
ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

ADDIS ABABA INSTITUTE OF TECHNOLOGY (AAiT)

ELECTRICAL AND COMPUTER ENGINEERING DEPARTMENT

# Design and Implementation of Predictive Text Entry Method for Afan Oromo on Mobile Phone

By

Gudisa Tesema

Advisor

Ato Yoseph Abate

A Thesis Submitted to the School of Graduate Studies of the Addis Ababa University in Partial Fulfilment of the Requirements for the Degree of Masters of Science in Electrical and Computer Engineering (Computer Engineering Stream)

February, 2013

Addis Ababa, Ethiopia

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

ADDIS ABABA INSTITUTE OF TECHNOLOGY (AAiT)

ELECTRICAL AND COMPUTER ENGINEERING DEPARTMENT

# Design and Implementation of Predictive Text Entry Method for Afan Oromo on Mobile Phone

Gudisa Tesema

## Approval by Board of Examiners

| | | |
|---|---|---|
| Prof. Kwon Ho Yeol | _____ | _____ |
| Chairman, Head. Committee | Signature | Date |
| Committee | | |
| Ato Yoseph Abate | _____ | _____ |
| Advisor | Signature | Date |
| Dr. Eng Getachew Alemu | _____ | _____ |
| Internal Examiner | Signature | Date |
| Ato Fistum Assamnew | _____ | _____ |
| External Examiner | Signature | Date |

## Declaration

I, the undersigned, declare that this thesis is my original work, has not been presented for a degree in this or any other university, and all sources of materials used for the thesis have been fully acknowledged.

Gudisa Tesema

_____ _____

Name Signature

Place: Addis Ababa

Date of Submission: _____

This thesis has been submitted for examination with my approval as a university advisor.

Ato Yoseph Abate

_____ _____

Advisor's Name Signature

# Acknowledgement

First of all, I would like to say "***Thank you my God***". Secondly, I would like to sincerely thank my advisor, *Yoseph Abate*, for his supervision, advice, patience, moral support throughout this thesis work, and for giving me variable and valuable ideas and guidance from the very beginning. I would like to thank *Mike Scott* for giving me the tool I used in this thesis to prepare the words statistics from the corpus I have collected.

I would also like to give special thanks to my family. My family members are always there to support me in every situation. On top of all, three persons, my lovely younger brothers Kenea and Regassa, and sister Kuli deserve very grateful thanks because without them I would not be who I am today.

Though it is difficult to mention the name of persons who gave me their hand while doing this thesis, it is necessary to thank those who gave their precious time to share ideas, and gave me moral and material support. Last but not least, I would like to sincerely thank all of my friends, colleagues, classmates for their assistance, encouragement, and inspiration during this thesis work.

I would also like to thank all the staff of ECE, AAiT, AAU for providing such an education and research environment.

## ABSTRACT

Language is a unique phenomenon that distinguishes man from other living things. It is our primary method of communication with each other, yet very little is understood about how language is acquired when we are infants. A greater understanding in this area would have the potential to improve man machine communication. Word prediction is an important NLP problem in which the correct word is predicted based on a given context. This paper presents a new word prediction approach based on context features and machine learning. The proposed method casts the problem as a learning-classification task by training word predictors with highly discriminating features selected by various feature selection techniques. The contribution of this work lies in the new way of presenting this problem, and the unique combination of a top performer in machine learning, svm, with various feature selection techniques and more. The method is implemented and evaluated using several datasets. The experimental results show clearly that the method is effective in predicting the correct words by using small contexts. For an advanced implementation of predictive text entry via machine learning, a multi-level feature based framework is developed. In order to use as much information as possible, features from the character level, word level, syntax level, and semantic level are included. The program that is capable of automatically inferring a grammar from a Natural Language Corpus is developed.

Afan Oromo words can be categorized into “Maqaa”, “Ibsa Maqaa”, “Xumura”, “Ibsa Xumuraa” fi “Firoomsee”. In addition the language has short and long voices that are dealt with vowels and strong and weak voices which are dealt with consonants. Due to these voices, Afan Oromo has long words containing many characters. And also Afan Oromo has got its own structure of writing sentences (Subject + Object+ Verb).



**Keywords**: Word prediction, word completion, machine learning, natural language processing.

## LIST OF ACRONYMS

| | |
|---|---|
| SVM | Support Vector Machine |
| NB | Naïve Bayes |
| EM | Expectation maximization |
| T9 | Text on 9 keys |
| OCR | Optical Character Recognition |
| ICR | Intelligent Character Recognition |
| HMM | Hidden Markov Model |
| 30K | 30 thousand |
| LOB | Lancaster-Oslo/Bergen |
| N-gram | N length word |
| PDA | Personal Digital Assistant |
| KS | Keystroke Saving |
| POS | Part Of Speech |
| AAC | Augmentative and Alternative Communication |
| HTK | Hidden Markov Model ToolKit |
| PAL | Predicative Adaptive Lexicon |
| RBF | Radial Basis Function |
| OS | Operating System |
| SDK | Software Development Kit |
| SE | Standard Edition |
| JVM | Java Virtual Machine |
| VM | Virtual Machine |
| XML | Extended Markup Language |
| UI | User Interface |
| ADT | Android Development Tool |
| IDE | Integrated Development Environment |
| API | Application Programming Interface |
| NLP | Natural Language Programming |
| MLE | Maximum Likelihood Estimate |
| wpm | words per minute |
| KSPC | KeyStroke Per Character |
| HR | Hit Rate |
| KuP | KeyStroke Until Prediction |

# LIST OF FIGURES

Figure 2.1: The standard buttons for text inputs to be pressed…….....5

Figure 2.2: The two key text input keypad display…….....7

Figure 3.1: Steps for grammar based word prediction…….....….....26

Figure 3.2: POS of words as they appear in a sentence…….....….....27

Figure 3.3: Basics of SVM…….....….....….....28

Figure 3.4: The word predictor Architecture……...….....31

Figure 3.5: HMM model for Parts of Speech in Afan Oromo…….....32

Figure 3.6: Best Hyperplane and Margin concept of SVM…….....….....33

Figure 3.7: Linear SVM…….....…….....…...34

Figure 3.8: Soft Margin or Classification in SVM…….....…...35

Figure 3.9: Low Data Dimension to Higher Data Dimension Mapping.....36

Figure 3.10: Kernel Function for Data Categorization…….....37

Figure 3.11: Context Based Word Prediction model……..….....39

Figure 4.1: Words unigram and bigram with their frequency…….....….....45

Figure 4.2: POS unigrams and Bigram statistics…….....…...46

Figure 4.3: The predicted words after two-prefix length (mo) is inserted to the text editor.....48

Figure 4.4: The predicted words after three prefix length (mor) is inserted to the text editor.....48

Figure 4.5: The screenshot of predicted words after 3 words and 4 prefixes.....……..48

## LIST OF TABLES

Table 2.1: The predictions of Dunlop and crossan (2000) for multi_press, Predictive and Predictive with word completion methods..…….....…….....9

Table 3.1: Feature encoding for SVM text classification…….....…….....38

Table 4.1: how N-gram model is represented in the dictionary…….....44

Table 4.2: The Feature encoding for the RBF kernel function…….....46

Table 4.3: The Features encoding for SVM Training…….....47

Table 4.4 : Evaluation of the accuracy of designed algorithm for context based prediction.....48

Content Page

CHAPTER ONE: Overview.....1

1.1 INTRODUCTION……..…….....….....1

1.2. Motivation …….....….....…2

1.3. Research Problem……...….....….....2

1.4. Objective……...……...…….....….....3

1.4.1. General Objective …….....3

1.4.2. Specific Objectives ……...….....3

1.5. Scope and Limitation …….....…….....3

1.6. Methodology ……...…….....4

1.7. Thesis Organization …….....….....4

CHAPTER TWO: Literature Review and Related Works.....5

2.1 Literature Review ……...…….....5

2.1.1 Data Entry Techniques ……...…...…….…….....5

2.2 Related Works ……..….....11

2.2.1 Character Recognition for Optical character input…….....11

2.2.2 Word Recognition …….…….…….....11

2.2.3 Text Prediction ……....…….....13

CHAPTER THREE: Word Prediction Model for Afan Oromo Language.....15

3.1. The Word Prediction Model …….….....15

3.1.1. Statistical Prediction …….…….....18

3.1.1.1. Fixed Lexicon ……....……..……..…….....18

3.1.1.2. Adaptive Lexicon …….....….....19

3.1.1.3. Bigrams and Trigrams ……...…….…….....….....20

3.1.2. Syntactic Prediction…….….....23

3.1.3. Semantic Prediction …….…….....24

3.1.4 SVM LEARNING AND PREDICTION…….....27

3.1.4.1 Parameter selection for support vector machines.....…….....29

3.2. Architecture of the System ....……...…….....30

3.3. Prediction Algorithm……..……...…….....31

3.4. Summary …….....….....39

CHAPTER FOUR: Implementation and Experiment.....….....40

4.1. Implementation …….....….....40

4.1.1. Used Tools and Development Environment …….....40

4.1.2. Android Emulator.....41

4.1.3. SQLite Databases.....41

4.1.4. User Interfaces …….....….....42

4.2. Experiment ……...…...…….....42

4.3. Data Collection :Corpus...…….....45

4.3.1. Result of the Experiment …….....45

4.3.2. SVM training.....46

4.4. Summary…….....52

CHAPTER FIVE: CONCLUSION AND FUTURE WORKS…...…….....53

REFERENCES ……...…...…….....54

APPENDIX A: THE STATISTICS GENERATED FOR THE COLLECTED CORPUS OF AFAN OROMO TEXT .....57

APPENDIX B: LIST AND DESCRIPTION OF DOCUMENTS USED TO PREPARE THE CORPUS …….…….....58

# CHAPTER ONE: Overview

## 1.1 Introduction

Text classification is the task of assigning a text to a pre-specified set of classes. Real world applications including spam filtering, e-mail routing, organizing web content into topical hierarchies, and news filtering rely on automatic means of classification [9]. Text classification can be broadly categorized into discriminative techniques, typified by support vector machines (SVMs), decision trees and neural networks and generative techniques, like Naive Bayes (NB) and Expectation Maximization (EM) based methods [13]. From a performance point of view, NB classifiers are known to be the **fastest**, learning a probabilistic generative model in just one pass of the training data. Their **accuracy** is however **relatively modest**. At the other end of the spectrum lie SVMs based on elegant foundations of statistical learning theory [7,8]. Their training time is quadratic to the number of training examples, but they are known to be the most accurate.

Support machine learning is used to classify the candidate words as acceptable or not based on the features that are supplied to it as a set of features that are populated from the text document and converted to numerical data. SVM always uses numerical inputs [7]. This thesis present an algorithm based on principles from design of experiments that can identify optimal or near optimal parameter settings in an order of magnitude faster than an exhaustive grid search for SVMs using the radial basis function kernel, and then evaluate its performance on a number of standard test problems [27, 29].

Word Prediction would be most useful for writers with physical disabilities like learning disability in reading and writing, attention deficit disorder and severe spelling problems. Word completion utilities, predictive text entry systems, writing aids, and language translation are some of common word prediction applications. Applications in modern mobile devices such as email clients, messaging tools and productivity software, require substantial text input and highlight the need for more efficient input systems.

## 1.2 Motivation

In the current world, the development of every country is dependent on the technology. The nations in a given country should have the technology implemented as per their culture, economic level, and social background like their language. If the nation is able to provide technology with its culture and make the use of the technology, it will get the road to the development in the technology. Therefore, in this thesis mobile application for Afan Oromo is developed and the features to enter the characters/word are created for the prediction to follow.

## 1.3 Research Problem

Many Oromo people use mobile phones. Oromo people use either in different level of depth. The less knowledgeable ones still use the phone without understanding the English language. So, why are those people who do not understand English not helped? Why the Oromo people cannot make their language part of the technology‟s language? In addition, the language can serve as an alternative text entry method for mobile phone messaging like SMS.

Word prediction for a language is challenging to develop, as it is necessary to study the neighboring characters, the context or meaning of the word, and the structures and occurrences of words. However, its usage can highly improve the efficiency of text inputting. The next problem we need to ask is to have  a better and fast text entry method for mobile application where the text to be entered involves lots of words or even sentences.

## 1.4 Objective

### 1.4.1. General Objective

The main objective of this research is to design a model and develop an algorithm for a word-based predictive Afan Oromo text input for systems that need text insertion, particularly for handheld devices.

### 1.4.2. Specific Objectives

To achieve the aforementioned general objectives the following specific tasks have been performed:

- Review relevant works in other languages
- Explore the existing word-based predictive models
- Develop a corpus upon which the models can be built
- Identify and adopt one of the appropriate prediction systems
- Design Afan Oromo word prediction model
- Develop the necessary algorithms for word prediction engine for Afan Oromo language
- Develop a prototype to demonstrate the effectiveness of the designed model and the developed algorithms.
- Evaluate the performance of the word prediction system.

## 1.6. Scope and Limitation

Word prediction is a tiresome, complex and a huge task that demands a lot of resources and effort. The main goal of this study is confined to the analysis and development of appropriate model for word prediction algorithm that is context aware for Afan Oromo. The model is also implemented and the prediction accuracy is estimated.

## 1.7. Methodology

The methodology to accomplish this thesis work will be as follows:

- **Literature review**: Related literatures from different sources (books, journals, Internet, etc.) will be reviewed to understand the existing predictive text entry methods and dealing with text inputs currently in use on different mobile phones.
- **Data collection**: the Afan Oromo words required for database creation.
- **Data analysis**: from words collected and database created, machines are made to learn from the analyzed data.
- **Select implementation tools**: in this study, a number of tools are used in order to come up with the solution for the problem that is going to be addressed.
- **Design**: an overall design of the text input method is done and implemented.
- **Experiment**: experiments are performed to evaluate the performance of the developed system.
- Conclusion and recommendations: with findings and recommendations for future works are given.

## 1.8. Thesis Organization

The remaining part of the thesis is organized in 5 chapters. Chapter 2 deals with Literature Review and Related Works, which are investigated on different issues. In Chapter 3, the Word Prediction Model of the system is presented. Chapter 4 deals with Implementation and Experimentation in order to show the result of the developed system. Finally Chapter 5 talks about Conclusion and Recommendations.

# CHAPTER TWO: Literature Review and Related Works

## 2.1 Literature Review

### 2.1.1 Data Entry Techniques

Most mobile applications used to apply the **Multi_press with time out, Multi_press with next button and two-key** with predictive text inputs. The earliest type of text input was the one that uses the keyboard which inserts a single character by applying Multi_press mechanism using the next button. It looked easy and has no complex process such as periodical database access**.** Multi_press with time out, Multi_press with next button and two_key inputs may access the database once the full word is entered and its purpose is mainly to check for spelling error. The predictive text input method was introduced after the artificial intelligence idea was matured. The artificial intelligence is a machine learning system that uses the learning algorithms proposed by different authors. Therefore, these input methods are largely applied for languages spoken in European countries like Dutch, Spanish, English and others [1, 2, 3, 4, 26].


Figure 2.1 The standard buttons for text inputs to be pressed.

#### 2.1.1.1 Multi-press with timeout

The multi-press with timeout technique uses a fixed period to decide when a user has finished cycling through letters on a key [Fig. 2.1]. Once the user presses another key the timeout restarts and if the same key is pressed before the timeout expires (usually 1 -1.5 seconds), the interface will cycle through the letters available for that key. For example, to enter „ABC", the user must press the „2"-key, and then wait for the time-out to expire. Next he/she presses the „2"-key twice, where the two presses are separated by less than the timeout interval. Once the timeout expires, he/she must press the „2"-key three times where the time gap between each pair of presses is less

than the timeout. In other words, „ABC“ can be entered using the key sequence 2-22-222 where a dash signifies waiting for the timeout to expire. For example, Nokia phones implement a 1.5 second timeout and a timeout-kill using the DOWN-ARROW key. Multitap is the established method for entering names into a mobile phone“s address book. As a general-purpose text input method, however, these mechanism is very slow, inefficient, and not even well received by many users [3,4].

