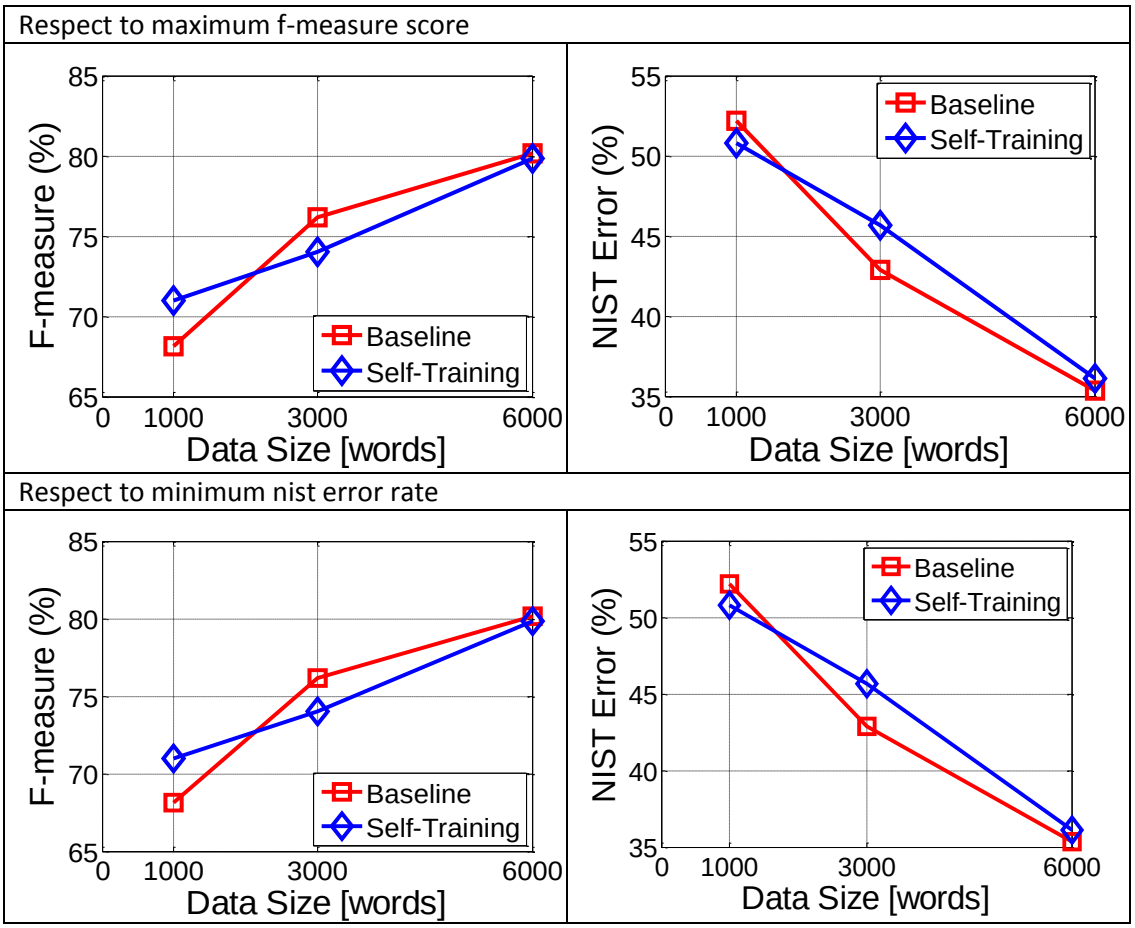


Figure 8.1 Graphical view of experiment 1 (Speaker 1 with Lexical Features)

	Maximum F-measure		Minimum Nist	
Man. Labeled Data=1000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	68.1200	52.1680	68.1200	52.1680
Self-Training	70.9630	50.7970	70.9630	50.7970
Self-Labeled (words)	13400	13400	13400	13400
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=3000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	76.1730	42.8690	76.1730	42.8690
Self-Training	74.0030	45.6770	74.0030	45.6770
Self-Labeled (words)	11400	11400	11400	11400
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=6000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	80.1420	35.3780	80.1420	35.3780
Self-Training	79.8600	36.0970	79.8600	36.0970
Self-Labeled (words)	8400	8400	8400	8400

Table 8.4 Table view of experiment 1 (Speaker 1 with Lexical Features)

8.2.2 Experimental Group 2

Speaker 2 data, lexical feature set and 10-fold cross validation method is used. In self-training, first 100 samples which gets the maximum confidence score is adding into training data set with decided labels from previous model, for each iteration and this samples are took out from the test set.

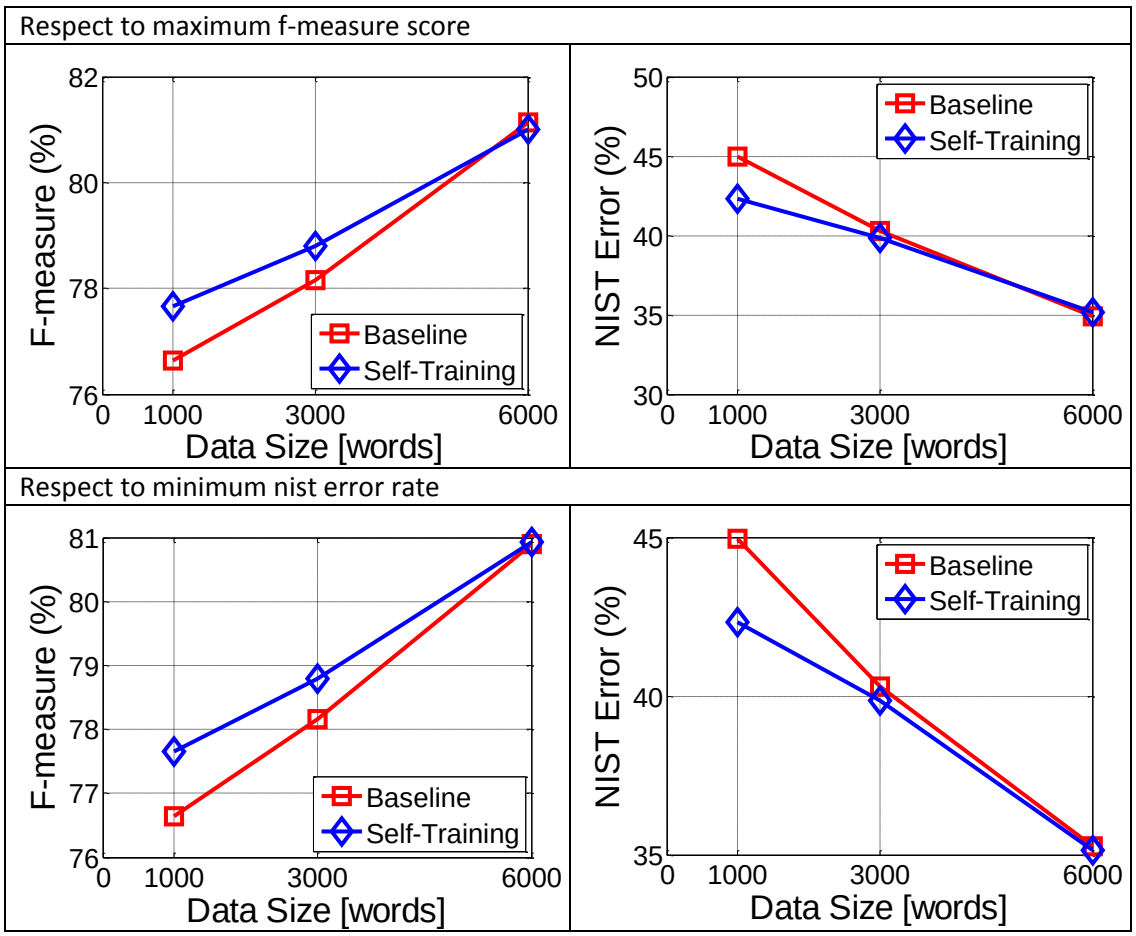


Figure 8.2 Graphical view of experiment 2 (Speaker 2 with Lexical Features)

	Maximum F-measure		Minimum Nist	
Man. Labeled Data=1000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	76.6400	44.9550	76.6400	44.9550
Self-Training	77.6520	42.3190	77.6520	42.3190
Self-Labeled (words)	13900	13900	13900	13900
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=3000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	78.1640	40.3050	78.1640	40.3050
Self-Training	78.7980	39.8630	78.7980	39.8630
Self-Labeled (words)	11900	11900	11900	11900
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=6000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	81.1410	34.9320	80.8950	35.2730
Self-Training	80.9950	35.1460	80.9340	35.1410
Self-Labeled (words)	8900	8900	8900	8900

Table 8.5 Table view of experiment 2 (Speaker 2 with Lexical Features)

8.2.3 Experimental Group 3

All speaker data, lexical feature set and 10-fold cross validation method is used. In self-training, first 100 samples which gets the maximum confidence score is adding into training data set with decided labels from previous model, for each iteration and this samples are took out from the test set.

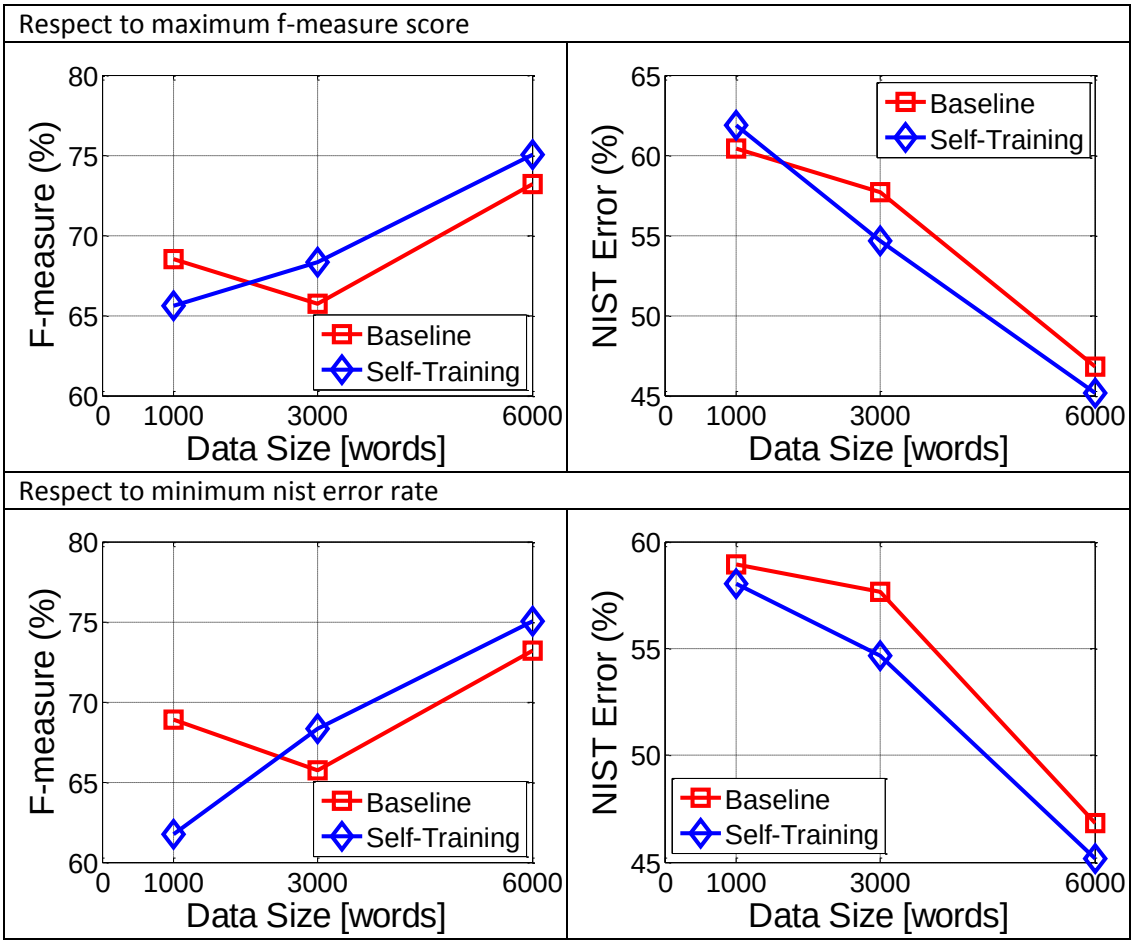


Figure 8.3 Graphical view of experiment 3 (All speakers with Lexical Features)

	Maximum F-measure		Minimum Nist	
Man. Labeled Data=1000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	68.5360	60.3830	68.9110	58.9310
Self-Training	65.6040	61.8470	61.7600	58.0510
Self-Labeled (words)	43100	43100	42700	42700
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=3000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	65.7170	57.6620	65.7170	57.6620
Self-Training	68.3330	54.6520	68.3330	54.6520
Self-Labeled (words)	39800	39800	39800	39800
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=6000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	73.1760	46.8280	73.1760	46.8280
Self-Training	75.0330	45.1650	75.0330	45.1650
Self-Labeled (words)	36500	36500	36500	36500

Table 8.6 Table view of experiment 3 (All speakers with Lexical Features)

8.2.4 Experimental Group 4

Speaker 1 data, DUR+M1 prosodic feature set and 10-fold cross validation method is used. In self-training, first 100 samples which gets the maximum confidence score is adding into training data set with decided labels from previous model, for each iteration and this samples are took out from the test set.

