St. Mary's University

School of Graduate Studies

Department of Computer Science

A Hybrid Sentiment Classification for Amharic Book Reviews

By
Musa Shikur Muktar

A Thesis Submitted to

The Department of Computer Science of St. Mary's University

in the Partial Fulfillment of the Requirements for the Degree of

Master of Science in Computer Science

January 2019

St. Mary's University

School of Graduate Studies

Department of Computer Science

A Hybrid Sentiment Classification for Amharic Book Reviews

By
Musa Shikur Muktar

Thesis Examination Committee:

Internal Examiner: _____ Signature_____Date_____

External Examiner: _____ Signature_____Date_____

Dean, Faculty of Informatics: _____ Signature_____Date_____

Declaration

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

_____

Full Name of Student

_____

Signature

Addis Ababa

Ethiopia

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

_____

Full Name of Advisor

_____

Signature

Addis Ababa

Ethiopia

January 2019

Acknowledgments

First of all, I would like to express great gratitude to my research advisor **Michael Melese (Ph.D.)** for his patience, friendly treatment, many insights, and support. I would also like to thank my wife **Zinet Adane** for her love and support throughout the research work. Her contribution to the research is greatly appreciated.

Musa Shikur

Table of Contents

List of Figures

List of Tables

List of Acronyms

NB	Naïve Bayes
SVM	Support Vector Machine
MAP	Maximum a Posteriori
MaxEnt	Maximum Entropy
CPU	Central Processing Unit
TrainSet	Training Dataset
DevSet	Development Dataset
TestSets	Testing Datasets
SMS	Short Message Service
GI	General Inquirer
NLTK	Natural Language Processing Toolkit
SERA	System for Ethiopic Representation in ASCII
MI	Mutual Information Gain

Abstract

The emergence of Web technology generated a massive amount of raw data by enabling Internet users to post their opinions, reviews, comments on the web. Processing this raw data to extract useful information can be a very challenging task. Sentiment Analysis involves extracting, understanding, classifying and presenting the emotions and opinions expressed by users. We explored opinion mining as a text classification task and employed unigram as a feature set. We have performed different experiments that can be grouped into three.

In the first group (lexical classifier), we developed an algorithm to classify reviews based on the number of count of opinion words. The performance of this algorithm has been evaluated by comparing the result of lexical classifier algorithm with the actual labels of the reviews. In the second group of experiments, three popular feature selection methods Chi-Square, Mutual-Information-Gain and Galavvotti-Sebastiani-Simi (GSS) coefficient have been compared for performance in selecting a better subset of feature set. For these comparisons, three supervised classifiers Nave Bayes, Logistic-Regression and SVM have been used. Experiments on these three classifiers have been done using all three of the above feature selection methods with 750, 1000, 1250, and 1500 numbers of features. Here, It enabled us to know which combinations of feature selection methods, classifier, and a number of features work best in our domain. In the third group of experiments, we combine the lexical classifier with machine learning sequentially.

In this research work, hybrid sentiment classification has been done for classifying Amharic book reviews into positive and negative. The experiments are conducted using 600 Amharic book reviews collected from different sources like facebook, personal blogs, and manually collected from individual book readers. For machine learning, the experiment indicates that the Naïve Bayes algorithm, using Mutual Information Gain feature selection method, with 1500 number of features perform best with an accuracy of **93.33%**. The experiment also indicates a **hybrid approach with accuracy (87%)** outperform **lexical approach with 74% accuracy** but not machine learning approach which performs with an accuracy of 93.33%.

Keywords: Opinion, Sentiment Analysis, Features, Lexicon-Based Classifier, Machine Learning, Hybrid Classifier.

CHAPTER ONE

INTRODUCTION

1.1 Overview

Opinions are central to almost all human activities and are key influencers of our behaviors. Our beliefs and perceptions of reality, and the choices we make, are, to a considerable degree, conditioned upon how others see and evaluate the world (Liu, 2012). For this reason, when we need to make a decision we often seek out the opinions of others. This is not only true for individuals but also true for organizations. Individuals, organizations, and government understand the influence of opinion on decision making and they were trying to use this for their advantage.

Before the emergence of the Internet, there was a very little written text opinion available in the market (Pawar, Jawale, & Kyatanavar, 2016) and opinions were collected and analyzed manually (Khan, Baharudin, Khan, & Ullah, 2014), which is expensive and time-consuming (Younis, 2015). In that time, if an individual needed to make a decision, he/she typically asked for opinions from friends and families. When an organization needed to find opinions of the general public about its products and services, it conducted surveys and focus groups. With the rapid expansion of e-Commence, more users are becoming comfortable with the Web and an increasing number of people are writing reviews (W. Wang & Zhou, 2009). The number of reviews can be in hundreds or even thousands for a popular product. This makes it difficult for a potential customer to read them, or to make an informed decision on whether to purchase the product. It also makes it difficult for the manufacturer of the product to keep track of customer opinions. However, doing this manually is only possible to a certain extent and time-consuming job. As an example, manufacturing organizations prefer information in a format that is easier to use, so automating this process is very useful (Hu & Liu, 2004). This is where opinion mining comes in to picture.

Sentiment analysis, also called opinion mining, is the field of study that analyzes people's opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards entities such as

products, services, organizations, individuals, issues, events, topics, and their attributes (Liu, 2012). Even though Sentiment analysis has been there since the 1990s, the outbreak of computer-based sentiment analysis only occurred with the availability of subjective texts on the Web (Mäntylä, Graziotin, & Kuutila, 2018). Nowadays it has become one of the prominent research areas over the past years in computer science, especially in Natural language processing.

1.2 Statements of the Problem

The presence of social media, blogs, forums, and e-commerce web sites encourages citizens to share their opinion, emotions, and feelings publicly (Haseena, 2014). The increased popularity of these sites resulted in the huge collection of people opinion on the web in an unstructured manner (Haseena, 2014; Khan et al., 2014; Pawar et al., 2016). These very large volumes of information are very difficult to process by individuals, leading to information overload and affecting decision-making processes in organizations (S. Wang & Wang, 2008). This situation creates a new area of research called opinion mining and sentiment analysis.

The sentiment analysis results are influenced by the differences in grammar and usage of language which makes opinion mining language and domain dependent task (Bal et al., 2011). Among these languages, English is the most studied language in the field of opinion mining because of the availability of linguistic resources for analyzing opinions in English language (Mhaske & Patil, 2016). As the internet is reaching to more and more people within the world, there is a tremendous increase in the web content of other languages because people feel comfortable with their native language (R. Sharma, Nigam, & Jain, 2014). According to Internet World User by Language (2017), 26.5% of the internet users are English speaker from the top language used in the web. The availability of data in a language other than English (R. Sharma et al., 2014), and the increasing need of automatic opinion mining systems (Mhaske & Patil, 2016), has motivated many researchers to study different languages. In addition to these, Ethiopia took 3.6 % of Internet users out of Africa's share and 0.4% out of a total population of internet users in the world in 2017. The statistics also show that there was an average increase of 966,323 users of the Internet in Ethiopia during the years 2000-2017. Due to this increase in Internet population within the country and a large number of population that speaks the language in diasporas, the number of web documents that are written in Amharic language and the Ethiopic script is increasing. Opinionated Amharic documents are among these web documents that show

increment on the web, though sentiment analysis research on the Amharic language is at its infant stage (Abreham, 2014; Philemon & Mulugeta, 2014). Collecting and analyzing opinions manually is expensive and time-consuming (Khan et al., 2014; Younis, 2015) and since opinion mining is language dependent task (Bal et al., 2011), we cannot use sentiment analysis works done for other languages directly for the Amharic language. This is due to the difference in grammar and other behavior of the language. For example, word order in Amharic is generally subject-object-verb (SOV), with subordinate clauses preceding the main clause. Therefore, we need sentiment analysis research works on Amharic. This study investigates the use of opinion mining on Amharic book review.

The reason why we used book reviews domain is that it is relatively easy to collect book reviews; this is because nowadays, there are different groups in social media like Facebook and personal blogs that freely discuss on Amharic books and give comments.

The reason why we choose to work on book reviews domain is that;

- First, books are one of the most common products to be sold, reviewed, and their sell is highly affected by reviews (Chevalier & Mayzlin, 2006). Senecal and Nantel (Senecal & Nantel, 2004) conducted a study across multiple product categories and found that consumers relied on recommendations for experiential products like movies, books, or music significantly more than other types of products. According to another study by Sunitha and Edwin (Sunitha & Edwin, 2014), based on the customer preference towards online shopping, 'books' has been ranked first. This is because most of the customers are interested in buying books online because they can access a variety of books by sitting before the computer.
- Secondly, as far as our knowledge, sentiment analysis on the book review domain has not been done in the Amharic language.

1.3 Objective of the Study

1.3.1 General Objective

The general objective of this research work is to design and develop a hybrid sentiment classification model for Amharic book reviews.

1.3.2 Specific Objectives

To realize the above mentioned general objective, the study aims to carry out the following specific objectives:

- To analyze the general structure of Amharic statements related to opinions and sentiments so as to identify negative and positive statements.
- To select appropriate algorithms, feature selection methods, and classification approaches on Amharic book review.
- Design a model for sentiment mining from Amharic book review.
- To evaluate the model for sentence level opinion mining on Amharic book review.

1.4 Scope and Limitations

The scope of the research is to develop a sentence level opinion mining model for Amharic book review. The system is designed to analyze 600 Amharic book reviews collected from social media, personal blogs and manually collected from individual book readers and identify the polarity into positive and negative. This includes preparation of book review data, selection of appropriate algorithms, feature selection methods, and classification approaches on Amharic book review.

The following are some of the limitations of our work:

- Human beings are a complex creature and they express their filling in different ways. In this research work, the researcher does not cover complex expressions like humor, sarcasm, irony and idiomatic expression.
- The research work focuses on the classification of sentiment in to positive or negative, it doesn't cover sentiment analysis tasks like subjective or objective classification.
- And fake review identification (Opinion spam detection) not parts of this research work.

1.5 Significance of the Study

Weather in the field of politics, business, or other fields, knowing what other people think, about some political ideas, services, product or other, is a major factor in making a reasonable and correct decision. Therefore, the following are the significance of hybrid sentiment classification for Amharic book review research work:

- Publishers spend a lot of money to know the reader's opinion about Books they published. But if they use hybrid sentiment classification for an Amharic book review, they can reduce their cost of finding what customers think about the books, and increase their sell-by the indirect promotion of books through review from customers.
- Help publishers and writers in identifying faults on the book and what improvement can be made on next print.
- It is difficult for readers to find information about books manually. But, by using hybrid sentiment classification for an Amharic book review, readers can make a decision on buying the book and save themselves from unnecessary cost and west of time.
- The review data and the results of the research can be used as an input to the development of a full-fledged opinion mining system for Amharic book review.
- The system can be used to classify book reviews as positive or negative.
- The research will give insight about which approach lexical, machine learning or hybrid approach gives good result in classifying Amharic book reviews in to positive or negative.

1.6 Organization of the Thesis

This thesis report is organized into six chapters consisting of Introduction, Literature review, related works, design, experiments and evaluation, and Conclusion and Recommendations.

The first chapter gives the general introduction of the thesis that contains an overview of the study, statement of the problem, objectives of the study, scope, limitations, and significance of the study. The second chapter is a literature review and in this chapter, opinion related principles/theories have been discussed. In addition to this, related works have been reviewed. In the third chapter, methodology and techniques have been discussed. The fourth chapter is about the Amharic language. In the fifth chapter, data collection and preparation discussed. In the sixth

chapter, the design of a hybrid sentiment classification has been done. In the seventh chapter, experiment and evaluation of results are presented. In the last chapter (chapter eight), conclusion and recommendation of future work have been discussed.

CHAPTER TWO

LITERATURE REVIEW

2.1 Overview

The first section mainly discusses opinion followed by basic concepts related to sentiment analysis, and then in third section features in sentiment analysis are discussed. In the fourth section approaches of sentiment classification with their advantages and challenges are discussed. In the last section, sentiment mining related researches done for a different language such as English, Hindi, Arabic, and Amharic using different techniques and approaches are reviewed.

2.2 Opinion

Nowadays people all over the world interconnected to each other through the internet and textual information is one of the ways that people in social media prefer to pass information. This information can be broadly categorized into facts and opinion (Ojokoh & Kayode, 2012). Facts are an objective statement that can be proven true or false but opinions are subjective statements or expressions of a person's feelings that cannot be proven. Humans are subjective creatures and opinions are important. In every aspect of life people's decisions are affected by the opinion of others, therefore there must be a way to handle and use these opinions to our advantage. Because of the availability of the huge amount of opinion document and the range of application that makes use of opinion to adjust marketing strategy, develop product quality, crisis management or other, automatic sentiment analysis or opinion mining attract people these days.

An opinion is a person's belief, view, feeling, or judgment the specific object (kasthuri, Jayasimman, & Jabaseeli, 2016; Liu, 2012). It is a subjective or value judgment, and it cannot be proven. In the sentence, "This camera sucks." The word suck indicates negative sentiment on the object camera (Liu, 2012). In our day to day life, we can see the effect of opinion in our decision, the way we feel about ourselves and others. According to (Liu, 2012), others' opinions greatly influence our decision and provide guidance for individuals, governments, and others. Therefore, because of the importance of opinion, researchers and organizations focus on automatic sentiment mining or opinion mining. In addition to this, opinion has three components, the

opinion holder, the object about which the opinion expressed and the opinion itself. Whenever we want to identify opinion all the three components are important (Khan et al., 2014).

Opinion Holder or Opinion Source

Mainly opinions on certain objects are expressed by users (Pawar et al., 2016). Users may be an individual person, group, and organization. It means that these users are authors of the opinions. In the field of sentiment analysis, such users are known as the holder of an opinion. These holders of opinion are also known as opinion sources. In the case of product reviews, forum posting and blogs, opinion holders are usually the author of the post (Liu, 2012). To understand it, consider the sentence, "John expressed his disagreement on the treaty." The opinion holder in this sentence is 'John" since he is the opinion source in this sentence as 'John' is mentioned explicitly in this sentence (Liu, 2012).

Object

It is mainly any entity which can be anything in the real world i.e. person, organization, event, product topic, etc (Liu, 2012). Consider the phone as a general class. So a particular brand of the phone can be considered as an object. While expressing the opinion, one can comment on the object i.e. the phone. These opinions may be like "I don't like this phone" (pawara, jawal, and kyatanavar, 2016).

2.3 Sentiment Analysis

Sentiment analysis, also called opinion mining, is the field of study that analyzes people's opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards entities such as products, services, organizations, individuals, issues, events, topics, and their attributes (Liu, 2012). Even though there was research on sentiment earlier (Pang, Lee, & Vaithyanathan, 2002; Satoshi, Kenji, Kenji, & Toshikazu, 2002; Turney, 2001; Wiebe, 2000), according to (D'Andrea, Ferri, Grifoni, & Guzzo, 2015) the term sentiment analysis first appeared in (Nasukawa & Yi, 2003).

Even though there may be slight difference on the overall steps that can be followed, most of the time, whenever we want to analyze sentiment we follow five phases (D'Andrea et al., 2015). These phases are:

- ***Data collection***: the first phase of sentiment classification is to collect data. Data might be collected manually or automatically from personal blogs, social media, and other data sources.
- **Text preparation**: the second phase of sentiment classification is text preparation. Text preparation is the process of filtering the extracted data before analysis. In this phase, the main thing to do is to identify and eliminate non-textual content and content that is irrelevant to the area of study from the data.
- **Sentiment detection**: the third phase of sentiment classification is sentiment detection. In this phase through carefully examination subjective and, objective sentences are identified. Sentences with subjective expressions are retained and that which conveys objective expressions are discarded.
- **Sentiment classification**: sentiment classification is the fourth phase. In this phase, each subjective sentence detected is classified into groups-positive, negative, good, bad, like dislike.
- **Presentation of output**: the last phase is the presentation of output. It means converting unstructured text into meaningful information.

Having these phase of sentiment classification, based on the level of granularities sentiment analysis has been investigated mainly at three levels; Document, Sentence, and Entity and Aspect level (Liu, 2012). The document-level analysis identifies whether the overall opinion expressed is positive or negative. According to (Liu, 2012), this level of analysis assumes that the whole document expresses an opinion about the single entity and not applicable to the document that contains the opinion about more than one entity. The second level of sentiment classification is a sentence level which is concerned about, identifying which sentences express a positive, negative or neutral opinion. It considers the sentence as a basic information unit. The third level of sentiment classification is an Entity and Aspect level. Entity and Aspect level classify opinion given by users about specific aspects of an entity. Aspect level performs finer-grained analysis and Instead of looking at language constructs (documents, paragraphs, sentences, clauses or phrases), aspect level directly looks at the opinion itself (Liu, 2012). Document-level analysis and sentence level analysis are not good in identifying what opinion holder feeling or opinion about specific future of entities.

2.4 Common Features in Sentiment Analysis

Converting a piece of text into a feature vector or other representation that makes its most salient and important features available is an important part of data-driven approaches to text processing (Pang & Lee, 2008). A set of documents is used as a training set to the classifier. These documents are represented as vectors (Ghag & Shah, 2014). The following are the common features used in sentiment analysis:

2.4.1 Term Presence and Frequency

It is common in information retrieval to represent a piece of text as a feature vector wherein the entries correspond to individual terms (Pang & Lee, 2008). These features include uni-grams or n-grams and their frequency or presence (Vohra & Teraiya, 2013).

Term frequencies have traditionally been important in standard IR, as the popularity of TF-IDF (term frequency-inverse document frequency weighting shows; but in contrast, Pang et al. (Pang et al., 2002) obtained better performance using presence rather than frequency (Pang & Lee, 2008).

Term Presence and Term Frequency are two popular techniques for Information Retrieval when representing documents as vectors (Cambria, Schuller, Liu, Wang, & Havasi, 2013).

In Term Presence technique an element can take a binary value. This element is set to one if the term is present in document otherwise set to zero if the term is not present in the document. In the Term Frequency technique, an element in the document vector is a non-negative integer that is set to count of the given term in a document (Ghag & Shah, 2014).

This finding may be indicative of an interesting difference between typical topic-based text categorization and polarity classification: While a topic is more likely to be emphasized by frequent occurrences of certain keywords, overall sentiment may not usually be highlighted through repeated use of the same terms (Pang & Lee, 2008).

2.4.2 Part of Speech Information

POS is used to disambiguate sense which in turn is used to guide feature selection (Pang & Lee, 2008).In POS tagging each term in sentences will be assigned a label, which represents its position/role in the grammatical context.

Among parts of speeches, adjectives are good indicators of sentiments. The fact that adjectives are good predictors of a sentence being subjective does not, however, imply that other parts of speech do not contribute to expressions of opinion or sentiment (Pang & Lee, 2008). For example, with POS tags, we can identify adjectives and adverbs which are usually used as sentiment indicators (Turney, 2001).In a study by Pang et al. (Pang & Lee, 2008) on movie-review polarity classification, using only adjectives as features were found to perform much worse than using the same number of most frequent unigrams. The researchers point out that, nouns (e.g., "gem") and verbs (e.g., "love") can be strong indicators for the sentiment.

2.4.3 Negations

Negation is also an important feature to take into account since it has the potential of reversing a sentiment (Pang & Lee, 2008).

Using a bag-of-words representation, the supervised classifier has to figure out by itself which words in the dataset, or more precisely feature set, are polar and which are not (Wiegand, Balahur, Roth, Klakow, & Montoyo, 2011).

The standard bag-of-words representation does not contain any explicit knowledge of polar expressions. As a consequence of this simple level of representation, the reversal of the polarity type of polar expressions as it is caused by a negation cannot be explicitly modeled (Wiegand et al., 2011).

Since standard bag-of-words representation does not contain any explicit knowledge of polar expressions, a simple bag of words is not enough to handle negation. There are different ways to handle negation. The usual way to handle negation is to attaché "NOT" to words occurring close to negation terms. such as "no" or "don't," so that in the sentence "I don't like deadlines," the token "like" is converted into the new token "like-NOT (Pang & Lee, 2008).

The other way to handle negation is to consider the usage of higher order n-grams. Imagine a labeled training set of documents contains frequent bigrams, such as not appealing or less entertaining. Then a feature set using higher order n-grams implicitly contains negation modeling (Wiegand et al., 2011). This also partially explains the effectiveness of bigrams and trigrams for this task as stated in (Ng, Dasgupta, & Arifin, 2006).