#### 2.1.1.2 Multi-press with next button

The multi-press with next button technique replaces the timeout with a „next“ button. Instead of waiting between successive letters, the user presses the „next“ button to signify that they have finished cycling through letters on that key. To enter the string „ABC“ the user would have to press the following key sequence: 2<next>22<next>222 [1].

#### 2.1.1.3 Two-key

The two-key method takes a substantially different approach. Instead of cycling through letters on a key, two key presses are used, where the first press indicates the desired key and the second press identifies the position on that key [Fig. 2.2]. This technique means that every letter takes exactly two presses to enter, unlike multi-press where the number of presses ranges from one to three (or more for punctuation). Except for S and Z, other characters in the English alphabet are entered in an analogous manner. For example, the letter „N“ is on key „6“ at the second position, so the user would press „62“. Likewise, a „G“ would be entered using „41“ („4“-key, first position) [1, 13].

The characters S and Z must be treated exceptionally as each is fourth in its respective group and there are only three columns of keys. Accordingly, KC specially associates Key # with the three characters S, space, and Z in that order left to right. Therefore, entering „S“ requires first pressing Key # and next pressing any one key in Column 1, and entering „Z“ requires first pressing Key # and next pressing any one key in Column 3 [13].

Two-key, however, is not suited for entering punctuation or special characters, as the user needs to be able to see all the letters mapped to each key in order to determine the position of the key they desire [13].

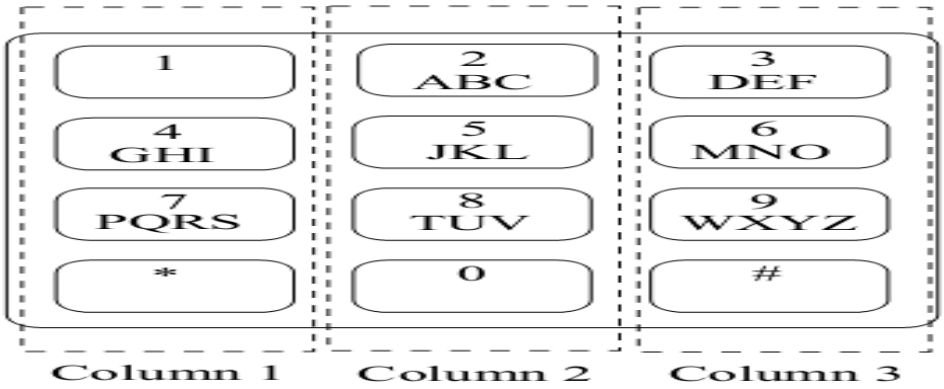



Figure 2.2 The two key text input keypad display

#### 2.1.1.4 T9

T9 by Tegic Communications is a word-based predictive method found on many commonly used phones (e.g. the Nokia 3210). As alphabet size is typically at least 26 letters, three or four letters are grouped on each key, and, so, ambiguity arises.

Dictionary-based Disambiguation can be used to overcome ambiguity by adding a dictionary to the system. Commercial examples include T9 by Tegic Communications, eZiText from ZiCorp, or iTAP from the Lexicus division of Motorola. With dictionary-based disambiguation, each key is pressed only once. After each word has been entered (any number of key presses followed by a space) the user is presented with the most likely word. If this is not the desired word, the user presses the down arrow or similar “next” key to cycle through the possible words. If the desired word is not in the list the user must delete the last group of key presses, change to multi-press mode and enter the word [10, 20, 26]. The word is then stored in the dictionary. When the assumptions used by previous research into T9 are removed (such as the user making no errors), T9's performance degrades significantly. For example, to enter “the”, the user enters 8-4-3-0. The 0 key, for SPACE, delimits words and terminates disambiguation of the preceding keys. The key sequence 8-4-3 has 3 * 3 * 3 = 27 possible renderings. The system compares the possibilities to a dictionary of words to guess the intended word. Naturally, disambiguation is not perfect since multiple words may have the same key sequence. In these cases the most-probable word is the default

or predicted first. However, if the desired word is not the most probable, overhead is incurred. For example, there are four words matching the key sequence 2-2-5-3. If the user intends calf, then three presses of a special NEXT key are required to reach the correct response. Clearly, "one key per letter" is an over simplification of user interaction with dictionary-based entry methods [20].

Prefix-based Disambiguation LetterWise was developed to avoid the problems just noted. It works with a stored database of probabilities of prefixes. A prefix is the letters preceding the current keystroke. For example, if the user presses 3 with prefix th, the most likely next letter is „e" because the in English is far more probable than either thd or thf.

#### 2.1.1.5 Letter Wise Prediction

Letter Wise by Eatoni Ergonomics takes a slightly different approach by using prefix-based disambiguation to make its predictions letter by letter. It predicts which letter is meant by a certain key-press based on the probability that a certain letter will follow the current set of letters. For example, if the text "th" has been entered, the next letter will very likely be an „e". If the user does not want an „e" they cycle through the 3 or 4 possible letters for that key until the correct letter is displayed. The statistical information is stored in a dictionary structure similar to T9. However, because LetterWise works letter by letter, there is no issue of "unknown" words. A 50% drop in required keystrokes and a 36% increase in entry rate after ten hours use has been achieved. LetterWise occasionally guesses the wrong letter, and in these cases the user must press a special NEXT key to choose the next mostly likely letter for the given key and context.

The performance of LetterWise improves with the number of preceding letters considered. In LetterWise, improved performance means fewer presses of the NEXT key. Increasing the number of preceding characters considered also increases the memory footprint of the implementation, an important consideration for mobile devices. Letterwise databases store information on a selected subset of prefixes. In practice, the memory requirements vary from about 500 bytes to 9000 bytes [20].

#### 2.1.1.6 Word Wise prediction

A second method developed by Eatoni is Word Wise, which takes a rather different approach. It uses a system where the letters „c", „e", „h", „l", „n", „s", „t", and „y" are entered unambiguously using an auxiliary key (like a "shift" key on a standard keyboard). For example, the 5-key is mapped to the letters „j", „k" and „l". Pressing 5 by itself means the user wishes to enter a „j" or a „k", while holding the auxiliary key and pressing 5 will always enter an „l". This allows words like "yes" to be entered without the system having to make any predictions. It also reduces the number of possible meanings of a group of key presses as the letters that are entered unambiguously narrow the possibilities [3, 20].

For example, the word "today" would be entered as <aux>8-6-3-2-<aux>9 which tells the system that the user wants a word matching the pattern "t???y". This reduces the amount of searching the system needs to do as well as improving the likelihood that the system predicts the word on the first attempt. The more often a predictive system can guess the right word without the user having to cycle through the possibilities, the closer the system will approach the one key press per character goal. Entire words (such as "the" and "then") can also be entered unambiguously, which means they do not need to be stored in the dictionary. However, there is still the issue of unknown words, which must be entered using multi-press, as with T9. Research shows that predictive text entry can insert more words per minute than other methods [Table 2.1].

| Method | Predicted Speed(wpm) |
|---|---|
| **Multi_press** | 14.9 |
| **Predictive_word based** | 17.6 |
| **Predictive_word completion** | 7.7 |

Table 2.1: The predictions of Dunlop and crossan (2000) for multi_press, Predictive and Predictive with word completion methods.

#### 2.1.1.7 Fitts' Law Based Prediction

Silfverberg, MacKenzie & Korhonen (2000) take a different approach to model prediction by applying Fitts' Law (Fitts 1954) to movement on the mobile phone keypad. The paper presents models for four input methods: two-key, multi-press with time-out, multi-press with next button and T9. The models are built from a series of

movement times (from Fitts' Law), $MT_i$, which are calculated using the following formula:

$$MT = a + b \log_2\left(A/W + 1\right) \quad (2.1)$$

The two constants, $a$ and $b$, are determined empirically. The log term is known as the index of difficulty and is based on $A$, the length (amplitude) of the movement, and $W$ the width of the target. The smallest dimension of each mobile phone key was used for $W$. Several versions of Fitts' Law are used, but this equation has been demonstrated to provide good predictions in a wide range of situations. The variations [eq. 2.1] are due to differences such as the direction of motion (horizontally or vertically), device weight (heavier devices are harder to move), device grasp, shape of target, and arm position (hand resting on a table or in the air) [4].

#### 2.1.1.8 Optical Character Recognition

**Optical character recognition**, usually abbreviated to **OCR**, is the mechanical or electronic conversion of scanned images of handwritten, typewritten or printed text into machine-encoded text. It is widely used as a form of data entry from some sort of original paper data source, whether documents, sales receipts, mail, or any number of printed records. It is a common method of digitizing printed texts so that they can be electronically searched, stored more compactly, displayed on-line, and used in machine processes such as machine translation, text to speech and text mining. OCR is a field of research in Pattern recognition, Artificial Intelligence and Computer Vision. Computer system equipped with such an OCR system can improve the speed of input operation and decrease some possible human errors.

Recognition of printed characters is itself a challenging problem since there is a variation of the same character due to change of fonts or introduction of different types of noises. Difference in font and sizes makes recognition task difficult if pre-processing, feature extraction and recognition are not robust. There may be noise pixels that are introduced due to scanning of the image. Besides, same font and size may also have bold face character as well as normal one. Thus, width of the stroke is also a factor that affects recognition. Therefore, a good character recognition approach must eliminate the noise after reading binary image data, smooth the image for better recognition, extract features efficiently, train the system and classify patterns.

Many people today are trying to write their own OCR System or to improve the quality of an existing one. OCR system is a complicated task and requires a lot of effort. Such systems usually are complicated and can hide a lot of logic. The use of artificial neural network in OCR applications improve quality of recognition while achieving good performance. There are two basic methods used for OCR: Matrix matching and feature extraction. Of the two ways to recognize characters, matrix matching is the simpler and more common. Matrix Matching compares what the OCR scanner sees as a character with a library of character matrices or templates. When an image matches one of these prescribed matrices of dots within a given level of similarity, the computer labels that image as the corresponding ASCII character.

Feature Extraction is OCR without strict matching to prescribed templates. Also known as Intelligent Character Recognition (ICR), or Topological Feature Analysis, this method varies by how much "computer intelligence" is applied by the manufacturer. The computer looks for general features such as open areas, closed shapes, diagonal lines, line intersections, etc. This method is much more versatile than matrix matching. Matrix matching works best when the OCR encounters a limited repertoire of type styles, with little or no variation within each style. Where the characters are less predictable, feature, or topographical analysis is superior [10, 17].

## 2.2 Related Works

### 2.2.1 Character Recognition for optical character input

The Prediction model that is going to be analyzed, uses the word and prefix contexts. So, the prefixes can be entered in different formats like OCR or digital character. In case of OCR, the character must be learned for recognition. The learning is performed by training the SVM for different characters with different styles. There will be no problem with digital characters since the characters are already recognized [10, 17].

### 2.2.2 Word Recognition

Since no work has been done on word prediction for afan Oromo using small context, in this section, the works done on other languages are discussed. And also, since the handwriting recognition uses the concept of text prediction, the researches done on handwriting recognition will be reviewed. In [34], the researchers proposed an online

handwriting recognition system for Turkish language. HTK software has been used for training and recognition purpose. To perform an experiment, two different models such as letter and word model have been designed. In the former model, a word in the lexicon is labeled according to the constituent letter models and forward-backward algorithm has been used to train the letter models and whereas in the case of word model, the model was trained for each word in the lexicon using different word samples.

The word model can also be created from consecutively lining up letter models. Esra *et. al.* used two stages to develop and test the system. In the first stage, a database of 1000 words collected from 20 people evenly has been used. In the second stage, a database of 3000 words has been used. The words of this database were collected from 30 people who have written 100 words each. Initially words have been collected from e-mail messages. Using these two databases the researchers made different experiments and achieved 97% and 95% performance accuracy for letter and word model respectively. They have recommended that to achieve a better performance there should be enough training data and because of the dots and cedillas found in Turkish alphabets the researchers concluded that stroke ordering is not suitable for online word recognition.

In [36] researchers have proposed an online handwriting recognition system for Arabic language. These researchers developed an algorithm for delayed-stroke projection which handles delayed strokes as Arabic language has dots and additional strokes with the alphabets. The delayed algorithm has involved two steps such as identifying of delayed-stroke and incorporating the stroke in the body of the word part. In addition to this, a discrete HMM (Hidden Markov Model) for the purpose of recognition task has been used. During the experiment, the training data set was used by collecting words from four persons who have written 800 words each. Similarly, the test data set has been collected from ten persons who have written 280 words each. Using these words the experiments have been conducted on five different dictionary sizes such as 5000 (5K), 10000 (10K), 20000 (20K), 30000 (30K) and 40000 (40K) words.

### 2.2.3 Text Prediction

#### *Effects of N-gram order and Training Text size on Word Prediction*

In [19, 33], the researchers believed that having predictions that are more accurate would provide a number of advantages like improving the quality as well as the quantity of message production for young people, persons with language impairments, and for those who have learning disabilities and disambiguate sequences from ambiguous keypads and correct spelling errors.

Lesher *et. al.* in addition to the establishment of a set of performance measures for word prediction using n-grams of higher orders (bigrams and trigrams) have also established experimental conditions by combining three different n-gram orders (unigram, bigram, and trigram) with seven representative testing texts of at least 2500 words for each. The content of the testing texts was independent from that of the training texts. To accomplish the above stated tasks they have used a corpus, which is evenly collected from Brown Corpus, the LOB (Lancaster-Oslo/Bergen) corpus, and the collection of time Megazine articles which has a size ranging from 100 thousand to 3 million words.

The researchers stated that irrespective of the n-gram order the performance of the system increases with an increase of **training text size**. However, the increase is much more on sounds for trigram (7.5 percentage units) than for unigrams (4.5 percentage units). Moreover, with higher n-gram orders the keystrokes savings for the given training texts increases constantly. A large jump in keystroke savings was realized when moving from unigram to bigram word prediction (6.4 percentage points at 3 million words) reflecting the transformation from context-insensitivity to context-sensitivity. The performance gain in moving from bigram to trigram prediction is considerably less dramatic (0.8 percentage points), although the difference grows for larger training texts.

Spot tests with higher order n-grams revealed an even smaller performance difference between trigram and quadgram (n = 4) word prediction, although with larger training texts this difference may also increase. This paper indicated that a corpus is the necessary element to work on a prediction system. Having larger corpus presents information about how frequently words are used in a document.

### *Ethiopic Keyboard Mapping and Predictive Text Inputting Algorithm in a Wireless Environment*

In [37], researchers proposed keyboard mapping scheme called ETWireless KeyBoard to perform keyboard mapping and also proposed new algorithm called EasyET for predictive text inputting in order to achieve practical and efficient composing of message texts with Ethiopic scripts on a wireless device in particular on a wireless phone. They have also tried to eliminate four sets of characters in order to reduce the number of characters in the font set. These reduced sets are „Superfluous" characters, replaceable characters, Ethiopic numerals and labialized characters. By reducing these sets they have reduced the number of characters used to write the messages by 40%. They have studied two entry methods such as Multi-press and Three-key input methods as well. In order to determine the frequency of words in the dictionary they have used three rating attributes. Moreover, they have developed a wireless application for Ethiopic text messaging.