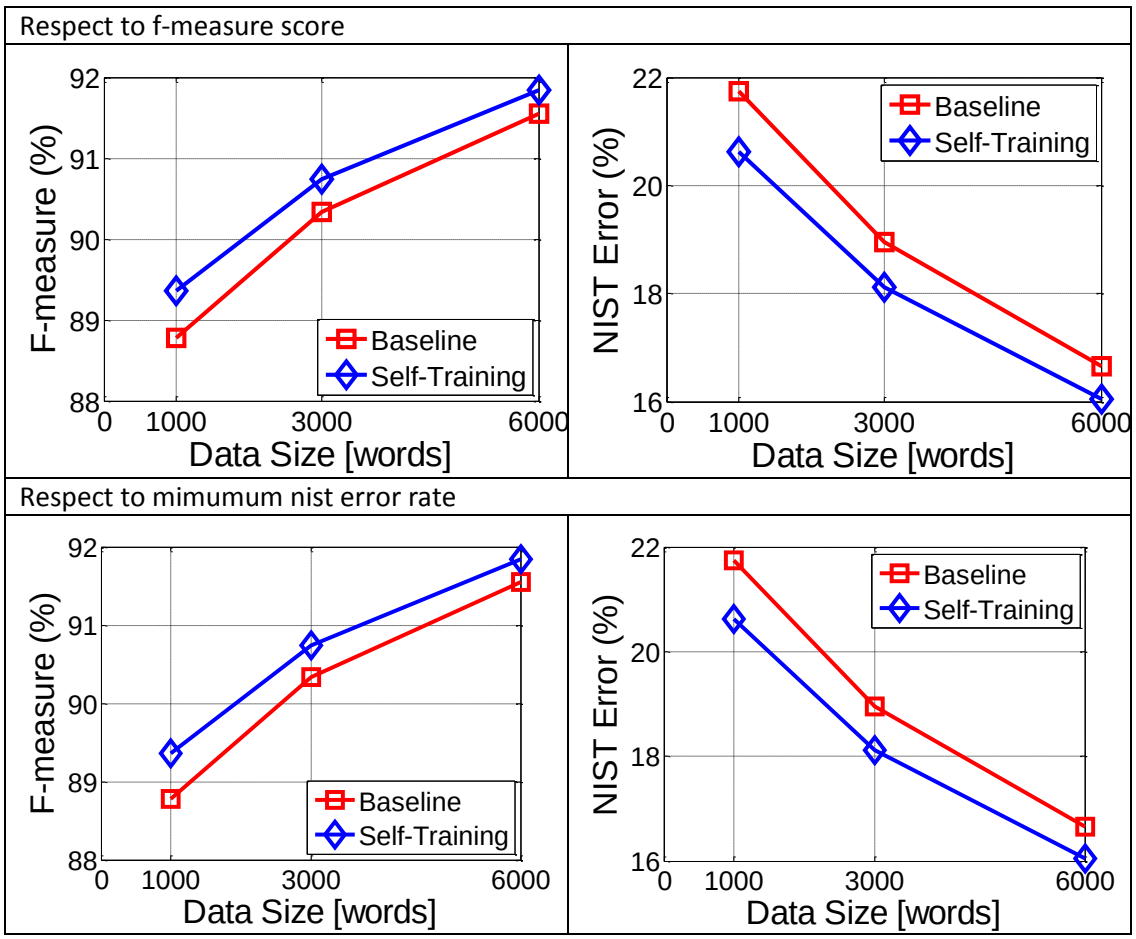


Figure 8.4 Graphical view of experiment 4 (Speaker 1 with DUR+M1 Prosodic Feature Set)

	Maximum F-measure		Minimum Nist	
Man. Labeled Data=1000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	88.7800	21.7370	88.7800	21.7370
Self-Training	89.3570	20.6210	89.3570	20.6210
Self-Labeled (words)	3300	3300	3300	3300
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=3000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	90.3350	18.9490	90.3350	18.9490
Self-Training	90.7460	18.1140	90.7460	18.1140
Self-Labeled (words)	1600	1600	1600	1600
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=6000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	91.5480	16.6520	91.5480	16.6520
Self-Training	91.8460	16.0420	91.8460	16.0420
Self-Labeled (words)	1100	1100	1100	1100

Table 8.7 Table view of experiment 4 (Speaker 1 with DUR+M1 Prosodic Feature Set)

8.2.5 Experimental Group 5

Speaker 2 data, DUR+M1 prosodic feature set and 10-fold cross validation method is used. In self-training, first 100 samples which gets the maximum confidence score is adding into training data set with decided labels from previous model, for each iteration and this samples are took out from the test set.

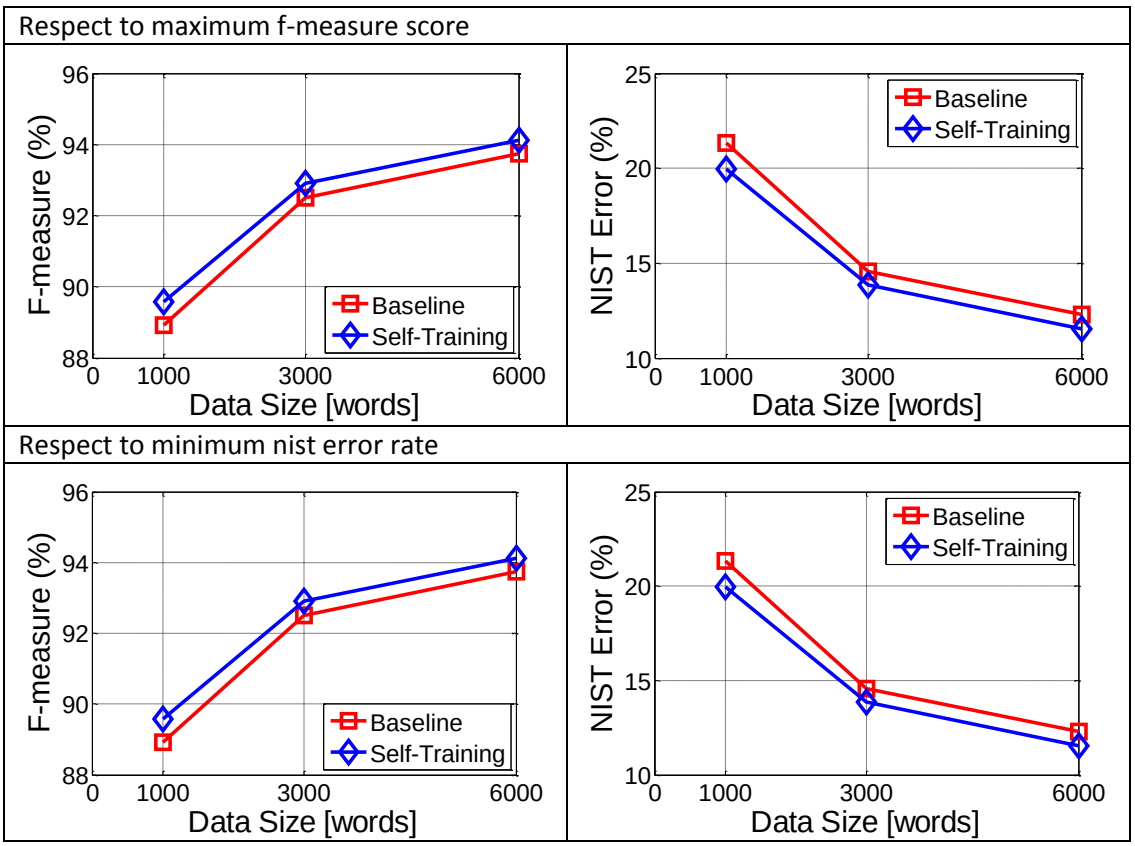


Figure 8.5 Graphical view of experiment 5 (Speaker 2 with DUR+M1 Prosodic Feature Set)

	Maximum F-measure		Minimum Nist	
Man. Labeled Data=1000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	88.9300	21.3150	88.9300	21.3150
Self-Training	89.5710	19.9500	89.5710	19.9500
Self-Labeled (words)	1900	1900	1900	1900
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=3000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	92.5170	14.5460	92.5170	14.5460
Self-Training	92.9050	13.8580	92.9050	13.8580
Self-Labeled (words)	4100	4100	4100	4100
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=6000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	93.7510	12.2820	93.7510	12.2820
Self-Training	94.1240	11.5330	94.1240	11.5330
Self-Labeled (words)	1800	1800	1800	1800

Table 8.8 Table view of experiment 5 (Speaker 2 with DUR+M1 Prosodic Feature Set)

8.2.6 Experimental Group 6

All speaker data, DUR+M1 prosodic feature set and 5-fold cross validation method is used. In self-training, first 100 samples which gets the maximum confidence score is adding into training data set with decided labels from previous model, for each iteration and this samples are took out from the test set.

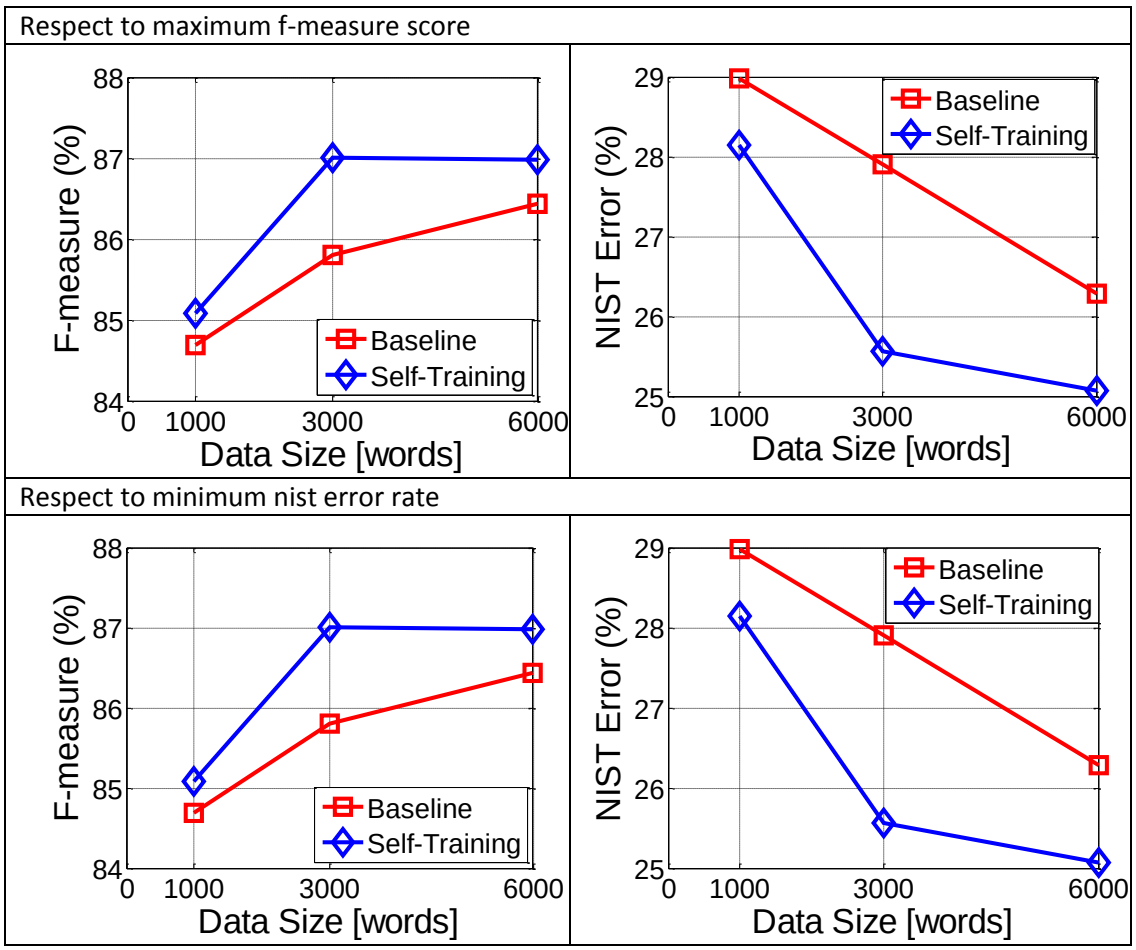


Figure 8.6 Graphical view of experiment 6 (All speakers with DUR+M1 Prosodic Feature Set)

	Maximum F-measure		Minimum Nist	
Man. Labeled Data=1000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	84.6940	28.9860	84.6940	28.9860
Self-Training	85.0860	28.1480	85.0860	28.1480
Self-Labeled (words)	1200	1200	1200	1200
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=3000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	85.8060	30.9540	85.8060	30.9540
Self-Training	87.0100	27.9000	87.0100	27.9000
Self-Labeled (words)	11000	11000	11000	11000
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=6000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	86.4360	27.1320	86.4360	27.1320
Self-Training	86.9820	26.2860	86.9820	26.2860
Self-Labeled (words)	100	100	100	100

Table 8.9 Table view of experiment 6 (All speakers with DUR+M1 Prosodic Feature Set)

8.2.7 Experimental Group 7

Speaker 1 data, DUR+F0 prosodic feature set and 10-fold cross validation method is used. In self-training, first 100 samples which gets the maximum confidence score is adding into training data set with decided labels from previous model, for each iteration and this samples are took out from the test set.