2.4.4 Opinion Words and Phrases

Opinion words and phrases are words and phrases that express positive or negative sentiments (Vohra & Teraiya, 2013). The main approaches to identify the semantic orientation of an opinion word are statistical-based or lexicon-based.

2.5 Feature Selection Methods

Feature selection is the process of selecting a subset of relevant features for use in model construction (Parlar, Özel, & Song, 2018). It is effective in the reduction of large data by removing irrelevant and noisy data and chooses a representative subset of all data to minimize the complexity of the classification process in sentiment classification (Sammut & Webb, 2016). Feature selection methods improve classification accuracy and decrease the running time of learning algorithms and better model interpretability (Sammut & Webb, 2016). Here we select possible feature selection methods:

Information Gain

Information gain represents the entropy reduction given a certain feature, that is, the number of bits of information gained about the category by knowing the presence or absence of a term in a document (Adel, Omar, & Al-Shabi, 2014):

$$IG(t) = -\sum_{i=1}^{|C|} P(C_i)*\log(P(C_i)) + P(t)*\sum_{i=1}^{|C|} P(C_i|t)*\log(P(C_i|t)) + P(\bar{t})*\sum_{i=1}^{|C|} P(C_i|\bar{t})*\log(P(C_i|\bar{t})) \quad \text{.....4.1}$$

Where, $P(C_i)$ represents the likelihood of the occurrence of C_i class; $P(t)$ represents the likelihood of the occurrence of t; $P(\bar{t})$ represents the likelihood of the non occurrence of t.

Since IG is a filter technique; it can scale well with the high dimensionality without a vast decrease in performance, and it is also applicable to several classifiers due to being classifier independent. So

it is useful in testing the effect of feature selection on the efficiency of more than one classifier (Saeys, Inza, & Larrañaga, 2007).

Chi-Square

Chi-square measures the dependence between a feature and a class (Parlar et al., 2018). The Chi-square statistics formula is related to information theoretic feature selection functions which try to capture the intuition that the best terms t_k for the class c_i are the ones distributed most differently in the sets of positive and negative examples of class c_i. A higher score of Chi-square implies that the related class is more dependent on the given feature (Adel et al., 2014). A feature with a low score is less informative and should be removed (Parlar et al., 2018). Terms or words will be selected as a feature if their Chi-Square value is higher (Adel et al., 2014). The Chi-Square value will be calculated as follows (A. Sharma & Dey, 2012):

Chi-Square $(t_k , c_i) = (\mathrm{N(AD - CB)})^2 / ((\mathrm{A + C})(\mathrm{B + D})(\mathrm{A + B})(\mathrm{C + D}))$…….4.2

Where,

N = The total number of training sentence,

A = The number of sentences that contain the term t in class c_i.

B = The number of sentences that contain the term t in other classes.

C = The number of sentences in class c_i that do not contain the term t.

D = The number of sentences that do not contain the term t in other classes.

Simplified Chi-Square

Simplified Chi-Square also called Galavotti-Sebastiani-Simi (GSS) Coefficient is the simplified version of Chi-Square in which The P and N factor and the denominator have completely removed (Kandarp, 2009). The denominators have also removed; because the denominator gives high correlation coefficient score to rare words and rare categories (Kandarp, 2009), therefore, the score for rare or low-frequency terms words is not reliable (Adel et al., 2014):

$\frac{(AD - CB)}{N^2}$……4.3

Where,

N = The total number of training sentence,

A = The number of sentences that contain the term t in class c_i.

B = The number of sentences that contains the term t in other classes.

C = The number of sentences in class c_i that do not contain the term t.

D = The number of sentences that do not contain the term t in other classes.

A. Mutual Information Gain

Mutual information (MI) of two random variables is a measure of the mutual dependence between the two variables. Due to its computational efficiency and simple interpretation, information gain is one of the most popular feature selection methods (Tang, Alelyani, & Liu, n.d.). A weakness of MI is that the score is strongly influenced by the marginal probabilities of terms (Matsumoto, Sproat, Wong, & Zhang, 2006), as can be seen in this equivalent form (Bramer, 2009):

$$I(t, c) = \log(P_r(t|c)) - \log \Pr(t) \quad \ldots\ldots 4.4$$

In another term, for terms with equal conditional probability $P_r(t|c)$, rare terms will have a higher score than common terms. The scores therefore are not comparable across terms of widely differing frequency. Since MI is one of the popular features selection methods the researcher is interested in evaluating its performance against other feature selection methods.

B. Combination of feature selection methods

When two or more feature selection methods combined, there is a chance to select better features, since by combining we may compensate for the shortcoming of individual feature selection methods. This is done by adding the weighted score of the top N selected feature, from different feature selection methods, and the weight can be calculated by identifying the rank of features in individual feature selection method and gives some weight based on their rank.

2.6 Approaches of Sentiment Classification

Whenever we want to classify sentiments or opinion into specific groups like positive or negative, we can perform this task by one of the three classification methods namely lexical

approaches, machine learning approaches and hybrid approaches (D'Andrea et al., 2015). These three approaches will be discussed as follows:

2.6.1 Lexical classification approach

The lexicon-based approach involves calculating orientation for a document from the semantic orientation of words or phrases in the document (Turney, 2001). According to (Palanisamy, Yadav, & Elchuri, 2013), the lexicon-based approach is based on the assumption that the contextual sentiment orientation is the sum of the sentiment orientation of each word or phrase. In other word, in lexicon based sentiment analysis, an attempt made to predict the sentiment of the sentence or document based on the overall sentiment of opinion words in the sentence or document.

Even though, sentiment words are very important in sentiment analysis using them alone is not enough due to the following issues (Liu, 2012):

1. Sentiment word in one domain may have opposite orientations on others. For example, suck usually indicates negative sentiment, e.g., this book sucks, but it can also imply positive sentiment, e.g., this vacuum cleaner really sucks.
2. A sentence containing sentiment words may not express any sentiment. e.g., Can you tell me where can I find a good meal? The above sentence contains the word good but does not express any sentiment.
3. The difficulty of sarcasm sentences.
4. Sentences without sentiment words can have sentiment. For example, the mobile phone I bought last night has a battery that needs charging every 5 minutes. In the example above, there is no sentiment word, but the sentence expresses sentiment.

Lexicon-based approaches have the advantage that labeled data and the procedure of learning is not required.

However, there are a number of drawbacks to lexicon-based classification (Jacob, 2017). First, while intuitively reasonable, lexicon-based classification lacks theoretical justification: it is not clear what conditions are necessary for it to work. Second, the lexicons may be incomplete, even for designers with strong substantive intuitions. Third, sentiment lexicons tend to assign a fixed

sentiment orientation and score to words, irrespective of how these words are used in a text, but some words may be more strongly predictive than others. Fourth, lexicon based classification ignores multi-word phenomena, such as negation (e.g., not so good) and discourse (e.g., the movie would be watchable if it had better acting). Supervised classification systems, which are trained on labeled examples, tend to outperform lexicon-based classifiers, even without accounting for multi-word phenomena (Liu, 2015; Pang & Lee, 2008). Fifth, Lexical don't need labeled data but is hard to create a unique lexical-based dictionary to be used for different contexts. For example, slang used in Social Networks is rarely supported in lexical methods (Xia, Jiliang, Huiji, & Huan, 2013).

Lexicon construction approaches

There are three approaches to construct a sentiment lexicon: manual construction, corpus-based approach and dictionary-based approach (Liu, 2012).

Manual approach

The manual construction the sentiment lexicons are constructed by human labor and, it is a difficult and time-consuming task. Given the time we have, it is difficult to prepare a huge corpus and use corpus-based approach and, also the dictionary based approach has a major disadvantage which is the inability to find opinion words with domain and context specific orientations. Therefore, in our research work, we use a manual approach to collect lexicons of seed words and, then, expand the number of lexica using a dictionary by search synonyms of the seed words.

Dictionary-based approach

In dictionary-based approach, the idea is to first collect a small set of opinion words manually with known orientations, and then the algorithm grows this set by searching in the WordNet dictionary for their synonyms and antonyms (Rajput & Solanki, 2016). In general, these methods assume that positive adjectives appear more frequently near a positive seed word and negative adjectives appear more frequently near a negative seed word (Liu, 2012). The dictionary-based approach has a major disadvantage which is the inability to find opinion words with domain and context specific orientations (Rajput & Solanki, 2016).

Corpus-based approach

The Corpus-based approach helps to solve the problem of finding opinion words with context-specific orientations. Its methods depend on syntactic patterns or patterns that occur together along with a seed list of opinion words to find other opinion words in a large corpus (Medhat, Hassan, & Korashy, 2014; Rajput & Solanki, 2016). Using the corpus-based approach alone is not as effective as the dictionary-based approach because it is hard to prepare a huge corpus to cover all English words (Liu, 2012; Rajput & Solanki, 2016).

2.6.2 Machine learning approaches

Machine Learning approach is a field of artificial intelligence that trains the model from the existing data in order to forecast future behaviors, outcomes, and trends with the new test data (Narayan, Roy, & Dash, 2016). The main advantage of machine learning approaches is the ability to adapt and create trained models for specific purposes and contexts (D'Andrea et al., 2015). This approach generally achieves higher accuracy than that of the unsupervised approach for sentiment analysis; however, it requires building a gigantic corpus (dataset) and labeling it manually by human experts. The process of manual annotation can be very difficult even for native speakers due to sarcasm and cultural references. It can also be expensive and time-consuming. Moreover, the model built may be a domain-biased. That is, it could give low accuracy when is applied to such a different domain (Read & Carroll, 2009).

The machine learning methods are applicable to sentiment analysis ordinarily belongs to supervised learning in trendy and textual classification strategies in particular (Singh & Agrawal, 2017).

Naive Bayes (NB)

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features. In assuming independence, the presence of a feature has no impact on the probability of another feature also being a member of the document vector (Smith, 2015). Despite its simplicity and the fact that its conditional independence assumption clearly does not hold in real-world situations, Naive Bayes-based text categorization still tends to perform surprisingly well (Pang et al., 2002). We get the model form by using Bayes Rule:

$$P(label|sentence) = p(sentence|label)p(label)/p(sentence) \ldots\ldots (2.1)$$

Each sentence is represented as a feature vector and by conditional independence assumption between features, we get (Abreham, 2014):

$$P(label|sentence) = \frac{\prod_i^F p(f_i|label)p(label)}{p(sentence)} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.2)$$

There are several variations in Naïve Bayes classifier (Gupte, Joshi, Gadgul, & Kadam, 2014):

- **Multinomial Naïve Bayes** – this is used when Multiple Occurrences of Word Matter a lot in Text Classification problems. Such an example is when we try topic classification.
- **Binarized Multinomial Naïve Bayes** – this is used when frequencies of the words don't pay a key role in our classification. Such an example is Sentiment analysis where it doesn't matter how many times someone enters the word 'bad' or 'good' but rather only the fact that he does.
- **Bernoulli Naïve Bayes** - this is used when in our problem the absence of a particular word matters, For example, Bernoulli is commonly used in Spam or Adult Content Detection with very good results.

Due to the intuitive motivation and speed of classification (Lewis, 1998), the Naıve Bayes (NB) classification model is one of the more frequently used models in the sentiment classification literature. When training, the NB classifier does not over-fit the training data, meaning a reliable classification model should be generated given a suitable input (Andrew & Michael, 2001).

Support vector machines (SVMs)

SVMs are a set of new supervised learning methods used for binary classification, regression and outlier's detection (Amarappa & Sathyanarayana, 2012). The basic concept of SVM is that it is looking for the Optimal Separating Hyper-plane between the two classes by maximizing the margin between the classes' closest points (see Figure 2.1) (Saud, 2015). The points are located on the boundaries are called support vectors.

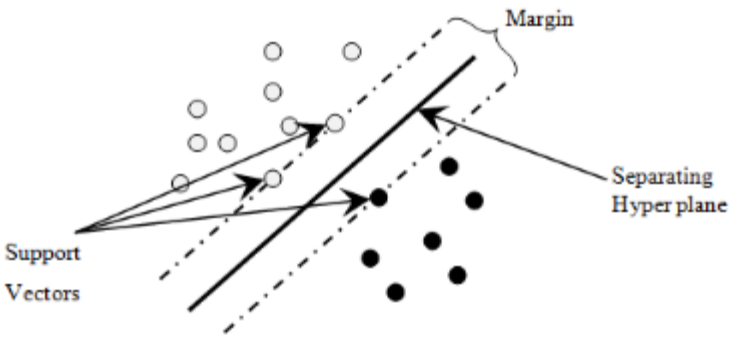


Fig. 2.1 support vector machine

When given a set of learning data, in which each example is marked to show which class this example belongs to, then SVM learning algorithm builds a model that assigns the new example to one of these classes. In general, the SVM model represents the training examples as points in spaces that are mapped such that the training examples belonging to different classes are separated by a gap as wide as possible. When a new example is given then based on which side of the gap it falls in, the SVM predicts the class to which this example belongs to (ShehlaKulsum & Vaidya, 2017).

The first advantage of SVM is effective in high dimensional spaces. It is effective in cases where a number of dimensions are greater than the number of samples. The second advantage is that it uses a subset of training points in the decision function (called support vectors), so it is memory efficient. The third advantage is that Different Kernel function can be specified for the decision function i.e. SVM is versatile (Amarappa & Sathyanarayana, 2012).

The disadvantage is that, if the number of features is much greater than the number of samples, the method is likely to give poor performances (Amarappa & Sathyanarayana, 2012).

Maximum Entropy (MaxEnt**)**

It is a probabilistic classifier which has a place with the class of exponential models (Joshi, Prajapati, Shaikh, & Vala, 2017). The maximum entropy principle is based on selecting the most uniform distribution which is to be known by the one having maximum entropy (Patel, Saxena, Verma, & Student, 2007). The model makes no assumptions about the independence of words. This means we can add features like bigrams and phrases to MaxEnt without worrying about feature overlapping (Gupte et al., 2014). Due to the minimum assumptions that the Maximum Entropy classifier makes, we regularly use it when we don't know anything about the prior distributions and when it is unsafe to make any such assumptions (Raghuwanshi & Pawar, 2017). However, it is computationally more expensive (Mehra, Khandelwal, & Patel, 2002). The Max Entropy requires more time to train compare to Naive Bayes, primarily due to the optimization problem that needs to be solved in order to estimate the parameters of the model. Nevertheless, after computing these parameters, the method provides robust results and it is competitive in terms of CPU and memory consumption (Raghuwanshi & Pawar, 2017).

This classifier always tries to maximize the entropy of the system by estimating the conditional distribution of the class label. The conditional distribution is defined as (Raghuwanshi & Pawar, 2017):

$$P_\lambda(y|X) = \frac{1}{Z(X)} \exp\left(\sum \lambda_i * f_i(X, y)\right) \quad \text{.....2.3}$$

'X' is the feature vector and 'y' is the class label. Z(X) is the normalization factor and λ_i is the weight coefficient. fi(X, y) is the feature function which is defined as(Raghuwanshi & Pawar, 2017) :

$$f_i(X, y) = \begin{cases} 1, & X = x_i \text{ and } y = y_i \\ 0, & \text{otherwise} \end{cases} \quad \text{.....2.4}$$

Logistic regression (LR)

Logistic regression sometimes called the logistic model or logit model analyzes the relationship between multiple independent variables and a categorical dependent variable and estimates the probability of occurrence of an event by fitting data to a logistic curve. There are two models of logistic regression, binary logistic regression, and multinomial logistic regression. Binary logistic regression is typically used when the dependent variable is dichotomous and the independent variables are either continuous or categorical. When the dependent variable is not dichotomous and is comprised of more than two categories, a multinomial logistic regression can be employed (Park, 2013).

The logistic model is popular because the logistic function, on which the logistic regression model is based, provides estimates in the range 0 to 1 and appealing S-shaped description of the combined effect of several risk factors on the risk for an event (Kleinbaum & Klein, 2002).

Using logistic regression for multiple predictors (numerical and categorical), the probability of the occurrence of the interested outcome can be calculated as follows (Park, 2013):

p=P(Y=interested outcome/X_1=x_1,… $X_k = x_k$)

$$p=\frac{e^{a+\beta_1 x_1+\cdots..+\beta_k x_k}}{1+e^{a+\beta_1 x_1+\cdots..+\beta_k x_k}} \quad \text{.....2.5}$$

Where p is the probability of interested outcome and x_1,.., x_k are the explanatory variable. The parameters of the logistic regression are α and β.

Advantages of Logistic Regression (Teja, Sai, Kumar, & Manikandan, 2018):

- It is much more robust to correlated features.
- If two features f1 and f2 are perfectly correlated, regression will simply assign half the weight to w1 and a half to w2.
- It is discriminative
- It works well on large datasets when compared with Naïve Bayes.

K-Nearest Neighbor Classifier (K-NN)

Nearest neighbor classifiers are based on learning by analogy. The training samples are described by n-dimensional numeric attributes. Each sample represents a point in an n-dimensional space. In this way, all of the training samples are stored in the n-dimensional pattern space. When given an unknown sample, a k-nearest neighbor classifier searches the pattern space for the k training samples that are closest to the unknown sample. The unknown sample is assigned the most common class among its k nearest neighbors (Phyu, 2009).

K-NN is a nonparametric method used for classification or regression. K-NN is powerful because it does not assume anything about the data, other than a distance measure can be calculated consistently between two instances. As such, it is called non-parametric as it does not assume a functional form (Dey, Chakraborty, Bose, & Tiwari, 2016).

But, nearest neighbor classifiers are instance-based or lazy learners in that they store all of the training samples and do not build a classifier until a new (unlabeled) sample needs to be classified. Therefore, it is slower at classification since all computation is delayed to that time. Lazy learners can incur expensive computational costs when the number of potential neighbors (i.e., stored training samples) with which to compare a given unlabeled sample is great. Nearest neighbor classifiers assign equal weight to each attribute. This may cause confusion when there are many irrelevant attributes in the data (Phyu, 2009).

Decision trees

Decision trees are trees that classify instances by sorting them based on feature values (Gullapelly & Shanmukhi, 2017).In this tree, the internal node represents a test on the attribute, each branch of the tree represents the outcome of the test and the leaf node represents a particular class label means the last decision after all computations (Kaur & Jindal, 2016). It categorizes a document by starting at the tree root and moving successfully downward via the branches (whose conditions are satisfied by the document) until a leaf node is reached. The document is then classified in the category that labels the leaf node (Chauhan, 2015).

Decision trees are the most widely used classifier (Manne, 2011) and consist of a set of rules which are applied in a sequential way and finally yield a decision (Manne, 2011). Their

robustness to noisy data and capability to learn disjunctive expressions seem suitable for document classification(Manne, 2011).

One of the most useful characteristics of decision trees is their comprehensibility. People can easily understand why a decision tree classifies an instance as belonging to a specific class. Since a decision tree constitutes a hierarchy of tests, an unknown feature value during classification is usually dealt with by passing the example down all branches of the node where the unknown feature value was detected, and each branch outputs a class distribution (Kotsiantis, 2007).

2.6.3 Hybrid approaches

Given the advantages and the disadvantages of both machine learning and lexicon based approaches, different researchers tried to combine them together so that they can take advantage from the benefits of each approach (Amira, 2013). The hybrid approach, the combination of both the machine learning and the lexicon based approaches has the potential to improve the sentiment classification performance (D'Andrea et al., 2015).

The main advantages of hybrid approaches are the lexicon/learning symbiosis, the detection and measurement of sentiment at the concept level and the lesser sensitivity to changes in the topic domain. While the main limitation is that reviews are with a lot of noise (irrelevant words for the subject of the review) are often assigned a neutral score because the method fails to detect any sentiment (D'Andrea et al., 2015).

2.7 Related Works

In this section, sentiment mining related researches done for different language opinionated documents such as English (Appel, Chiclana, Carter, & Fujita, 2016; Pedro, Balage, & Thiago, 2013), Arabic (Amira, 2013), French(Hamdan, Bellot, & Bechet, 2016), (Hussaini, Padmaja, & Sameen, 2018) and Amharic (Abreham, 2014; Gebremeskel, 2010; Tulu, 2013) using different techniques and approaches are reviewed.

2.7.1 Sentiment Mining from Opinionated English Texts

Pedro, Balage, and Thiago (Pedro et al., 2013) develop a system that adopts a hybrid classification process that uses three classification approaches rule-based, lexicon-based, and machine learning approaches. The purpose is to better understand the use of a hybrid system in

Twitter text and to verify the performance of this approach in an open evaluation contest. The researcher in this work suggests a pipeline architecture that extracts the best characteristics of each classifier. In this pipeline architecture, each classifier may assign a sentiment class, if it achieves a particular confidence threshold, otherwise, it will pass to the next classifier.