# CHAPTER THREE: Word Prediction Model for Afan Oromo

## 3.1. The Word Prediction Model

Word prediction is a research area where a very challenging and ambitious task is faced, with methods coming from Artificial Intelligence, Natural Language Processing and Machine Learning. The main goal of word prediction is guessing and completing the word a user is willing to type. Word predictors are intended to support writing and are commonly used in combination with assistive devices such as keyboards, virtual keyboards, touchpad and pointing devices. Another potential application is in text-entry interfaces for messaging on mobile phones and typing on handheld and ubiquitous devices (e.g. PDAs or smartphones).

Prediction methods have become quite known as largely adopted in mobile phones and PDAs, where multitap is the input method. Nuance T9 (formerly Tegic Communications T9) and Zi Corporation eZiText are commercial systems that adopt a very simple method of prediction based on dictionary disambiguation. At each user keystroke the system selects the letter between the ones associated with the key guessing it from a dictionary of words: hence they are commonly referred to as letter predictors. Letter predictors bring a Keystroke Saving (KS) but it has been proven to be not completely free from ambiguities that are more frequent for inflected languages. Therefore, it is not surprising that these methods had a great success for none inflected languages such as English and Afan Oromo. The limited number of inflectional forms lead to very high KS that, now, are above 40%.

Word prediction is a more sophisticated technique within recent research. Differently from letter predictors, word predictors typically make use of language modeling techniques, namely stochastic models that are able to give context information in order to improve the prediction quality. Most of the literature related to word prediction concerns non-inflected languages. Language Models and prediction techniques are presented that allow the user to save more than 50% of keystrokes. The language that the system has to model influences the prediction techniques; inflected languages pose a harder challenge to prediction algorithms, since they have to deal with a usually high number of inflected forms that dramatically decrease Keystroke Saving.

To simplify the task of predicting the correct form, some techniques provide a two-step procedure, choosing first only among word "roots", and proposing all the possible word forms only when the user selects a root. Fast Type relies instead on Part-of-Speech (POS) and related morpho-syntactic information to provide a one-step procedure, presenting to the user a list of word forms. This procedure, combined with on-the-fly POS tagging, enables Fast-Type to boost performances, cutting off the prediction list all words whose gender, number, tense or mood are not consistent with the sentence context. The prediction list becomes also a "guide tool" to write syntactically correct sentences [10, 11, 12, 26].

Word prediction techniques are well-established methods in the field of AAC (Augmentative and Alternative Communication) that are frequently used as communication aids for people with disabilities.

- Accelerate the writing;
- Reduce the effort needed to type;
- Suggest the correct word (no misspellings).

Word prediction uses a language model to statistically predict the most likely next word based on what has been typed so far.

Since this chapter is dealing with the language model, the common approaches in Modeling Language are:

1. **Usual Approach**: Learn or use a model of the language
2. **Classical Approach**: Grammars
    - Every language has a grammar(regular, context free)
    - Grammars can be learned from examples
    - Deterministic grammars do not help in deciding which is the **most probable one**
3. **Better**: Probabilistic grammars
    - Probabilistic automata- Transitions have a relative frequency
    - Also called "Sequential Models"

Various methods of improving text input rates have been tried. However, the most successful and widely accepted method is word prediction. When the system is given a word by the user through the normal selection methods, it tries to guess which word or words will follow it. Most current systems use some form of statistical analysis, examining the likelihood of certain words being used in a sentence. There are two main statistical methodologies employed by current prediction systems. The first is frequency-of-usage or word counting which is used to decide which words are most often used after given words and, therefore, most likely to be used next. Research has shown that improvements in excess of 50% can be obtained from prediction using word counting. However, word counting tends not to adapt to an individual's usage, but are based on generalities about the language as a whole.

Further improvements can be obtained using recency of usage. This uses the hypothesis that if a word has been used recently, then it is more likely to be used again. The technique can be combined with frequency of usage or used independently. Assuming that the key stroke saving can be passed on directly to time saving, then this will increase the persons communication ability, which is an advance although there is still room for significant improvement. However, keystroke-savings may not result in comparable time savings because of other factors that come into play causing penalties in the usage of prediction systems. These include cognitive loading, eye-movement and other stresses. The disadvantage of „word-counters" is that they do not take into account the syntactic status of the sentence. In other words, they do not examine the sentence itself and how it is being structured, as the normal human brain does when formulating speech. Instead, it is merely looking at the overall probabilities of words being employed in a sentence and governs the prediction as such. Consequently, although the hit rate of such systems is high, it is possible for the words being predicted to be grammatically incorrect. Such words have no possibility of being used in the sentence and as such are useless to the user. However, words that are of the correct type, and would replace incorrect ones in the prediction list, have more chance of being the word the user wishes to use next.

If a filter can be placed over the statistically generated word list that removes words which are of a type which is inappropriate, given the current syntactic status, then only grammatically correct words can be placed in the prediction list. Consequently, words

of no syntactic use to the user will never appear in the list. It has been found in various studies that removal of words, which were grammatically incorrect, improved the comfort and pleasure of the user even though the actual keystroke saving was not that large. This can only lead to a decrease in cognitive loading and frustration [30].

### 3.1.1. Statistical Prediction

The majority of current commercial systems and many of the research systems employ numerical analysis for their prediction. The choice of words for placement in the prediction list is based upon the probability that they will appear in the text. The probabilities can be fixed, based on language in general, or they can be based on the user's own style and altered as the system is used. In the simplest systems, the probabilities relate only to isolated words and their likelihood of use. In more complex approaches, the statistics are based upon the probability of a word appearing given what has gone before. There are several approaches to statistical prediction and each of these is discussed in this section.

#### *3.1.1.1. Fixed Lexicon*

The simplest form of prediction is that which uses a fixed lexicon. Each of the words contained within it will have a frequency score associated with it relating to how often it is used in the language in general. These corpora are usually based on textual or written Language although they are sometimes derived from spoken language. The former form of lexicon is often at a disadvantage to the latter since the language, which is most likely to be used in a communicator device, will be more informal.

Two distinct methods are used to produce the predictions. The simpler method sorts the complete lexicon into their frequency order, and offers the few at the top of the list to the user as predictions. If the prediction that is required does not appear on the list straight away then the user types in the first letter. The list is then reduced to only those words, which have the initial letter just entered, and again, the top few are offered as predictions. This process continues until either the word has been spelt out in its entirety, or it has appeared on the prediction list, at which point the user can add it to the sentence in whichever way is made available to him by the device. The second method of employing a fixed lexicon requires more detailed corpus data. Words stored in the dictionary are tagged with the frequencies with which they appear after other given words. Consequently, when a word is entered, the most frequently

used words, which follow it, can be extracted from the corpus to produce a prediction list. The advantage of this method over the previously discussed method is that the prediction list will be much more dependent on the current status of the sentence and therefore more likely to contain correct predictions. This makes it a far more common choice for designers of prediction systems. The process of selecting a word is the same as for the previous method, although the list, which is initially drawn up, is changed each time a new word is entered rather than being a fixed list as was seen before. Regardless of which corpus the frequency statistics are based upon, there is a distinct disadvantage with this form of prediction method. Since the lexicon is fixed at the design stage of the system, it cannot be changed as it is used, so no allowance can be made for the user's habitual use of language. This means that for a given word, the predictions following it will always be the same no matter how it has been used in the past. In a few isolated cases, such as Professor Stephen Hawking, this is actually a preferred method of operation since the user has learnt to expect words to be in a certain slot on the prediction list following a given word. Consequently, they do not have to stop and read the list before commencing their selection run [4, 8, 9].

Taking an imaginary example, the researcher might know that two clicks followed by a short pause followed by another click leads to the word `namaa' after the word `mana'. If the system were to change the order in which the predictions appear, the user would have scan the list to find the position of the word every time before working out what sequence of `clicks' is required. If the user is a `fast thinker' then this could in fact slow him down considerably. However, this only applies to a limited number of potential users so a more adaptable methodology is required which can be found with the use of an adaptive lexicon.

#### *3.1.1.2. Adaptive Lexicon*

An adaptive lexicon is one, which alters the frequency tags attached to the words contained in the dictionary as the user constructs sentences. In addition to this, an adaptive lexicon may also use recency tags. These provide an indication of how recently a word has been used as this will increase the likelihood of it being used again [9]. In a similar manner to the fixed lexicon, the adaptive lexicon can provide statistics of words independently of each other, paying no attention to preceding words, or it can provide information about the likelihood of a word following another

word and how recently this was done. The only difference is that each time a word is used, its frequency and recency will be updated and stored in the dictionary for later reference. If a new word is encountered during a session, then it is added to the lexicon with a frequency of one but a high recency, so that when it is used again, it will appear acceptably high up the prediction list.

An adaptive lexicon is often prebuilt being based on similar corpora to the fixed lexicon, although this is not always the case. This gives it a base from which to work so that it can provide useful predictions from the outset. It is then altered to suit the user's individual needs as it goes along. This form of prediction is the most common in modern communication systems, including Equalizer, Profet and PAL [9]. The advantages of this method include its ability to adapt to the user's requirements, and the manner in which new words are stored without the need for any interaction with the user. However, the disadvantage of it is that no reference is made to the grammatical structure of the sentence when making predictions. Consequently, for a given word, the prediction list, which will follow it, will always be the same, regardless of the syntactical situation. This can lead to the frustrating situation of a prediction list containing grammatically incorrect words, which are of no help to the user and can exclude others, which may be of more use. Therefore, a method is required, which can bar these incorrect words from the list to make way for those with a high potential for matching the user's requirements.

#### *3.1.1.3. Bigrams and Trigrams*

The most successful form of statistical prediction examines the probabilities of words given the previous word or two words in the sentence. A lexicon, which stores the probabilities of word-pairs, is known as a **bigram**. One, which uses word-triples, is known as a trigram although it drastically increases the required number of entries in the dictionaries, since the number of combinations of three words is much higher than that of just two. In fact, it is possible to extend the number of words in probable sequences to as many as one wish although the combinations will soon become unmanageable [33, 35]. The term used for this form of lexicon is an n-gram where n is the number of words used in the probability sequences. The use of n-grams is a methodology, which is common in word recognition and is employed to narrow down the „envelope" of words from which the system can make its choice.

It is a successful method in both prediction and recognition since it relies on the fact that most European languages generally contain standard word orders, which are often repeated. In addition, the use of this technique embodies the syntax and context of a given topic of discussion without having to directly address any specific problems of either. However, the smaller n becomes, the more difficult it is to use this technique to represent these concepts accurately. One method, which has been experimented with in speech recognition, is the use of long-distance n-grams. This approach spots „key" words in the sentence, which usually give rise to certain grammatical constructs and therefore certain word sequences. Examples of such key words are relative pronouns before relative clauses or a preposition before a prepositional phrase in case of English language. Knowledge of the presence of such a key word allows the system to increase the probabilities of normal n-grams, which are relevant to the grammatical construct associated with the given key word.

The main disadvantage of all n-grams is that they require training. This means that they must be „exposed" to many examples of text in order to gather enough information to be useful to the application. The consequence of this is that the success of a given n-gram in a given application depends greatly on the text used in the training sessions. Thus in order to use an n-gram for prediction for disabled persons, it must have been trained on appropriate transcriptions of spoken discourse preferably recorded from the conversations of disabled persons and even more preferably, from the person for which the system is intended. The problems related to this form of information gathering are obvious.

### *Problems with n-grams*

The drawback of these methods is the amount of text needed to train the model. Training corpus has to be large enough to ensure that each valid word sequence appears a relevant number of times. A great amount of computational resources is needed especially if the number of words in the lexicon is big. Since the number of possible words is very large, there is a need to focus attention on a smaller subset of these. The word completion is based on a linear interpolation of n-gram models.

Finding the most likely word sequence $w_{t+1}$ , . . . , $w_{t+T}$ given a word n-gram model and an initial sequence $w_1$, . . . , $w_t$ will be determined by eq. 3.1. The decoding problem is mathematically defined as follows:

$$P(w_{t+1}, \ldots, w_{t+T} / w_1, \ldots, w_t) \quad (3.1)$$

The $n^{th}$ order Markov assumption constrains each $w_t$ to be dependent on at most $w_{t-n+1}$ through $w_{t-1}$ as shown in eq. 3.2.

The parameters of the problem are:

$$P(w_t / w_{t-n+1}, \ldots, w_{t-1}) \quad (3.2)$$

An n-gram model is learned by estimating the probability of all possible combinations of n words. The solution to overcome data sparseness problem is to use a weighted linear mixture of n-gram models.

### *N-gram POS Models*

One proposed solution consists in generalizing the n-gram model, by grouping the words in category according to the context. A mapping $\phi$ is defined to approximate a context by means of the equivalence class it belongs to (eq. 3.3):

$$P(w_i|\phi[w_{i-n+1}, \ldots, w_{i-1}]) \quad (3.3)$$

Usually, Part-of-Speech (POS) tags are used as mapping function, replacing each word with the corresponding POS tag (i.e. classification). POS tags have the potential of allowing generalization over similar words, as well as reducing the size of the language model. Clearly, a constrained application context improves the accuracy of prediction.

### *N-gram Models for Inflected Languages*

Many word prediction methods are focused on non-inflected languages (English and Afan Oromo) that have a small amount of variation. Inflected languages can have a huge amount of affixes that affect the syntactic function of every word. It is difficult to include every variation of a word in the dictionary. A morpho-syntactic component is needed to compose inflections in accordance with the context.

### *Hybrid Approach to Prediction*

Prediction can be based on either text statistics or linguistic rules. Two Markov models can be included: one for word classes (POS tag unigrams, bigrams and trigrams) and one for words (word unigrams and bigrams). A linear combination algorithm may combine these two models, incorporating morpho-syntactic information to enforce prediction accuracy.

### 3.1.2. Syntactic Prediction

In order to produce a list of predictions that excludes those, which is grammatically incorrect, one must employ a representation of the grammar of the language one is using, which can be done in many ways. One method employed in several prediction systems is the use of a statistical model, which attaches probabilities to word types indicating the likelihood of them following the current word type. This does not provide true grammatical representation, although it does allow for the possibility of the user straying from the normal rules of grammar, for instance dropping articles, which is common and serves to save time when text-entry is of the order of fifteen words per minute.

A human listener can easily understand a sentence, which is missing smaller, relatively unknown words, because the brain of a person with normal cognitive abilities can fill in the gaps, in the same way that it can recognize an incomplete picture of an object. The other advantage of this method is that it is not significantly processor intensive, which can be the case with other techniques, which employ more linguistic forms of grammatical representation. Syntax PAL and Profet[16] use this method to assemble their prediction lists. An alternative method of implementing syntactic prediction is to use a system of rules. Like most human-discourse languages, English is rigorously defined and structured. Consequently, despite the ability to concatenate infinite combinations of words into sentences, the ways in which this can be done can be described with a finite set of logical rules. If one can follow these rules whilst making a sentence rather like a chef might follow a recipe, one should be able to predict with some degree of accuracy at least the type of word, which will come next.

Syntactic prediction requires much more information than a simple syntactic approach. Primarily, it needs a grammar detailing the structure of the sentences being created in order to make choices about the types of words it can offer. Processing such a grammar will require a significant workload on the machine since it must re-parse potential sentences each time a new word is entered. In addition, the dictionary must be type-tagged. Although it is possible to ascertain a word's type automatically by examination, this is by no means so in all cases. Consequently, the majority of the

words added to the dictionary will have to be hand tagged, although the resulting predictions should be more accurate [21].