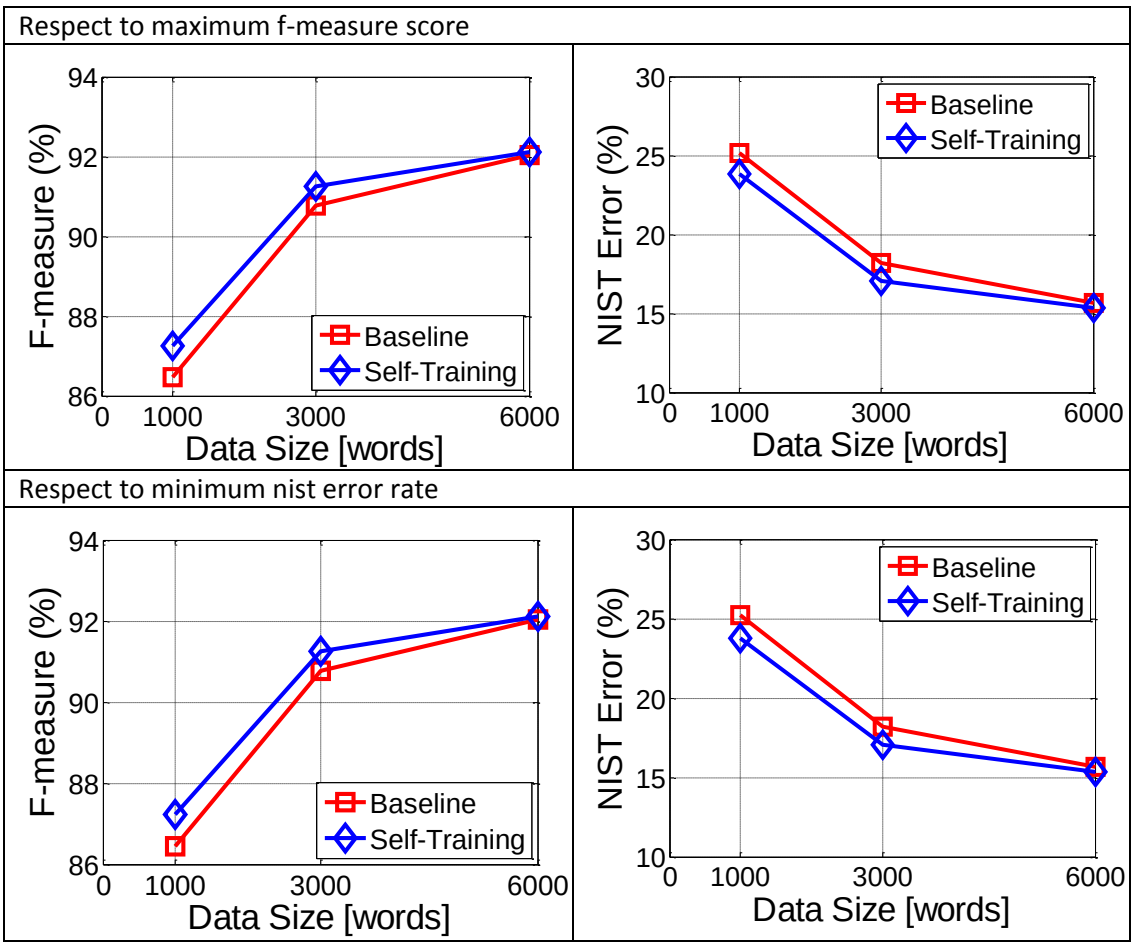


Figure 8.7 Graphical view of experiment 7 (Speaker 1 with DUR+F0 Prosodic Feature Set)

	Maximum F-measure		Minimum Nist	
Man. Labeled Data=1000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	86.4720	25.1710	86.4530	25.1990
Self-Training	87.2620	23.8320	87.2180	23.7910
Self-Labeled (words)	4300	4300	3600	3600
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=3000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	90.7830	18.2080	90.7830	18.2080
Self-Training	91.2600	17.0790	91.2600	17.0790
Self-Labeled (words)	5800	5800	5800	5800
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=6000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	92.0390	15.6570	92.0390	15.6570
Self-Training	92.1260	15.3670	92.1260	15.3670
Self-Labeled (words)	1700	1700	1700	1700

Table 8.10 Table view of experiment 7 (Speaker 1 with DUR+F0 Prosodic Feature Set)

8.2.8 Experimental Group 8

Speaker 2 data, DUR+F0 prosodic feature set and 10-fold cross validation method is used. In self-training, first 100 samples which gets the maximum confidence score is adding into training data set with decided labels from previous model, for each iteration and this samples are took out from the test set.

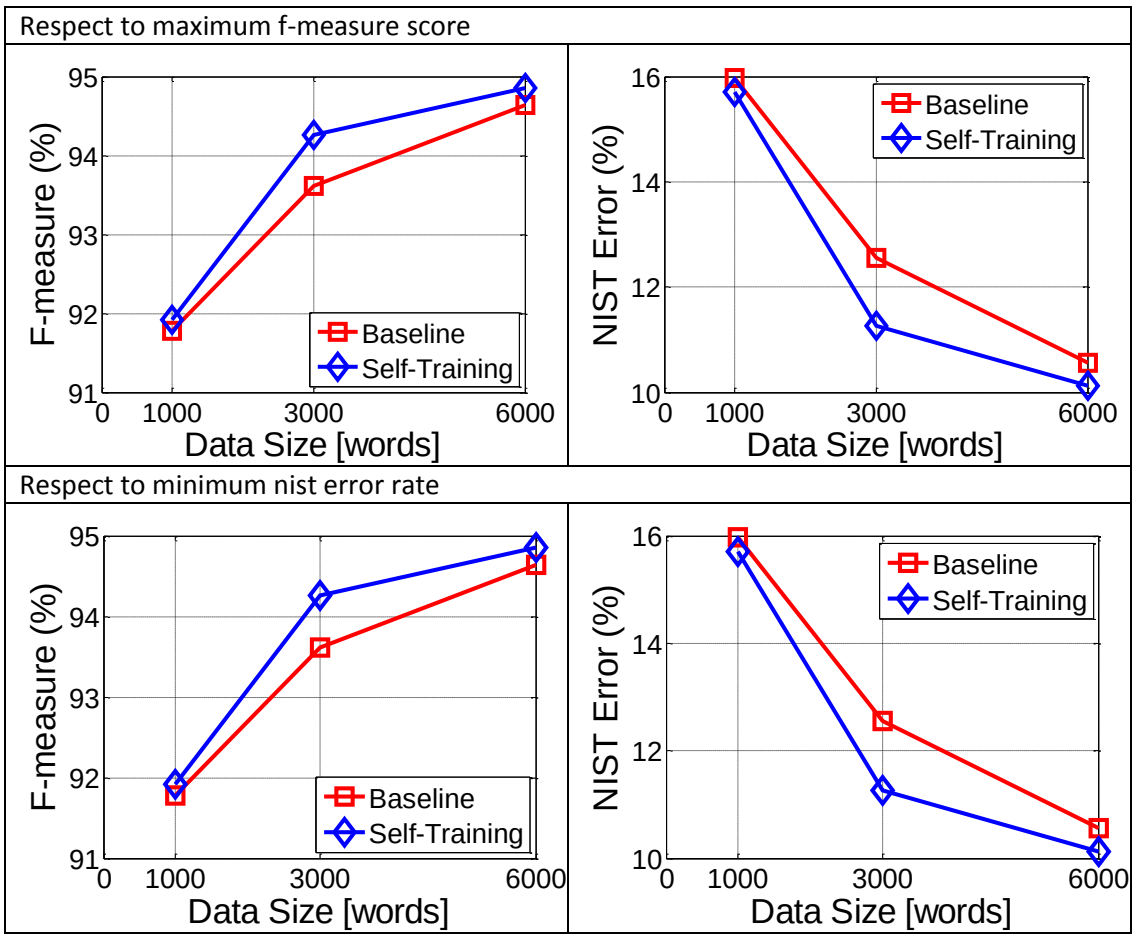


Figure 8.8 Graphical view of experiment 8 (Speaker 2 with DUR+F0) Prosodic Feature Set)

	Maximum F-measure		Minimum Nist	
Man. Labeled Data=1000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	91.7840	15.9650	91.7840	15.9650
Self-Training	91.9240	15.6990	91.9240	15.6990
Self-Labeled (words)	4000	4000	4000	4000
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=3000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	93.6150	12.5500	93.6150	12.5500
Self-Training	94.2590	11.2670	94.2590	11.2670
Self-Labeled (words)	2000	2000	2000	2000
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=6000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	94.6360	10.5590	94.6360	10.5590
Self-Training	94.8600	10.1300	94.8600	10.1300
Self-Labeled (words)	1400	1400	1400	1400

Table 8.11 Table view of experiment 8 (Speaker 2 with DUR+F0 Prosodic Feature Set)

8.2.9 Experimental Group 9

All speaker data, DUR+F0 prosodic feature set and 5-fold cross validation method is used. In self-training, first 100 samples which gets the maximum confidence score is adding into training data set with decided labels from previous model, for each iteration and this samples are took out from the test set.

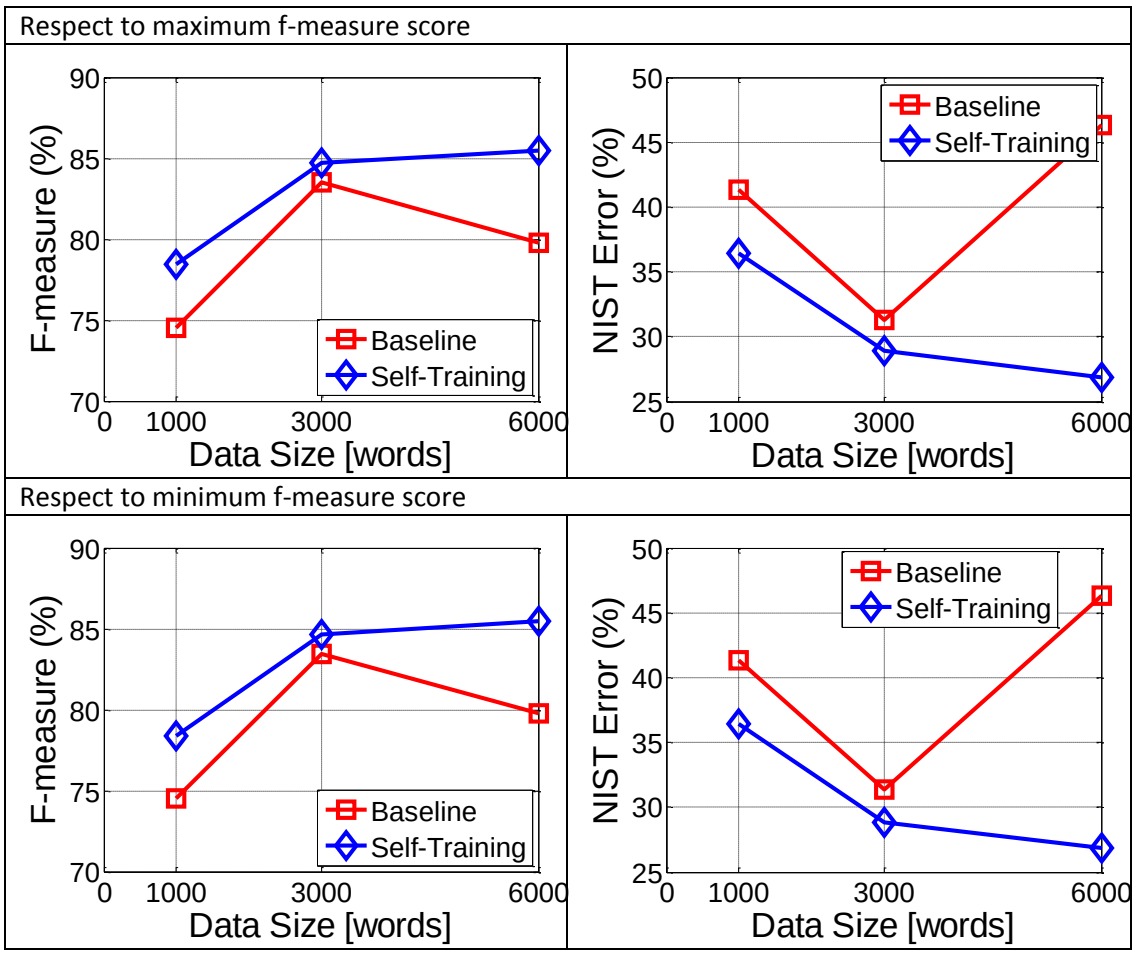


Figure 8.9 Graphical view of experiment 9 (All speakers with DUR+F0) Prosodic Feature Set)

	Maximum F-measure		Minimum Nist	
Man. Labeled Data=1000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	74.5440	41.3540	74.5280	41.3740
Self-Training	78.4260	36.4480	78.3740	36.4300
Self-Labeled (words)	5800	5800	4700	4700
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=3000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	83.4840	31.3220	83.4660	31.3720
Self-Training	84.7040	28.8920	84.6400	28.8200
Self-Labeled (words)	5100	5100	7200	7200
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=6000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	79.7540	46.2820	79.7540	46.2820
Self-Training	85.5020	26.8520	85.5020	26.8520
Self-Labeled (words)	1700	1700	1700	1700

Table 8.12 Table view of experiment 9 (All speakers with DUR+F0 Prosodic Feature Set)

8.2.10 Experimental Group 10

Speaker 1 data, M1 prosodic feature set and 10-fold cross validation method is used. In self-training, first 100 samples which gets the maximum confidence score is adding into training data set with decided labels from previous model, for each iteration and this samples are took out from the test set.