A training dataset, with 6,686 messages, a development dataset, with 1,654 messages, and two test datasets, with 3,813 Twitter Test test data and 2,094 SMS based test data messages are used. With this data different experiment has been done and the hybrid system achieved an F-score of 56.31% in the Twitter message-level subtask, which is better than rule-based, lexicon based or machine learning alone. Sentiment Analysis in Twitter compares the systems by the average F-score for positive and negative classes. As shown in the table below

Table 2.1: Average F-score

classifier	Twitter TestSet	SMS TestSet
Rule-based	0.1437	0.0665
Lexicon-Based	0.4487	0.4282
Machine learning	0.4999	0.4029
Hybrid Approach	**0.5631**	**0.5012**

The reason why their system improves the classification process because it takes advantage of the multiple approaches. For example, the rule-based classifier is the most reliable classifier. It achieves good results when the text is matched by a high-confidence rule. However, due to the freedom of language, rules may not match 100% of the unseen examples; consequently, it has a low recall rate. Lexicon-based classifiers, for example, are very confident in the process to determine if a text is polar or neutral. Finally, machine learning is known to be highly domain adaptive and to be able to find deep correlations (Taboada, Brooke, Tofiloski, Voll, & Stede, 2011).

Appel et al. (2016), proposed an approach that uses a hybrid approach to Sentiment Analysis encompassing the use of Semantic Rules, Fuzzy Sets, and an enriched Sentiment Lexicon, improved with the support of SentiWordNet is described. The proposed hybrid method is applied to three different data-sets and the results achieved are compared to those obtained using Naïve Bayes and Maximum Entropy techniques. It is demonstrated that the presented hybrid approach

is more accurate and precise than both Naïve Bayes and Maximum Entropy techniques when the later are utilized in isolation. The result obtained has shown that their hybrid method performs well than the other approaches.

Table 2.2 Naïve Bayes Table

Accuracy	0.67
Precision	0.63
Recall	0.85
F1-score	0.72

2.3 Maximum Entropy

Accuracy	0.68
Precision	0.63
Recall	0.86
F1-score	0.73

Table 2.4 Hybrid

Accuracy	0.76
Precision	0.73
Recall	0.83
F1-score	0.77

The creation of an improved Sentiment Lexicon was decisive in obtaining good experimental results and SentiWordNet became an important component of their proposed solution and certainly enriched dramatically the quality of their Lexicon.

Sentiment mining from opinionated Arabic language Amira (Amira, 2013) combined semantic orientation and machine learning. In this research work, her aim was to improve the performance measures of *Egyptian dialect sentence-level sentiment analysis* by proposing a hybrid approach which combines both the machine learning approach using support vector machines and the semantic orientation approach. Two methodologies were proposed, one for each approach, which was then joined, creating the hybrid proposed approach. The corpus used contains more than 20,000 Egyptian dialect tweets collected from Twitter, from which 4800 manually annotated tweets are used (1600 positive tweets, 1600 negative tweets, and 1600 neutral tweets). Several experiments have been performed to; 1) compare the results of each approach individually with regards to her case which is dealing with the Egyptian dialect before and after preprocessing; 2) compare the performance of merging both approaches together generating the hybrid approach against the performance of each approach separately; and 3) evaluate the effectiveness of considering negation on the performance of the hybrid approach. The results obtained show significant improvements in terms of accuracy, precision, recall, and F-measure, indicating that the proposed hybrid approach is effective in sentence-level sentiment classification. Also, the results are very promising which encourages continuing in this line of research.

2.7.2 Sentiment Mining from Opinionated French Language

Hamdan et al. (2016) work on Sentiment Analysis in Scholarly *Book Reviews in the French language*. Their objective is to extract the opinion expressed towards a book in all its reviews. They chose aspect level of sentiment analysis. Therefore, given a collection of book reviews, they aim at finding out the aspects of the book and the sentiment expressed towards each aspect. The aspects are determined by asking domain experts to extract the aspects of books found in book reviews of social and human sciences. They have listed aspects like Book presentation, Problematic, Scientific context, Scientific method, Author's arguments, Book organization, and Judgment about the book. Different features are combined in order to be presented to supervise classifiers which extract the opinion target expressions and detect their polarities in scholarly book reviews. For creating an annotated corpus of French book reviews, 200 book reviews in the French language have been selected with the help of domain experts.

Different experiments have been done for opinion target extraction and determine whether the polarity of each opinion target is positive, negative or neutral. In the experiments for opinion target extraction, Conditional Random Fields (CRF) suite tool is used for target extraction with the L-BFGS algorithm. Part of speech tag to each term (Pos), the type of the word (uppercase, digit, symbol, and combination), the shape of each character in the word (capital letter, small letter, digit, punctuation, and other symbol), prefixes and suffixes, are used for target extraction. They found 61.53% when using all the above features together. From the experiment, we understand that the word and POS features seem to be enough to produce a good result, 61.02%.

In the experiments for Sentiment Polarity determination, they train Logistic regression classifier on the training dataset using N-grams and Z score as a feature with the three polarities (positive, negative, and neutral) as labels. In these experiments in addition to books review, restaurant and laptop reviews also considered. The best result is given when using terms and Z score (or standard score) features with Z threshold of -0.5. The accuracy is 79% which seems fair enough when comparing with the results produced in restaurant reviews (about 75.5%). Where Z-score (or standard score) represents how many standard deviations a given measurement deviates from the mean.

2.7.3 Sentiment Mining from Opinionated Hindi Language

Hussaini et al (Hussaini et al., 2018), apply a score-based approach for sentiment classification of *book reviews in the Hindi language*. Opinion words were extracted from individual sentences using parts tagger, incorporated within the Hindi shallow parser. They consider adjectives, adverbs, nouns and verbs for extraction. Their approach uses subjectivity lexicons for retrieving polarity scores of the extracted words. The overall positive and negative scores were calculated for each sentence, the higher value between the two determining the polarity of the sentence.

A dataset of 700 sentences pertaining to book reviews was considered for this work. These sentences were first annotated by three Hindi-speaking annotators. The mutual agreement between them was calculated and the kappa value was found to be 79.4%. The results obtained from the system were tested against these human annotations. An accuracy of 86.3% was achieved working with H-SWN, after applying word-sense disambiguation (WSD) and handling morphological variations. An accuracy of 87.4% was achieved working with HSL.

2.7.4 Sentiment Analysis for Amharic Language

Selama Gebremeskel (Gebremeskel, 2010), proposed a sentiment mining model for determining the sentiments expressed in opinionated Amharic texts or reviews. They Used Lexicon based approach and the proposed model has the following components: preprocessing, sentiment words detection, weight assignment and propagation, polarity classification, polarity strength representation, and sentiment lexica. The system designed based on the proposed model detects positive and negative sentiment terms including contextual valence shifters such as negations and the lexica of Amharic sentiment terms are used to identify and assign initial polarity value to the sentiment terms detected in order to determine the polarity classification of the opinionated text. Based on the weights of the sentiment values, the reviews are classified into predefined categories: positive, negative or neutral. Finally, the polarity strength of the reviews is rated. A prototype system is developed to validate the proposed model and the algorithms designed. Tests on the prototype are done using movie and newspaper reviews where the result obtained with these test data is very much encouraging. Experiment, using contextual valence shifter, terms such as negations give better results.

The sentiment lexica are built manually from different sources and after having lexicons, the review document is preprocessed and relevant term in the review is checked whether it is a sentiment word or not, by scanning the whole lexicon for every term. If the term exists in the dictionary, then the term is a polarity word (positive or negative). The total polarity weight of a review is calculated by adding the polarity weight of the individual sentiment terms in the review if the summation of polarity is greater than zero, then the review is categorized into predefined category positive. Similarly, if the summation of polarity is less than zero then the review is categorized into a predefined category negative. Otherwise, if the weight of all the individual terms is equal to zero, the review is categorized into the neutral category.

As we can show in the table below, the researcher gets a better result using both general lexica, specific lexicons and considering contextual valance shifter terms. Summary of the experimental result of Gebremeskel (Gebremeskel, 2010) is as shown in the table below:

Table 2.5: Summary of experimental result

system	Class	Precision	Recall	F-measure
General purpose Amharic sentiment terms(Basic system)	Positive	0.929	0.823	0.867
	Negative	0.6	0.573	0.589
Basic + Domain lexicon	Positive	0.937	0.943	0.939
	Negative	0.62	0.78	0.69
Both lexica + contextual valance shifter terms	Positive	0.943	0.949	0.945
	Negative	0.666	0.842	0.743

Selama Gebremeskel (2010) uses lexicon approach alone. Therefore, sentiment lexicons assign a fixed sentiment orientation and score to words, irrespective of how these words are used in a text, but some words may be more strongly predictive than others. These can affect the performance, and if we combine machine learning with lexicon based classifier, there is a possibility of improving the performance. Machine learning can give us the ability to learn from training data, how much a word lean towards positive or negative, instead of assigning fixed orientation. In our research work, we combine machine learning and lexicon classifiers and create a hybrid approach. Therefore, first, a review classified as positive or negative based on

fixed sentiment orientation and a score of words (lexical classifier). If it can not be classified, the review passed to machine learning, which assigns sentiment orientation and a score of words after learning through training, instead of just use fixed sentiment orientation like a lexical classifier.

Tulu (Tulu, 2013) work on feature-level opinion mining model for the Amharic language by employing manually crafted rules and lexicon. The proposed model consists of five major components that can extract features, determine opinion words regarding identified features with their semantic orientation, aggregate multiple opinions and generate a structured summary.

Two experiments have been conducted for features extraction and opinion words determination by using 484 reviews from three different domains. The first experiment indicated that an average precision of 95.2% and recall of 26.1% were achieved in the features extraction and an average precision of 78.1% and recall of 66.8% were achieved in the determination of opinion words. The precision of the second experiment in features extraction gets lower by 15.4% whereas the precision of opinion words determination gets higher by 1.9% and the recall of both features extraction and opinion words determination gets higher by 7.8% and 25.9% respectively when compared to the first experiment. Thus, their experimental results demonstrate the effectiveness of the techniques they have applied.

Abreham (Abreham, 2014), develop an Opinion Mining model for classifying the Amharic opinionated text into positive and negative. Two simple feature sets are employed (all unigram and the most informative bag-of-words of the review) and Information Gain feature selection method used to calculate most informative words from the document. For classification, three supervised classifiers implemented from the Natural Language Toolkit (the Naïve Bayes, Decision Tree, and Maximum Entropy classifiers).

The datasets for conducting the experiment are manually collected from Ethiopia Broadcasting Corporation in Addis Ababa. The rest of the dataset is collected from diretube.com and habesha.com sites. A total amount of 616 reviews were collected for experiments.

Different experiments have been done and the Experiment indicates that Information Gain feature selection methods perform the best through all algorithms (Naïve Bayes, Decision Tree,

and Maximum Entropy). Based on their relative performance of classification, NB with 90.9% accuracy outperforms Decision Tree with 83.1% and Maximum Entropy with 89.6%. The result obtained is encouraging.

2.7.5 Summary

But both machines learning and lexical have their own advantage and disadvantage. Machine learning has the ability to adapt and create a trained model for specific purpose and context but, disadvantages of low applicability to new data because it is necessary availability of labeled data that could be costly. Lexicon based advantages wider term coverage but, their disadvantages are, Finite number of words in the lexicons and the assignation of a fixed sentiment orientation and score to words.

Table 2.6: Overview of some previous work

Author	Language	Domain	Labeling	Approach	accuracy
(Filho & Pardo, 2013)	English	Twitter	manual	rule-based, lexicon-based and machine learning (svm)	61.85%
(Amira, 2013)	Arabic	Twitter	manual	(SVM,NB, and ME), and semantic orientation	75.4%
(Appel et al., 2016)	English	movie	manual	NLP-techniques, semantic rules and fuzzy sets	76%

To gain the advantages and avoid disadvantages of machine learning many researchers in English, Arabic, and other languages, have done research by combining lexicon based and machine learning approaches. This kind of combination (hybrid approaches) become the focus of researchers.

But sentiment analysis is language dependent, and the work done by one language is not directly applicable to another language (Bal et al., 2011). This is due to differences in grammar and other behavior of the language. For example, word order in Amharic is generally subject-object-verb (SOV), with subordinate clauses preceding the main clause. Noun phrases are also generally headed final with modifiers, including relative clauses, preceding the noun. In addition to that, Amharic language has a complex inflectional morphology, particularly in the verbal system, employing not only prefixes and suffixes but also internal modification of the typical Semitic consonantal root-and-pattern type (Keith & Sara, 2010).

But, as far as my knowledge is concerned, sentiment classification researches on Amharic language focus only on lexical, machine learning like or feature based sentiment analysis. Therefore, the researcher is interested in exploring the performance of Hybrid classification approach in this research work.

Even in machine learning classification, almost all researches in Amharic language try to explore only the performance of sentiment classification algorithms, but not on between feature selection methods. But machine learning algorithms performance in addition to the quality of data used; partially depend on the kind of feature selection method used. Therefore, in my opinion, there is a need to compare the performance of different feature selection methods like **Chi-square**, **Mutual Information gain**, **simplified Chi-Square** and other popular feature selection method on Amharic language.

In addition to the above two tasks (exploring the performance of Hybrid Approach and popular feature selection methods), the performance of some machine learning algorithms like Logistic Regression, SVM, and Naïve Bayes has been explored.

Challenges of sentiment analysis for Amharic language

There are different factors that make sentiment analysis in Amharic. The first factor is that, for Amharic language, there is no standardized corpus (both for review data and lexicons) for opinion mining. Secondly, People usually use positive words in negative reviews, but the word is followed by valance shifters (negation) words like aydel_em or "አይደለም", in this research work we attempt to handle negation using rules developed for a lexical component of the hybrid

classifier. Also, the use of Amharic slang words makes sentiment classification challenging. To reduce this effect, we incorporated slang words that have sentiment. The other factor is that sometimes people use objective text to express their opinion but the classifier did not identify those facts from opinions.

CHAPTER THREE

METHODOLOGY

The following wide-spread methodologies have been employed to design and develop a hybrid sentiment classification for Amharic book review.

3.1 Data Collection Methodology

In this research work, we use **Qualitative Data Collection** Method in collecting Amharic book review data. Even though most of Amharic book review data have been collected manually from different online sources, one **Open-ended qualitative questionnaire** designed for the purpose of analysis, and classification. This simple questioner has been adapted from Gebremeskel (Gebremeskel, 2010) and slightly modified to fit with the domain we are working. The opinion of selected readers, who at least read one of the selected Amharic books, have been collected using this questionnaire and opinion holders fill their positive and negative opinion about selected books in a space provided.

These reviews are qualitative data, which means that these data are unstructured and usually textual. But these qualitative data are not in a form to be manipulated and analyzed by a computer. Therefore, *qualitative data transformed into quantitative data*, which involves turning the data from words into numbers (Bernard, 1996), thereby unleashing the full power of **quantitative analytics** on the qualitative data.

3.2 Data Sources

In this research work, two types of data are collected. These data consist of 1370 lexicons and 600 book review. From 1300 lexicon used in this research work, we collect 870 lexicon or opinion words by translating from English source (Liu, Hu, & Cheng, 2005) and the rest, 500, lexicons are adapted from Gebremeskel (Gebremeskel, 2010). The 600 reviews (dataset) for conducting the experiment are manually collected from different sources such as **Facebook, personal blogs, book review sites** and also, through a questionnaire distributed for randomly selected book readers. With the limited time we have, we can collect only 500 reviews manually. Therefore, it was necessary to collect additional reviews using **Open-ended qualitative questionnaire** distributed for book readers**.** The questionnaire distributed for 150 book readers,

but only 101 people respond positively, fill the questionnaire and returned. We use 100 of these reviews collected using questionnaire which is filled correctly. 500 reviews collected manually and 100 reviews collected using questionnaire. Make up total review used in the research work 600. These reviews are stored in a text file and, any time, if we want to incorporate new data, we simply append on a text file that stored these reviews.

The amount of reviews used in our research work is small. However, data extraction for Amharic was difficult due to lack of Amharic web content and it took a significant proportion of the time. In other research works like (Aggarwal & Gupta, 2017) in English, (Mittal, Agarwal, Chouhan, Bania, & Pareek, 2013) in Hindi, and (Abreham, 2014; Gebremeskel, 2010; Tulu, 2013) in Amharic language use a similar amount of dataset.

3.3 Tools

A number of tools have been used to design and develop a hybrid sentiment classification. These tools include NLTK, SERA, Hornmorpho, and python as a programming language:

Python is a simple yet powerful programming language with excellent functionality for processing linguistic data (Bird, Klein, & Loper, 2009). In this study, preprocessing activities like stop word removal, transliteration, and stemming are done using python programming, version 3.5. In addition to preprocessing activities, tools like NLTK and HORNMORPHO are implemented using python programming. The reason why we chose python is that; Python is a programming language with clear and readable syntax. The way Python's syntax is organized imposes some order to programmers as a result; experts and beginners can easily understand the code.

NLTK is the natural language toolkit, a comprehensive python library for natural language processing and text analytics. Although Python already has most of the functionality needed to perform simple NLP tasks, it's still not powerful enough for most standard NLP tasks (Madnani, 2007). This is where the Natural Language Toolkit (NLTK) comes. NLTK defines an infrastructure that can be used to build NLP programs in Python (Bird et al., 2009). Since NLTK provides basic standard modules for performing classification tasks and have extensive

documentation for reference at (Ojokoh & Kayode, 2012), we chose to work with NLTK. All classification tasks have been done using NLTK in python programming.

In this research work, we use NLTK, but NLTK does not support Ge'ez characters. To represent Ge'ez character using its equivalent English characters, we use a convention called SERA. SERA is a convention for the transcription of Ethiopic script into the seven bit American Standard for computer information interchange (ASCII) (Firdyiwek & Yaqob, 1997).

HORNMORPHO is a system for morphological processing of Amharic, Oromo, and Tigrigna. According to (Gasser, 2012**),** HORNMORPHO is a Python program that analyzes Amharic, Oromo, and Tigrinya words into their constituent morphemes (meaningful parts) and generates words, given root or stem and a representation of the word's grammatical structure.

In this research work, the researcher uses HORNMORPHO for two purposes. The first purpose we use HORNMORPHO is to convert every word to their base forms to avoid data sparseness. The second purpose we use HORNMORPHO is to convert an input word in Ge'ez characters to a phonetic representation of the word. The phonetic representations conform to the Romanization conventions of the SERA system.

3.4 Feature Selection Methods

Feature selection methods reduce the original feature set by removing irrelevant features for text sentiment classification to improve classification accuracy and decrease the running time of learning algorithms (A. Sharma & Dey, 2012). Before using machine learning methods in text categorization, it is essential to choose which features are the most suited for this task. Feature selection can be done using different feature selection methods. We select Chi-Square, Mutual Information, and GSS coefficient, as feature selection methods.

The reason we select to work with these three feature selection methods is that first because they have a reputation to have good performance in English language and we are interested to know the performance of these algorithms in Amharic language. According to Oystein (Oystein, 2009) among feature selection methods, Chi-Square and Mutual Information perform best also, the Chi-Square variant GSS coefficient was also among the top performers. The second reason is that, in addition to performing well, they have contrasting and different behavior. Chi-Square and

Mutual Information have a tendency to gave a high score for common features that have a high frequency of occurrence in the corpus. In another hand, the GSS coefficient gives a high score for rare words. Therefore by considering these feature selection methods, we can easily see the importance of common words and rare words on the performance of machine learning. The performance of these three feature selection methods has been compared against each other. For comparing these three algorithms we use three machine learning algorithms namely NAÏVE BAYES, SVM, and LOGISTIC REGRESSION.

3.5 Algorithms

Three machine learning algorithms were employed for classifying reviews as positive and negative such as Naïve Bayes, SVM, and Logistic Regression. Testing with more than one classification algorithms provides comparison clues for determining algorithm with the best performance for Amharic opinionated text in the domain and many research works in opinion mining achieved high performance using them. For example, in research works like (Ashari, Paryudi, & Min, 2013; Pang et al., 2002) SVM and Naïve Bayes outperform other algorithms like MaximumEntropy and *K*-Nearest Neighbors. In another research (Hao & Priestley, 2016) Logistic Regression outperforms *K*-Nearest Neighbors and Random Forest classifier algorithm. Before deciding to use these three algorithms, we checked their performance against the other two popular algorithms *K*-Nearest Neighbors and Random Forest classifier algorithm. *K*-Nearest Neighbors and Random Forest classifier algorithm performs with an accuracy of 59.4% and 60.1% respectively, which is very low compared to the performance achieved by Naïve Bayes (93.3%), SVM (88%) and Logistic Regression (86%).