### 3.1.3. Semantic Prediction

Instead of using grammar as the main basis for one's prediction, it is also possible to draw information from the semantics of the discourse. Sometimes this is combined with syntactic prediction and serves as an extra refinement, although it can be used in isolation. When one is trying to identify the topic being discussed in a conversation, one can gain a significant amount of information just from examining certain words or sequences of words being used. If the context of the conversation is known, one will have a much greater understanding of the likely vocabulary, which will be used. A prediction system, which is able to obtain contextual information about the conversation, and has a dictionary of words, which are tagged with contextual information, will be in an excellent position to provide useful predictions. The disadvantage of this method is that it requires the representation of an enormous amount of a priori knowledge. Collecting together sets of words, which represent a particular concept, is a potentially vast task. In addition, it may be the case that this programming will have to be performed in accordance with a particular case's needs, since the potential topics of discussion of one person will not necessarily be those of another. However, this method does have uses when considering it as a refinement to another methodology [16, 21]. For instance: In the statement “Gamoo sana **calla** dheerata.” the word “calla” is semantically wrong.

#### *Prediction and Grammar*

When speech is produced, far more techniques and information are employed than is immediately obvious. It is not only linguistic knowledge which governs the phrases that we use when speaking, but also information about the topic area being discussed, knowledge of oneself and of the person or people being talked to as well as conversational and discourse plans. It is important to consider just how complex some of these processes and information stores can be before attempting to design a system as prosthesis. It is unlikely that an artificial intelligence system could be produced to replace all of these faculties, although it could be made to complement and support them. The definition of prosthesis in context of computing is a symbiosis between human and machine to perform tasks that neither could do alone. This means that each has to help the other. Thus when creating a communication prosthesis, a device is

required which can help the user to produce speech but a human mind (i.e. that of the user) is required in order to achieve this. In the case of prediction, this help is given in the form of offering single words to the user in order to speed up the rate at which words can be „spoken". However, communication prosthesis requires the user to make selections or rejections so that the system can derive the next set of predictions. Grammatical analysis can improve this symbiosis by bringing the system's manner of processing closer to that of the user by teaching it the rules of language, which people use every day.

### *Grammars for Prediction*

The majority of linguistic theories examine language from an analytic point of view. That is, they wish to understand all levels of meaning, from contextual concepts through semantic and syntactic structure to phonetic relations. However, when looking at prediction this is not necessarily the case. The aim of syntactic prediction is to ensure that the system does not offer the user grammatically incorrect words. However, the other required feature of grammatical prediction is that it must be as fast as possible. A system which provides an exact list of correct words but which takes half an hour to do so is of no use to someone who wishes to improve the speed of their communication. A model of language should be generative meaning that it should describe it completely and accurately. However, this theory was hypothesized in the context of language analysis rather than word prediction.

A trade-off must be made between the accuracy of the predictions produced and the speed at which they are derived. For instance, there is little need to deal with semantics since the user will guide this. Extra improvements could be made by modeling contextual information but the overheads involved with this would be significant. Since the speed of operation of such a system is paramount, a semantic predictor of any note can be ruled out at this stage. Therefore, efforts need only be made in the direction of a syntactic model of language.

The idea that redefining the domain of choosable words from which classical numeric prediction systems make their selections was also introduced. By passing the entire lexicon through a syntactic process in order to remove incorrect words given the current sentence situation, a pool of correct words is left. If a useful list of predictions

is to be defined then some order must be applied to this pool. Examination of current methodologies reveals that statistical prediction can offer the user a 50% keystroke-saving which in theory could double their prediction rate. These classical numeric approaches apply statistical processing to the pool of words representing the entire corpus available to the user. Obviously this includes the grammatically incorrect ones for a given sentence structure, as well as the correct ones. However, if these methods were applied to a pool of correct words then it is possible that the prediction rate would increase. In order to achieve this, a sub-lexicon must be set up which contains only syntactically correct words, given the current sentence situation. Statistical methods can then be applied to this pool to order them in rank of likelihood [Fig. 3.1]. The fact that a pool is being created means that there can be a certain degree of tolerance of incorrect words, given that statistical heuristics can be applied to ensure that the most likely ones appear at the head of the list [16].

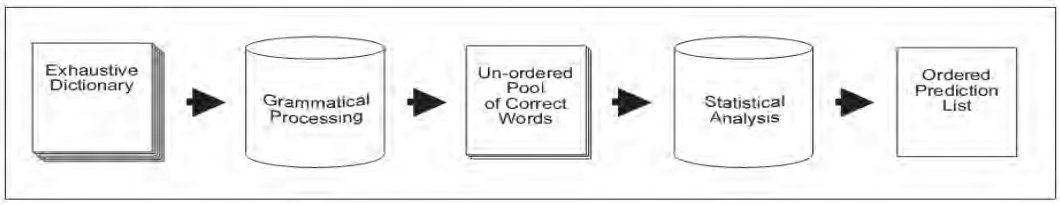



Figure 3.1 Steps for grammar based word prediction [8].

A syntactic pool, which is processed by a statistical engine, will tolerate a degree of incorrect words. If the lexicon is adaptive then the probability that these words will be used will drop to insignificance. Consequently, their presence will not adversely affect the usefulness of the system provided they are not too high in number. Thus, a grammar, which is suitable for prediction, does not necessarily need to produce exclusively correct words provided that the majority of them are. This allows the required high speed to be realized. The next step is to decide the format, which a grammar suitable for prediction should take [Fig. 3.2]. Word prediction can save significant time entering words [23].

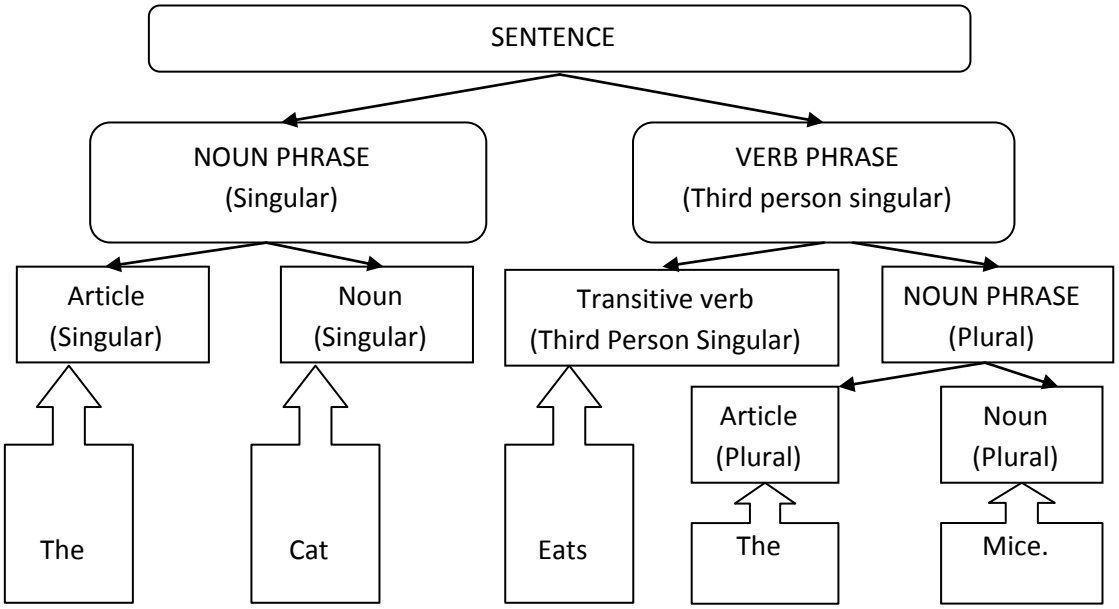



Figure 3.2 POS of words as they appear in a sentence [8].

### 3.1.4 SVM LEARNING AND PREDICTION

SVM is an inductive learning technique for two-class classification. A significant elaboration of SVM theoretical and empirical justification has been presented in the following review. Moreover, SVM was extensively applied in various areas and achieved remarkable results. For example, in text categorization, SVM was investigated extensively and proved to be one of the best learning algorithms. In the present method, for a given confusion set $\{w_x , w_y\}$, we construct one feature vector for each $w_x$ and each $w_y$ instance in the training text. Thus, these vectors will be the training examples, and we divide them into two classes, one for $w_x$ vectors and one for $w_y$ vectors. Then SVM trains on these two classes and produces a classifier (model). Thus, we construct with SVM a classifier for each confusion set. The created classifier is then used in the prediction phase to predict the word in the given context. Of course, in the prediction process, we construct a feature vector in the same way as in the training process [6, 7, 8, 17, 18, 23].

#### *Basic concept of SVM*

SVMs were developed by Cortes & Vapnik (1995) for binary classification. Their approach may be roughly sketched as follows:

**Class separation**: basically, the optimal separating hyper-plane between the two classes by maximizing the margin between the classes' closest points is created. The

points lying on the boundaries are called support vectors [Fig. 3.3], and the middle of the margin is the optimal separating hyperplane [Fig. 3.3];

**Overlapping classes**: data points on the wrong side of the discriminant margin are weighted down to reduce their influence (which is called "soft margin")[Fig 3.8].

**Nonlinearity**: when a linear separator cannot be found, data points are projected into an (usually) higher-dimensional space where the data points effectively become linearly separable (this projection is realized via kernel techniques);

**Problem solution**: the whole task can be formulated as a quadratic optimization problem which can be solved by known techniques.

A program able to perform all these tasks is called a **Support Vector Machine**.

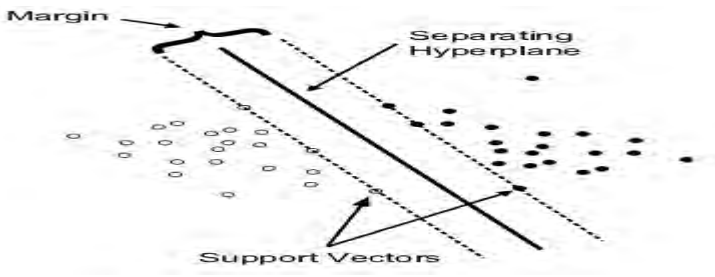



Figure 3.3 Basics of SVM [5].

### *Why Should SVMs Work Well for Text Categorization?*

To find out what methods are promising for learning text classifiers, one should find out more about the properties of text [7, 24, 27, 32].

**High dimensional input space:** When learning text classifiers, one has to deal with very many (more than 10000) features. Since SVMs use over fitting protection, which does not necessarily depend on the number of features, they have the potential to handle these large feature spaces.

**Few irrelevant features:** One way to avoid these high dimensional input spaces is to assume that most of the features are irrelevant. Feature selection tries to determine these irrelevant features. Unfortunately, in text categorization there are only very few irrelevant features. All features are ranked according to their (binary) information gain. Features ranked lowest can still contain considerable information and are somewhat relevant. A classifier using only those worst features has a performance much better than random. Since it seems unlikely that all those features are completely redundant, this leads to the conjecture that a good classifier should combine many

features (learn a dense concept) and that aggressive feature selection may result in a loss of information.

**Document vectors are sparse**: For each document, the corresponding document vector contains only few entries, which are not zero. Both theoretical and empirical evidence for the mistake bound model that additive algorithms, which have a similar inductive bias like SVMs, are well suited for problems with dense concepts and sparse instances.

C**ategorization problems are linearly separable**: Most text categories are linearly separable [Fig 3.7]. The idea of SVMs is to find such linear (or polynomial, RBF, etc.) separators. These arguments give theoretical evidence that SVMs should perform well for text categorization.

#### *3.1.4.1 Parameter selection for support vector machines*

Support vector machines (SVM) are a powerful machine learning method for both regression and classification problems. However, the various SVM formulations each require the user to set two or more parameters that govern the training process, and those parameter settings can have a profound effect on the resulting engine‟s performance. The first issue is deciding how to evaluate the parameter‟s effectiveness, which is just the standard problem of evaluating machine learning method performance. The most common method is some form of N-fold cross-validation, but other approaches such as leave-one-out test are also possible. All these methods require that the machine learning engine to be trained multiple times in order to obtain a single performance number for a single parameter setting. The most common and reliable approach to parameter selection is to decide on parameter ranges, and to then do an exhaustive grid search over the parameter space to find the best setting. Unfortunately, even moderately high-resolution searches can result in a large number of evaluations and unacceptably long run times. The best approach is to start with a very coarse grid covering the whole search space and iteratively refine both the grid resolution and search boundaries, keeping the number of samples at each iteration roughly constant. This is a variant on a standard search method from the design of experiments field. SVM parameter selection can be viewed as an optimization process [7].

### *3.2. Architecture of the System*

The architecture of the system has two parts. The first part, Character Recognition Engine and The second part of the architecture, Word Prediction Engine, which will be discussed in detail. The recognized characters from the character recognition engine and the list of all words in the Afan Oromo lexicon will be provided as an input to the proposed Word Prediction Engine. As it has been mentioned in this chapter earlier, every word pairs in the lexicon has a frequency value, which is assigned from the corpus. To start predicting the intended word the prediction engine needs two or more characters. Hence, the character recognition process will be done repeatedly at least twice and at most n times, where n is the length of the word, as there will be a probability that a user writes a word which does not match any word in the lexicon (dictionary) [10].

The Word Prediction Engine has three components, which are participating in the prediction process. These are Start Engine, Word Selector and Word Ranker. The ***Start Engine*** component, for the first time, gets two recognized characters one by one from the character recognition engine. After these two characters are received, this component waits for the period set up, to initiate the Word Selector component. On the next process, in case the intended word is not predicted, the Start Engine component initiates Word Selector component after getting every recognized character. Once the ***Word Selector*** is initiated, it will start searching for words in the dictionary (lexicon) of which their first two or more characters match with the recognized characters. The list of words found and their corresponding frequency will be delivered to the next component, Word Ranker, of the word prediction engine**.**

***Word Ranker,*** by considering the frequency of each word, provides a rank to each word in the list of found words. Words with highest frequency will get highest rank and those with least frequency will get the least rank. In the case of two or more words having the same frequency, the Word Ranker will decide the rank of the words by considering their alphabetical order. These ranking policies are used to determine the word(s) to be predicted. As mobile devices have screen size constraints, it is not possible to display large number of predicted words so that a user can select his/her desired word from the list. Thus, some trials are done to decide the convenient number of words to be displayed. Because of these trials, it is found that displaying four words

at a time will be much convenient. Hence, the Word Prediction Engine will display the top ranked words as an output of the system [Fig 3.4].

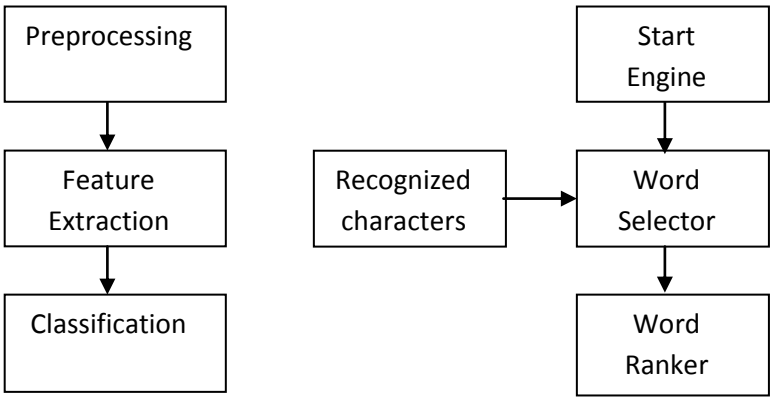



Figure 3.4: The word predictor Architecture

## 3.3. Prediction Algorithm

Creating Better Language Models is to increase the information about the language contained in the model structure

- **Lexical** information: parts of speech, word classes, semantics…
- **Structural** information: phrase construction, attachment, hierarchy …

The "grammatical [1] type" of word:

- Verb, Noun, Adjective, Adverb, Article, … for English language
- Xumura, Maqaa, Ibsa Maqaa, Ibsa Xumuraa, walqabsiiftuu,… for Oromo

The goal of Part-Of-Speech Tagging is to assign the correct part-of-speech to each word (and punctuation) in a text. Then, Learn a *local* model of POS dependencies, usually from pre-tagged data.