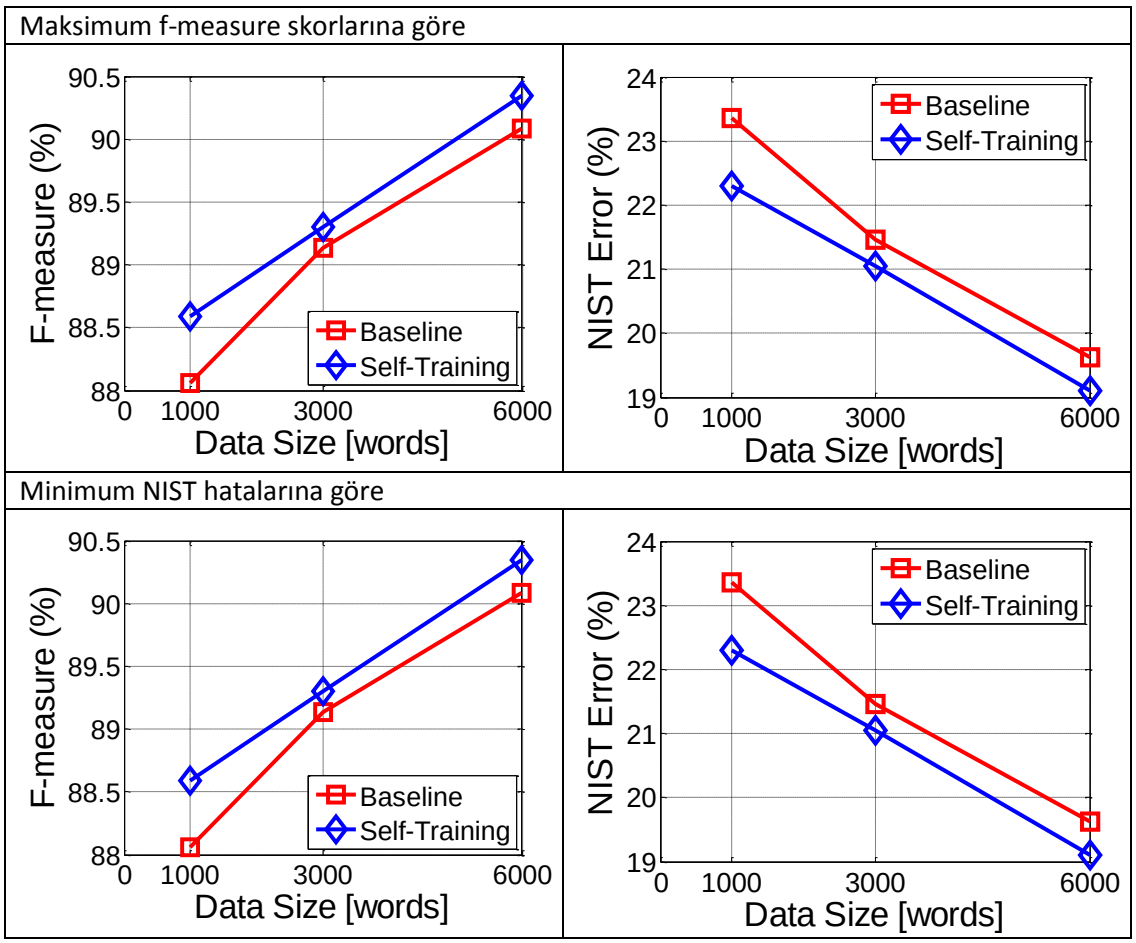


Figure 8.10 Graphical view of experiment 10 (Speaker 1 with M1 Prosodic Feature Set)

	Maximum F-measure		Minimum Nist	
Man. Labeled Data=1000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	88.0620	23.3650	88.0620	23.3650
Self-Training	88.5860	22.2980	88.5860	22.2980
Self-Labeled (words)	2600	2600	2600	2600
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=3000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	89.1390	21.4630	89.1390	21.4630
Self-Training	89.3010	21.0440	89.3010	21.0440
Self-Labeled (words)	1300	1300	1300	1300
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=6000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	90.0810	19.6230	90.0810	19.6230
Self-Training	90.3460	19.1040	90.3460	19.1040
Self-Labeled (words)	700	700	700	700

Table 8.13 Table view of experiment 10 (Speaker 1 with M1 Prosodic Feature Set)

8.2.11 Experimental Group 11

Speaker 2 data, M1 prosodic feature set and 10-fold cross validation method is used. In self-training, first 100 samples which gets the maximum confidence score is adding into training data set with decided labels from previous model, for each iteration and this samples are took out from the test set.

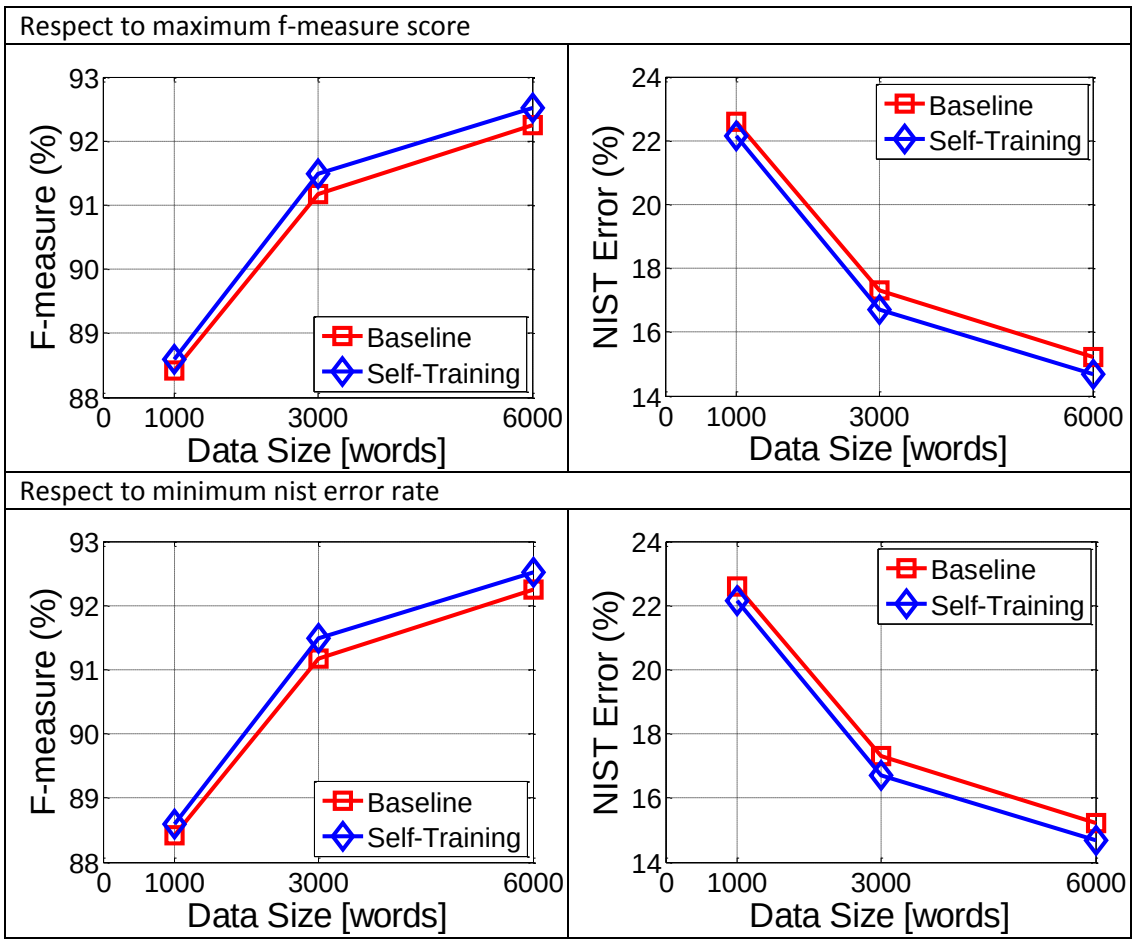


Figure 8.11 Graphical view of experiment 11 (Speaker 2 with M1 Prosodic Feature Set)

	Maximum F-measure		Minimum Nist	
Man. Labeled Data=1000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	88.4160	22.5800	88.4160	22.5800
Self-Training	88.5880	22.1580	88.5880	22.1580
Self-Labeled (words)	1900	1900	1900	1900
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=3000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	91.1740	17.3200	91.1740	17.3200
Self-Training	91.4890	16.6940	91.4890	16.6940
Self-Labeled (words)	300	300	300	300
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=6000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	92.2510	15.2320	92.2510	15.2320
Self-Training	92.5230	14.6770	92.5230	14.6770
Self-Labeled (words)	300	300	300	300

Table 8.14 Table view of experiment 11 (Speaker 2 with M1 Prosodic Feature Set)

8.2.12 Experimental Group 12

All speaker data, M1 prosodic feature set and 5-fold cross validation method is used. In self-training, first 100 samples which gets the maximum confidence score is adding into training data set with decided labels from previous model, for each iteration and this samples are took out from the test set.

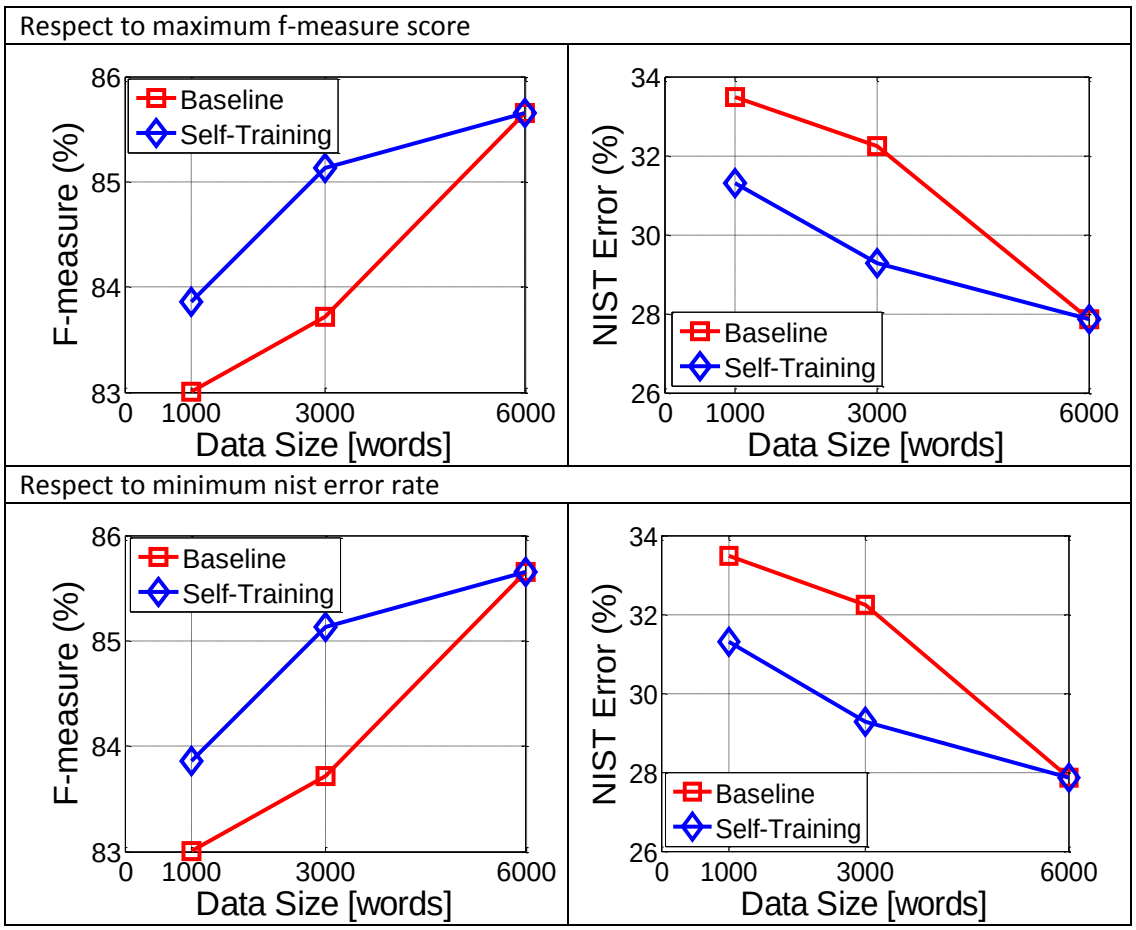


Figure 8.12 Graphical view of experiment 12 (All speakers with M1 Prosodic Feature Set)

	Maximum F-measure		Minimum Nist	
Man. Labeled Data=1000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	83.0020	33.4800	83.0020	33.4800
Self-Training	83.8600	31.3080	83.8600	31.3080
Self-Labeled (words)	7600	7600	7600	7600
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=3000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	83.7140	32.2460	83.7140	32.2460
Self-Training	85.1300	29.2800	85.1300	29.2800
Self-Labeled (words)	2500	2500	2500	2500
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=6000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	85.6500	27.8680	85.6500	27.8680
Self-Training	85.6500	27.8680	85.6500	27.8680
Self-Labeled (words)	0	0	0	0

Table 8.15 Table view of experiment 12 (All speakers with M1 Prosodic Feature Set)

8.2.13 Experimental Group 13

Speaker 1 data, morphological feature set and 5-fold cross validation method is used. In self-training, first 100 samples which gets the maximum confidence score is adding into training data set with decided labels from previous model, for each iteration and this samples are took out from the test set.