Naive Bayes

Naive Bayes is a supervised classification method developed using Bayes' Theorem of conditional probability with a Naive assumption that every pair of feature is mutually independent (Das, Behera, & Tech, 2017). Bayesian classifiers are often used for classification because they require far less computing power than other methods (Abreham, 2014). Also, Naive Bayes need only a small amount of training data. Beside, NB classifiers have considerably outperformed even highly advanced classification techniques (Das et al., 2017). Therefore, we choose Naïve Bayes to be one of the algorithms for our experiments.

SVM

SVM is a supervised classification algorithm, proposed by Vapnik in the 1960s have recently attracted major attention of researchers (Das et al., 2017). The main idea of SVM is to construct the hyperplane in a high dimensional space which can be used for classification (Kaur & Jindal, 2016). One remarkable property of SVMs is that their ability to learn can be independent of the dimensionality of the feature space (Joachims, 1998). That means SVM is effective in high dimensional spaces (Muhammad & Yan, 2015). Since SVM uses a subset of training points in the decision function (called support vectors), so it is also memory efficient (Muhammad & Yan, 2015). SVM classifier method is outstanding from other with its effectiveness (Yaming & Xin, 1999) and according to Vohra and Teraiya (Vohra & Teraiya, 2013), most of the researchers reported that Support Vector Machines (SVM) has high accuracy than other algorithms. Because of its effectiveness in high dimensional spaces and performance, we select SVM to be one of the candidate algorithms in this research work.

Logistic Regression

Logistic regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome (Hao & Priestley, 2016). In addition to logistic function, since it is logistic regression is efficient to train, does not require too many computational resources, highly interpretable, and it doesn't require any tuning logistic regression is the most widely used model in biomedicine (Dreiseitl & Ohno-Machado, 2002). In the research (Hao & Priestley, 2016; Teja et al., 2018), Logistic Regression outperforms other algorithms like *K*-Nearest Neighbors, SVM, and Naïve Bayes. Despite its popularity and efficiency, as far as the researcher knowledge concern, the performance of Logistic Regression against the two baseline algorithms, Naïve Bayes and SVM is not known in Amharic language. Therefore, we are interested to explore the performance of these three algorithms.

3.6 Numbers of Features

Even though different researchers use different numbers of features, the range typically considered for text classification is between 500 to 3000 numbers of features (Parlar et al., 2018). In this research work, for each feature selection method, we tried four feature sizes at 750, 1000, 1250, and 1500. We chose to show results for 750 features because it is low enough so the

classification is well possible and for the numbers of features, less than 750 the accuracy is too small. We stop at 1500, because, for the number of features greater than 1500, the accuracy of classifiers starts to drop and the maximum accuracy we found is at this point.

CHAPTER FOUR

AMHARIC LANGUAGE

4.1 Overview

Amharic is the working language of the government of Ethiopia and some of the federal states (Martha, 2010), with 25,873,820 (Eberhard, David, Simons, & Fennig, 2019). It is also the native language of several million Ethiopian immigrants, especially in North American and Israel (Gelbukh, 2018).

Amharic has been the language of government and the ruling group in Ethiopia since the end of the thirteenth century. Despite its long history, Amharic really only became the written language of Ethiopia from the second half of the nineteenth century onwards, when Emperor Tewodros II actively encouraged its use in the government bureaucracy. Prior to that, though, there are some examples of writing Amharic going back some six hundred years. The language of literacy in Ethiopia was Ge'ez (Appleyard, 2015).

Amharic belongs to the Semitic language family (Leslau, 1995) and it is the second most spoken Semitic language after Arabic (Solomon & Menzel, 2007). The Amharic language is related to Hebrew, Arabic, and Syrian (Martha & Menzel, 2009), but unlike Arabic, Hebrew or Syrian, the language is written from left to right (Leslau, 1995). Amharic exhibits typical Semitic behavior, in particular, the pattern of inflectional and derivational morphology, along with some characteristics Ethiopian Semitic features, such as subject–object–verb **(**SOV) word order, which are generally thought to have resulted from long contact with Cushitic languages (Gelbukh, 2018).

Amharic is a syllabic language which uses a script which inherited from the Ge'ez alphabet(Isenberg, 1842; Leslau, 1995). Geez is an ancient South Semitic language which now serves only as liturgical language of the Ethiopian Orthodox Tewahedo Church (MARKOS, 2010). Amharic has five dialectical variations (Addis Ababa, Gojjam, Gonder, Wollo, and Menz) spoken in different regions of the country (Rosenhouse & Kowner, 2008; Solomon & Menzel, 2007). Of the five, the Addis Ababa dialect has emerged as the accepted standard since its

introduction as the medium of instruction and press during the reign of Emperor Menelik II (Goldenberg, 2013; Rosenhouse & Kowner, 2008).

4.2 The Amharic Characters (Fidel)

Amharic has 33 basic characters with each having 7 forms for each consonant-vowel combination (MARKOS, 2010). Out of 33 basic characters, 26 are from Ge'ez (Isenberg, 1842). These 26 characters are shown in figure 3.1 below (Scelta, 2001):

Table 4.1 Ge'ez Alphabet

ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ
መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ
ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ
ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ
ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ
ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ
ነ	ኑ	ኒ	ና	ኔ	ን	ኖ
አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ
ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ
ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ
ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ
ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ
የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ
ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ
ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ
ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ
ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ
ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ
ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ
ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ
ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ

But besides the 26 Ge'ez Characters, the Amharic language has 7 peculiar Orders of Letters, Which serve to express sounds not existing in former (Isenberg, 1842).These 7 Characters Which serve to express sounds not existing in Ge'ez are ሸ [xe] , ጨ [Ce], ቸ [ce], ጀ [je], ኘ [Ne], ኸ [Ke] and ዠ [Ze] .

4.3 Deficiencies of the Amharic Alphabet

The deficiencies of the Amharic alphabet are (Leslau, 1995):

1. Lack of a special symbol for germination. Thus a word such as አለ may be read either ale 'he said' or 'alle' 'there is'; ገና may be read gena 'still' or 'genna' Christmas'. In writing of western scholars, germination is marked by two dots placed above the letter.
2. The 6th order designates both a constant followed by the vowel e and a constant without a vowel. Unless the word or the principles underlying the syllabic structure are known, one does not know how to pronounce it properly. Thus a word such as ይንገር, whose actual pronunciation is 'yengar', maybe read mistakenly 'yenegar', or 'yenneggaar'.

4.4 Characteristics of the Amharic Language

4.4.1 Amharic Alphabet Orthography

On the whole, no real problems exist in Amharic orthography, as there is, more or less, a one-to-one correspondence between the sounds and the graphic symbols. Since, however, a few sounds are expressed by more than one symbol; some confusion occasionally arises in the spelling. Sounds that are represented by more than one symbol are: s (ሰ፡ ሠ), s (ጸ፡θ), h (ሀ፡ሐ፡ኀ፡ኸ), and the vowels carry (አ፡ዐ). Moreover, ሀ፡ሐ፡ኀ፡አ፡ዐ written in the 1st order are pronounced as ሃ፡ሓ፡ኃ፡ኣ፡ዓ, that is, with the vowel a. Thus, the word 'seyyum', proper name, maybe spelled ሥዩም or ስዩም (Leslau, 1995). Although these different 'fidels' give each word different meaning in Ge'ez, in Amharic language they have been used interchangeably (Mindaye, Redwan, & Atnafu, 2010).

Amharic orthography reflects the spoken phonetic features to a large extent. The rule generally followed is "if a word sounds right when reading aloud then it was rightly written". Amharic spelling has rules, even though, it is not strict. There are acceptable levels of precision, a phono-orthographical radius that renderings may fall within to be considered recognizable and acceptable. Some phonetically spelling variations are more acceptable than others. Wider degree of spelling variance considered acceptable at the basic level (a working medium for all informal exchanges) and At the Advance level of writing, the canonical forms of words must be used and words can no longer be written merely as they would be spoken (Yacob, 2004).

4.4.2 Amharic Compound Word

The Amharic writing system uses multitudes of ways to denote compound words and there is no agreed upon, spelling standard for compounds (Gelbukh, 2018). For example, the word school can be represented in Amharic like "ትምህርት-ቤት", "ትምህርት ቤት" and "ትምህርትቤት". This kind of non-uniform representation of the same word is not suitable for sentiment analysis; therefore, we must form uniformity by choosing only one of the representations.

4.4.3 Amharic Short form of Words

For suitability or other reason, people like to write words in short form. But the way how to write a short form of a word or phrase is not uniform. In Amharic language, people use dot operator "." Or forward slash "/" to make a short form of a word or phrase and this create confusion in sentiment mining. Let us see this with an example, the capital city of Ethiopia, Addis Ababa can be written in Amharic like "አ.አ" or "አ/አ". Therefore we must deal with this kind of issues in this research work.

4.4.4 Words adapted from foreign languages

Non-uniform representation is also the problem we may face in opinion mining in Amharic language. For example, the word director represents in different ways, sometimes as "ዲሬክተር" or "ዳይሬክተር". To reduce confusion, Mersehaizen (Mersehaizen W/Mariam, 1934) suggest using a uniform way of representing words adapted from a foreign language.

4.5 Morphology

Amharic is a morphologically rich language where up to 120 words can be conflated to a single stem (Mindaye et al., 2010). Amharic has a complex inflectional morphology, particularly in the verbal system, employing not only prefixes and suffixes but also internal modification of the typical Semitic consonantal root-and-pattern type. In general, the morphology of Amharic has been less influenced by the Cushitic substratum than, for instance, syntax or the lexicon. The inflectional morphology of noun, on the other hand, is relatively simple (Keith & Sara, 2010).

4.6 Slang Words

Nowadays it is common to use Amharic slang words in social media. These slang words make challenging the detection of opinionated expressions. For example, in the word 'aynefam' or

“አይነፋም” have similar meaning with English phrase not good. Dealing with Amharic slang words will help in sentiment analysis. To handle this kind of slang words, some slang words are added to our lexicon.

Table 4.2 Amharic slang words

Amharic Slang	Phonetic	Meaning in common Amharic	Phonetic
ዝገት	zIget	አሰልቺ	asel_Ici
አላስ	al_as	ምቀኛ	mIqeNa
የማይባትት	yemaybatt	የማይገባው	yem_aygebaw
ፋርጣ	farTa	ፋራ	fara
መንጬ	menCu	መቀማት	meqem_at
ገጀረቸ	gejerece	አምቢ አለቸ	'ambi 'alece
አርካ	arka	ጉራ	gur_a
ቦከማ	bokema	ሰረቀ	ser_eqe
ሸመጠጠ	xemeT_eTe	ዋሸ	wax_e
ጨማቂ	Cemaqi	ወረኛ	wereNa
ነቀለ	neq_ele	ተናደደ	tenad_ede
ቦርኮ	borko	ዝርክርክ	zIrkIrk
ነቄ	neqE	አዋቂ	awaqi
አሰፉ	asef_u	ወረኛ	wereNa
ይሸክላል	yIxekkal	ይደብራል	yIdebral
ድንች	dIn_Ic_	ሀሰተኛ	has_eteN_a

4.7 Punctuation

Punctuation in Amharic language consisting of word-divider or hulet-netib (፡), end of the sentence indicator or arat-netib (።), drib serez (፤), netela serez (፥), and other symbols inherited from the Latin language like (?), exclamation mark (!), quotes (“”) and parenthesis (Mesay, 2003).

4.8 Amharic Numbers

The original Amharic character set has no symbol for representing zero, Negative numbers, decimal points, and mathematical operators for performing a mathematical operation (Mesay,

2003). Amharic numbers that are borrowed from Ge'ez are not suitable for mathematical operations (Mersehaizen W/Mariam, 1934). Consequently, Arabic numerals are used for the representation of numbers and Latin based scripts for operators (Mesay, 2003).

Table 4.3 Amharic numbers (Ge'ez) with Arabic equivalent (Isenberg, 1842)

Amharic numrals	፩	፪	፫	፬	፭	፮	፯	፰	፱	፲	፻	፲፻	፼	፲፼
Arabic numrals	1	2	3	4	5	6	7	8	9	10	100	1000	10000	100000

CHAPTER FIVE

DATA COLLECTION AND PREPROCESSING

5.1 Data collection

Building the list of Sentiment Words

A very basic and simple idea to build a classifier for unannotated data is to use a lexicon of words. A lexicon is a dictionary of words, each word associated with a score showing its polarity. For our work, we gathered 1370 sentiment words out of which, 570 of them are positive sentiment word and 800 of them are negative sentiment words. Out of the 1370 sentiment words 870 of them are collected by translating from English lexicons compiled by Liu (Liu et al., 2005) and then increase their numbers by looking for synonyms from Amharic to Amharic dictionaries. And the rest, 500 lexicons were adapted from Gebremeskel (Gebremeskel, 2010).

Book review data collection

Amharic is one of the languages that have scarcity in labeled reviews, especially in a domain like a book review. As a solution for this, we first collect manually 600 unlabeled book reviews from sources like "facebook", "ethiopaizare.com","cyberethiopia.com", and manually collect from book readers using questionnaire. After collecting these 600 reviews, we have removed unnecessary symbols like '#', and '@'. We also removed subjective sentences or side talks.

5.2 Preprocessing

The majority of the text produced by social websites is considered to have an unstructured or noisy nature (Amira, 2013). The reason for this can be a lack of standardization, spelling mistakes, missing punctuations, nonstandard words, and repetitions. We perform four activities in preprocessing our data. These four activities are tokenization, normalization, stemming and transliteration, and stop word removal.

Tokenization

Given a review, tokenization is the task of chopping it up into pieces, called tokens, perhaps at the same time throwing away certain characters, such as punctuation and numbers. In English, word tokens are delimited by a blank space. But in Amharic language, punctuation marks like

word-divider or (ሁለት ነጥብ) (፡), end of the sentence indicator or (አራት ነጥብ) (።), semicolon or (ድርብ ሰረዝ) (፤), comma or (ነጠላ ሰረዝ) (፣), and other symbols inherited from the Latin language like question mark or (የጥያቄ ምልክት) (?), and exclamation mark or (ቃለ አጋኖ) (!) can be used as word separators.

Open the file

While not end of file do

For each character in the file

If the character is ' ', '፡', '።', '፤', '፣', '?', and '!'

Split at that point

End if

End for

End while

Fig.5.1 Tokenization algorithm `

Normalization

The normalizing process puts the text in a consistent form, thus converting all the various forms of a word to a common form. Amharic language alphabet contains letters with the same sound but different shapes called homophones. Homophone has been represented with only one symbol, for example, homophones "ሀ","ሐ","ሃ","ኅ","ኻ" are represented only by the symbol "ሀ". Table 4.1 shows the list of homophone in our sentiment classification model (Yacob, 2004).

Table 5.1 Simplification of Phonetically Equivalent Syllables

Phonemic Equivalents	**Simplification**
ሀ, ሃ, ሐ, ሓ, ኀ, ኃ, ኻ and their family	ሀ and it's family
ሰ, ሠ and their family	ሰ and it's family
አ, ኣ, ዐ, ዓ and their family	አ and it's family
ጸ, ፀ and their family	ጸ and it's family

The next step is to handle abbreviations. Punctuation marks like dot operator "።", or forward slash "/" is used interchangeably to form abbreviation. Accepted standards for many abbreviations and acronyms do not yet exist. Instead of just removing dot operator and forward slash, for common abbreviations like "ኢ.አ" or "ኢ/አ" has been replaced with "ኡዲስ አበባ".

While not end of file do

For each character in the file

If the character is ሐ, ኀ, ሃ, ሓ or their family then

Changed to ሀ

Else if it is ሠ or their family then

Changed to ሰ

Else if it is ፀ or their family then

Changed it to ጸ

Else if it ኧ, ኣ, ዐ, ዓ or their family then

Changed it to አ

Else if it is ቍ then

Changed it to ቁ

Else if it is ቈ then

Changed it to ቆ

Else if it is ኵ then

Changed it to ኩ

Else if it is ጎ then

Changed it to ጐ

Else if it is "/" then

*Changed it to"." *

Else if it is "-" then

Changed it to " "

End if

End for

End while

Fig.5.2 Normalization algorithm

Stop words removal

Stop words are most common words found in any natural language which carries very little or no significant semantic context in a sentence (Lemaire, Salperwyck, & Bondu, 2015) and so that the process is not over-influenced by very frequent words (Ceska & Fox, 2009).

Open file-1(corpus)
Open file-2(stop-word)
While not the end of file file-1
    For each term in the file-1
        If the term is in file-2(stop-word) then
            Remove the term
        End if
    End for
End while

Fig.5.3 Stop words removal

Stemming and Transliteration Using Horn-Morpho

In this step, Amharic book review has been transliterated and change to stem. Machine learning algorithms NLTK do not work with Amharic language and we shall change Amharic symbols to English Equivalent representative in Horn-Morpho. Horn-Morpho, use SERA (System for Ethiopic Representation in ASCII) to represent Amharic characters with English character. The input to Horn-Morpho is a file that contains Amharic book reviews and the output is also file but in ASCII representation.

Amharic is a morphologically rich language where up to 120 words can be conflated to a single stem this clearly shows that stemming has a profound effect on the retrieval process of the language documents and (Mindaye et al., 2010). After transliteration complete, we use the output of transliteration, by Horn-Morphon as input to stemming process, Stemming also is done by Horn-Morphon. After Horn-Morphon completes stemming, the output will further be changed to a form suitable for the next process. In this process, special attention is given to the words like ‘ayIm_ec_Im’ or “አይመችም”, because it has another form ‘temec_e’ or “ተመቸ” with a positive orientation which is opposite to original words ‘ayIm_ec_Im’ or “አይመችም”, with negative sentiment orientation. Therefore, we represent two words containing the stem or root and

negative, as “temec_e nEgativ” so that it will keep its original sentiment orientation. In Amharic, there are a lot of words of this type, therefore we have to be careful when we stem.

After installing Horn-Morpho you have to import it like, import l3, and then, pass the file to Horn- Morphine like, l3.anal_file ('am', 'file-1.txt', file-2.txt'), the output from Horn-Morpho note in a format suitable for processing, it contains detain analysis result that we do not want in addition, it contains English word, punctuations, and numbers. Therefore we have to remove unnecessary elements, keeping only the stem and its label indicating the word is negative. The following section shows its process.

BEGIN Store reviews into file-1.txt

Give file-1.txt to Horn-Morpho as input and save the output as file-2.txt
Open file-2.txt
While not end of file-2.txt Do
For each character in file-2.txt
Else the character is Amharic punctuation-marks or digits then
Remove character
End if
End for
For each word in a file-2
If the word is equal to the English word ‘negative’ then
Replace the English word ‘negative’ with Amharic word ‘አሉታዊ’
End if
End for
For each character in file-2.txt
Else the character is Amharic English character then
Remove character
End if
End for
End while

Fig.5.4 Stemming

For transliteration or to produce phonetic representations of the Amharic words in a file, use the function phon_file, and pass it like l3.phon_file (‘am’, input_file, output_file). The reason for transliteration is that classification algorithms in NLTK do not work with Amharic alphabets.

Give file-2.txt to Horn-Morpho as input and save the output as phonetic.*txt*

Open phonetic.*txt*

While not end of phonetic.*txt Do*

For each character in phonetic.*txt*

Else the character is Amharic punctuation-marks or digit or non-English character then

Remove character

End if

End for

End while

Fig.5.5 Translitration

CHAPTER SIX

DESIGN

6.1 Introduction

In this chapter data preparation and design of a hybrid sentiment classification model for Amharic book reviews will be discussed. This will involve combining lexical based classifier with machine-learning classifier in a sequential manner (lexical and then machine learning).

This chapter is organized as follows: section 6.2 discusses lexicon based classifier, section 6.3 then describes the Machine learning classifiers, section 6.4 clarifies how these two methodologies are combined together, and finally, section 6.5 lists the evaluation measures followed in order to evaluate our proposed approaches.