### 3.3.1 Hidden Markov Models

HMM is a graphical model that involves two sequences that are associated based on probabilistic dependencies. One of these sequences, called an observation symbol sequence, is observable (known), while the other sequence, called a hidden state sequence, is not directly observable. The process of inferring the most likely hidden state sequence from an observation symbol sequence is called decoding. Because all possible values of a hidden state are known in most HMMs, decoding is a process of finding the most likely state value assignment to a hidden state sequence from an

observation symbol sequence using probabilistic dependencies. The N-most likely sequences found by the decoding process are returned as auto-completion candidates. All possible values of a hidden state are collected from a corpus of text. HMM is used to extract word pairs and part of speech pairs[Fig. 3.5]. Since words are ambiguous in terms of their parts of speech, the correct part of speech is usually identified from the context the word appears in.

POS is generated as candidate POS of the word, and each POS randomly generates a word.

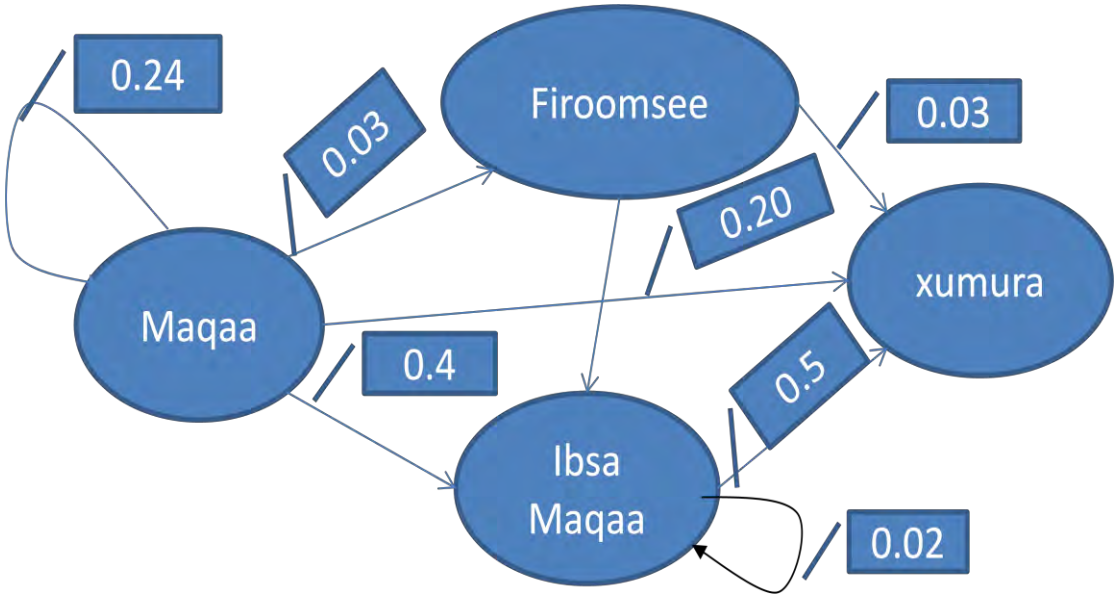



Figure 3.5 HMM model for Parts of Speech in Afan Oromo

In the figure above each states (POS) have their own words associated with them.

### *HMMs For Tagging*

First-order (bigram) Markov assumptions:

- Limited Horizon: Tag/POS depends only on previous tag $P(t_{i+1} \mid t_i)$
- Time invariance: No change over time [eq. 3.4]. So Data Base can be used for the order.

$$P(t_{i+1} = t^k \mid t_i = t^j) = P(t_2 = t^k \mid t_1 = t^j) = P(t^j \rightarrow t^k) \qquad (3.4)$$

- Probability of getting word $w^k$ for tag $t^j$: $P(w^k \mid t^j)$

  **Assumption:** *Not* dependent on other tags or words!

Combining Probabilities that shows tag sequence probability given their bigram database and how much the word bigrams are relevant in relation to the given tag sequences given are determined as follows:

- Probability of a tag sequence[eq. 3.5]:

$$P(t_1 t_2 \ldots t_n) = P(t_1)P(t_1 \rightarrow t_2)P(t_2 \rightarrow t_3)\ldots P(t_{n-1} \rightarrow t_n) \quad (3.5)$$

Assume $t_0$ – starting tag:

$$= P(t_0 \rightarrow t_1)P(t_1 \rightarrow t_2)P(t_2 \rightarrow t_3)\ldots P(t_{n-1} \rightarrow t_n) \quad (3.6)$$

- Probability of word sequence *and* tag sequence[eq. 3.7]:

$$P(W,T) = \Pi_i \, P(t_{i-1} \rightarrow t_i) \, P(w_i \mid t_i) \quad (3.7)$$

$$P_{MLE}(w_i / w_{i-1}) = \frac{C(w_{i-1}, w_i)}{C(w_{i-1})} \quad (3.8)$$

$$P(t_i / t_{i-1}) = \frac{C(t_{i-1},\ t_i)}{C(t_{i-1})} \quad (3.9)$$

Equations 3.8 and 3.9 shows how the probabilities, that are going to be used as feature in SVM feature vector, of word pairs and POS pairs are calculated.

### 3.3.2 SVM classification: Support Vector Machines basics

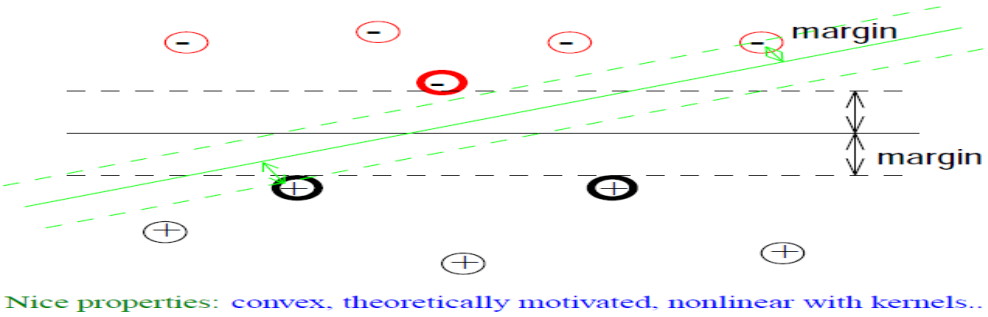



Figure 3.6 Best Hyperplane and Margin concept of SVM [5].

Machine learning is about learning structure from data. SVM is a supervised learning method to build the model a number of training examples need to be prepared and labeled with the appropriate category identifier. SVM is a well-established technique for case categorization, each described by a feature vector (i.e. a set of numerical attributes), is used to construct the optimal hyperplane that separates the original set into clusters of vectors of the same category, with the minimum error [Fig 3.6].

SVM is known for structural risk minimization. A fancy term, but it simply means; we should find a classifier that minimizes the sum of training error (empirical risk) and a term that is a function of the flexibility of the classifier (model complexity). Some training algorithm may try to minimize the training error, which in turn may increase

the test error. SVM is very good in avoiding this problem. The basic idea behind the text classifier is to supply the classifier with a set of feature vectors and the desired outcomes ($X_i$ ,$Y_i$ ) [eq. 3.10 and 3.11] where

$X_i = <f_1 \ldots.fn>$ which is a tuple of n features (3.10)

$Y_i = \{\pm 1\}$, ($y_1$ $y_2$ ….. $y_k$ ) is the desired output given $X_i$ (3.11)

The set $S=\{<X_o ,Y_o >,\ldots.,<X_l ,Y_l >\}$ will be referred as the training set of l instances. The classifier then tries to generalize this information in a training phase to be able to make statements about previously unobserved examples. SVMs have terrific classification performance but are very slow both when it comes to training and the actual classification phase [7, 15, 18].

### *Finding the Decision Boundary*

Let $\{x_1 , ..., x_n \}$ be our data set and let $y_i \in \{1,-1\}$ be the class label of $x_i =(f_1 ,f_2 ,\ldots,f_n )$ where $f_n$ is the nth feature binary value.

The decision boundary should classify all points correctly[Fig. 3.7 and eq. 3.11].

$y_i (w^T x_i +b) \geq 1$, for all inputs and outputs. The decision boundary can be found by solving the following constrained optimization problem.

The decision boundary should be as far away from the data of both classes as possible, the optimization should maximize the margin between the hyperplane. Therefore, the mapping: X, Y needs to be learned, where x ε X is some object and y ε Y is a class label. Training sets and prediction models take input/output sets X, Y for training set ($x_1$ , $y_1$),…, ($x_m$, $y_m$) and make ”generalization”: given a previously seen x Є X, find a suitable y ЄY i.e., want to learn a classifier: y = f(x,α), where α are the parameters of the function.

### *Linear SVM*

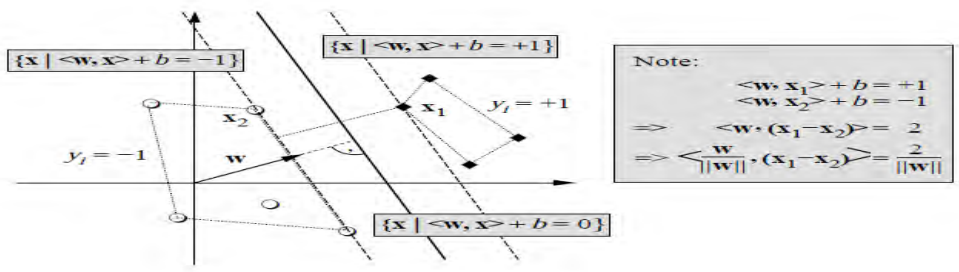



Figure 3.7 Linear SVM [5].

### *SVM non-separable case*

To deal with the non-separable case, one can write the problem as:

Minimize:

$$\|w\|^2 + C\sum_{i=1}^{m} \xi_i \,, \tag{3.12}$$

C: tradeoff parameter between error and margin

$\xi_i$ , are "slack variables" in optimization

Subject to: $y_i(w.x_i + b) \geq 1 - \xi_i \,, \quad \xi_i \geq 0$ (3.13)

This is just the mean error:

$$\frac{1}{m}\sum_{i=1}^{m} l(w.x_i + b,\ y_i) + \|w\|^2 \tag{3.14}$$

Except $l$ is no longer the zero-one loss, but it is called the "hinge-loss":

$l(y,\ \hat{y})$ = max(0, 1-y$\hat{y}$ ). This is still a quadratic program (y$\hat{y}$)!

### *Non-separable data points*

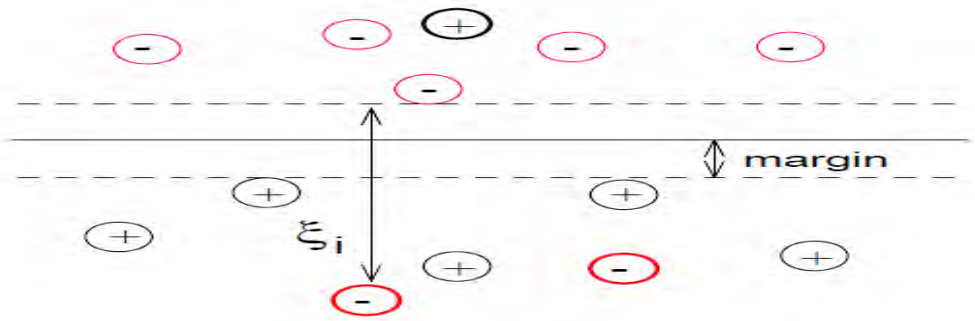



Figure 3.8 Soft Margin or Classification in SVM [27].

Decision function:

$$f(x) = w.x + b \tag{3.15}$$

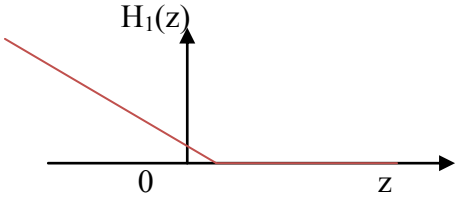
Primal formulation :

$$\text{Min P(w, b)} = \frac{1}{2}\|w\|^2 + C\sum_i H_1[y_i f(x_i)] \tag{3.16}$$

Maximize margin  minimize training error

Ideally, $H_1$ would count the number of errors, approximate with:

Hinge Loss $H_1(z)$ = max(0, 1-z)



### *SVM: non-linear case*

Linear classifiers are not complex enough sometimes.

**SVM solution**: Map data into a richer feature space including nonlinear features, then construct a hyperplane in that space so all other equations are the same!

Formally, preprocess the data with: $x \longrightarrow \Phi(x)$ and

then the map from Φ(x) to y [eq. 3.17]:

$$f(x) = w.\ \Phi(x) + b \qquad (3.17)$$

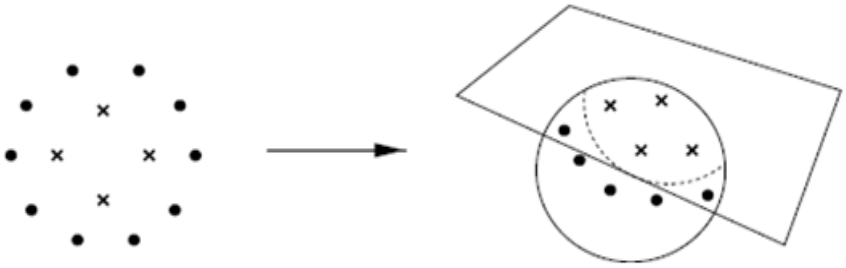


Figure 3.9 Low Data Dimension to Higher Data Dimension Mapping [27].

### *SVM : the kernel trick*

Problem: the dimensionality of Φ(x) can be very large, making w hard to represent explicitly in memory and hard for the quadratic programming to solve.

The Representer theorem (Kimeldorf and Wahba, 1971) shows that (for SVM as a special case):

$$W = \sum_{i=1}^{m} \alpha_i \Phi(x_i) \quad \text{for some variables } \alpha. \qquad (3.18)$$

Instead of optimizing w directly, we can thus optimize α.

The decision rule is now:

$$f(x) = \sum_{i=1}^{m} \alpha_i\ \Phi(x_i)\ .\Phi(x) + b \qquad (3.19)$$

The $K(x_i, x) = \Phi(x_i)\ .\Phi(x)$ is called the kernel function. Kernel function K (.,. ) is used to make (implicit) nonlinear feature map,

Example

- Polynomial kernel: $K(x, x') = (x.x'+1)^d$ (3.20)
- RBF kernel: $K(x, x') = \exp(-\gamma\|x - x'\|^2)$ (3.21)

The RBF kernel [eq. 3.21] is one of the most popular kernel functions. It adds a "bump" around each data point [Fig 3.10]:

$$f(x) = \sum_{i=1}^{m} \alpha_i \exp(-\gamma\|x_i - x\|^2) + b \qquad (3.22)$$

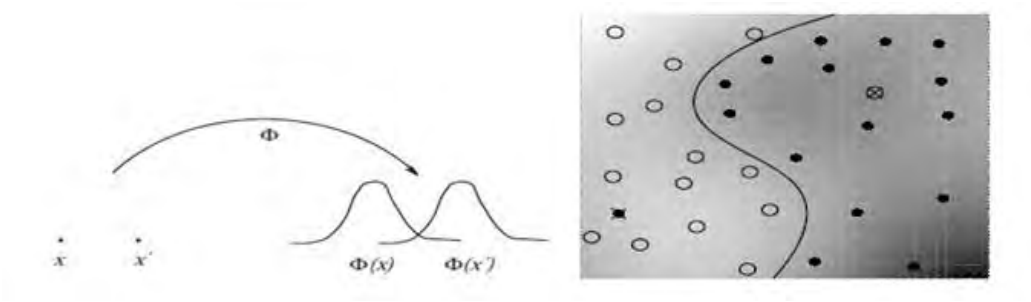



Figure 3.10 Kernel Function for Data Categorization [24].