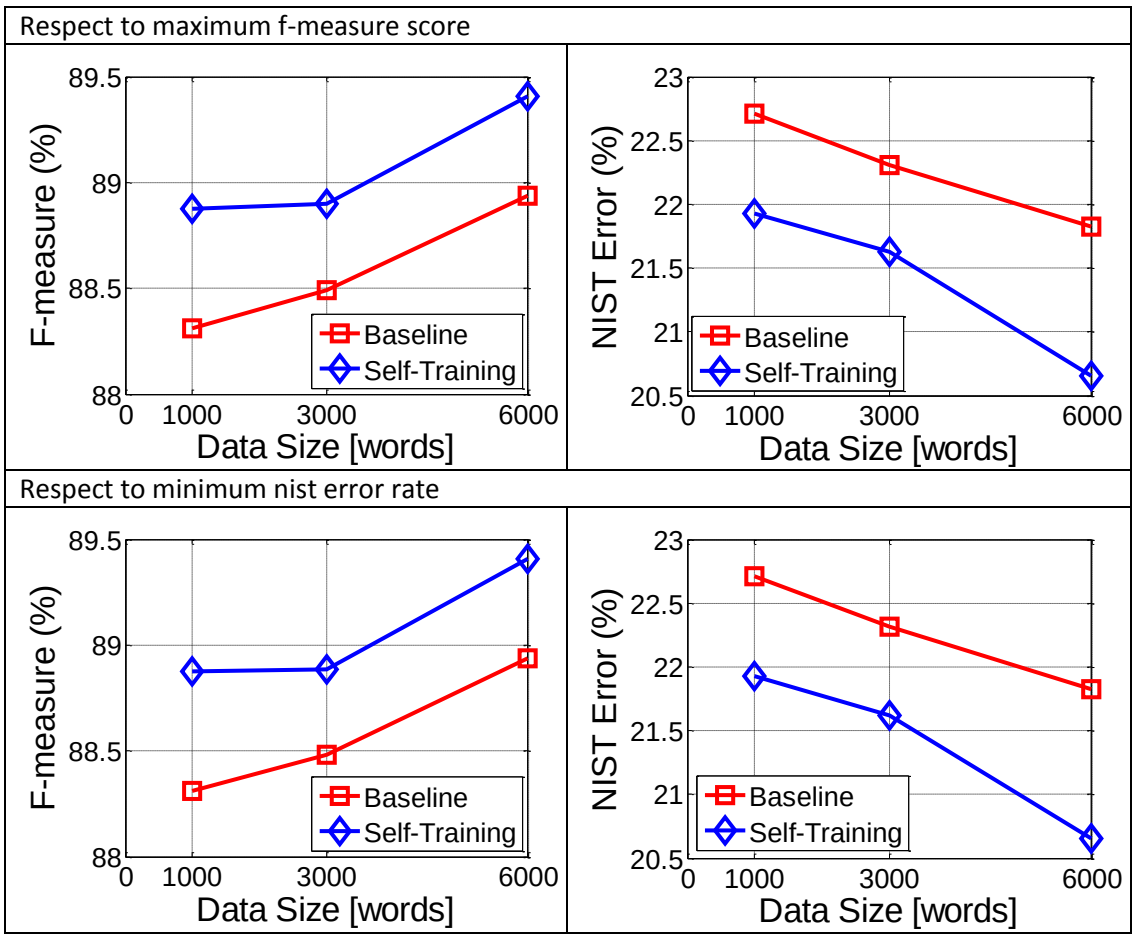


Figure 8.13 Graphical view of experiment 13 (Speaker 1 with Morphological Feature Set)

	Maximum F-measure		Minimum Nist	
Man. Labeled Data=1000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	88.3100	22.7120	88.3100	22.7120
Self-Training	88.8740	21.9280	88.8740	21.9280
Self-Labeled (words)	7300	7300	7300	7300
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=3000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	88.4920	22.3100	88.4840	22.3120
Self-Training	88.8980	21.6240	88.8840	21.6220
Self-Labeled (words)	1000	1000	400	400
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=6000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	88.9360	21.8260	88.9360	21.8260
Self-Training	89.4080	20.6580	89.4080	20.6580
Self-Labeled (words)	4500	4500	4500	4500

Table 8.16 Table view of experiment 13 (Speaker 1 with Morphological Feature Set)

8.2.14 Experimental Group 14

Speaker 2 data, morphological feature set and 10-fold cross validation method is used. In self-training, first 100 samples which gets the maximum confidence score is adding into training data set with decided labels from previous model, for each iteration and this samples are took out from the test set.

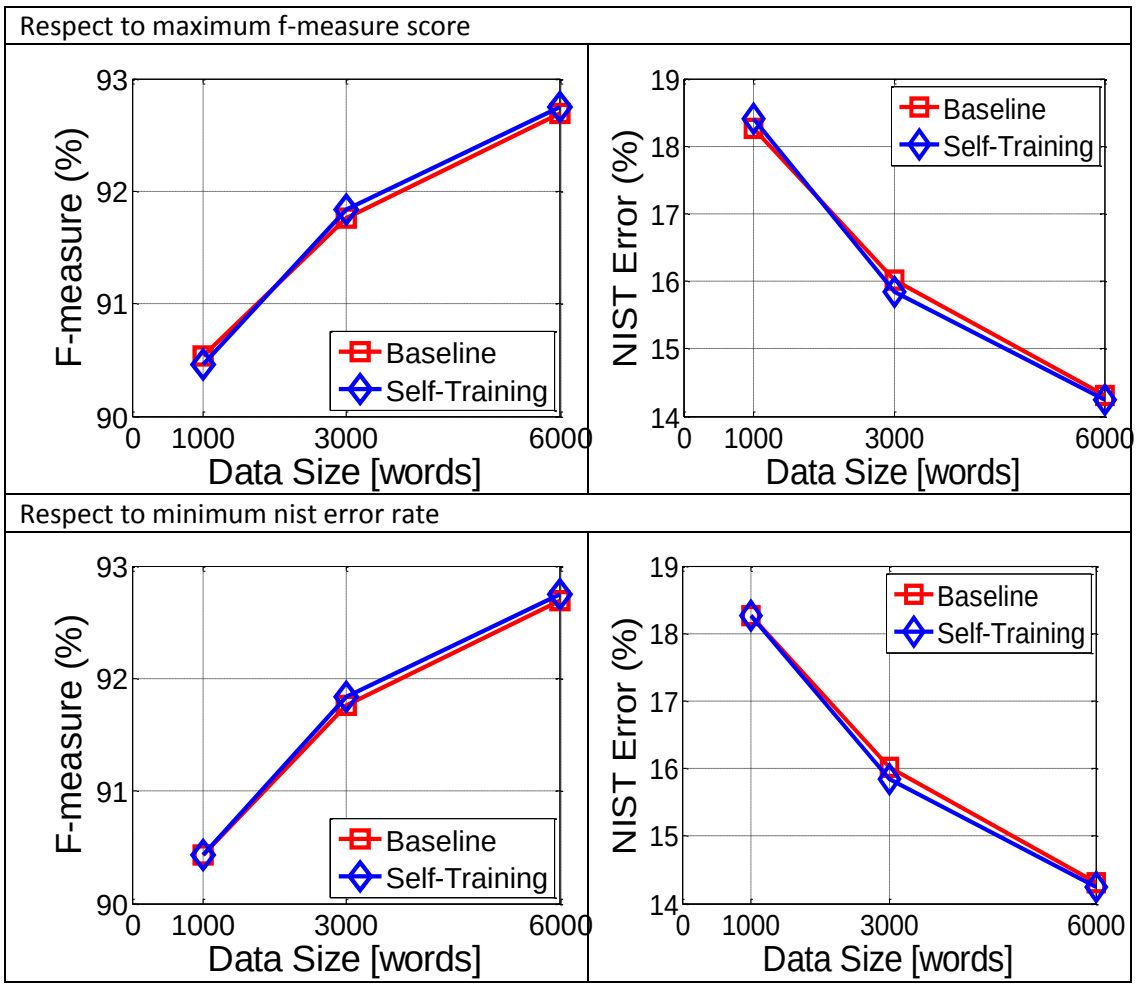


Figure 8.14 Graphical view of experiment 14 (Speaker 2 with Morphological Feature Set)

	Maximum F-measure		Minimum Nist	
Man. Labeled Data=1000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	90.5380	18.2580	90.4330	18.2680
Self-Training	90.4650	18.4100	90.4330	18.2680
Self-Labeled (words)	5400	5400	0	0
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=3000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	91.7590	16.0140	91.7590	16.0140
Self-Training	91.8380	15.8480	91.8380	15.8480
Self-Labeled (words)	100	100	100	100
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=6000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	92.6950	14.3010	92.6950	14.3010
Self-Training	92.7520	14.2450	92.7520	14.2450
Self-Labeled (words)	100	100	100	100

Table 8.17 Table view of experiment 14 (Speaker 2 with Morphological Feature Set)

8.2.15 Experimental Group 15

All speaker data, morphological feature set and 5-fold cross validation method is used. In self-training, first 100 samples which gets the maximum confidence score is adding into training data set with decided labels from previous model, for each iteration and this samples are took out from the test set.

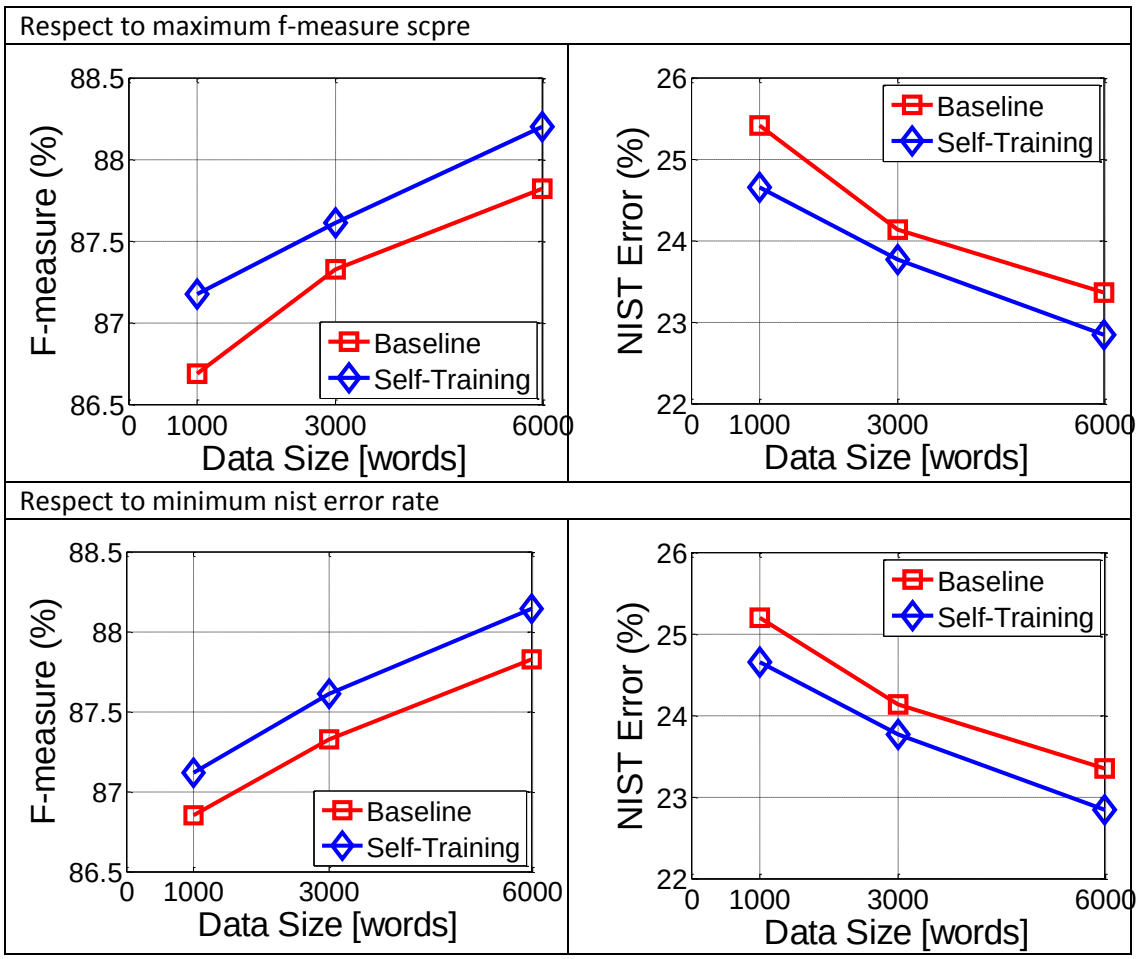


Figure 15 Graphical view of experiment 15 (All speakers with Morphological Feature Set)

	Maximum F-measure		Minimum Nist	
Man. Labeled Data=1000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	86.6860	25.4120	86.8500	25.1920
Self-Training	87.1720	24.6580	87.1200	24.6540
Self-Labeled (words)	100	100	7500	7500
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=3000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	87.3240	24.1300	87.3240	24.1300
Self-Training	87.6120	23.7660	87.6120	23.7660
Self-Labeled (words)	800	800	800	800
	Maximum F-measure		**Minimum Nist**	
Man. Labeled Data=6000	**F-measure (%)**	**NIST (%)**	**F-measure (%)**	**NIST (%)**
Baseline	87.8180	23.3600	87.8240	23.3500
Self-Training	88.2020	22.8400	88.1440	22.8360
Self-Labeled (words)	1500	1500	100	100

Table 8.18 Table view of experiment 15 (All speakers with Morphological Feature Set)

8.3 Conclusion

To conclude, we have to try increase feature types to solve sentence segmentation problem. In addition boosting algorithm is used to get a strong learner with respect to prosodic features, morphological features and lexical features; we worked for increase the scores according to previous works. For receiving much more scores, co-training algorithm can be used which is multi view learning algorithm against to single view self-training algorithm. All of the system tools which we used, are open source and we modified all these tools in to Turkish Spoken language. It is also mean that these tools could be used for all the other language with small effort. We have observed that morphological feature set, M1 prosodic feature set and DUR+F0 prosodic feature set are performed maximum performance and system is given similar response both single speaker and multi speaker. This work could be considered as the first step of further ASR applications such as topic segmentation, sentence translation and summarization.