6.2 Lexical based classifier

6.2.1 Pre-processing

In this research work, the first sub-component of the system is preprocessing. Data acquired from various sources often need to be preprocessed before analysis. We have done preprocessing activities like tokenization, normalization, steaming and transliteration on 600 review data and 1370 lexicons.

6.2.2 Sentiment Word Detection

In this process, every word in review checks if it exists in sentiment word collected. Count the number of positive and negative sentiment words in each review and the same time incorporate the effect of valance shifters.

Sentiment Words

Sentiment words are words that express an opinion like ‘gIrum’ or “ግሩም” which means marvelous and this kind of sentiment words are collected and put into two separate files containing positive and negative sentiment words.

Valance Shifters (negations and intensifiers)

As the name indicates, valance shifters are words that change the strength or orientation of opinion word. The first kind (negations) have the effect of changing the orientation from negative to positive and positive to negative. Amharic words like ‘aydel_em’ or “አይደለም” belong to this group. For example, the sentence ‘asdes_ac meShaf ‘ or “አስደሳች መጽሐፍ” roughly means the book is interesting, but if it is followed by the word ‘aydel_em’ or “አይደለም” like in the following sentence ‘asdes_ac meShaf aydel_em“ or “አስደሳች መጽሐፍ አይደለም” the polarity change from positive to negative. The other group of valance shifters, intensifiers, changes only the strength of sentiment-bearing word and words like ‘beTam’ or “በጣም” and ‘Ij_Ig’ or “እጅግ” belongs to this group.

6.2.3 Polarity Word count and valance shifter incorporation

In this step, the 570 positive and 800 negative sentiment words collected are put into two separate files. These opinion words transliterated, stemmed and used in lexicon based classification. To determine the weight of a review, each word in review has been check if it is sentiment word or valance shifter, and the following rules are used:

Rule 1: if a word in the review found in positive lexicon file and not followed by negation

- Positive_count=Positive_count+1

Rule 2: if a word in the review found in positive lexicon file and followed by negations

- Negative_count=Negative_count+1

Rule 3: if a word in the review found in negative lexicon file and not followed by negation

- Negative_count=Negative_count+1

Rule 4: if a word in the review found in negative lexicon file and followed by negations

- Positive_count=Positive_count+1

Rule 5: if a word in the review found in negative lexicon file, followed by none opinion word, and then followed by Negation.

- Positive_count=Positive_count+1

Rule 6: if a word in the review found in positive lexicon file, followed by none opinion word, and then followed by Negation.

- Negative_count=Negative_count+1

Rule 7: if a word in the review found in positive lexicon file and proceeded by intensifiers

- Positive_count=Positive_count+2

Rule 8: if a word in the review found in negative lexicon file and proceeded by intensifiers

- Negative_count = Negative_count +2

Rule 9: if a sentence contains the word 'bihonIm' or "ቢሆንም" disregard all previous sentiment and only take the sentiment of the one after 'bihonIm' or "ቢሆንም".

6.2.4 Sentiment Classification

Here we will decide whether the polarity of each review, in such a way that:

- If the number of negative terms (negative_count) greater than the number of positive counts the review has been labeled as negative.
- If the number of positive terms (Positive_count) greater than the number of negative counts the review has been labeled as positive.
- Otherwise, it is unclassified

6.3 Machine Learning Component Design

Data Annotation

We have manually annotated 600 reviews consisting of 300 positive, 300 negative reviews to be our training corpus. We already preprocess the review data in the lexical component of the hybrid model we do not have to do preprocessing here.

6.3.1 Feature selection

Feature selection is an important preprocessing stage of text classification, which increases the performance of a predictive model (Adel et al., 2014). To choose a subset of high discriminative features and eliminate the non-discriminative features, in this study, we investigate the performance of three common feature selection methods namely, **Chi-Square**, **Mutual-Information** and **Galavotti-Sebastiani-Simi (GSS) Coefficient** and combinations of the two highest performing feature selection methods for Amharic text classification. For this purpose, three classifiers are used to conduct the experiments namely **Naïve Bayes**, S**upport Vector Machine**, and L**ogistic-Regression**. At the end of this experiment, we identified which selection

method and classification algorithm, perform best and use this selection method for a hybrid model for Amharic book review as sub-component.

6.3.2 Training and Testing Classifiers

In this step, we have put labeled reviews into feature vectors, a format understandable by the classifier. By feature, we mean that to capture the pattern of the data selected and the entire dataset must be represented in terms of them before it is fed to a machine learning algorithm (Abreham, 2014). We chose to work with NLTK classification packages, scikit-learn library and python programming. NLTK and scikit-learn library together they provide several machine learning algorithms such as SVM, Naïve Bayes, and Logistic-Regression and others. It also provides a number of test options, such as cross-validation, test set, and percentage split.

We split our data into training and testing in a ratio of 9:1 and use cross-validation for evaluation. By using different feature selection methods we select feature that supposed to represent the data at hand. Then, we transform the review into a collection of selected unigram features. After that, all three algorithms are trained and tested. Finally, we select the best combination of feature selection method and algorithm to use for machine learning component of the hybrid classifier.

6.4 Proposed Hybrid Approach

To take advantage of the benefits of each approach, we combined lexical based and machine learning classifiers in a sequential manner to form a hybrid approach for sentence-level sentiment analysis.

6.4.1 Architecture of Hybrid Approach for Amharic Book Review

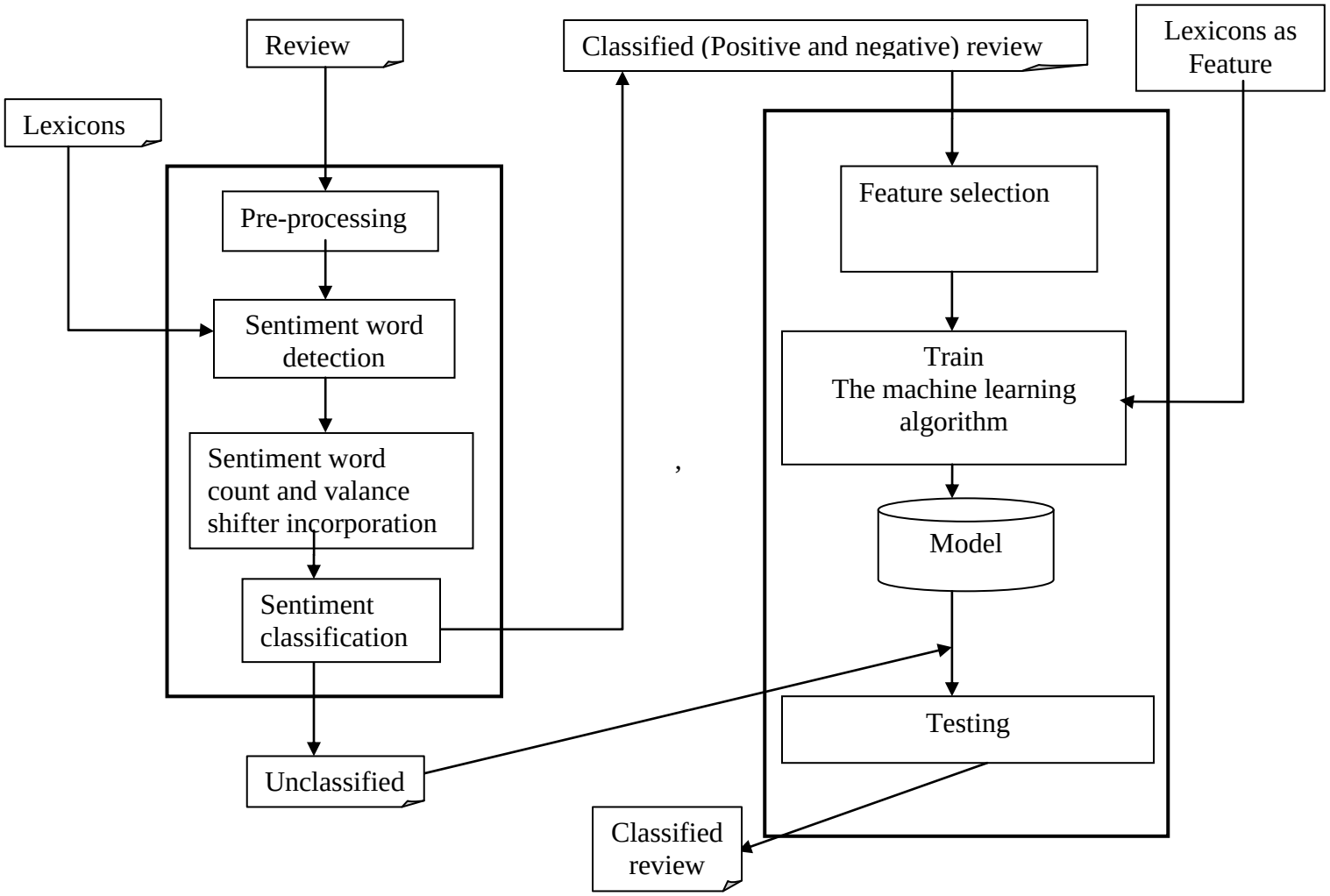


Fig. 6.1 Architecture of hybrid approach for Amharic book review

The general architecture for a hybrid approach for Amharic book review is shown in figure 4.1. As shown in the figure, the system contains different component based on the process required. Generally, the proposed model composed of two major components, lexicon based component and machine learning component.

The lexicon component contains sub-components like preprocessing, opinion word detection and counting, polarity classification. After collecting the review the first thing we have done is cleaning the data by identifying and eliminate non-textual content and, content that is irrelevant to the area of study from the data, and normalization of the data, this is what we call preprocessing. Then using python programming, sentiment words and valance shifters identified, count the number of positive and negative words, incorporate the effect of valance shifters and based the result classify the review as positive and negative. The review classified as positive or negative used to train the machine learning algorithm and the review that cannot be classified by lexicon based part is the input for the machine learning for further analysis.

When we come to the second component (machine learning part) it contains feature selection, machine learning algorithm training, and testing. The model uses the output of lexical based component as training data for a machine learning component and also incorporates the knowledge of lexicon to improve performance. The machine learning algorithm decides the categories of unclassified review.

6.4.1.1 Lexical Component

Since lexical component discussed in sections before and no new thing is added here we will not discuss it again. But, one thing to recall is that the lexical classifier has two types of output; these are classified review and unclassified reviews. These two outputs are given as input to a machine learning algorithm. In the proposed hybrid approach, we do not train our machine learning algorithm using manually labeled training data. Instead, we use classified reviews, which is the output lexical classifier. In this way, we save time that otherwise would be spent to classify the review.

6.4.1.2 Machine Learning Component

At this stage, we already identified which feature selection method and algorithm to use. We also have review annotated by the lexical classifier for Training the algorithm selected. And finally, we have unclassified reviews that cannot be classified by the lexical component, which will be used for testing purpose.

Feature selection

Unigram is used as a feature, for that reviews have chopped down into unigram words. And then, using the best performing feature selection method, among chi-square, galavotti-sebastiani-simi (GSS) coefficient and mutual information gain, we select a subset of unigram features. In addition to the feature selected by feature selection method (let us call them feature-group-1), to improve the performance of the classifier, we incorporate lexicon as a feature (let us call them feature-group-2).

Training machine learning algorithm

By using review labeled by the lexical classifier and using the above two group of features combined as a feature, we train a machine learning algorithm.

Testing machine learning algorithm

After training, the performance of machine learning algorithm tested by the reviews which could not be classified by the lexical component.

6.5 Evaluation Measures

In this research work classification models mainly evaluated with an accuracy of the model against test data that contain labeled positive and negative classes. In addition to accuracy, other measures like precision, recall and f1 sore will be used.

For comparing the performance of the classifying algorithms we use K-Fold cross-validation method. With K-fold cross-validation, the available data is partitioned into k separate sets of approximately equal size (Craven, 1996). The cross-validation procedure involves k iterations in which the learning method is given k-1 as the training data and the rest used as the testing data. Iteration leaves out a different subset so that each is used as the test set once (Craven, 1996). Since the training and testing are repeated k times with different parts of the original dataset, it is

possible to average all test errors in order to obtain a reliable estimate of the model performance on the test data (Nelles, 2001). This approach is advantageous as each test set is independent of the others (Omary & Mtenzi, 2010). In the experiments performed, 10-fold cross-validation (k=10) has been used to evaluate classifiers performance.

For the purpose of evaluation, four metrics are used these are accuracy, precision, recall, and f1-measure.

Table 6.1 Confusion Matrix

		predicted class	
		p	n
actual class	p	true positive (tp)	false negative (fn)
	n	false positive (fp)	true negative (tn)

Accuracy

Accuracy is the ratio of correctly predicted instances. Accuracy can be calculated as follows:

Accuracy = $\frac{tp+tn}{tp+tn+fp+fn}$……6.1

Precision

Precision is about how precise our model is out of those predicted positive, how many actually positive. Precision calculated as:

Precision = $\frac{tp}{tp+fp}$……6.2

Recall

Calculate how many of the actual positives, our model capture through labeling it as positive (true positive). Recall calculated as:

Recall = $\frac{tp}{tp+fn}$……6.3

Where:

True Positives (TP): Predicted as positive instances that were actually positive.
True Negatives (TN): Predicted as negative instances that were actually negatives.
False Positives (FP): Predicted as positive but were negative instances.
False Negatives (FN): Predicted as negative but were positive instances.

CHAPTER SEVEN

EXPERIMENTS AND EVALUATION

7.1 Experimental Setup

Since the proposed hybrid model for Amharic book review combine lexical and machine learning, the experiment includes both machine learning (supervised) and lexical (unsupervised).

Generally, the experiments done can be classified into three groups these are lexical experiment (unsupervised), Machine learning experiment (supervised), and Combining lexical and machine learning. In the first group, only one experiment has been done, and the aim of this experiment is to know the performance of our lexical classifier. In the second group, 36 experiments have been done, with the combination of 3 feature selection methods, 3 machine learning classifiers, and with 4 different numbers of features. The aim of this second group of experiments is to select a combination of feature selection method and machine learning algorithm pair. In the last groups of the experiment, 2 experiments have been done with or without incorporating lexical knowledge in machine learning as a feature. The aim of these last groups of experiments is to see the effect of incorporating lexical knowledge in machine learning. These three groups of experiments will be discussed below:

A. lexical experiment(unsupervised)

The researcher writes an algorithm to classify reviews, based on sentiment word counting using python programming and test the performance of this algorithm based on different performance metrics like accuracy, precision, recall, and f-measure.

The outputs of this step are classified and unclassified reviews. And classified reviews are used to train in the latter experiment, in hybrid classifier as training dataset and, the unclassified reviews are input to the third group of experiments, as testing dataset.

B. Machine learning experiment(supervised)

In this experiment, three supervised machine learning algorithms were used which are**:** Naive Bayes, Logistic Regression, and SVM Classifier. All the above classifiers were tested using different feature selection methods (Chi-Square, Mutual Information Gain, and GSS) and a different number of features (750, 1000, 1250 and 1500 features). We test each technique individually and evaluate its performance. The procedure is, as is standard in supervised machine learning tasks, first training a classifier on pre-classified training data and then evaluating the performance of the classifier on an unlabeled set of test data.

For the purpose of these experiments, the researcher uses NLTK and SCIKIT-LEARN library with python as a programming language. At the end of the experiment, we will identify the performance of each feature selection method mentioned above. We also know which classification algorithm performs well.

C. Combining lexical and machine learning

Here, we combine the lexical and machine learning technique for performing the experiment. First machine learning algorithm, which was selected in supervised machine learning experiment above, will be trained with review data that is labeled positive or negative by unsupervised technique (lexical technique). Second machine learning algorithm selected in the supervised experiment will be tested with an unclassified review of lexical output. For testing purpose, the researchers classify unclassified review manually and use it as criteria to check the performance of the machine learning algorithm.

7.2 Experimental Result

7.2.1 Lexical Experiment Result

Based on rules developed in chapter five, section 6.2.3, algorithm to classify the review as positive or negative was written using python programming. All 600 reviews are given to this algorithm, and the algorithm classifies the reviews. Then the result is evaluated against the actual label (manually labeled). The table below shows the result from lexical based classifier:

Table 7.1 result of lexical classifier

Class	Recall	Precision	F1-measure	Accuracy
positive	0.937	0.845	0.888	74%
negative	0.820	0.926	0.870	

Explanation of the result

- Most of the file that is positive correctly identified as such, with 93.7 % recall. This means very few false negatives in positive class.
- But, a file given a positive classification is only 84.5 % likely to be correct. Not so good precision leads to 14.5% false positives for the pos label.
- Any file that is identified as negative is 88.8 % likely to be correct. This means a few false positive in the negative class.
- But, many files that are negative are incorrectly classified.
- The positive class has higher F1-measure 88.8%.

7.2.2 Supervised Machine Learning

Machine learning works on one principle that, if you give it garbage data it will give you garbage as output. In addition to cleaned data, the features you use highly affect the result you get from machine learning. There are different feature selection methods available, and identifying which feature selection method to use is very important. In this regard, there are different researches in English, Arabic and other languages. But as far as the researchers are concerned, there is no research on the comparison of feature selection methods in Amharic language. Most sentiment classification researches on Amharic language focus only on comparing the performance of machine learning algorithms. That is why the researcher interested to investigate different feature selection methods.

It is necessary to extract clues from the text that may lead to correct classification, In order to perform machine learning (Abreham, 2014). Based on previous works unigrams outperform bigrams in works like (Tan, 2007; Ramdass, 2009) and according to Oystein (Oystein, 2009), among feature selection methods Chi-Square, and Mutual Information Gain perform best also, the Chi-Square variant GSS coefficient was also among the top performers. But the performance

of these feature selection methods against each other is not known in Amharic book review domain. Therefore, we are interested in exploring the performance of these feature selection methods.

For the purpose of this experiment three feature selection methods (Chi-square, Galavotti-Sebastiani-Simi (GSS) Coefficient, and Mutual Information gain) and three machine learning algorithms like Naive Bayes, Logistic Regression, and SVM Classifier algorithms are used.

The experiment will be done using 750, 1000, 1250 and 1500 number of feature on all feature selection methods and machine learning algorithm and compare the result. The next sections present the result:

7.2.2.1 Experimental result using basic naïve bayes

The researcher, conduct the first experiment by using Naïve Bayes in three stages. In all three stages, the three feature selection methods used one at a time and see the result.

A. 750 numbers of features

In the first step, we use 750 numbers of features and conduct an experiment on Naïve Bayes by using different feature selection methods like Chi-Square, GSS, and Mutual Information Gain, and combining Chi-Square and Mutual Information Gain feature selection methods.

Table 7.2 below present experimental result of Naïve Bayes with 750 numbers of feature, and different feature selection methods. Based on the result, Naïve Bayes works well with Mutual Information Gain feature selection method with 76.81% of Accuracy.

Table 7.2 Experimental result of Naïve Bayes with 750 numbers of features

	Chi-Square	Combined(MI+Chi-Square)	simplified chi	MI
Accuracy	**66.6666667**	**59.4202899**	**55.0724638**	**76.8115942**
Precision-Positive	0.62903226	0.58208955	0.64285714	0.70909091
Recall- Positive	1	1	0.46153846	1
F-measure- Positive	**0.77227723**	**0.73584906**	**0.53731343**	**0.82978723**
Precision-Negative	1	1	0.48780488	1
Recall- Negative	0.23333333	0.06666667	0.66666667	0.46666667
F-measure- Negative	**0.37837838**	**0.125**	**0.56338028**	**0.63636364**

When we come to positive reviews when we see F-measure using MI as a feature selection method, every review that is identified as positive is 82.97% likely to be correct. That means among review labeled as positive 17.03% are falsely identified as positive. For negative using MI feature selection method, every review that is identified as negative is 63.63% likely to be correct. That means among review labeled as negative 36.37% are falsely identified as negative.

B. 1000 numbers of features

In the second stage, we use 1000 numbers of features to experiment with different feature selection methods on Naïve Bayes.

Table 7.3 Experimental result of Naïve Bayes with 1000 numbers of features

	Chi-Square	Combined(MI+Chi-Square)	simplified chi	MI
Accuracy	**72.4637681**	**65.2173913**	**47.826087**	**85.5072464**
Precision-Positive	0.75	0.62711864	0.56	0.79591837
Recall- Positive	0.76923077	0.94871795	0.35897436	1
F-measure- Positive	**0.75949367**	**0.75510204**	**0.4375**	**0.88636364**
Precision-Negative	0.68965517	0.8	0.43181818	1
Recall- Negative	0.66666667	0.26666667	0.63333333	0.66666667
F-measure- Negative	**0.6779661**	**0.4**	**0.51351351**	**0.8**

As can be seen from the result, MI feature selection method out-performs the other feature selection methods. When using 1000 numbers of features Naïve Bayes performs best with an accuracy of 85.50% which is better than we get when we use 750 numbers of features. In positive class when using MI as a feature selection method F-measure improve when we increase the number of features from 750 to 1000.Among the review labeled as positive 88.63% likely to be correct. The rest 11.34% of the time it is falsely labeled as positive. In negative review when using MI as a feature selection method, among the review labeled as negative 80% likely to be correct, and the rest 20% of a time it is falsely identified as negative.