### *Recap of Constrained Optimization*

There must exist $\alpha_i \geq 0$ for i=1… m such that $x_o$ satisfy

$$\frac{\partial(f(x)+\sum_i \alpha_i g_i(x))}{\partial x}\Big|_{x=x_o} \quad (3.23)$$

subject to $g_i(x) \leq 0$. The function $f(x) + \sum_i \alpha_i\, g_i(x)$ is called the Lagrangian and $\alpha_i$ is the Lagrangian multiplier. The gradient is set to zero. So, the optimization problem becomes

$$\text{Minimize} \quad \frac{1}{2}\|w\|^2 \quad (3.24)$$

$$\text{Subject to } 1\text{- } y_i\,(w^T x_i + b) \leq 0$$

Eq. 3.24 implies maximizing the margin($2/\|w\|$) of the hyper plane.

The Lagrangian is:

$$L(\alpha, w) = \frac{1}{2}\|w\|^2 + \sum_i^n \alpha_i\,(1\text{- } y_i\,(w^T x_i + b)) \quad (3.25)$$

Note that $\|w\|^2 = w^T w$;

Setting the gradient of L with respect to w to zero [eq. 3.25], we have:

$$W + \sum_{i=1}^n \alpha_i(-y_i)x_i = 0, \quad (3.26)$$

From eq. 2.26: $W = \sum_{i=1}^n \alpha_i y_i x_i$

Setting the gradient of L with respect to b to zero [eq. 3.25], we have:

$$\sum_{i=1}^n \alpha_i y_i = 0; \quad (3.27)$$

Change all inner products to kernel functions. For training,

$$\text{Original: } \begin{cases} \text{Maximize} \quad W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1,j=1}^n \alpha_i \alpha_j y_i y_j x_i^T x_j \\ \text{Subject to :} \qquad C \geq \alpha_i \geq 0, \sum_{i=1}^n \alpha_i y_i = 0 \end{cases} \quad (3.28)$$

$$\text{With Kernel Function: } \begin{cases} \text{Maximize } W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1,j=1}^n \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\ \text{Subject to :} \qquad C \geq \alpha_i \geq 0, \sum_{i=1}^n \alpha_i y_i = 0 \end{cases} \quad (\mathbf{3.29})$$

Larger C (Penalty factor) [eq. 3.29] helps us to generate smaller number of support vectors.

### *SVM Features For Text Categorization*

A feature can be a characteristic quantity at different linguistic levels. To transform a document, this can be regarded, as a string of tokens, into another set of tokens will lose some linguistic information such as word sequence. Word sequence is crucial for a human being to understand a document and should be crucial for a computer. Using phrases as features is a partial solution for incorporating word sequence information into text categorization. The effectiveness of different classifiers by using single tokens, phrases, stemmed tokens, etc. as features is the first problem in text categorization. The second problem is how to quantify a feature. A feature weight should show the degree of information represented by local feature occurrences in a document, at a minimum. A slightly more complicated feature weight scheme may also represent statistical information of the feature‟s occurrence within the whole training set or in a pre-existing knowledge base (taxonomy or ontology). A yet more complicated feature weight may also include information about feature distribution among different classes [7].

**Important features for SVM:**

- The trigram words validity
- The bigram words validity
- Being dictionary entry
- Number of prefixes used
- Pos trigram validity
- Pos bigram validity

### *Feature encoding for SVM (See Table 3.1)*

| Feature | value |
|---|---|
| **Words Trigram valid?** | 0/1 |
| **Words Bigram?** | 0/1 |
| **In Dictionary?** | 0/1 |
| **Abs(prefix)<=3?** | 0/1 |
| **Pos bigram?** | 0/1 |
| **Pos trigram?** | 0/1 |

Table 3.1: Feature encoding for SVM text classification.

***How to implement/ the algorithm?***

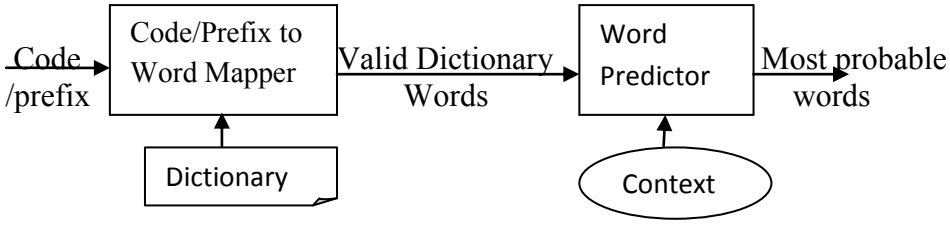



Figure 3.11 Context Based Word Prediction model

Code is the numerical sequences used to represent characters in the mobile phones. Example, 6-2-6-2 may be the code for words like “mana”, “nama” on standard 12-key mobile phones.

Dictionary is created for different words in the language and the corpus is used to take the context words into consideration and determine the most frequent next word for the context provided so far (for learning). Context is the words entered so far to the system. The word prediction is like context aware in that it senses the possible word that the user is intending to enter to the editor. SVM is used to create class of the candidate words while HMM is used to determine the most probable word based on the context, **so far words** entered to the system [Fig 3.11].

## 3.4. Summary

Linguistics is the study of language structures. Computational linguistics is a research area that combines knowledge from linguistics and computer world. By creating statistical models of languages one can make language processing automatic by a computer. Training data is used to train these models so that conclusions about previously unseen data can be drawn. So, in this chapter the SVM with RBF is used to model the prediction system and different features are identified and the methods of feature encoding are also sited to implement the system. The SVM parameter identification is identified too. The implementation is going to be discussed in the next chapter.

# CHAPTER FOUR: Implementation and Experiment

In this chapter, the tools and environments that are used to implement the designed algorithm and the experiment that is conducted to demonstrate the word prediction accuracy will be presented.

## 4.1. Implementation

Developing a prototype to demonstrate the validity and usability of the proposed word prediction system is one of the objectives of this work. In order to implement the algorithms and make the necessary experiment on the system, different tools and development environments are used. This section will talk about the tools and development environments used to implement the prediction algorithm.

### 4.1.1. Used Tools and Development Environment

Mobile phones use a variety of operating systems such as Symbian OS, Microsoft‟s Windows Mobile, Mobile Linux, iPhone OS (based on Mac OS X), Moblin (from Intel), and many other proprietary OSs. So far no single OS has become the de facto standard. The available APIs and environments for developing mobile applications are too restrictive and seem to fall behind when compared to desktop frameworks. The Android platform promised openness, affordability, open source code, and a high-end development framework. The Android Platform embraces the idea of general-purpose computing for handheld devices. It is a comprehensive platform features that is based on a Linux-based operating system stack for managing devices, memory, and processes. Android‟s libraries cover telephony, video, graphics, UI programming, and a number of other aspects of the device. The Android SDK supports most of the Java Platform, Standard Edition (Java SE) except for the Abstract Window Toolkit (AWT) and Swing. In place of AWT and Swing, Android SDK has its own extensive modern UI framework. Because the programming of the applications is in Java, you could expect that you need a Java Virtual Machine (JVM) that is responsible for interpreting the runtime Java byte code. A JVM typically provides the necessary optimization to help Java reach performance levels comparable to compiled languages such as C and C++. Android offers its own optimized JVM to run the compiled Java class files in order to counter the handheld device limitations such as memory, processor speed, and

power. This virtual machine is called the Dalvik VM. The familiarity and simplicity of the Java programming language coupled with Android‟s extensive class library makes Android a compelling platform to write programs. The Dalvik VM uses a different kind of assembly-code generation, in which it uses registers as the primary units of data storage instead of the stack. We should point out that the final executable code in Android, because of the Dalvik VM, is based not on Java byte code but on .dex files instead. This means you cannot directly execute Java byte code; you have to start with Java class files and then convert them to linkable .dex files.

This performance paranoia extends into the rest of the Android SDK. For example, the Android SDK uses XML extensively to define UI layouts. However, all of this XML is compiled to binary files before these binary files become resident on the devices. Android provides special mechanisms to use this XML data [38].

### 4.1.2 Android Emulator

Android SDK ships with an Eclipse plug-in called Android Development Tools (ADT). You will use this Integrated Development Environment (IDE) tool for developing, debugging, and testing your Java applications [38]. You can also use the Android SDK without using ADT; you would use command-line tools instead. Both approaches support an emulator that you can use to run, debug, and test your applications. You will not even need the real device for 90 percent of your application development. The full-featured Android emulator mimics most of the device features .

### 4.1.3 SQLite Databases

SQLite is a very popular embedded database, as it combines a clean SQL interface with a very small memory footprint and decent speed. Moreover, it is public domain, so everyone can use it. Lots of firms (Adobe, Apple, Google, Sun, and Symbian) and open source projects (Mozilla, PHP, Python) all ship products with SQLite. For Android, SQLite is "baked into" the Android runtime, so every Android application can create SQLite databases. Since SQLite uses a SQL interface, it is straightforward to use for people with experience in other SQL-based databases. However, its native API is not JDBC, and JDBC might be too much overhead for a memory-limited device like a phone, anyway. Hence, Android programmers have a different API to learn – the good news being is that it is not that difficult [38].

### 4.1.4. User Interfaces

Since the IDE used for the prototype design of the predictive system is Android, Android comes with XML file to create the user interfaces. Everything like font, text color, text style and other properties of a given text are set using xml file editor.

The user interfaces used in this prototype are TextEditor, TextView, Autotextcompletion, buttons. Autotextcompletion uses Different Adapters (Cursor or Array) to complete the text prefix. Text Editor is used to insert the texts. The buttons are used to insert the extracted HMM for words including the important statistics needed for prediction and also to update the Sqlite Database created to support the prediction process.

## 4.2. Experiment

Word prediction (WP) is an important Natural Language Processing (NLP) task in which the correct word is predicted (determined) in a given context. A common approach to handle word disambiguation-like problem is to train and apply **word bigram or n-gram model**. The word prediction problem is casted as a word classification task in which multiple candidate words are classified to determine the most correct one in the given context.

For example, in this word prediction instance: $[w_n \ldots w_3\ w_2\ w_1$ -?-] to predict the word in place of "-?-". The prediction uses word as the feature vector and then using machine learning (SVM) to train the word classifiers during the prediction phase. The candidate words are collected to the same class and in their class the words have their own frequency of occurrence, the words are listed according to their frequency in their class. The training uses the text taken from some corpus. Then their probability can also be determined from their frequency in a large corpus text. The machine learning used for text classification is support vector machine. SVM has very important property of creating clear class of some features with nearly same property.

### 4.2.1 Basic Hypothesis for multi-word

Many methods could be applied to extract the multi-words from text, such as the frequency approach, correlation approach and mutual information approach, etc. A basic hypothesis was conceived with a multi-word, that if a multi-word appears in a text and has the power enough to discriminate the category of this text from others, it should occur more than once in all the texts of this category.

### 4.2.2 Multi-word extraction

The usually adopted preprocessing methods in text mining area were employed, such as stop word elimination, stemming, sentence boundary determination. Furthermore, a handcrafted method for multi-word extraction and post-processing of the multi-words into normalized features were included in this section as well. The following sections give the details of each procedure. Based on the above hypothesis, the multi-words were extracted by the comparison between any two sentences in the same category so to find out the same consecutive matching in both sentences.

### 4.2.3 Text representation with multi-word features

The frequency of the multi-word feature in a text is considered, because long multi-word features were decomposed into short ones and texts were represented in vector space model.

$$\bigcup_{i=1}^{k} s_i \text{ , } s_i \text{ is } i^{th} \text{ sentence in the training text T.} \qquad (4.1)$$

The frequency is calculated by matching count and normalizing the count result of the multi-word. The set of functions that evaluates and assign scores to words consists of:

- Language model
- Semantic affinity
- Part of speech validity
- Dependency validity

Language models are based on sequences of words and are the most commonly used in text input today. Semantic affinity is based on the likelihood of two words appearing together in a sentence. Part of speech (POS) is a way of dividing words into sets where each set represents common grammatical properties, for example verb, adjective and noun. Just as some sequences of words are more common than others, some sequences of POS tags are more common than others. Gathering statistics on these sequences creates a sort of implicit grammar for the language. Dependency

grammars are more linguistically rooted theories about the inherent structure of a sentence and the dependency validity is very important to exploit this structure, when finding the probability of a given prediction candidate being the user intended word.

### 4.2.4 Language model

Input methods based on word frequencies are not context sensitive. This means that previous words in a sentence are not considered when sorting the word alternatives. A better way would be to increase the context and thereby achieving better disambiguation. From statistical theory comes the notion of the maximum likelihood estimate (MLE). This can be used to make estimations of how probable a sequence of words is. The estimation is based on how frequent the sequence is in a corpus.

For a sequence of n words the MLE is:

$P_{MLE}(S) = C(w_1 \ldots\ldots\ldots w_n)/N$ where N is the number of sequences of length n in the corpus. For the estimate, one needs a corpus, which contains all possible sequences to produce the probability. This is practically impossible and no corpus big enough exists.

One may decompose P(S) by

$$P(S) = p(w_1)p(w_2/w_1)\ldots p(w_n/w_1,\ldots,w_{n-1}) = \prod_{i=1}^{n} p(w_i/w_1,\ldots,w_{i-1}) \quad \textbf{(4.2)}$$

N-grams are exponential in memory usage [Fig 4.1]. Consider a corpus containing 10, 000 distinct words. To save all possible combinations and their occurrences, a matrix of size $10, 000^2$ for bigrams, $10, 000^3$ for trigrams and so forth up to $10, 000^i$ for n-grams of size I are needed. This means that a model based on n-grams larger than 3 is practically impossible to keep in memory during runtime.

| $W_{i-2}$ | $W_{i-1}$ | $W_i$ | Count($w_{i-2}$ $w_{i-1}$ $w_i$) |
|---|---|---|---|
| Nama | Jaalachuun | Gaaridha | 20 |

Table 4.1: how N-gram model is represented in the dictionary.

Common N-grams

- **Unigrams**: Always the most frequent word in the corpus is used, does not differentiate.
- **Bi-grams**: Correct word often ranks high, but not always
- **Tri-grams**: Has a perfect hit, but already suffers from data sparsity.
- **Four-grams**: Unusable.

The experiment conducted in this work, will determine the accuracy of the word prediction system for Afan Oromo language. This section presents how the test data has been collected and the word prediction accuracy of the system.