References

[1] E. Shriberg, A. Stolcke, D. Hakkani-Tur, G. Tur, *Prosody-based automatic segmentation of speech into sentences and topics*, 2000.

[2] Y. Gotoh and S. Renals, *Sentence boundary detection in broadcast speech transcripts*, 2000.

[3] J. Huang, G. Zweig, *Maximum entropy model for punctuation annotation from speech*, 2002.

[4] Y. Liu, E. Shriberg, A. Stolcke, B. Peskin, J. Ang, D. Hillard, M. Ostendorf, M. Tomalin, P. Woodland, and M. Harper, *Structural metadata research in the EARS program*, 2000.

[5] Zimmermann M., Hakkani-Tür D., Fung J., Mirghafori N., Gottlieb L., Liu Y. and Shriberg E., *The ICSI+ multi-lingual sentence segmentation*, 2006.

[6] Dalva D., D.Revidi I., Guz U., Gurkan H., *Extractionf and Comparison of Various Prosodic Feature Sets on Sentence Segmentation Task for Turkish Broadcast News Data*, 2014.

[7] Coltekin C., *A Freely Available Morphological Analyzer For Turkish*, 2010.

[8] Tur G., Oflazer Kemal A *statistical information extraction system for*, 2000.

[9] Dalva D., *Automatic Speech Recognition System for Turkish Spoken Language, MSc. Thesis*, June 2012.

[10] Huang Z., Chen L., Harper M.P., *Purdue Prosodic Feature Extraction Tool on Praat*, 2006.

[11] Schapire R.E., *The Boosting Approach to Machine Learning An Overview*, 2001.

[12] Turk Dil Kurumu,
http://www.tdk.gov.tr

[13] Turkish Alphabet
http://en.wikipedia.org/wiki/Turkish_alphabet

[14] Papoulis A., Pillai S. U, *Probability, Random Variables and Stochastic Processes*, 4th edition, 2002.

[15] Young S., Everman G., Gales M., Hain T., Kershaw D., Liu X., Moore G., Odell J., Ollason D., Povey D., Valtchev V. Woodland P., *The HTK Book*, 2009.

[16] Glass J., *A Brief Introduction to Automatic Speech Recognition*, 2007.

[17] Guz U., Favre B., Tur D., *Tur G., Generative and Discriminative Methods Using Morphological Information for Sentence Segmentation of Turkish*, 2009.

[18] Oflazer K., *Two-Level Description of Turkish Morphology*, 1993.

[19] Ritchie G., *Languages Genereated by Two-Level Morphological Rules*, 1992.

[20] R.Beesley K., Karttunen L., *Two-Level Rule Compilter*, 2003.

[21]Oflazer K., *Computational Morphology*, 2009.

[22] Koskenniemi K. , *Two Level Morphology: A General Computational Model for Word-Form Recgonition and Production*, 1983.

[23] Foma
http://code.google.com/p/foma/

[24]Trmorph
http://code.google.com/p/dkpro-core-gpl/

[25] Yuret D., Yatbaz M. A., *Unsupervised Morphological Disambiguation using Statistical Language Models*, 2009.

[26] Nather P., *N-gram Based Text Categorization*, 2005.

[27] Brill E., *Discovering the Lexical Features of a Language*, 1991.

[28] J.Mooney R., *Natural Language Processing: N-Gram Language Models*.

[29] Guz U., Cuendet S., Tur D., Tur G., *Multi-View Semi-Supervised Learning for Dialog Act Segmentation of Speech*, 2010.

[30] Guz U., Gurkan H., Dalva D., *Comparing Performances of Various Prosodic Feature Sets on Sentence Segmentation Task for Turkish Broadcast News*, 2013.

[31] Matas J., Sochman J., *Adaboost*.

[32] Icsiboost
http://code.google.com/p/icsiboost

[33] Amerika'nin Sesi
http://www.voanews.com/turkish

[34] BUSIM
http://www.ee.boun.edu.tr/busim

[35] Mihalcea R., *Co-training and Self-training for Word Sense Disambiguation*, 2004.

[36] Coltekin C., *Trmorph: A morphological analyzer for Turkish (User Manual)*. 2013.

Appendix A Prosodic Feature List

A.1 Basic Features

A.1.1 Base Features

- WAV : Location information of current audio file.
- SPK_ID: Identification information of speaker.
- SPK_GEN: Gender information of speaker.

A.1.2 Duration Features

- WORD: Recent word boundary.
- WORD_START: Starting time of the recent word boundary.
- WORD_END: Ending time of the recent word boundary.
- FWORD: Next word boundary.
- FWORD_START: Starting of the next word boundary.
- FWORD_END: Ending time of the next word boundary.
- PAUSE_START: Starting time of the pause boundary.
- PAUSE_END: Ending time of the pause.
- PAUSE_DUR: Duration time of the pause.
- WORD_PHONES: Phone and durations in the word(phone1:duration1…)
- FLAG: Specify reliable phone duration according to threshold. (SUSP=Suspicious word, ?=Error or not include phones, 0=otherwise)
- LAST_VOWEL: Last vowel in recent word boundary.
- LAST_VOWEL_START: Starting time of the last vowel in the recent word boundary.

- LAST_VOWEL_END: Ending time of the last vowel in the recent word boundary.
- LAST_VOWEL_DUR: Duration of the last vowel in the previous word boundary.
- LAST_RHYME START: Starting time of the last rhyme in the recent word boundary.
- LAST_RHYME_END: Ending time of the last rhyme in the recent word boundary.
- NORM_LAST_RHYME_DUR:

$$\sum_{every\_phone\_in\_word} \frac{dur(phone) - mean(phone)}{std\_dev(phone)}$$

- PHONES_IN_LAST_RHYME: Whole number of phones in the last rhyme.

A.1.3 F_0 Features

- MIN_F0: The minimum raw F_0 value of current word.
- MAX_F0: The maximum raw F_0 value of current word.
- MEAN_F0: The mean raw F_0 value of current word.
- MIN_F0_NEXT: The minimum raw F_0 value of the word after boundary.
- MAX_F0_NEXT: The maximum raw F_0 value of the word after boundary.
- MEAN_F0_NEXT: The mean raw F_0 value of the word after boundary.
- MIN_F0_WIN: The minimum raw F_0 value of the word N frames before a boundary. (If there isn't enough data, maximum number of frames are used.)
- MAX_F0_WIN: The maximum raw F_0 value of the word N frames before a boundary. (If there isn't enough data, maximum number of frames are used.)
- MEAN_F0_WIN: The mean raw F_0 value of the word N frames before a boundary. (If there isn't enough data, maximum number of frames are used.)
- MIN_F0_NEXT_WIN: The minimum raw F_0 value of the word N frames before from after boundary. (If there isn't enough data, maximum number of frames are used.)
- MAX_F0_NEXT_WIN: The maximum raw F_0 value of the word N frames before from after boundary. (If there isn't enough data, maximum number of frames are used.)

- MEAN_F0_NEXT_WIN: The mean raw F_0 value of the word N frames before from after boundary. (If there isn't enough data, maximum number of frames are used.)
- MIN_STYLIFT_F0: The minimum stylized F_0 value of current word.
- MAX_STYLIFT_F0: The maximum stylized F_0 value of current word.
- MEAN_STYLIFT_F0: The mean stylized F_0 value of current word.
- FIRST_STYLIFT_F0: The first stylized F_0 value of current word.
- LAST_STYLIFT_F0: The last stylized F_0 value of current word.
- MIN_STYLIFT_F0_NEXT: The minimum stylized F_0 value of the word after a boundary.
- MAX_STYLIFT_F0_NEXT: The maximum stylized F_0 value of the word after a boundary.
- MEAN_STYLIFT_F0_NEXT: The mean stylized F_0 value of the word after a boundary.
- FIRST_STYLIFT_F0_NEXT: The first stylized F_0 value of the word after a boundary.
- LAST_STYLIFT_F0_NEXT: The last stylized F_0 value of the word after a boundary.
- MIN_STYLIFT_F0_WIN: The minimum stylized F_0 value of the word N frames before a boundary. (If there isn't enough data, maximum number of frames are used.)
- MAX_STYLIFT_F0_WIN: The maximum stylized F_0 value of the word N frames before a boundary. (If there isn't enough data, maximum number of frames are used.)
- MEAN_STYLIFT_F0_WIN: The mean stylized F_0 value of the word N frames before a boundary. (If there isn't enough data, maximum number of frames are used.)
- FIRST_STYLIFT_F0_WIN: The first stylized F_0 value of the word N frames before a boundary. (If there isn't enough data, maximum number of frames are used.)
- LAST_STYLIFT_F0_WIN: The last stylized F_0 value of the word N frames before a boundary. (If there isn't enough data, maximum number of frames are used.)

- MIN_STYLIFT_F0_NEXT_WIN: The minimum stylized F_0 value of the word N frames before from after boundary. (If there isn't enough data, maximum number of frames are used.)
- MAX_STYLIFT_F0_NEXT_WIN: The maximum stylized F_0 value of the word N frames before from after boundary. (If there isn't enough data, maximum number of frames are used.)
- MEAN_STYLIFT_F0_NEXT_WIN: The mean stylized F_0 value of the word N frames before from after boundary. (If there isn't enough data, maximum number of frames are used.)
- FIRST_STYLIFT_F0_NEXT_WIN: The first stylized F_0 value of the word N frames before from after boundary. (If there isn't enough data, maximum number of frames are used.)
- LAST_STYLIFT_F0_NEXT_WIN: The last stylized F_0 value of the word N frames before from after boundary. (If there isn't enough data, maximum number of frames are used.)
- PATTERN_WORD: Detects falling slope, unvoiced section and rising slope which represent by '' *f* '', '' *u* '' and '' *r* ''.
- PATTERN_WORD_CALLAPSED: Similar with PATTERN_WORD; sequence are represented by one symbol('' *f* '', '' *u* '' and '' *r* '').
- PATTERN_SLOPE: Similar with PATTERN_WORD; values are listed.
- PATTERN_WORD_NEXT: Detects falling slope, unvoiced section and rising slope which represent by '' *f* '', '' *u* '' and '' *r* '' of the word after a boundary.
- PATTERN_WORD_CALLAPSED_NEXT: Similar with PATTERN_WORD_NEXT; sequence are represented by one symbol('' *f* '', '' *u* '' and '' *r* '') .
- PATTERN_SLOPE_NEXT: Similar with PATTERN_WORD_NEXT; values are listed.
- PATTERN_WORD_WIN: Detects falling slope, unvoiced section and rising slope which represent by '' *f* '', '' *u* '' and '' *r* '' of the word N frames before a boundary. (If there isn't enough data, maximum number of frames are used.)

- PATTERN_WORD_CALLAPSED_WIN: Similar with PATTERN_WORD_WIN; sequence are represented by one symbol(‘’ *f* ’’, ‘’ *u* ’’ and ‘’ *r* ’’). (If there isn’t enough data, maximum number of frames are used.)
- PATTERN_SLOPE_WIN: Similar with PATTERN_WIN; values are listed. (If there isn’t enough data, maximum number of frames are used.)
- PATTERN_WORD_NEXT_WIN: Detects falling slope, unvoiced section and rising slope which represent by ‘’ *f* ’’, ‘’ *u* ’’ and ‘’ *r* ’’ of the word N frames before from after a boundary. (If there isn’t enough data, maximum number of frames are used.)
- PATTERN_WORD_CALLAPSED_NEXT_WIN: Similar with PATTERN_WORD_NEXT_WIN; sequence are represented by one symbol(‘’ *f* ’’, ‘’ *u* ’’ and ‘’ *r* ’’) . (If there isn’t enough data, maximum number of frames are used.)
- PATTERN_SLOPE_NEXT_WIN: Similar with PATTERN_WORD_NEXT_WIN; values are listed. (If there isn’t enough data, maximum number of frames are used.)
- NO_PREVIOUS_SSF: Number of sequentially frames with in the word which have same slope as last voiced frame in previous word.
- NO_PREVIOUS_VF: Number of sequentially voiced frames with in the word from last voiced frame in the word backwards.
- NO_FRAMES_I_S_WE: Number of sequentially frames between the last voiced frame which have a proper to a sequence of voiced frames larger than min_frame_length in the current word and at the end of that word.
- NO_SUCESSOR_SSF: Number of successor sequentially frames with in the word which have same slope as the first voiced frame in current word.
- NO_SUCCESSOR_VF: Number of sequentially voiced frames with in the word from the first voiced frame in the following word.
- NO_FRAMES_WS_FS: Number of sequentially frames between the first frame of the current word and the first voiced frame in that word which have a proper to a sequence of voiced frames larger than min_frame_length.