C. 1250 numbers of features

In the third stage, we use 1250 numbers of features to experiment with different feature selection methods on Naïve Bayes.

Table 7.4 Experimental result of Naïve Bayes with 1250 numbers of features

	Chi-Square	Combined(MI+Chi-Square)	simplified chi	MI
Accuracy	**66.47**	**78.23**	**72.74**	**92.94**
Precision-Positive	0.681	0.736	0.901	0.910
Recall- Positive	0.809	0.976	0.593	0.973
F-measure- Positive	**0.737**	**0.838**	**0.713**	**0.940**
Precision-Negative	0.620	0.933	0.901	0.910
Recall- Negative	0.460	0.503	0.913	0.870
F-measure- Negative	**0.524**	**0.650**	**0.733**	**0.910**

As can be seen from the result MI feature selection method out-performs the other feature selection methods. When using 1250 numbers of features Naïve Bayes performs best with an accuracy of **92.94**%. In positive class when using MI as a feature selection method F-measure improve when we increase the number of features from 1000 to 1250.Among the review labeled as positive 94% likely to be correct. The rest 11.34% of the time it is falsely labeled as positive. In negative review when using MI as a feature selection method, among the review labeled as negative 91% likely to be correct, and the rest 9% of a time it is falsely identified as negative.

D.1500 numbers of features

As shown in Table 7.5 below, when we increase the number of features to 1500, the result is improved.

Table 7.5 Experimental result of Naïve Bayes with 1500 numbers of features

	Chi-Square	Combined(MI+Chi-Square)	simplified chi	MI
Accuracy	**91.96**	**92.74**	**79.21**	**93.33**
Precision-Positive	0.911	0.932	0.907	0.941
Recall- Positive	0.956	0.941	0.709	0.941
F-measure- Positive	**0.932**	**0.936**	**0.793**	**0.940**
Precision-Negative	0.929	0.917	0.907	0.941
Recall- Negative	0.862	0.908	0.904	0.920
F-measure- Negative	**0.893**	**0.911**	**0.782**	**0.919**

In the third stage, using 1500 numbers of features there is a big improvement in the result in Chi-Square. This is because chi-square gave a high score to rare words, and, these rare words are selected, before frequent words that matter but if you increase the number of feature this effect is reduced since frequent words that matter will be included and finally the result will be improved.

In this stage feature selection method that combined Chi-Square and MI perform better with Accuracy of 93.33%. Positive class using a combined feature selection method, review labeled as positive is 94% likely to be correct, but 6% of a time it is falsely labeled as positive.

For negative class 91.9% of time likely to be correct and 8.1% of a time falsely labeled as negative. To summarize the above three experiment, Naïve Bayes performs well in a different number of features and Chi-Square and combined feature selection methods work well when the number of features increased. At a high number of features, MI perform better than others.

7.2.2.2 *Experimental result using Logistic Regression classifier*

The researcher, conduct the second experiment by using Logistic Regression classifier like the experiment in naïve Bayes, in three stages. In all three stages, the three feature selection methods used one at a time and see the result.

A. 750 numbers of features

The result when using 750 features on Logistic Regression classifier and different feature selection methods presented in table 7.6 below:

Table 7.6 Experimental result of Logistic Regression with 750 numbers of features

	Chi-Square	Combined(MI+Chi-Square)	simplified chi(GSS)	MI
Accuracy	**81.1594203**	**68.115942**	**69.5652174**	**81.1594203**
Precision-Positive	0.80952381	0.68888889	0.73684211	0.75
Recall- Positive	0.87179487	0.79487179	0.71794872	1
F-measure- Positive	**0.83950617**	**0.73809524**	**0.72727273**	**0.85714286**
Precision-Negative	0.81481481	0.66666667	0.64516129	1
Recall- Negative	0.73333333	0.53333333	0.66666667	0.56666667
F-measure- Negative	**0.77192982**	**0.59259259**	**0.6557377**	**0.72340426**

From the table above, we notice that Logistic Regression perform well even when the number of features is less. Both Chi-Square and MI perform 81.15% of accuracy. But when we see their result, in terms of f-measure MI is 85.71% to be accurate on positive class labeling, and out of the reviews labeled as positive, 14.69% of a time it is falsely labeled as positive. On the other hand, Chi-Square performs better when it comes to negative class labeling. Chi-Square label with 77.19% outperforms MI which scores only 72.34% in f-measure.

B. 1000 numbers of features

In the second stage of the experiment we use 1000 numbers of features and with a different number of feature selection methods one at a time. In this step, MI increase in performance and score 82.60 of accuracy. For positive class Logistic Regression reviews are 86.67% are likely to be correctly labeled as positive. But negative class label assignment is only 75% likely to be correct. Generally, MI still shows improvement as the number of features increase.

Table 7.7 Experimental result of Logistic Regression with 1000 numbers of features

	Chi-Square	Combined(MI+Chi-Square)	simplified chi	MI
Accuracy	**73.9130435**	**69.5652174**	**62.3188406**	**82.6086957**
Precision-Positive	0.73333333	0.6875	0.65116279	0.76470588
Recall- Positive	0.84615385	0.84615385	0.71794872	1
F-measure- Positive	**0.78571429**	**0.75862069**	**0.68292683**	**0.86666667**
Precision-Negative	0.75	0.71428571	0.57692308	1
Recall- Negative	0.6	0.5	0.5	0.6
F-measure- Negative	**0.66666667**	**0.58823529**	**0.53571429**	**0.75**

C. 1250 numbers of features

In the third stage of the experiment we use 1250 numbers of features and with a different number of feature selection methods one at a time. In this step, MI increase in performance and score 87.25 of accuracy. For positive class Logistic Regression reviews are 88.7% are likely to be correctly labeled as positive. But negative class label assignment is only 84.7% likely to be correct. Generally, MI still shows improvement as the number of features increase.

Table 7.8 Experimental result of Logistic Regression with 1250 numbers of features

	Chi-Square	Combined(MI+Chi-Square)	simplified chi	MI
Accuracy	**66.47**	**71.56**	**86.27**	**87.25**
Precision-Positive	0.681	0.672	0.870	0.899
Recall- Positive	0.809	1.0	0.897	0.876
F-measure- Positive	**0.737**	**0.803**	**0.883**	**0.887**
Precision-Negative	0.620	1.0	0.870	0.899
Recall- Negative	0.460	0.309	0.807	0.861
F-measure- Negative	**0.524**	**0.465**	**0.826**	**0.847**

D. 1500 numbers of features

When using 1500 features MI decrease in performance, but MI score 86.27 in accuracy. For positive class, 88.1% likely to be correct and 11.9% of a time reviews classified as positive are labeled falsely. For negative class only 83% correctly identified as negative and 17% of a time it is falsely classified as negative.

Table 7.9 Experimental result of Logistic Regression with 1500 numbers of features

	Chi-Square	Combined(MI+Chi-Square)	simplified chi	MI
Accuracy	**85.88**	**85.88**	**80.58**	**86.27**
Precision-Positive	0.861	0.857	0.764	0.862
Recall- Positive	0.901	0.898	0.958	0.904
F-measure- Positive	**0.878**	**0.876**	**0.847**	**0.881**
Precision-Negative	0.858	0.856	0.764	0.862
Recall- Negative	0.800	0.804	0.605	0.809
F-measure- Negative	**0.824**	**0.827**	**0.723**	**0.830**

7.2.2.3 *Experimental result using SVM classifier*

Here the same as experiments before, the experiment done in 3 stages using 750, 1000 and 1500 numbers of features.

A. 750 numbers of features

Based on the result on table MI still outperforms other feature selection methods and score 82.60 of accuracy.

Table 7.10 Experimental result of SVM classifier with 750 numbers of features

	Chi-Square	Combined(MI+Chi-Square)	simplified chi	MI
Accuracy	**76.8115942**	**68.115942**	**56.5217391**	**82.6086957**
Precision-Positive	0.79487179	0.74285714	0.63636364	0.76470588
Recall- Positive	0.79487179	0.66666667	0.53846154	1
F-measure- Positive	**0.79487179**	**0.7027027**	**0.58333333**	**0.86666667**
Precision-Negative	0.73333333	0.61764706	0.5	1
Recall- Negative	0.73333333	0.7	0.6	0.6
F-measure- Negative	**0.73333333**	**0.65625**	**0.54545455**	**0.75**

B. 1000 numbers of features

When using **SVM** classifier, Increasing the number of feature from 750 to 1000 shows a negative impact on Chi-Square, MI and Combined (MI+Chi-Square). Here MI still performs better with an accuracy of 78.26%.

Table 7.11 Experimental result of SVM classifier with 1000 numbers of features

	Chi-Square	Combined(MI+Chi-Square)	simplified chi	MI
Accuracy	**72.4637681**	**66.6666667**	**56.5217391**	**78.2608696**
Precision-Positive	0.7173913	0.65384615	0.61538462	0.72222222
Recall- Positive	0.84615385	0.87179487	0.61538462	1
F-measure- Positive	**0.77647059**	**0.74725275**	**0.61538462**	**0.83870968**
Precision-Negative	0.73913043	0.70588235	0.5	1
Recall- Negative	0.56666667	0.4	0.5	0.5
F-measure- Negative	**0.64150943**	**0.5106383**	**0.5**	**0.66666667**

C. 1250 numbers of features

In the third stage of experiment we use 1250 numbers of features and with a different number of feature selection methods one at a time. In this step, MI increase in performance and score 87.25 of accuracy. For positive class Logistic Regression reviews are 89.6% are likely to be correctly labeled as positive. But negative class label assignment is only 87% likely to be correct. Generally, MI still shows improvement as the number of features increase.

Table 7.12 Experimental result of SVM classifier with 1250 numbers of features

	Chi-Square	Combined(MI+Chi-Square)	simplified chi	MI
Accuracy	**67.05**	**79.60**	**77.45**	**88.62**
Precision-Positive	0.639	0.741	0.827	0.929
Recall- Positive	1.0	1.0	0.779	0.867
F-measure- Positive	**0.778**	**0.850**	**0.800**	**0.896**
Precision-Negative	1.0	1.0	0.827	0.929
Recall- Negative	0.201	0.501	0.757	0.914
F-measure- Negative	**0.332**	**0.660**	**0.727**	**0.870**

D. 1500 numbers of features

When we increase the numbers of features from 1250 to 1500, a combination of MI and Chi-square perform better with an accuracy of 88.03. For positive class, 89.40% likely to be classified as positive correctly, and only 10.6% of classified as positive are falsely categorized as positive.

Table 7.13 Experimental result of SVM classifier with 1500 numbers of features

	Chi-Square	Combined(MI+Chi-Square)	simplified chi	MI
Accuracy	**85.49**	**88.03**	**77.84**	**86.47**
Precision-Positive	0.844	0.882	0.819	0.863
Recall- Positive	0.921	0.912	0.803	0.905
F-measure- Positive	**0.877**	**0.894**	**0.805**	**0.882**
Precision-Negative	0.877	0.883	0.819	0.863
Recall- Negative	0.775	0.838	0.748	0.817
F-measure- Negative	**0.815**	**0.855**	**0.730**	**0.835**

7.2.3 Combining Lexical and Machine

Both lexical and machine learning approaches have their own drawback. To compensate this by taking the advantages of the two approaches the researcher combined them. From previous experiments on machine learning algorithms, we identified a classification algorithm to use (i.e. Naïve Bayes algorithm) and feature selection method (i.e. Mutual-Information) in our hybrid model as a machine learning component.

Unlabeled reviews are given to lexical classifier. Then if the review cannot be classified as positive or negative, it will be passed to a machine learning algorithm that is Naïve Bayes with MI feature selection method. The machine learning decides on the label of this unclassified review.

Data for Training

507 Reviews, that are classified as either positive or negative by the lexical classifier, used for training for the machine learning component of a hybrid model for Amharic book review. By doing this ***we avoided the need for labeling the data manually***, this is important for a language like Amharic that has a scarcity of labeled data especially domains like a book review.

Data for Testing

93 Reviews that could not be classified as either positive or negative previously by the lexical classifier, are used for testing the performance of the machine learning component of a hybrid model for Amharic book review. We manually labeled unclassified review and use it for testing purpose.

Experiment and result for the machine learning part trained by the lexical output

Lexical component of a hybrid classifier performance is already known from the previous experiment and 93 of unclassified review which could not be classified by lexical component pass to Naïve Bayes algorithm with MI feature selection method and the final decision on classifying these unclassified reviews. The result of the experiment is shown in the table below:

Table 7.14 machine learning using naïve Bayes trained by the output of lexical

Class	Recall	Precision	F1-measure	Accuracy
positive	**0.918**	**0.739**	**0.819**	79.45
negative	**0.666**	**0.888**	**0.761**	

- Most of the file that is positive correctly identified as such, with 91.8% recall. This means a few false negatives in positive class.
- But, a file given a positive classification is only 73.9% likely to be correct. Not so good precision leads to 26.1% false positives for the pos label.
- Any file that is identified as negative is 82% likely to be correct. This means a few false positive in the negative class.
- But, many files that are negative are incorrectly classified. Low recall causes 38% false negatives for the negative label.
- The positive class has higher F1-measure of 81.9%.

Experiment and result for machine learning part trained by lexical output (Effect of lexicon incorporation)

At this experiment, we are incorporating lexicons as features into the machine learning algorithm. The following table shows the effect of incorporating lexicon knowledge on machine learning:

Table 7.15 effect of lexicon incorporation on machine learning

Class	Recall	Precision	F1-measure	Accuracy
positive	1.0	0.732	0.845	83.87
negative	0.711	1.0	0.831	

We can see from the result that, the accuracy of machine learning component improved 79.45 to 83.87.

7.2.4 Summary of the Results

The combined result of section B and C above will be as follows.

Accuracy for hybrid= (Accuracy for LX*total+ (accuracy of ML component*No test data))/total

= (0.74*600+0.83.87*93)/600=0.8711

Where

Accuracy for lexical=0.7411

Total=600

Accuracy of ML component=83.87

No of test data=93

LX: Lexical component

ML: Machine learning component

Table 7.16 Comparison of lexical vs. machine vs. hybrid performance

Accuracy		
Lexical	Hybrid	Machine learning
74%	**87%**	**93%**

From the above result, we can see that the proposed hybrid approach is better than lexical based classifier, but the result found shows that machine learning is still better in accuracy. Even though supervised machine learning perform better than the Hybrid model we develop, one thing to remember that the result of Hybrid model use unsupervised machine learning approach which reduce the time and cost because there is no need of manually labeled data in our hybrid model, therefore hybrid model developed is more useful in language like Amharic which lake organized and labeled data resource. In addition to that, most of the time machine learning classifiers trained in one domain do not perform well in another domain (Ding & Pan, 2016). That means, if the target domain is very different from the source domain, the sentiment analysis performance can deteriorate significantly.

7.3 Findings of the Study

Based on different experiments which are grouped into lexical, machine learning and combination (hybrid) we get the following:

Based on the first group of experiment lexical classifier perform with an accuracy of 74%. And then, from the second group experiment, we found that the combination of Mutual-Information-Gain, Nayev Bayes algorithm and using 1500 numbers of feature perform the best with 93.3% accuracy.

In the third group of experiments, by combining lexical and machine learning we found that hybrid approach performs with an accuracy of 87%.

Finally, by comparing the result found in the three groups of experiments, we found that our hybrid approach with an accuracy of 87%, outperform lexical classifier which performs only 74%. But, our hybrid approaches, outperformed by machine learning approach which performs 93.3%.

In this research work, there are more false positive labeled reviews as compared to false negative reviews. We have learned some reasons for the slanted results. The first reason is that when writing reviews in Amharic, many reviewers use positive opinion terms to express negative opinions. For example: in the review "*መጽሐፉ ልክ መልካም አና ጥሉም ምግብ ቀርቦ ለመብላት ጓጉቶ ጸያፍ ነገር እንዳየበት ሰው ዘጋኝ*" *Polarity: Positive*", the expressed opinion is negative but the system labeled it as positive. This is because the reviewer used the positive opinion terms '*መልካም*' (good), '*ጥሉም*' (delicious), and '*ጓጉቶ*' (interested) in the sentence to express negative opinion towards the film. The second reason we have learned is that most of the lexicons we have used for the experimental purpose are negative lexicons that may contribute to this slanted results.

CHAPTER EIGHT

CONCLUSIONS AND RECOMMENDATIONS

8.1 Conclusions

Nowadays, with the growth of social media like reviews, forum discussions, blogs, micro-blogs, Twitter, comments, and postings in social network sites like Facebook on the Web, individuals, and organizations are increasingly using the content in these media for decision making. But Social Media contains a huge volume of opinion text that is not always easily deciphered. The average human reader will have difficulty identifying relevant sites and extracting and summarizing the opinions in them. Automated sentiment analysis systems are thus needed.

In this research work, we made a sentence-level hybrid sentiment classification for Amharic book reviews. For this, we combined lexical classifier with machine learning in a sequential manner, first lexica and then machine learning. To accomplish our aim, we perform 1 experiment on the lexical classifier, 27 experiments to know the performance of three classifiers (Naïve Bayes, Logistic-Regression, and SVM), three feature selection methods, and three different numbers of features. For hybrid classifier, we also perform two experiments with and without incorporating manually crafted lexicons into machine learning component.

As we can observe from 1 experiment done on the lexical classifier, the lexical classifier performs with an accuracy of 74%. Also, we can observe from the 27 experiments made for machine learning, **mutual information gain feature selection method** with **naïve Bayes** using **1500 numbers of features** perform best with **93.3%** accuracy. In addition, from the 2 experiments done for the hybrid approach with and without incorporating the effect of lexicon on the machine learning component of the hybrid approach, the overall performance of hybrid classifier increased. Finally, by comparing the output of lexical classifier, machine learning, and hybrid approach, the **hybrid approach** with an accuracy of **87%**, outperforming **lexical classifier** with **74% accuracy**. But machine learning with an accuracy of 93.3%, outperforms both lexical and hybrid approach.

Here we conclude by examining factors that makes the sentiment classification problem challenging in Amharic language. For Amharic language, there is no standardized corpus (both

for review data and lexicons) for opinion mining. People usually use positive words in negative reviews, but the word is followed by valance shifters (negation) words like aydel_em or "አይደለም", in this research work we attempt to handle negation using rules developed for a lexical component of the hybrid classifier. Also, the use of Amharic slang words make sentiment classification challenging. To reduce this effect, we incorporated slang words that have sentiment.

8.2 Recommendations

Sentiment analysis research work in Amharic language is at the beginning stage. Developing a full-fledge system in Amharic language needs the contribution of many researchers. In this regard, the researcher recommends the following area as a work in sentiment analysis task for feature work.

The first possible sentiment analysis area to work is opinion spam detection, according to (Yadollahi, Shahraki, & Zaiane, 2017) opinion spam detection is the task detecting opinions in favor of or against a product or a service that malicious users intentionally write to make their target popular or unpopular.

The second possible area to work in sentiment analysis for Amharic language in the feature is subjectivity detection which the task of detecting if a text is objective or subjective. Objective texts carry some factual information, while subjective texts express somebody's personal views or opinion (Liu, 2012). Since sometimes people expressed their view by combining objective and subjective text and identifying objective and subjective sentence is the major task to be accomplished for developing full-fledged sentiment mining system.

The third recommendation is that since sentiment analysis research works highly dependent on the availability of corpus, and Amharic language don't have publicly available corpus. We recommend the development of corpus as feature work.

Based on research for language like English, there is a chance to improve the performance of sentiment classification by combining lexical and machine learning. So I recommend working further research work in hybrid sentiment classification for future work.

References

Abreham, G. (2014). Opinion mining from Amharic entertainment texts. Addis Ababa University, Addis Ababa, Ethiopia.