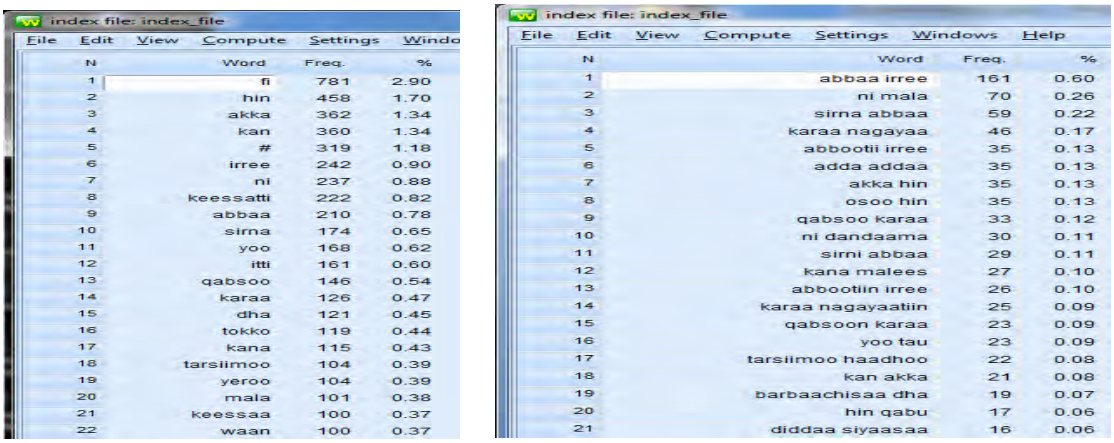
## 4.3. Data Collection: Corpus

A corpus (plural corpora) is a large collection of natural language text. It can be collected from different sources such as novels, newspapers, discussion forums, etc. Some corpora are only collected from one source while others come from multiple sources. Different corpora apply to different needs, e.g. an application used in the financial industry would not benefit from a corpus based on sports articles since the different lingo match poorly. A suitable choice of corpus is therefore important and has an impact on the application. Some corpora hold meta-information about the words in them, so called annotated corpora. Here each word is associated with a set of tags that describes, for example, the grammatical function of the word. Usage of the Meta information could for example involve training of models.

Words occurring fewer than two times in the corpus were used as irrelevant words. Punctuation marks were removed from the corpus, so they do not appear in the suggested sentence and must be entered when needed.

The test data of this work is collected from the newspaper "maxxansa Oromiyaa" ,which is published on September 4, 2010 and the book called "Hibbo fi Mammaaksa Afaan Oromoo" published in 2010. The Figures 4.2 and 4.3 are generated from a corpus collected and the statistics are used to calculate probabilities in eq. 3.8 and 3.9.

### 4.3.1. Result of the Experiment



index file: index_file

File Edit View Compute Settings Windo

| N | Word | Freq. | % |
|---|---|---|---|
| 1 | fi | 781 | 2.90 |
| 2 | hin | 458 | 1.70 |
| 3 | akka | 362 | 1.34 |
| 4 | kan | 360 | 1.34 |
| 5 | # | 319 | 1.18 |
| 6 | irree | 242 | 0.90 |
| 7 | ni | 237 | 0.88 |
| 8 | keessatti | 222 | 0.82 |
| 9 | abbaa | 210 | 0.78 |
| 10 | sirna | 174 | 0.65 |
| 11 | yoo | 168 | 0.62 |
| 12 | itti | 161 | 0.60 |
| 13 | qabsoo | 146 | 0.54 |
| 14 | karaa | 126 | 0.47 |
| 15 | dha | 121 | 0.45 |
| 16 | tokko | 119 | 0.44 |
| 17 | kana | 115 | 0.43 |
| 18 | tarsiimoo | 104 | 0.39 |
| 19 | yeroo | 104 | 0.39 |
| 20 | mala | 101 | 0.38 |
| 21 | keessaa | 100 | 0.37 |
| 22 | waan | 100 | 0.37 |

index file: index_file

File Edit View Compute Settings Windows Help

| N | Word | Freq. | % |
|---|---|---|---|
| 1 | abbaa irree | 161 | 0.60 |
| 2 | ni mala | 70 | 0.26 |
| 3 | sirna abbaa | 59 | 0.22 |
| 4 | karaa nagayaa | 46 | 0.17 |
| 5 | abbootii irree | 35 | 0.13 |
| 6 | adda addaa | 35 | 0.13 |
| 7 | akka hin | 35 | 0.13 |
| 8 | osoo hin | 35 | 0.13 |
| 9 | qabsoo karaa | 33 | 0.12 |
| 10 | ni dandaama | 30 | 0.11 |
| 11 | sirni abbaa | 29 | 0.11 |
| 12 | kana malees | 27 | 0.10 |
| 13 | abbootiin irree | 26 | 0.10 |
| 14 | karaa nagayaatiin | 25 | 0.09 |
| 15 | qabsoon karaa | 23 | 0.09 |
| 16 | yoo tau | 23 | 0.09 |
| 17 | tarsiimoo haadhoo | 22 | 0.08 |
| 18 | kan akka | 21 | 0.08 |
| 19 | barbaachisaa dha | 19 | 0.07 |
| 20 | hin qabu | 17 | 0.06 |
| 21 | diddaa siyaasaa | 16 | 0.06 |

Figure 4.1 Words unigram and bigram with their frequency

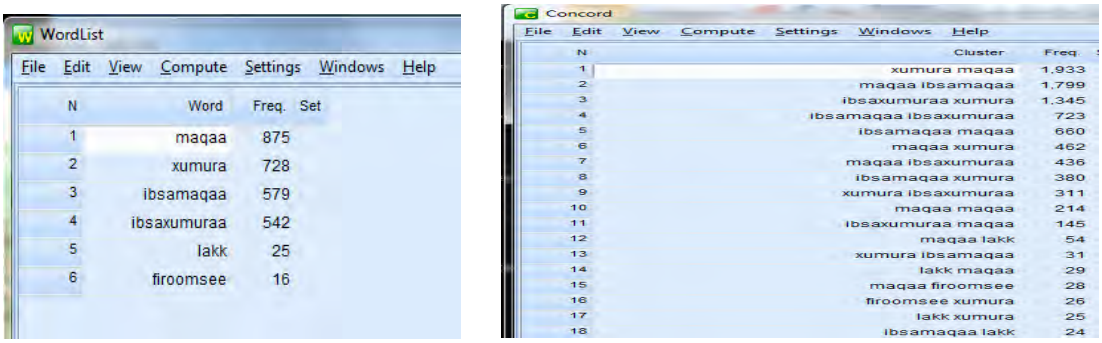



Figure 4.2 POS unigrams and Bigram statistics

### 4.3.2 SVM training

SVM training involves the parameter value selection for radial bias function or kernel and the alpha values for the decision making function. So the SVM training is setting the appropriate values for the penalty value(C) and the RBF kernel parameter ($\gamma$). Once these values are selected, they are used to determine the prediction model parameters, which are alphas (α‟s) in the decision function. The decision functions determine the appropriate words for the context the user is in at a given point of a sentence. After several trial of C and γ, the following values are selected, C=100 and γ=0.5, considering the fact that big values are used to set the number of the support vectors to a smaller number [eq. 3.21].

| Feature | value |
|---|---|
| Words Bigram? | 0/1 |
| In Dictionary? | 0/1 |
| Abs(prefix)<=3? | 0/1 |
| Pos bigram? | 0/1 |
| Relevance of words with POS? | 0/1 |

Table 4.2: The Feature encoding for the RBF kernel function

Once this encodings are done, they are used as a feature vector for the kernel function that is the X parameter of RBF [See Table 4.2]. This formulation of the SVM optimization problem is called the hard margin formulation since no training errors are allowed. Every training point satisfies the inequality $y_i f(x_i) \geq 1$ and for points xi with corresponding $\alpha_i > 0$ an equality is satisfied. Notice that one may require the separating

hyperplane to pass through the origin by choosing a fixed b=0. This variant is called the hard margin SVM without threshold. In that case, the optimization problem remains the same as above except that the constraint $\sum \alpha_i y_i$ =0 disappears.

So, the following combinations are used to train the SVM [eq. 3.29].

| Feature | $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | $X_6$ | $X_7$ |
|---|---|---|---|---|---|---|---|
| Word Bigram | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| POS Bigram | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Relevance | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| Prefix Matches | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| Decision | R | R | R | R | R | R | A |

$X_n$ >> Feature n
R>> Reject
A>> Accept

Table 4.3: The Features encoding for SVM Training

This section presents the result that is found after the experiment has been conducted. The values obtained for the $K(X_i, X_j)$ using C=100 and γ=0.5 is [eq. 3.21]:

K =

| | | | | | | |
|---|---|---|---|---|---|---|
| 1.0000 | 0.6065 | 0.6065 | 0.3679 | 0.6065 | 0.3679 | 0.2231 |
| 0.6065 | 1.0000 | 0.3679 | 0.6065 | 0.3679 | 0.2231 | 0.3679 |
| 0.6065 | 0.3679 | 1.0000 | 0.6065 | 0.3679 | 0.6065 | 0.3679 |
| 0.3679 | 0.6065 | 0.6065 | 1.0000 | 0.2231 | 0.3679 | 0.6065 |
| 0.6065 | 0.3679 | 0.3679 | 0.2231 | 1.0000 | 0.6065 | 0.3679 |
| 0.3679 | 0.2231 | 0.6065 | 0.3679 | 0.6065 | 1.0000 | 0.6065 |
| 0.2231 | 0.3679 | 0.3679 | 0.6065 | 0.3679 | 0.6065 | 1.0000 |

The Results obtained for alphas using K above [eq. 3.29]:

alphas = { 0.0000, 0.1206, 0.0000, 2.0109, 0.1206, 2.0109, 4.2630}

The value for bias (b) is b=-0.7349;

The result, shown in Table 4.4, depicts the prediction accuracy in the case the character length of the prefix is two. At prefix length two, some of words are fully written and some are left with some more characters to appear at the top of the predicted list. The prediction accuracy determined can be further improved by increasing the length of the prefix.

As it is shown in this table, the result of the experiment indicated that the average word prediction accuracy of the system is 56.88% for prefix length of two.

| words | 2-word Accuracy | | | 3-word Accuracy | | | 4-words Accuracy | | | 5-words Accuracy | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Top-1 | Top-4 | Top-10 | Top-1 | Top-4 | Top-10 | Top-1 | Top-4 | Top-10 | Top-1 | Top-4 | Top-10 |
| 8081 | 100 | 100 | 100 | 13.29 | 22.25 | 100 | 9.75 | 23.47 | 98.04 | 0 | 19.56 | 96.14 |
| | Avg =100% | | | Avg =45.18% | | | Avg =43.47% | | | Avg =38.57% | | |
| OverAll Average Accuracy is: **56.88%** | | | | | | | | | | | | |

Table 4.4: Evaluation of the accuracy of designed algorithm for context based prediction.


Figure 4.3: The predicted words after two-prefix length (mo) is inserted to the text editor

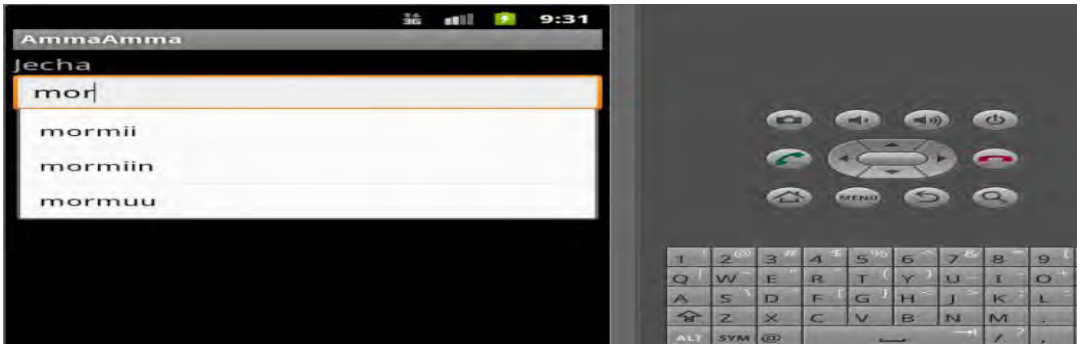



Figure 4.4 The predicted words after three prefix length (mor) is inserted to the text editor

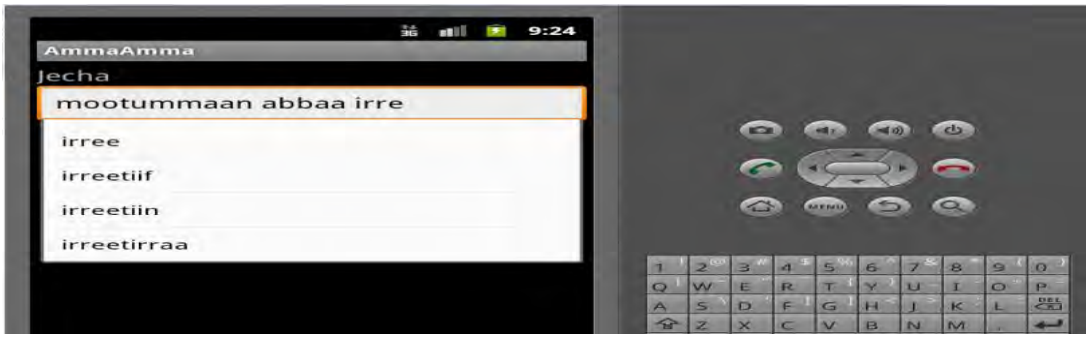



Figure 4.5 The screenshot of predicted words after 3 words and 4 prefixes.

Figures 4.3, 4.4 and 4.5 show the prediction lists obtained after the prediction algorithm SVM and HMM are implemented on android using the parameters obtained for C, gamma of RBF kernel function and alphas of SVM with RBF kernel. The first word prediction is simply predicted without feature extraction and others are listed based on the context.

### 4.3.3 Comparison with Other Input Methods

T9 was significantly faster when entering dictionary sentences (10.5 wpm vs. 7.2 wpm), while multi-press was significantly faster when entering non-dictionary sentences (8.4 wpm vs. 4.9 wpm). Subjectively, a majority (60%) of users preferred T9. As T9 remembers non-dictionary words entered by a user, it was concluded that T9's performance would tend to the 10.5 wpm speed for dictionary words with continued use. As a result, T9 was considered the better method of the two. The designed algorithm (13wpm vs. 7wpm) is significantly faster than T9.

#### *Performance Measures*

To evaluate the system, four standard performance metrics have been used in this research. Evaluation of word prediction systems considers the keystroke savings, time savings, and cognitive overload (length of choice list vs. accuracy). A predictor is considered to be adequate if its **hit ratio** is high as the required number of selections decreases.

#### *Justification for Grammatical Prediction*

One of the disadvantages of existing prediction systems is that the words produced are often grammatically incorrect. Despite being useless, these words must still be read so that the user can decide that the required word is not contained in the list. This increases the cognitive load due to the associated frustration, as well as the time overhead involved with reading through the list. Thus incorrect predictions mean the user must spend extra time, potentially without gain, but resulting in extra and unnecessary stress. In this situation, it may have been preferable for the prediction system not to be there in the first place. The aim of grammatical prediction is to reduce the envelope of search used by more conventional methods for providing predictions. This envelope should contain only words, which are syntactically correct given the current sentence structure. This subset of the lexicon can then be ordered,

according to more traditional statistical analysis, in order to produce the final prediction list. Thus the resulting words should be no less accurate whilst eliminating those which could not follow in a given sentence structure. Since unusable words are removed, there is more room on the list for words that can be used. This can only mean an increase in the likelihood of the required word being included in the list at an earlier stage, known as the prediction hit-rate. A further consequence of removing the likelihood of unusable words from the prediction list is a reduction in the levels of cognitive load experienced by the user. Grammatical prediction systems thus have the potential to offer a higher prediction hit-rate than conventional systems that use statistical analysis alone. This can result in an increase in the rate at which communication can take place. An additional advantage is the likely decrease in users' cognitive loading allowing longer periods of communicator use and lower levels of frustration.