- NO_PREVIOUS_SSF_NEXT: Number of sequentially frames with in the word which have same slope as last voiced frame in previous word for the word after a boundary.
- NO_PREVIOUS_VF_NEXT: Number of sequentially voiced frames with in the word from last voiced frame in the word backwards for the word after a boundary.
- NO_FRAMES_I_S_WE_NEXT: Number of sequentially frames between the last voiced frame which have a proper to a sequence of voiced frames larger than min_frame_length in the current word and at the end of that word for the word after a boundary.
- NO_SUCESSOR_SSF_NEXT: Number of successor sequentially frames with in the word which have same slope as the first voiced frame in current word for the word after a boundary.
- NO_SUCCESSOR_VF_NEXT: Number of sequentially voiced frames with in the word from the first voiced frame in the following word for the word after a boundary.
- NO_FRAMES_WS_FS_NEXT: Number of sequentially frames between the first frame of the current word and the first voiced frame in that word which have a proper to a sequence of voiced frames larger than min_frame_length for the word after a boundary.
- NO_PREVIOUS_SSF_WIN: Number of sequentially frames with in the word which have same slope as last voiced frame in previous word for the word N frames before a boundary . (If there isn't enough data, maximum number of frames are used.)
- NO_PREVIOUS_VF_WIN: Number of sequentially voiced frames with in the word from last voiced frame in the word backwards for the word N frames before a boundary. (If there isn't enough data, maximum number of frames are used.)
- NO_FRAMES_I_S_WE_WIN: Number of sequentially frames between the last voiced frame which have a proper to a sequence of voiced frames larger than min_frame_length in the current word and at the end of that word for the word N frames before a boundary. (If there isn't enough data, maximum number of frames are used.)

- NO_SUCESSOR_SSF_WIN: Number of successor sequentially frames with in the word which have same slope as the first voiced frame in current word for the word N frames before a boundary. (If there isn't enough data, maximum number of frames are used.)
- NO_SUCCESSOR_VF_WIN: Number of sequentially voiced frames with in the word from the first voiced frame in the following word for the word N frames before a boundary. (If there isn't enough data, maximum number of frames are used.)
- NO_FRAMES_WS_FS_WIN: Number of sequentially frames between the first frame of the current word and the first voiced frame in that word which have a proper to a sequence of voiced frames larger than min_frame_length for the word N frames before a boundary. (If there isn't enough data, maximum number of frames are used.)
- NO_PREVIOUS_SSF_NEXT_WIN: Number of sequentially frames with in the word which have same slope as last voiced frame in previous word for the word N frames before from after a boundary. (If there isn't enough data, maximum number of frames are used.)
- NO_PREVIOUS_VF_NEXT_WIN: Number of sequentially voiced frames with in the word from last voiced frame in the word backwards for the word N frames before from after a boundary. (If there isn't enough data, maximum number of frames are used.)
- NO_FRAMES_I_S_WE_NEXT_WIN: Number of sequentially frames between the last voiced frame which have a proper to a sequence of voiced frames larger than min_frame_length in the current word and at the end of that word N frames before from after a boundary. (If there isn't enough data, maximum number of frames are used.)
- NO_SUCESSOR_SSF_NEXT_WIN: Number of successor sequentially frames with in the word which have same slope as the first voiced frame in current word N frames before from after a boundary. (If there isn't enough data, maximum number of frames are used.)
- NO_SUCCESSOR_VF_NEXT_WIN: Number of sequentially voiced frames with in the word from the first voiced frame in the following word for the

word N frames before from after a boundary. (If there isn't enough data, maximum number of frames are used.)

- NO_FRAMES_WS_FS_NEXT_WIN: Number of sequentially frames between the first frame of the current word and the first voiced frame in that word which have a proper to a sequence of voiced frames larger than min_frame_length for the word N frames before from after a boundary. (If there isn't enough data, maximum number of frames are used.)
- PATTERN_BOUNDARY: Combine of PATTERN_WORD and PATTERN_NEXT_WORD.
- SLOPE_DIFF: The difference between the last non-zero slope of the word and the first non-zero slope of the following word. ''?' is default value for not found features.

A.1.4 Energy Features

Energy features are formeed similar with F_0 features as listed below,

- MIN_ENERGY
- MAX_ENERGY
- MEAN_ENERGY
- MIN_ENERGY_NEXT
- MAX_ENERGY_NEXT
- MEAN_ENERGY_NEXT
- MIN_ENERGY_WIN
- MAX_ENERGY_WIN
- MEAN_ENERGY_WIN
- MIN_ENERGY_NEXT_WIN
- MAX_ENERGY_NEXT_WIN
- MEAN_ENERGY_NEXT_WIN
- MIN_STYLIFT_ENERGY
- MAX_STYLIFT_ENERGY
- MEAN_STYLIFT_ENERGY
- FIRST_STYLIFT_ENERGY
- LAST_STYLIFT_ENERGY

- MIN_STYLIFT_ENERGY_NEXT
- MAX_STYLIFT_ENERGY_NEXT
- MEAN_STYLIFT_ENERGY_NEXT
- FIRST_STYLIFT_ENERGY_NEXT
- LAST_STYLIFT_ENERGY_NEXT
- MIN_STYLIFT_ENERGY_WIN
- MAX_STYLIFT_ENERGY_WIN
- MEAN_STYLIFT_ENERGY_WIN
- FIRST_STYLIFT_ENERGY_WIN
- LAST_STYLIFT_ENERGY_WIN
- MIN_STYLIFT_ENERGY_NEXT_WIN
- MAX_STYLIFT_ENERGY_NEXT_WIN
- MEAN_STYLIFT_ENERGY_NEXT_WIN
- FIRST_STYLIFT_ENERGY_NEXT_WIN
- LAST_STYLIFT_ENERGY_NEXT_WIN
- ENERGY_PATTERN_WORD
- ENERGY_PATTERN_WOD_CALLAPSED
- ENERGY_PATTERN_SLOPE
- ENERGY_PATTERN_WORD_NEXT
- ENERGY_PATTERN_WOD_CALLAPSED_NEXT
- ENERGY_PATTERN_SLOPE_NEXT
- ENERGY_PATTERN_WORD_WIN
- ENERGY_PATTERN_WOD_CALLAPSED_WIN
- ENERGY_PATTERN_SLOPE_WIN
- ENERGY_PATTERN_WORD_NEXT_WIN
- ENERGY_PATTERN_WOD_CALLAPSED_NEXT_WIN
- ENERGY_PATTERN_SLOPE_NEXT_WIN
- ENERGY_PATTERN_BOUNDARY
- ENERGY_SLOPE_DIFF

A.2 Statistical Tables

- Phone_dur_stats: Table includes the mean phone duration, the standard deviation of the phone duration, the number of occurrences of that phone in the training database and the phone duration threshold for each phone. Computation as shown in Equation XX.

$$threshold(phone) = mean(phone) + 10 \times std\_dev(phone) \quad \text{(A.1)}$$

- Pause_dur.stats: Table includes mean and standard deviation of the pauses in the training database for each audio.
- Spkr_feat.stats: Table has rows as number of speakers. Each row includes voiced and unvoiced frames, F_0 and F_0 slope, energy and energy slope. Detailed features as listed below,
  - MEAN_VOICED: The average length of voiced frames.
  - STDEV_VOICED: The standard deviation of voiced frames.
  - COUNT_VOICED: The number of voiced frames.
  - MEAN_UNVOICED: The average length of unvoiced frames.
  - STDEV_UNVOICED: The standard deviation of unvoiced frames.
  - COUNT_UNVOICED: The number of unvoiced frames.
  - MEAN_PITCH: The average F_0 value.
  - STDEV_PITCH: The standard deviation F_0 value.
  - COUNT_PITCH: The number of F_0 value.
  - MEAN_SLOPE: The mean pitch slope.
  - STDEV_SLOPE: The standard deviation of pitch slope.
  - COUNT_SLOPE: The number of pitch slope.
  - MEAN_ENERGY: The average energy value.
  - STDEV_ENERGY: The standard deviation of energy value.
  - COUNT_ENERGY: The number of energy value.
  - MEAN_ENERGY_SLOPE: The mean slope.
  - STDEV_ENERGY_SLOPE: The standard deviation of the slope.
  - COUNT_ENERGY_SLOPE: The number of energy slope.
- Spkr_phone_dur.stats: Feature has tables as number of speakers. Similar with phone_dur_stats, it considers all of the speakers.

- Last_rhyme_dur.stats: Table includes mean phone duration for the phones in the last rhyme, the standard deviation of the phone duration for the ohones in the last rhyme, and the number of last rhymes used for each audio.
- Pause_dur.stats: Table has rows as number of speakers. Detailed features as listed below,
    - MEAN: The mean duration of the pauses.
    - STDEV: The standard deviation of the duration of the pauses.
    - MEAN_LOG: The mean of the log duration of the pauses.
    - STDEV_LOG: The standard deviation of the log duration of the pauses.
    - COUNT_PAUSE: The number of the pauses.

A.3 Derived Features

Derive features are formed from basic features and statistics.

A.3.1 Normalized Word Duration

- WORD_DUR = WORD_END – WORD_START (A.2)
- WORD_AV_DUR = $\sum_{every\_phone\_in\_word} mean(phone)$ (A.3)
- NORM_WORD_DUR = WORD_DUR / WORD_AV_DUR (A.4)

A.3.2 Normalized Pause

- PAU_DUR_N = PAU_DUR / PAUSE_MEAN (A.5)

A.3.3 Normalized Vowel Duration

- LAST_VOWEL_DUR_Z = (LAST_VOWEL_DUR – ALL_PHONE_DUR_MEAN) / ALL_PHONE_DUR_STDEV (A.6)
- LAST_VOWEL_DUR_N = LAST_VOWEL_DUR / ALL_PHONE_DUR_MEAN (A.7)
- LAST_VOWEL_DUR_ZSP = (LAST_VOWEL_DUR – SPKR_PHONE_DUR_MEAN) / SPKR_PHONE_DUR_STDEV (A.8)
- LAST_VOWEL_DUR_NSP = LAST_VOW_DUR / SPKR_PHONE_DUR_MEAN (A.9)

A.3.4 Normalized Rhyme Duration

- LAST_RHYME_DUR_PH = LAST_RHYME_DUR / PHONES_IN_LAST_RHYME (A.10)
- LAST_RHYME_DUR_PH_ND = (LAST_RHYME_DUR / PHONES_IN_LAST_RHYME) – LAST_RHYME_PHONE_DUR_MEAN (A.11)
- LAST_RHYME_DUR_PH_NR = (Last_RHYME_DUR / PHONES_IN_LAST_RHYME) / LAST_RHYME_PHONE_DUR_MEAN (A.12)
- LAST_RHYME_NORM_DUR_PH = NORM_LAST_RHYME_DUR / PHONES_IN_LAST_RHYME (A.13)
- LAST_RHYME_NORM_DUR_PH_ND = (NORM_LAST_RHYME_DUR / PHONES_IN_LAST_RHYME) – NORM_LAST_RHYME_PHONE_DUR_MEAN (A.14)
- LAST_RHYME_NORM_DUR_PH_NR= (NORM_LAST_RHYME_DUR / PHONES_IN_LAST_RHYME) / NORM_LAST_RHYME_PHONE_DUR_MEAN (A.15)
- LAST_RHYME_DUR_WHOLE_ND= LAST_RHYME_DUR – LAST_RHYME_WHOLE_DUR_MEAN (A.16)
- LAST_RHYME_WHOLE_DUR_NR = LAST_RHYME_DUR / LAST_RHYME_WHOLE_DUR_MEAN (A.17)
- LAST_RHYME_WHOLE_DUR_Z = (LAST_RHYME_DUR – LAST_RHYME_WHOLE_DUR_MEAN) / LAST_RHYME_WHOLE_DUR_STDEV (A.18)

A.3.5 F_0 Derived Features

F_0 derived features are formed from F_0 features.