Adel, A., Omar, N., & Al-Shabi, A. (2014). A comparative study of combined feature selection methods for Arabic text classification. *Journal of Computer Science*, *10*(11), 2232–2239. https://doi.org/10.3844/jcssp.2014.2232.2239

Aggarwal, R., & Gupta, L. (2017). A Hybrid Approach for Sentiment Analysis using Classification Algorithm. *International Journal of Computer Science and Mobile Computing*, *6*, 9.

Amarappa, S., & Sathyanarayana, D. S. V. (2012). Data classification using Support Vector Machine (SVM), a simplified approach. *International Journal of Electronics and Computer Science Engineering*, *3*, 11.

Amira, M. (2013). *Arabic sentence-level sentiment analysis*. The American University, Cairo, Egypt.

Andrew, Y., & Michael, I. (2001). On Discriminative vs. Generative Classifiers: A comparison of logistic regression and naive Bayes. *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, 8.

Appel, O., Chiclana, F., Carter, J., & Fujita, H. (2016). A Hybrid Approach to Sentiment Analysis with Benchmarking Results. In H. Fujita, M. Ali, A. Selamat, J. Sasaki, & M. Kurematsu (Eds.), *Trends in Applied Knowledge-Based Systems and Data Science* (Vol. 9799, pp. 242–254). https://doi.org/10.1007/978-3-319-42007-3_21

Appleyard, D. (2015). *Colloquial Amharic* (2nd ed.). Routledge.

Ashari, A., Paryudi, I., & Min, A. (2013). Performance Comparison between Naïve Bayes, Decision Tree and k-Nearest Neighbor in Searching Alternative Design in an Energy Simulation Tool. *International Journal of Advanced Computer Science and Applications*, *4*(11). https://doi.org/10.14569/IJACSA.2013.041105

Bal, D., Bal, M., van Bunningen, A., Hogenboom, A., Hogenboom, F., & Frasincar, F. (2011). Sentiment Analysis with a Multilingual Pipeline. In A. Bouguettaya, M. Hauswirth, & L. Liu (Eds.), *Web Information System Engineering – WISE 2011* (pp. 129–142). Springer Berlin Heidelberg.

Bernard, H. R. (1996). Qualitative Data, Quantitative Analysis. *CAM Journal*, *8*(1), 9–11. https://doi.org/10.1177/1525822X960080010401

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python* (1st ed). Beijing ; Cambridge [Mass.]: O'Reilly.

Bramer, M. (2009). *Artificial Intelligence. An International Perspective: An International Perspective*. Springer.

Cambria, E., Schuller, B., Liu, B., Wang, H., & Havasi, C. (2013). Knowledge-Based Approaches to Concept-Level Sentiment Analysis. *IEEE Intelligent Systems*, *28*(2), 12–14. https://doi.org/10.1109/MIS.2013.45

Ceska, Z., & Fox, C. (2009). The Influence of Text Pre-processing on Plagiarism Detection. *International Conference RANLP*, 5.

Chauhan, A. (2015). Sentiment Analysis Using Hybrid Approach: A Survey. *Proceedings of Ashish Sentiment AU*, *5*(1).

Chevalier, J. A., & Mayzlin, D. (2006). The Effect of Word of Mouth on Sales: Online Book Reviews. *Journal of Marketing Research*, *43*(3), 345–354.

Craven, W. (1996). *Extracting comprehensible models from trained neural networks*. University of Wisconsin, Madison, USA.

D'Andrea, A., Ferri, F., Grifoni, P., & Guzzo, T. (2015). Approaches, Tools and Applications for Sentiment Analysis Implementation. *International Journal of Computer Applications*, *125*(3), 26–33. https://doi.org/10.5120/ijca2015905866

Das, K., Behera, R. N., & Tech, B. (2017). A Survey on Machine Learning: Concept, Algorithms, and Applications. *International Journal of Innovative Research in Computer and Communication Engineering*, *5*(2), 9.

Dey, L., Chakraborty, S., Bose, B., & Tiwari, S. (2016). Sentiment Analysis of Review Datasets Using Naïve Bayes' and K-NN Classifier. *International Journal of Information Engineering and Electronic Business*, *8*(4), 54–62. https://doi.org/10.5815/ijieeb.2016.04.07

Ding, T., & Pan, S. (2016). An Empirical Study of the Effectiveness of using Sentiment Analysis Tools for Opinion Mining: *Proceedings of the 12th International Conference on Web Information Systems and Technologies*, 53–62. https://doi.org/10.5220/0005760000530062

Dreiseitl, S., & Ohno-Machado, L. (2002). Logistic regression and artificial neural network classification models: a methodology review. *Journal of Biomedical Informatics*, *35*(5–6), 352–359. https://doi.org/10.1016/S1532-0464(03)00034-0

Eberhard, David, M., Simons, F., & Fennig, D. (2019). Ethnologue: Languages of the World. Retrieved February 28, 2019, from Ethnologue website: https://www.ethnologue.com/

Firdyiwek, Y., & Yaqob, D. (1997). *The System for Ethiopic Representation in ASCII*. 9.

Gasser, M. (2012). *HORNMORPHO 2.5 User's Guide*. 55.

Gebremeskel, S. (2010). *Sentiment mining model for opinionated Amharic texts*. Addis Ababa University, Addis Ababa, Ethiopia.

Gelbukh, A. (2018). *Computational Linguistics and Intelligent Text Processing: 18th International Conference, CICLing 2017, Budapest, Hungary, April 17-23, 2017, Revised Selected Papers*. Springer.

Ghag, K., & Shah, K. (2014). SentiTFIDF – Sentiment Classification using Relative Term Frequency Inverse Document Frequency. *International Journal of Advanced Computer Science and Applications*, *5*(2). https://doi.org/10.14569/IJACSA.2014.050206

Goldenberg, G. (2013). *Semitic Languages: Features, Structures, Relations, Processes* (1st ed.). Oxford, UK: OUP Oxford.

Gullapelly, A., & Shanmukhi, M. (2017). A Survey on Supervised Classification Techniques in Machine Learning. *International Journal of Computer & Mathematical Sciences*, *6*(11), 8.

Gupte, A., Joshi, S., Gadgul, P., & Kadam, A. (2014). Comparative Study of Classification Algorithms used in Sentiment Analysis. *International Journal of Computer Science and Information Technologies*, *5*, 4.

Hamdan, H., Bellot, P., & Bechet, F. (2016). *Sentiment Analysis in Scholarly Book Reviews*. 11.

Hao, J., & Priestley, J. L. (2016). *A Comparison of Machine Learning Techniques and Logistic Regression Method for the Prediction of Past-Due Amount*. 7.

Haseena, R. (2014). Opinion Mining and Sentiment Analysis - Challenges and Applications. *International Journal of Application or Innovation in Engineerin g & Management (IJ AI E M), Vol ume 3*(Issue 5).

Hu, M., & Liu, B. (2004). *Mining and Summarizing Customer Reviews*. 10.

Hussaini, F., Padmaja, S., & Sameen, S. (2018). Score-based sentiment analysis of book reviews in Hindi language. *International Journal on Natural Language Computing*, *7*.

Isenberg, K. W. (1842). *Grammar of the Amharic Language*. Watts.

Jacob, E. (2017). Unsupervised Learning for Lexicon-Based Classification. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. *C. Nédellec & C. Rouveirol (Eds.), Proceedings of 10th European Conference on Machine Learning (ECML-98), Chemnitz, Germany*.

Joshi, M., Prajapati, P., Shaikh, A., & Vala, V. (2017). A Survey on Sentiment Analysis. *International Journal of Computer Applications*, *163*(6), 34–38. https://doi.org/10.5120/ijca2017913552

Kandarp, D. (2009). *Study of feature selection algorithms for text-categorization*. University of Nevada, Las Vegas, USA.

kasthuri, S., Jayasimman, L., & Jabaseeli, N. (2016). An Opinion Mining and Sentiment Analysis Technique: A Survey. *International Research Journal of Engineering and Technology(IRJET), 03*.

Kaur, S., & Jindal, S. (2016). A Survey on Machine Learning Algorithms. *International Journal of Innovative Research in Advanced Engineering*, *3*(11), 9.

Keith, B., & Sara, O. (2010). *Concise Encyclopedia of Languages of the World* (1st ed.). Oxford, UK: Elsevier.

Khan, K., Baharudin, B., Khan, A., & Ullah, A. (2014). Mining opinion components from unstructured reviews: A review. *Journal of King Saud University - Computer and Information Sciences*, *26*(3), 258–275. https://doi.org/10.1016/j.jksuci.2014.03.009

Kleinbaum, D., & Klein, M. (2002). *Logistic Regression A Self-Learning Text* (Second). New York: Springer-Verla.

Kotsiantis, S. B. (2007). Supervised Machine Learning: A Review of Classification Techniques. *Informatica 31*, 20.

Lemaire, V., Salperwyck, C., & Bondu, A. (2015). A Survey on Supervised Classification on Data Streams. In E. Zimányi & R.-D. Kutsche (Eds.), *Business Intelligence* (Vol. 205, pp. 88–125). https://doi.org/10.1007/978-3-319-17551-5_4

Leslau, W. (1995). *Reference Grammar of Amharic*. Otto Harrassowitz Verlag.

Lewis, D. D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. In C. Nédellec & C. Rouveirol (Eds.), *Machine Learning: ECML-98* (Vol. 1398, pp. 4–15). https://doi.org/10.1007/BFb0026666

Liu, B. (2012). *Sentiment Analysis and Opinion Mining*. Morgan & Claypool.

Liu, B. (2015). *Sentiment Analysis: Mining Opinions, Sentiments, and Emotions*. https://doi.org/10.1017/CBO9781139084789

Liu, B., Hu, M., & Cheng, J. (2005). Opinion observer: analyzing and comparing opinions on the Web. *Proceedings of the 14th International Conference on World Wide Web - WWW '05*, 342. https://doi.org/10.1145/1060745.1060797

Madnani, N. (2007). Getting started on natural language processing with Python. *Crossroads*, *13*(4), 5–5. https://doi.org/10.1145/1315325.1315330

Manne, S. (2011). A Query based Text Categorization using K-Nearest Neighbor Approach. *International Journal of Computer Applications*, *32*, 6.

Mäntylä, M. V., Graziotin, D., & Kuutila, M. (2018). The evolution of sentiment analysis—A review of research topics, venues, and top cited papers. *Computer Science Review*, *27*, 16–32. https://doi.org/10.1016/j.cosrev.2017.10.002

MARKOS, G. (2010). *Implementing an open source Amharic resource grammar in gf*. Chalmers University of Technology, Göteborg, Sweden.

Martha, Y., & Menzel, W. (2009). Amharic part-of-speech tagger for factored language modeling. *International Conference RANLP*, 428–433.

Matsumoto, Y., Sproat, R., Wong, K.-F., & Zhang, M. (2006). *Computer Processing of Oriental Languages. Beyond the Orient: The Research Challenges Ahead: 21st International Conference, ICCPOL 2006, Singapore, December 17-19, 2006, Proceedings*. Springer.

Medhat, W., Hassan, A., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, *5*(4), 1093–1113. https://doi.org/10.1016/j.asej.2014.04.011

Mehra, N., Khandelwal, S., & Patel, P. (2002). *Sentiment Identification Using Maximum Entropy Analysis of Movie Reviews*. 7.

Mersehaizen W/Mariam. (1934). *ያማርኛ ሰዋሰው* (4th ed.). Berhanenaselam printing.

Mesay, H. (2003). *Line fitting to Amharic ocr : the case of postal address*. Addis Ababa University, Addis Ababa.

Mhaske, N., & Patil, A. (2016). Opinion Mining Techniques for Non-English Languages: An Overview. *International Journal of Computational Linguistics*, *7*(2).

Mindaye, T., Redwan, H., & Atnafu, S. (2010). Searching the Web for Amharic Content. *Journal of Multimedia Processing and Technologies*, *1*(1), 13.

Mittal, N., Agarwal, B., Chouhan, G., Bania, N., & Pareek, P. (2013). Sentiment Analysis of Hindi Review based on Negation and Discourse Relation. *International Joint Conference on Natural Language Processing*, 6.

Muhammad, I., & Yan, Z. (2015). Supervised machine learning approaches: a survey. *ICTACT Journal on Soft Computing*, *05*(03), 946–952. https://doi.org/10.21917/ijsc.2015.0133

Narayan, R., Roy, M., & Dash, S. (2016). Ensemble based Hybrid Machine Learning Approach for Sentiment Classification- A Review. *International Journal of Computer Applications*, *146*(6), 31–36. https://doi.org/10.5120/ijca2016910813

Nasukawa, T., & Yi, J. (2003). Sentiment analysis: capturing favorability using natural language processing. *Proceedings of the International Conference on Knowledge Capture - K-CAP '03*, 70. https://doi.org/10.1145/945645.945658

Nelles, O. (2001). *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models*. Springer Science & Business Media.

Ng, V., Dasgupta, S., & Arifin, S. M. N. (2006). Examining the role of linguistic knowledge sources in the automatic identification and classification of reviews. *Proceedings of the COLING/ACL on Main Conference Poster Sessions* -, 611–618. https://doi.org/10.3115/1273073.1273152

Ojokoh, B. A., & Kayode, O. (2012). A feature–opinion extraction approach to opinion mining. *Journal of Web Engineering*, *11*, 14.

Omary, Z., & Mtenzi, F. (2010). Machine Learning Approach to Identifying the Dataset Threshold for the Performance Estimators in Supervised Learning. *International Journal for Infonomics*, *3*(3), 12.

Oystein, L. (2009). *Feature selection for text categorization*. Norwegian University of Science and Technology, Norwegian.

Palanisamy, P., Yadav, V., & Elchuri, H. (2013). Serendio: Simple and Practical lexicon based approach to Sentiment Analysis. *Second Joint Conference on Lexical and Computational Semantics*, 6. Atlanta, Georgia.

Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, *2*, 1–135.

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up?: sentiment classification using machine learning techniques. *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - EMNLP '02*, *10*, 79–86. https://doi.org/10.3115/1118693.1118704

Park, H.-A. (2013). An Introduction to Logistic Regression: From Basic Concepts to Interpretation with Particular Attention to Nursing Domain. *Journal of Korean Academy of Nursing*, *43*(2), 154. https://doi.org/10.4040/jkan.2013.43.2.154

Parlar, T., Özel, S. A., & Song, F. (2018). QER: a new feature selection method for sentiment analysis. *Human-Centric Computing and Information Sciences*, *8*(1). https://doi.org/10.1186/s13673-018-0135-8

Patel, D., Saxena, S., Verma, T., & Student, P. G. (2007). Sentiment Analysis using Maximum Entropy Algorithm in Big Data. *International Journal of Innovative Research in Science, Engineering and Technology*, *5*(5), 7.

Pawar, A. B., Jawale, M. A., & Kyatanavar, D. N. (2016). Fundamentals of Sentiment Analysis: Concepts and Methodology. In W. Pedrycz & S.-M. Chen (Eds.), *Sentiment Analysis and*

Ontology Engineering (Vol. 639, pp. 25–48). https://doi.org/10.1007/978-3-319-30319-2_2

Pedro, P., Balage, F., & Thiago, A. (2013). NILC USP: A Hybrid System for Sentiment Analysis in Twitter Messages. *Second Joint Conference on Lexical and Computational Semantics*, *2*.

Philemon, W., & Mulugeta, W. (2014). A Machine Learning Approach to Multi-Scale Sentiment Analysis of Amharic Online Posts. *HiLCoE Journal of Computer Science and Technology*, *2*(2), 8.

Phyu, T. N. (2009). Survey of Classification Techniques in Data Mining. *Proceedings of the International MultiConference of Engineers and Computer Scientists*, *1*, 5.

Raghuwanshi, A. S., & Pawar, S. K. (2017). Polarity Classification of Twitter Data using Sentiment Analysis. *International Journal on Recent and Innovation Trends in Computing and Communication*, *5*(6), 6.

Rajput, R., & Solanki, A. K. (2016). Review of Sentimental Analysis Methods using Lexicon Based Approach. *International Journal of Computer Science and Mobile Computing*, 8.

Read, J., & Carroll, J. (2009). Weakly supervised techniques for domain-independent sentiment classification. *Proceeding of the 1st International CIKM Workshop on Topic-Sentiment Analysis for Mass Opinion - TSA '09*, 45. https://doi.org/10.1145/1651461.1651470

Rosenhouse, P. J., & Kowner, R. (2008). *Globally Speaking: Motives for Adopting English Vocabulary in Other Languages*. Multilingual Matters.

Saeys, Y., Inza, I., & Larrañaga, P. (2007). A review of feature selection techniques in bioinformatics. *Bioinformatics*, *23*(19), 2507–2517. https://doi.org/10.1093/bioinformatics/btm344

Sammut, C., & Webb, G. I. (Eds.). (2016). *Encyclopedia of Machine Learning and Data Mining*. https://doi.org/10.1007/978-1-4899-7502-7

Satoshi, M., Kenji, Y., Kenji, T., & Toshikazu, F. (2002). Mining Produ ct Repu tations on the Web. *SIGKDD*.

Saud, S. (2015). *Sentiment Analysis in the Arabic Language Using Machine Learning*. Colorado State University, Colorado.

Scelta, G. F. (2001). *The Comparative Origin and Usage of the Ge'ez writing system of Ethiopia*.

Senecal, S., & Nantel, J. (2004). The influence of online product recommendations on consumers' online choices. *Journal of Retailing*, *80*(2), 159–169. https://doi.org/10.1016/j.jretai.2004.04.001

Sharma, A., & Dey, S. (2012). *Performance Investigation of Feature Selection Methods and Sentiment Lexicons for Sentiment Analysis*. 6.

Sharma, R., Nigam, S., & Jain, R. (2014). Opinion Mining In Hindi Language: A Survey. *International Journal in Foundations of Computer Science & Technology*, *4*(2), 41–47. https://doi.org/10.5121/ijfcst.2014.4205

ShehlaKulsum, K., & Vaidya, S. G. (2017). An Efficient Approach for Sarcasm Recognition on Twitterusing Pattern-Based Method. *International Journal of Advanced Research in Computer Engineering & Technology*, *6*(1), 9.

Singh, K. P., & Agrawal, S. (2017). Sentiment Classification using Machine Learning: A Survey. *International Journal on Recent and Innovation Trends in Computing and Communication*, *5*(5), 5.

Smith, P. (2015). *Sentiment analysis of patient feedback*. University of Birmingham.

Solomon, A., & Menzel, W. (2007). Syllable-based speech recognition for Amharic. *Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages Common Issues and Resources - Semitic '07*, 33. https://doi.org/10.3115/1654576.1654583

Sunitha, c. k., & Edwin, G. (2014). Online Shopping - An Overview. *B-DIGEST*, 6.

Taboada, M., Brooke, J., Tofiloski, M., Voll, K., & Stede, M. (2011). Lexicon-Based Methods for Sentiment Analysis. *Computational Linguistics*, *37*(2), 267–307. https://doi.org/10.1162/COLI_a_00049

Tang, J., Alelyani, S., & Liu, H. (n.d.). *Feature Selection for Classification: A Review*. 33.

Teja, J. S., Sai, G. K., Kumar, M. D., & Manikandan, R. (2018). *Sentiment Analysis of Movie Reviews Using Machine Learning Algorithms - A Survey*. *118*, 8.

Tulu, T. (2013). *OPINION MINING FROM AMHARIC BLOG*. Addis Ababa University, ADDIS ABABA.

Turney, P. D. (2001). Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, 417. https://doi.org/10.3115/1073083.1073153

Vohra, M. S. M., & Teraiya, J. B. (2013). A COMPARATIVE STUDY OF SENTIMENT ANALYSIS TECHNIQUES. *JOURNAL OF INFORMATION*, 5.

Wang, S., & Wang, H. (2008). *A Knowledge Management Approach to Data Mining*. *108*, 622–634.

Wang, W., & Zhou, Y. (2009). E-business Websites Evaluation Based on Opinion Mining. *2009 International Conference on Electronic Commerce and Business Intelligence*, 87–90. https://doi.org/10.1109/ECBI.2009.93

Wiebe, J. M. (2000). Learning Subjective Adjectives from Corpora. *American Association for Artificial Intelligence*, 6.

Wiegand, M., Balahur, A., Roth, B., Klakow, D., & Montoyo, A. (2011). *A Survey on the Role of Negation in Sentiment Analysis*. 9.

Xia, H., Jiliang, T., Huiji, G., & Huan, L. (2013). *Unsupervised Sentiment Analysis with Emotional Signals*.