**1) Keystroke Saving (KSS)**: It is the percentage of keystrokes that the user saves by using the word prediction system. A higher value for keystroke saving implies a better performance. Keystrokes per character (KSPC) are a useful metric for characterizing overall text entry behavior. KSPC is the number of keystrokes, on average, required to produce each character using a given input method. As a baseline, consider KSPC = 1. This is a reasonable measure for a Qwerty keyboard, because each letter has a dedicated key. KSPC < 1 is possible, for example, with word prediction techniques. KSPC > 1 is likely if the keyboard has fewer keys than symbols in the target language. An extreme example of KSPC > 1 is text input using a 5-button two-way pager. With these devices, the cursor is maneuvered over letters using four arrow buttons and then a letter is selected using the ENTER button. If letters are presented alphabetically in two rows, the effect is KSPC = 6.18. Multitap, T9, LetterWise all have KSPC> 1.

$$\text{KSPC} = \frac{\sum_{w \in L} K_w . OCC_w}{\sum_{w \in L} C_w . OCC_w} \qquad (4.3)$$

where $C_w$ is the number of characters in w, $K_w$ the number of keystrokes to enter w and $OCC_w$ the number of occurrences of w in the language L.

**2) Hit Rate (HR)**: The percentage of correct words that are predicted entering any letters of the next word. A higher hit rate implies a better performance. The Hit Rate for the designed algorithm is 100% for dictionary words.

**3) Keystroke until Prediction (KuP)**: The average number of keystrokes that the user enters for each word before it appears in the prediction list. A lower value for this measure implies a better performance. For this research it is Kup>=2 for QWERTY Keypad.

4) **Complexity**: There are two intuitive lower bounds on the computational cost of any algorithm that solves the SVM problem for arbitrary kernel matrices $K_{ij}$. Suppose that an oracle reveals which examples are not support vectors ($\alpha_i$ = 0), and which examples are bounded support vectors ($\alpha_i$ = C). The coefficients of the R remaining free support vectors are determined by a system of R linear equations representing the derivatives of the objective function. Their calculation amounts to solving such a system. This typically requires a number of operations proportional to $R^3$. Simply verifying that a vector is a solution of the SVM problem involves computing the gradient of the dual and checking the optimality conditions. With n examples and S support vectors, this requires a number of operations proportional to nS.

Few support vectors reach the upper bound C when it gets large. The cost is then dominated by the $R^3$. Otherwise, the term nS is usually larger. The final number of support vectors therefore is the critical component of the computational cost of solving the dual problem. Since the asymptotic number of support vectors grows linearly with the number of examples, the computational cost of solving the SVM problem has both a quadratic and a cubic component. It grows at least like $n^2$ when C is small and $n^3$ when C gets large. Empirical evidence shows that modern SVM solvers come close to these scaling laws.

### 4.1) Computation of the Kernel Values:

Although computing the n2 components of the kernel matrix $K_{ij} = K(x_i, x_j)$ seems to be a simple quadratic in air, a more detailed analysis reveals a much more complicated picture.

**Computing kernels is expensive**: Computing each kernel value usually involves the manipulation of sizable chunks of data representing the patterns. Images have thousands of pixels (e.g., Boser et al., 1992). Documents have thousands of words (e.g., Joachim‟s, 1999a). In practice, computing kernel values often accounts for more than half the total computing time.

**Computing the full kernel matrix is wasteful:** The expression of the gradient only depends on kernel values $K_{ij}$ that involve at least one support vector (the other kernel values are multiplied by zero). All three optimality criteria can be verified with these kernel values only. The remaining kernel values have no impact on the solution. To determine which kernel values are actually needed, efficient SVM solvers compute no more than 15% to 50% additional kernel values. The total training time is usually smaller than the time needed to compute the whole kernel matrix. SVM programs that precompute the full kernel matrix are not competitive.

**The kernel matrix does not fit in memory**: When the number of examples grows, the kernel matrix $K_{ij}$ becomes very large and cannot be stored in memory. Kernel values must be computed on the fly or retrieved from a cache of often accessed values. The kernel cache hit rate becomes a major factor of the training time. These issues only appear as constant factors in the asymptotic complexity of solving the SVM problem. However, practice is dominated by these constant factors.

**Applications of word prediction**:

- Spelling Checkers
- Mobile Phone/PDA Texting
- Disabled Users
- Handwriting Recognition
- Word-sense Disambiguation

## 4.4. Summary

The fact that a computer-based emulator was used instead of an actual mobile phone was regrettable. Users had to enter text using a keyboard and shift their eyes from the keyboard to the screen, which is a much greater distance than a user would usually have to deal with using a real mobile phone. Using a keyboard is also different to holding a mobile phone in one's hand. The keyboard may have increased the speed of text input due to the large size and slightly different layout from a mobile phone keypad. However, the extra time needed to look up at the screen will have reduced the calculated speeds of each method. Although the use of a keyboard is not ideal, the same keyboard was used for all input mechanisms, and there is no reason to believe that it will have a different impact across any of the input methods.

# CHAPTER FIVE: Conclusion and Future Works

Word prediction is a process of predicting a word that a user intends to write after the first few characters and based on a lexicon and statistical information obtained from the corpus. To accomplish the task of word prediction process for Afan Oromo language, a good collection of words, which will be used as a corpus, is necessary. Analyses have been done on the corpus prepared. These analyses used to get information like the average word-length of Afan Oromo language, the most frequently used Afan Oromo word-length and the like. These information have been used to decide the core element of word prediction engine which is N for N-gram model, where N is the number of characters after which the prediction process starts. Based on the analyses done, the value of N has been decided to be two (N=2) considering the fact that the predicted word may not come early and as compensation for prediction accuracy. It is also because of the fact that the average word length is 6.71 and many afan oromo words are of 4-letter length. Using the designed algorithm, the prediction accuracy of **56.88%** was achieved at the prefix length of 2 for words and it was seen that as the length of prefix increases, the prediction accuracy increases significantly.

As this is the first ever work in designing on Afan Oromo word prediction system that is small context aware, it is recommended to suggest the possible future works in this domain. Thus, the following works are identified as a future work:

- As it has been mentioned so many times in this work, having a complete corpus is the base to do word prediction. Thus, preparing error adequate and better size corpus must be one of the tasks that need to be done in the future. In addition to this, having a standard dictionary with maximum possible word size is very important and will increase the accuracy of the word prediction system.
- This work can be done for other local languages like Afar, Somali etc. which are languages spoken in Ethiopia and use Latin character sets.
- Integrate this word prediction system with Afan Oromo text editing systems.
- Extend this work to phrase and sentence level prediction system.

# REFERENCES


[1]. Saied B. Nesbat, "A System for Fast, Full-Text Entry for Small lectronic Devices", Proceedings of the Fifth International Conference on Multimodal Interfaces, ICMI2003 (ACM-sponsored), Vancouver, November 5-7, 2003.

[2]. Sornlertlamvanich V.et al, The State of the Art in Thai Language processing"Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL 2000), Hong Kong, pp 597-598, October 2000.

[3]. Barry McCaul and Alistair Sutherland, "Predictive Text Entry in Immersive Environments", Proceedings of the IEEE Virtual Reality 2004 (VR'04), P: 241, 2004

[4]. Hedy Kober, Eugene Skepner1, Terry Jones1, Howard Gutowitz1, and Scott MacKenzie, "Linguistically Optimized Text Entry on a Mobile Phone", http://www.eatoni.com/research/chi.pdf, Last Referenced Date: April 30, 2007

[5]. C. Cortes and V.Vapnik. Support vector networks. Machine Learning, 20:273 {297, November 1995.}

[6]. T. Joachim‟s, a probabilistic analysis of the rocchio algorithm with tdf for text categorization. In International Conference on Machine Learning (ICML), 1997.

[7]. T. Joachims. Text categorization with support vector machines: Learning with many relevant features. Technical Report 23, University at Dortmund, LS VIII, 1997.

[8]. Vladimir N. Vapnik. The Nature of Statistical Learning Theory. Springer, New York, 1995.

[9]. Y. Yang. An evaluation of statistical approaches to text categorization. Technical Report CMU-CS-7--27, Carnegie Mellon University, April 1997

[10]. Nesredien Suleiman, Word Prediction for Amharic Online Handwriting Recognition, Addis Ababa University, Computer Science department, July, 2008.

[11]. Altun, Y., Tsochantaridis, I., & Hofmann, T. (2003).Hidden Markov Support Vector Machines. In Proceedings of 20th International Conference on Machine Learning.

[12]. Rabiner, L.R., (1989) A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.

[13]. Chen S. and Goodman J., "An Empirical study of smoothing Techniques for Language Modelling," center for Research in Computing Technology, Harvard University, Cambridge, 1998.

[14]. Dumais S. T., Platt J., Heckerman D., and Sahami M., "Inductive Learning Algorithms and Representations for Text Categorization," in proceedings of the 7th International Conference on Information and Knowledge Management, USA.

[15]. EvenZohar Y. and Roth D., "A classification Approach to Word Prediction," USA, May 2000

[16]. Fazly A., "The Use of Syntax in Word Completion Utilities," Master‟s thesis, University of Toronto, Canada, 2002.

[17]. BAHLMANN, C.H, B. BURKHARDT, H. " Online handwriting recognition with Support Vector Machines- a kernel approach." Eighth International Workshop on Frontiers in handwriting Recognition,2002.

[18]. Ha, J.,Zheng, Y., Lee, G. G., "High Speed Unknown Word Prediction Using Support Vector Machines for Chinese Text-to-Speech Systems", Proc. IJCNLP, 2004.

[19]. Brown, P.F., Pietra, V.J.D., deSouza, P.V., Lai, J.C, and Mercer, R.L, "Class based N-gram Models of Natural Language," Computational linguistics 18(4):467-479, 1992.

[20]. Dunlop, M.D. and Crossan, A. "Predictive Text Entry Methods for Mobile Phones.", Personal Technologies, pp. 134-143, 2000.

[21]. Forman G., "An Extensive Empirical Study of Feature Selection Metrics for text Categorization," JMLR, vol. 3, Number one, pp. 1289-1305, 2003.

[22]. B. Scholkopf. Support Vector Learning. PhD thesis, Technical University of Berlin, 1997

[23]. T. Nakgawa, T.Kudoh, Y. Matsumoto. Unknown Word Guessing and Part of Speech Tagging using Support Vector Machines, In the Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium, 2001.

[24]. Yang, Y.M., Pedersen,(1997). A Comparative Study on Feature Selection in Text Categorization, Proceedings of the Fourteenth International Conference on Machine Learning, 412-420.

[25]. I. S. MacKenzie, R. W. Soukoreff, Text Entry for Mobile Computing: Models and Methods, Theory and Practice, Human Computer Interaction 17 (2002), pp. 147-198.

[26]. James, C. L. & Reischel, K. M. (2001), Text input for mobile devices: comparing model prediction to actual performance, in Proc of CHI2001, ACM, New York, pp. 365-371.

[27]. N. Cristianini and J. Shawe Taylor, Support Vector Machines and Other Kernel-based Learning methods. Cambridge University Press, 2000.

[28]. M. Jeon, H. Park, and J. B. Rosen. Dimensional reduction based on centroid and least squares for efficient processing of text data. In Proceedings for the First SIAM International Workshop on Text Mining. Chicago, IL, 2001.

[29]. C. Park and H. Park, Nonlinear feature extraction based on centroid and kernel functions. Pattern Recognition, to appear.

[30]. S. Shieber and R. Nelken. Abbreviated text input using language modeling. Natural Language Engineering, 13(02):165–183, 2007.

[31]. S. Han, D. Wallace, and R. Miller. Code Completion from Abbreviated Input. International Conference on Automated Software Engineering, 0:332–343, 2009.

[32]. C. Hsu, C. Chang, and C. Lin "A Practical Guide to Support Vector Classification," Technical report, 2003.

[33]. M. Damashek "Gauging Similarity with n-Grams: Language-Independent Categorization of Text," Science, 267(5199):843–848, 1995.

[34]. Esra Vural, Hakan Erdogan, Kemal Oflazer, Berrin Yanikoglu, "An Online Handwriting recognition system for Turkish", in Proceedings of SPIE Electronic Imaging Symposium, 2005.

[35]. W. Cavnar and J. Trenkle "N-Gram-Based Text Categorization," In Proceedings of SDAIR-94, pages 161–175, 1994.

[36]. Fadi Biadsy, Jihad El-Sana, and Nizar Habash, "Online Arabic handwriting recognition using Hidden Markov Models", In Proceedings of the 10th International Workshop on Frontiers of Handwriting and Recognition, 2006.

[37]. Shiferaw Abebe, Tewodros Seyoum, Solomon Atnafu, Samuel Kinde Kassegne, "Ethiopic Keyboard Mapping and Predictive Text Inputting Algorithm in a Wireless Environment", ITEs-2004, Addis Ababa, Ethiopia, 2004.

[38]. Mark L.Murphy, The Busy Coder"s Guide to Android Development, August 2008.

**Appendix A: The Statistics Generated for the collected Corpus of Afan Oromo Text**

| Text file | overall | corpusOne | corpusTwo | corpusThree | kan |
|---|---|---|---|---|---|
| Distinct words | 8081 | 2792 | 4695 | 1623 | 86 |
| MeanWordlength | 6.71 | 6.79 | 6.86 | 5.71 | 6.25 |
| 2_letter words | 1963 | 516 | 1060 | 376 | 11 |
| 3_letter words | 2809 | 809 | 1580 | 409 | 11 |
| 4_letter words | 4354 | 935 | 2722 | 675 | 22 |
| 5_letter words | 3256 | 876 | 1833 | 535 | 12 |
| 6_letter words | 3366 | 957 | 1982 | 412 | 15 |
| 7_letter words | 2825 | 786 | 1775 | 253 | 11 |
| 8_letter words | 2629 | 805 | 1654 | 158 | 12 |
| 9_letter words | 1665 | 536 | 1035 | 91 | 3 |
| 10_letter words | 1198 | 321 | 840 | 34 | 3 |
| 11_letter words | 679 | 171 | 494 | 13 | 1 |
| 12_letter words | 427 | 162 | 256 | 7 | 2 |
| 13_letter words | 217 | 59 | 155 | 3 | |
| 14_letter words | 112 | 21 | 90 | 1 | |
| 15_letter words | 35 | 11 | 24 | | |
| 16_letter words | 36 | 18 | 18 | | |
| 17_letter words | 4 | 2 | 2 | | |
| 18_letter words | 5 | 4 | 1 | | |
| 19_letter words | 5 | 3 | 2 | | |
| 20_letter words | 3 | 3 | | | |
| 21_letter words | 2 | 2 | | | |
| 22_letter words | 1 | 1 | | | |
| | | | | | |
| | | | | | |
| | | | | | |

APPENDIX B: LIST AND DESCRIPTION OF DOCUMENTS USED TO PREPARE THE CORPUS

| S.N | Title | Type | Author | Date/year |
|---|---|---|---|---|
| 1 | Maxxansa Oromiyaa | News Paper | | 2004 A.L.I |
| 2 | Kallacha Oromiyaa | " | | 2004 A.L.I |
| 3 | Bariisaa | " | | 2004 A.L.I |
| 4 | Sirna Abbaa Irreerraa Gara Dimokraasiitti | Book (Translated) | Dr. Abdulsamad, Boruu Barraaqaa | Ebla 2011 |
| 5 | Sirna Gadaa Oromoo | " | Warqinaa A/Soorii | Fulbaana, 2010 |
| 6 | Godaannisa | " | Dhaabaa Wayyeessaa | 1992 A.L.I |
| 7 | Afoola Oromoo fi Walaloowwan | " | Marshaa Yilmaa | Hagayya 2003 |
| 8 | Hiikaa Suuraa Yusuuf Dhaamsa Isaa Waliin | " | Kadir Ibraahim | 2002 A.L.I |