- SPKR_FEAT_F0_MODE = exp(SPKR_F0_MEAN) (A.19)
- SPKR_FEAT_F0_TOPLN = 0.75x(exp(SPKR_F0_MEAN) (A.20)
- SPKR_FEAT_F0_BASELN = 1.5x(exp(SPKR_F0_MEAN) (A.21)
- SPKR_FEAT_F0_STDLN = exp(SPKR_F0_STDEV) (A.22)

- SPKR_FEAT_F0_RANGE = SPKR_FEAT_F0_TOPLN – SPKR_FEAT_F0_BASELN (A.23)
- F0K_WORD_DIFF_HIHI_N = log(MAX_STYLFIT_F0 / MAX_STYLIFT_F0_NEXT) (A.24)
- F0K_WORD_DIFF_HILO_N = log(MAX_STYLFIT_F0 / MIN_STYLFIT_F0_NEXT) (A.25)
- F0K_WORD_DIFF_LOLO_N = log(MIN_STYLFIT_F0 / MIN_STYLFIT_F0_NEXT) (A.26)
- F0K_WORD_DIFF_LOHI_N = log(MIN_STYLFIT_F0 / MAX_STYLFIT_F0_NEXT) (A.27)
- F0K_WORD_DIFF_NNMN_N = log(MEAN_STYLFIT_F0 / MEAN_STYLFIT_F0_NEXT) (A.28)
- F0K_WORD_DIFF_HIHI_NG = (log (MAX_STYLFIT_F0) / log (MAX_STYLFIT_F0_NEXT)) / SPKR_FEAT_F0_RANGE (A.29)
- F0K_WORD_DIFF_HILO_NG = (log (MAX_STYLFIT_F0) / log (MIN_STYLFIT_F0_NEXT))/ SPKR_FEAT_F0_RANGE (A.30)
- F0K_WORD_DIFF_LOLO_NG = (log (MIN_STYLFIT_F0) / log (MIN_STYLFIT_F0_NEXT))/ SPKR_FEAT_F0_RANGE (A.31)
- F0K_WORD_DIFF_LOHI_NG = (log (MIN_STYLFIT_F0) / log (MAX_STYLFIT_F0_NEXT))/ SPKR_FEAT_F0_RANGE (A.32)
- F0K_WORD_DIFF_MNMN_NG = (log (MEAN_STYLFIT_F0) / log (MEAN_STYLFIT_F0_NEXT))/ SPKR_FEAT_F0_RANGE (A.33)
- F0K_WIN_DIFF_HIHI_N = log (MAX_STYLFIT_F0_WIN / MAX_STYLFIT F0_WIN_NEXT) (A.34)
- F0K_WIN_DIFF_HILO_N = log (MAX_STYLFIT_F0_WIN / MIN_STYLFIT_F0_WIN_NEXT) (A.35)
- F0K_WIN_DIFF_LOLO_N = log (MIN_STYLFIT_F0_WIN / MIN_STYLFIT_F0_WIN_NEXT) (A.36)
- F0K_WIN_DIFF_LOHI_N = log (MIN_STYLFIT_F0_WIN / MAX_STYLFIT_F0_WIN_NEXT) (A.37)
- F0K_WIN_DIFF_MNMN_NG = log (MEAN_STYLFIT_F0_WIN / MEAN_STYLFIT_F0_WIN_NEXT) (A.38)

- F0K_WIN_DIFF_HIHI_NG = (log (MAX_STYLFIT_F0_WIN) / log (MAX_STYLFIT_F0_WIN_NEXT)) / SPKR_FEAT_F0_RANGE (A.39)
- F0K_WIN_DIFF_HILO_NG = (log (MAX_STYLFIT_F0_WIN) / log (MIN_STYLFIT_F0_WIN_NEXT)) / SPKR_FEAT_F0_RANGE (A.40)
- F0K_WIN_DIFF_LOLO_NG = (log (MIN_STYLFIT_F0_WIN) / log (MIN_STYLFIT_F0_WIN_NEXT)) / SPKR_FEAT_F0_RANGE (A.41)
- F0K_WIN_DIFF_LOHI_NG = (log (MIN_STYLFIT_F0_WIN) / log (MAX_STYLFIT_F0_WIN_NEXT)) / SPKR_FEAT_F0_RANGE (A.42)
- F0K_WIN_DIFF_MNMN_NG = (log (MEAN STYLFIT F0 WIN) / log (MEAN STYLFIT F0 WIN NEXT)) / SPKR FEAT F0 RANGE (A.43)
- F0K_DIFF_LAST_KBASELN = LAST_STYLFIT_F0 – SPKR_FEAT_F0_BASELN (A.44)
- F0K DIFF MEAN KBASELN = MEAN STYLFIT F0 - SPKR FEAT F0 BASELN (A.45)
- F0K_DIFF_WINMIN_KBASELN = MIN_STYLFIT_F0_WIN – SPKR_FEAT_F0_ BASELN (A.46)
- F0K_LR_LAST_KBASELN = log (LAST_STYLFIT_F0 / SPKR_FEAT_F0_BASELN) (A.47)
- F0K_LR_MEAN_KBASELN = log (MEAN_STYLFIT_F0 / SPKR_FEAT_F0_BASELN) (A.48)
- F0K_LR_WINMIN_KBASELN = log (MIN_STYLFIT_F0_WIN / SPKR_FEAT_F0_BASELN) (A.49)
- F0K_ZRANGE_MEAN_KBASELN = (MEAN_STYLFIT_F0 – SPKR_FEAT_F0_BASELN) / SPKR_FEAT_F0_RANGE (A.50)
- F0K ZRANGE_MEAN_KTOPLN = (SPKR_FEAT_F0_TOPLN – MEAN_STYLFIT_F0) / SPKR_FEAT_F0_RANGE (A.51)
- F0K_ZRANGE_MEANNEXT_KBASELN = (MEAN_STYLFIT_F0_NEXT –SPKR_FEAT_F0_BASELN) / SPKR_FEAT_F0_RANGE (A.52)
- F0K_ZRANGE_MEANNEXT_KTOPLN = (SPKR_FEAT_F0_TOPLN – MEAN_FEAT_F0_NEXT) / SPKR_FEAT_F0_RANGE (A.53)
- F0K_DIFF_MEANNEXT_KTOPLN = MEAN_STYLFIT_F0_NEXT – SPKR_FEAT_F0_TOPLN (A.54)

- F0K_DIFF_MAXNEXT_KTOPLN = MAX_STYLFIT_F0_NEXT – SPKR_FEAT_F0_TOPLN (A.55)
- F0K_DIFF_WINMAXNEXT_KTOPLN = MAX STYLFIT_F0_NEXT_WIN – SPKR_FEAT_F0_TOPLN (A.56)
- F0K_LR_MEANNEXT_KTOPLN = log (MEAN_STYLFIT_F0_NEXT / SPKR_FEAT_F0_TOPLN) (A.57)
- F0K_LR_MAXNEXT_KTOPLN = log (MAX_STYLFIT_F0_NEXT / SPKR_FEAT_F0_TOPLN) (A.58)
- F0K_LR_WINMAXNEXT_KTOPLN = log (MAX_STYLFIT_F0_NEXT_WIN / SPKR_FEAT_F0_TOPLN) (A.59)
- F0K_MAXK_MODE_N = log (MAX_STYLFIT_F0 / SPKR_FEAT_F0_MODE) (A.60)
- F0K_MAXK_NEXT_MODE_N = log (MAX_STYLFIT_F0_NEXT / SPKR_FEAT_F0_MODE) (A.61)
- F0K_MAXK_MODE_Z = (MAX_STYLFIT_F0 – SPKR_FEAT_F0_MODE) / SPKR_FEAT_F0_RANGE (A.62)
- F0K_MAXK_NEXT_MODE_Z = (MAX_STYLFIT_F0_NEXT – SPKR_FEAT_F0_MODE) / SPKR_FEAT_F0_RANGE (A.63)
- F0K_WORD_DIFF_BEGBEG = log (FIRST_STYLFIT_F0 / FIRST_STYLFIT_F0_NEXT) (A.64)
- F0K_WORD_DIFF_ENDBEG = log (LAST_STYLFIT_F0 / FIRST_STYLFIT_F0_NEXT) (A.65)
- F0K_INWRD_DIFF = log (FIRST_STYLFIT_F0 / LAST_STYLFIT_F0) (A.66)
- LAST_SLOPE: The last ‘’f’’ or’’ r’’ slope in PATTERN_SLOPE.
- FIRST SLOPE NEXT: The first ’’ f’’ or ‘’ r’’ slope in PATTERN_SLOPE_NEXT.
- SLOPE_DIFF_N = SLOPE_DIFF / SKPR_FEAT_F0_SD_SLOPE (A.67)
- LAST_SLOPE_N = LAST_SLOPE/ LAST_STYLFIT_F0 (A.68)

A.3.6 Energy Derived Features

Energy derived features are formed from energy features. They are computed similarly as the derived F features as listed below.

- ENERGY_WORD_DIFF_HIHI_N
- ENERGY_WORD_DIFF_HILO_N
- ENERGY_WORD_DIFF_LOLO_N
- ENERGY_WORD_DIFF_LOHI_N
- ENERGY_WORD_DIFF_MNMN_N
- ENERGY_WORD_DIFF_HIHI_NG
- ENERGY_WORD_DIFF_HILO_NG
- ENERGY_WORD_DIFF_LOLO_NG
- ENERGY_WORD_DIFF_LOHI_NG
- ENERGY_WORD_DIFF_MNMN_NG
- ENERGY_WIN_DIFF_HIHI_N
- ENERGY_WIN_DIFF_HILO_N
- ENERGY_WIN_DIFF_LOLO_N
- ENERGY_WIN_DIFF_LOHI_N
- ENERGY_WIN_DIFF_MNMN_NG
- ENERGY_WIN_DIFF_HIHI_NG
- ENERGY_WIN_DIFF_HILO_NG
- ENERGY_WIN_DIFF_LOLO_NG
- ENERGY_WIN_DIFF_LOHI_NG
- ENERGY_WIN_DIFF_MNMN_NG
- ENERGY_DIFF_LAST_KBASELN
- ENERGY_DIFF_MEAN_KBASELN
- ENERGY_DIFF_WINMIN_KBASELN
- ENERGY_LR_LAST_KBASELN
- ENERGY_LR_MEAN_KBASELN
- ENERGY_LR_WINMIN_KBASELN
- ENERGY_ZRANGE_MEAN_KBASELN
- ENERGY_ZRANGE_MEAN_KTOPLN
- ENERGY_ZRANGE_MEANNEXT_KBASELN

- ENERGY_ZRANGE_MEANNEXT_KTOPLN
- ENERGY_DIFF_MEANNEXT_KTOPLN
- ENERGY_DIFF_MAXNEXT_KTOPLN
- ENERGY_DIFF_WINMAXNEXT_KTOPLN
- ENERGY_LR_MEANNEXT_KTOPLN
- ENERGY_LR_MAXNEXT_KTOPLN
- ENERGY_LR_WINMAXNEXT_KTOPLN
- ENERGY_MAXK_MODE_N
- ENERGY_MAXK_NEXT_MODE_N
- ENERGY_MAXK_MODE_Z
- ENERGY_MAXK_NEXT_MODE_Z
- ENERGY_WORD_DIFF_BEGBEG
- ENERGY_WORD_DIFF_ENDBEG
- ENERGY_INWRD_DIFF
- ENERGY_LAST_SLOPE
- ENERGY_SLOPE_DIFF_N
- ENERGY_LAST_SLOPE_N

A.3.7 Average Phone Duration

- AVG_PHONE_DUR_Z = $\sum_{every\_phone\_in\_word} phone\_z(phone)/\#\ phones$

(A.69)

- MAX_PHONE_DUR_Z = $\max_{every\_phone\_in\_word} phone\_z(phone)$

(A.70)

- AVG_PHONE_DUR_N = $\sum_{every\_phone\_in\_word} phone\_n[phone]/\#\ phones$

(A.71)

A.3.8 Speaker Specific Normalization

- AVG_PHONE_DUR_ZSP =

  $\sum_{every\_phone\_in\_word} phone\_zsp[phone]/\#\ phones$ (A.72)

- MAX_PHONE_DUR_ZSP = $\max_{every\_phone\_in\_word} phone\_zsp[phone]$ (A.73)

- AVG_PHONE_DUR_NSP =

  $\sum_{every\_phone\_in\_word} phone\_nsp[phone]/\#\ phones$ (A.74)

- MAX_PHONE_DUR_NSP = $\max_{every\_phone\_in\_word} phone\_nsp[phone]$ (A.75)

To be performed features with over only the vowels listed below (similar to _PHONE_DUR_) ;

- AVG_VOWEL_DUR_Z
- MAX_VOWEL_DUR_Z
- AVG_VOWEL_DUR_N
- MAX_VOWEL_DUR_N
- AVG_VOWEL_DUR_ZSP
- MAX_VOWEL_DUR_ZSP
- AVG_VOWEL_DUR_NSP
- MAX_VOWEL_DUR_NSP

Appendix B Praat Scripts

B.1 Scripts For Computing Global Statistics (‘’code/stats’’)

- stats_batch_praat: The interface for accepts inputs and controls the statistics.
- operations.praat: Highest level of the operation flow.
- io.praat: Controller for input and output files.
- table.praat: Controller for table operations.
- stats.praat: Routines for computing statistics.
- routine.praat: Routines for obtaining various basic elements.
- utils.praat: Routines for some miscellaneous utility.
- config.praat: Configuration of the pre-defined parameter values, such as frame and window size, default file names, etc.

B.2 Scripts for Extracting Prosodic Features (‘’code/’’)

- main_batch.praat: The interface for accepts inputs and controls the statistics.
- operations.praat: Highest level of operation flow.
- io.praat: Controller for input and output files.
- table.praat: Controller for table operations.
- fetch.praat: Higher level routines for extracting basic prosodic features.
- routine.praat: Routines for obtaining various basic elements, and lower level routines that implement feature extraction.
- derive.praat: Routines for computing derived features.
- utils.praat: Routines for some miscellaneous utility.
- config.praat: Configuration of the pre-defined parameter values, such as frame and window size, default file names, etc.
- pf_list_files/feature_name_table.Tab: Contains a list of feature names.

Appendix C Morphological Feature List

- Alpha: Symbols of the alphabet
- Adj: Adjective
- Adv: Adverb
- Cnj: Conjunction
- Det: Determiner
- Exist: The word var and yok
- Ij: Interjection
- N: Noun
- Not: The word değil
- Num: Number
- Onom: Onomatopoeia
- Postp: Postposition
- Prn: Pronoun
- Punc: Punctuation
- Q: Question particle mI
- V: Verb
- 1s: First person single
- 2s: Second person single
- 3s: Third person single
- 1p: First person plural
- 2p: Second person plural
- 3p: Third person plural

Curriculum Vitae

Izel Revidi was born on 25th May 1990 in Istanbul, Turkey. He received his BS degree in Electrical and Electronics Engineering from Işık University, Istanbul, Turkey in 2012. He is an engineer in Electronic Security Solutions market since 2013. His research interest covers speech processing, speech modeling, automatic speech recognition and machine learning.