Yacob, D. (2004). *Application of the Double Metaphone Algorithm to Amharic Orthography*. 13.

Yadollahi, A., Shahraki, A. G., & Zaiane, O. R. (2017). Current State of Text Sentiment Analysis from Opinion to Emotion Mining. *ACM Computing Surveys*, *50*(2), 1–33. https://doi.org/10.1145/3057270

Yaming, Y., & Xin, L. (1999). *A re-examination of text catagorization methods*.

Younis, E. (2015). Sentiment Analysis and Text Mining for Social Media Microblogs using Open Source Tools: An Empirical Study. *International Journal of Computer Applications*, *112*.

APPENDICES

Appendix A: List of Amharic positive sentiment terms in SERA (System for Ethiopic Representation in ASCII)

	ageg_eme	ar_egag_eTe
ab_erar_a	ageN_e	ar_eme
ab_ere	agwagi	ar'aya
ab_eretat_a	agWagWa	arbeN_a
abeleS_ege	aj_ebe	arek_a
aber_ede	akeb_ere	areka
abeT_ere	al_efe	arif
abIn_et	alama	arIn_et
abronet	aleqa	as_am_ere
aC_awac	almaz	as_ebe
aC_Ir	am_elak_ete	as_elas_ele
ad_ab_ere	amare	aS_enan_a
ad_ege	amenEta	as_IqiN_
ad_ele	a'mIro	asad_ege
ad_uN_a	amW_al_a	asam_ene
adane	anbes_a	asaqe
adem_eTe	andafta	asdem_eme
aden_eqe	andeN_a	asden_eqe
adlawi	aneholele	asdes_ete
adnaqot	aneSe	asedesac
afel_eqe	anSebar_eqe	aSefa
afez_eze	anTeleT_ele	asela
ag_elab_eTe	anTelTaynet	asfel_ege
ag_enaz_ebe	aq_erar_ebe	asger_eme
ag_eze	aqIl	asmare
agbab_a	aqIm	asmeseg_ene

asred_a
asrek_ebe
astar_eqe
astekak_ele
astemam_ene
astemari
asTeneq_eqe
astewaS'o
asteway
aTare
aTgabi
aw_aT_a
awenta
awentawi
awTeneTene
ax_en_efe
axeber_eqe
aynafar
ayneteNa
az_eze
aznan_a
balemuya
balewleta
balnIjera
befit
beg_o
bel_a
bel_eTe
belal_a
beq_a
beq_ele
ber_a
ber_ere
bereket
beret_a
bes_ele
bIl_ICa
bIl_ICta
bIlCIlC
bIlha
bIlhat
bIlhateN_a
bIlT
bIlTablIT
bIqat
bIrhan
bIruk
bor_eqe
cale
Cem_ere
cer
cere
cerIn_et
Cewa
cIlota
CIm_Ir
Col_E
def_ar
defer_ese
deg_
deg_efe
deg_In_et
dehna
dehnIn_et
del_ebe
dem_aq
dem_eqe
demam_Im
denb
denbeN_a
denta
des
des_Ita
des_IteN_a
desyIlal
desyIlel
dIbaq
dIl
dIleNa
dIlot
dImqet
dIngIl
dInq
dIrq
faf_a
fana
fata
fayda
fegegta
fek_a
fel_ege
feneT_eze
fenTezIy_a

feqadeN_a
fet_a
feT_an
fetena
fews
fIl_agot
fIlqIlq
fIndeqa
fIqad
fIqadeNa
fIrE
fIrE'ama
fIs_Iha
fIseha
fIthawi
gab_eze
gale
geb_a
gebey_e
gebirawi
gedeb
gehad
gelelteN_a
gen_et
genananet
geneb_a
ger
ger_eme
geseSe
gET
geT_eme
gEta
gETagET
gez_a
gIbIZ_a
gIbregeb
gIlS
gIlS
gIN_It
gIrma
gIrmawi
gIrum
gIzuf
gobez
gola
gox
gug_ut
gulbet
gWadeNa
habt
habtam
halafin_et
hamelmal
haq_
haq_eN_a
harnet
hawariya
hay_al
hayleN_a
haymanoteN_a
hIbret
hIgawi
hIk_ImIn_a
hIli
hIyaw
hIywetawi
hIzbawi
Id_Il
Idget
Im_ebEt
Im_Iq
Imerta
Imnet
In_at
IngIda
InkIbIk_abE
Iq_Id
IqC
Ir_Imat
IrgITeN_a
Iwnet
IwneteN_a
Iwq
jegna
kabete
kase
keberEta
kef_IteN_a
kWale
laqe
lef_a
leg_ese
lega

lemel_eme
lemlem
leslas_a
lez_a
lez_a
lIb_
lIbawi
lIdet
lIkeNa
lI'lIn_a
lImat
lImd
lImlamE
lImuT_
liq
lIy_u
loga
ma'areg
mahteb
mar_eke
ma'rege
medhanit
meftIhE
megneTisawi
mehandis
mehari
mek_ere
meketa
mel_a
melekotawi
mel'Ikt
melkam
melkemelkam
menfesawi
merzeN_a
mes_aC
mes_eTe
mesebe
mIc_ot
mIgb
mIhret
mIhur
mIhurawi
mina
mIr_
mIr_Iqat
mIr_uq
mIrT
mIs_alE
mIsgana
mIT_anE
mizan
mizanawi
moges
mol_a
moq
moqe
moya
mudeNa
mulu
muya
muyawi
naf_eqe
nam_una
naN_e
nebelbal
nek_a
neq_a
neSa
neSan_et
nIbret
nISuh
nWay
qeb_a
qel_al
qeldeN_a
qelTaf_a
qen_a
qEnTeN_a
qeT_Ita
qeTteNa
qICWan
qId_us
qIdmiya
qIlT
qIlTIfIn_a
qImem
qIn
qonjo
qub
qunCo
qurT
qurTeNa

qurTeNnet
quT_eba
quT_Ib
quTbInet
quTIT_Ir
qWam_eTe
qWeT_ebe
ra'Iy
reb_a
red_a
reqiq
ret_a
rIdata
rIgb
rIgum
rIkata
rItu'
ruhru
sabe
seb'awi
sebeb
Sebele
Sed_a
Sedal
sef_a
Seg_a
selam
selamawi
selamteNa
sem_ere
sema't
Sen_a
seq_ele
sereSe
seT_e
sIb'Ina
SIdq
sIkEtama
sIl_IT_anE
sIlt
sIlTun
sImmIn_et
sIr'at
sisay
sITota
Taf_eTe
tag_ele
tag_ese
Ta'Im
tal_aq
tal_eme
tam_ene
tam_Ir
tam_IreN_a
Tame
Tare
tas_ebe
tat_ere
taw_eqe
taw_ese
taz_eze
te'amani
te'amaninet
te'amrawi
Teb_eqe
tebar_eke
Tebib
tebrara
teCawac
tedelad_ele
tedem_eme
tedem_eTe
teden_eqe
tederaj_e
tedla
tefCereC_ere
tefel_ege
teg_a
Teg_ene
tegag_eze
tegbaba
tegbarawi
tegeb_a
tegenez_ebe
teger_eme
tegeTeg_eTe
tegeza
tegsaS
tehadso
tek_a
tekane
Tel_eqe
telem_ede

telew_eTe
telewaw_eTe
tel'Iko
temam_ene
temare
temec_e
temer_eTe
temeseg_ene
temeT_ene
temeTaT_ene
temWal_a
TEna
TEnam_a
tenbogeb_oge
TEneN_a
tenes_a
tenesaxnet
Tenkar_a
Tenkaranet
tenketek_ete
TEnnet
Tenqaq_a
Teq_eme
Teqaminet
teqeb_ele
teqebaynet
TeqemEta
Ter_a
teram_ede
tered_a
teredad_a
teregag_eTe
tergaga
tesak_a
tesemam_a
teseT'o
tesfa
testekakele
teTeb_eqe
teTey_eqe
tetrefer_efe
tewaTa
tewed_ede
tewedad_ere
tewehade
texag_ere
texale
texaxale
TIbebeN_a
tIbIb_Ir
TIbq
TIgab
tIgat
tI'gIst
tI'gIsteNa
tIhtIn_a
tIhut
tIkIk_IleN_a
tIkkIleNanet
TIlq
tIlq
TIm_ona
tImhIrt
tIngIrt
TInquq
TIqIm
TIret
TIrt
tIz_Ita
tub_a
waga
wan_a
wan_eN_a
wastIn_a
waw
webete
wed_ede
wedajIn_et
wedjewalehu
weg
wel_ad
wendIm_amacIn_et
werota
werq
werqam_a
wez
wIb
wId_
wId_asE
wIl
wIleta
wIT_Et
wITEtama

wIyIy_It
wubet
wunet
xeN_e
xIl_Imat
xum
yaze
yewah
yIfa
yIhunta
yIqrIta
zedEN_a
zelalemawi
zelaqinet
zele'alemawi
zeleqEta
zemenawi
zemenawinet
zemeneNa
zenkat_a
zerez_ere
zew_ere
zIgIj_u
zIn_a
zIn_eN_a

Appendix B: List of Amharic negative sentiment terms In SERA (System for Ethiopic Representation in ASCII)

ab_ak_ene
ab_ar_ere
ab_ede
ab_eTe
abase
abelax_e
abesa
abesaC_e
abEtuta
aC_enag_efe
aCbereb_ere
ad_af_ene
ad_al_a
adag_ete
adbeseb_ese
adega
adegeN_a
adek_eme
adenag_ere
adenaq_efe
adma
admeN_a
af_ene
af_ere
afeneg_eTe
afer_ese
ag_aC_e
ag_ad_ele
ag_al_eSe
ag_an_ene
agel_ele
agodefe
agodele
ahIy_a
ahzab
ak_erak_ere
akes_ere
akrari
al_ak_eke
al_efe
al_eqe
alagbab
alageTe
alasfelagi
ale'agbab
aleqT
aleqT
alIb_alE
alubalta
aluta
alutawi
am_a
am_ar_ere
am_etat_a
amaSi
amaSya
ambagen_en
ambagWaro
amel
amel_eTe
amenet_a
amenzari
amer_ere
ameS
an_ad_ede
an_ag_a
an_eqe
an_ese
anaweSe
aneb_a
anek_ese
angebeg_ebe
anqWax_exe
anzar_eTe
aq_as_ete
aq_aT_ele
aq_aT_ere
aq_at_ete
aqaqir
aqate
aqebet

aqlexel_exe
aqWeref_ede
ar_ere
aremenE
arenqWa
arogE
as_asate
as_elec_e
as_eq_aqi
aS_ey_efe
asaf_ere
asas_abi
asceg_ere
asCen_eqe
asdeneg_eTe
asecegari
aselec_e
asfer_a
asged_ede
askef_i
asleq_ese
asmes_ele
asqey_eme
astebab_ele
astegab_a
astegWagole
asTel_a
asTey_efe
asweg_ede
asweT_a
aT_a
aT_abqiN_
aT_ad_efe
aT_am_eme
aT_eraT_ere
atal_ele
aTed_efe
aTefa
aTeqa
aw_ar_ede
aw_elaw_ele
aw_enab_ede
awdelday
awed_eme
awegeze
awezagebe
awrEn_et
ax_eb_ari
ax_ekaka
ax_emaq_eqe
ax_eme
axangul_it
axofe
ayb
az_ab_a
azenEta
ba'd
bado
bado
balegE
barIn_et
bariya
base
bed_el
bedeleN_a
bedIn
bel_eze
beqel
beseb_ese
bet_ene
beTeb_eTe
bex_Ita
bex_IteN_a
bezeb_eze
bIc_a
bIc_eN_a
bIc_eN_In_et
bIceNnete
bId_Ir
bIklet
bIkun
bIlgIn_a
bIlIx_u
bIlSIg_In_a
bIrd
bIsIC_It
bIskIsk
bITIb_IT
bIZta
bok_a
bokete
bozenE
bukata

CaCata
Cana
Care
CefeC_efe
Cefgag_a
Cek_ene
Cel_ema
cel_IteN_a
Celameme
CelemteNa
Cemdad_a
CeqCaq_a
CIfCefa
cIg_Ir
cIk_ul
cIkola
cIkyale
CImCImta
CInq
CInqet
CInqInq
CIq_un
CIqCIq
CIqone
CIr
CIraq
CIret
cIsta
Cohe
Cuhet
daget
dateN_a
debed_ebe
debere
debez_eze
debzaz_a
ded_eb
deg_eme
dek_eme
dekama
dem
demeN_a
demenef
demes_ese
denbar_a
denef_a
deneg_eTe
denez
denqoro
deq_aq
dereq
dewE
dIb_Iq
dIfam
dIggImo
dIha
dIhIn_et
dIkam
dIkmet
dIl_ela
dIl_Iz
dIngeteN_a
dInIg_aTE
dInk
diqala
dIrito
dubda
durye
faqe
fel_a
fened_a
fer_a
fer_ese
fes_ese
fet_
feTaTa
fEz
fezaz_a
fICt
fId_a
fIj_It
fIrhat
fITCa
fogere
funga
gagata
gan
gaTeweT
gebgab_a
ged_ele
gedel
gef_a
gef_efe

gegema
gehan_em
gEja
geleb_eTe
geleba
geljaja
gem_ete
gememteNa
gena
geneT_ele
ger_efe
get_a
gIb
gIb_Iz
gIbsIbs
gIdEle
gIdfet
gIdIy_a
gIf
gIf_it
gIfeN_a
gIl_Ib_aC
gIlfIteN_a
gIm
gImatam
gIr_Ifat
gIr_IgIr_
gIra
gIra
gIra'agabi
gIrfiya
gIt_Ir
gI'uz
gIxbet
godolo
gomIz_aza
gorbaTa
goriT
goseNa
gosqIwala
gub_o
gud
gudeN_a
gudf
gudlete
gul
gur_eN_a
gurmIrmIta
gusqul
gWeda
gWedele
gWesegose
hafret
ham_Et
haraj
has_et
has_eteN_a
haTi'at
haTyat
hazen
hIgeweT
hImem
hIq_Ita
his
hISeS
hIwalaqer
hIwalaqernet
hukata
huket
Ibab
Ibd
Iblet
IbriteN_a
Ida
Idf
iftIhawi
iftIhawinet
Ig_eda
Ik_ek_am
Ik_Il
Ikuy
IlhaNa
Ilqit_
Imba
ImbiteN_a
imnIt
Inba
Inken
InqIfat
InqIlf
InqoqIl_Ix
InToroTo
IrbItbIt

IrgIC_a
IrgIman
Irita
Irqan
Is_Ir
iseb'awi
IssIt
IT_abi
ITot
ITret
Ix_Iruru
jaj_e
jegnIn_et
jelgaga
jIl
jIlIn_et
jIn_In
kade
keb_ad
keb_ede
ked_a
kedateN_a
kef_ele
kefaf_ele
kehadi
kenek_ene
kentu
kerdad_a
kerek_ere
kes_eme
kes_ere
kes_ese
kewkaw_a
kex_efe
keysi
kIftet
kIftet
kIfu
kIhdet
kIlkIl
kIs_
kisara
kIw
komTaTa
korma
kostar_a
kotet
kotetam
kumtIr
kur_eja
kurateN_a
kWebel_ele
kWen_ene
kWer_a
kWer_ete
kWerekW_ere
kWesas_a
lafe
lEba
lebel_ebe
lefel_efe
lElit
lem_ene
leyay_e
lIfsIfs
lIfya
lIgmeN_a
lIksIks
lolE
maq_eqe
ma'qeb
mat
mehay_Im
mehay_ImIn_et
mekera
melti
men_a
menem_ene
menman_a
meqIn
meqseft
meqseft
mer_eze
merar_
merara
merz
mes_ele
mesenakIl
metecete
metet
meTfo
mex_e
mexewed

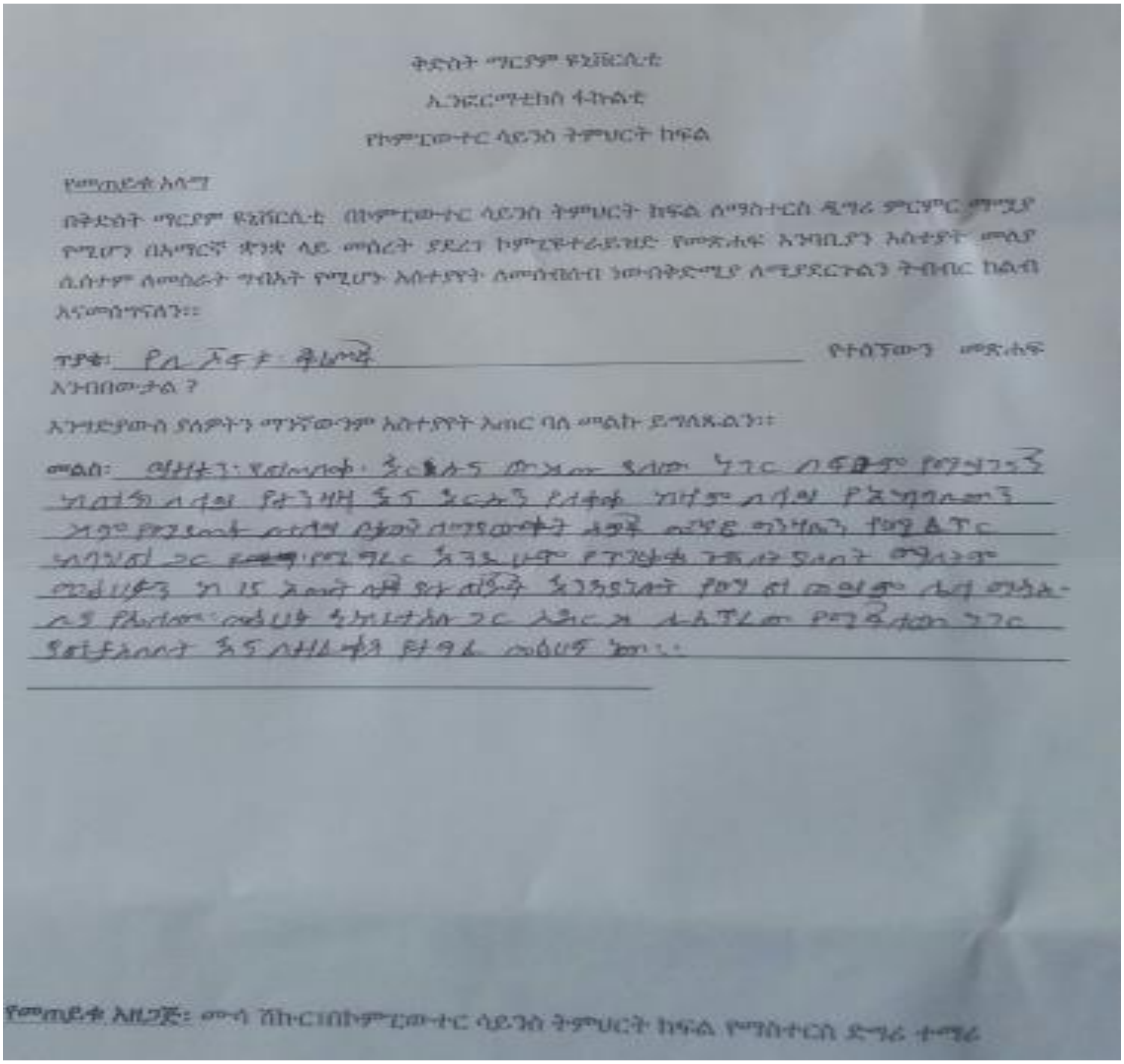
mezez
meZger
mIci
mIl_ax
mIq_eN_In_et
mIqeNa
mIS_et
mIskin
mIsqIlqIl
misTir
mIsTir
miTiTi
mIzbera
molfaT_a
molqaq_a
moN_e
moNnet
mote
muCC
museNa
musna
muT_IN_
naqe
neCnaC_a
nefnaf_a
nefTeN_a
negereN_a
nehulala
nek_ese
neqefa
nes_a
neT_efe
new_ere
newT
newTeNa
neznaz_a
nICnIC
nId_Et
nIfg
nIfug
nIqet
nItIr_Ik
nIznIz
ona
qaTelo
qebaT_ere
qebeT
qebIr
qed_a
qedada
qef_efe
qefo
qelaTE
qelqal_a
qem_aN_a
qen_ateN_a
qen_ese
qenber
qer_e
qerfaf_a
qes_efe
qeT_a
qeT_efe
qewlal_a
qews
qeZqaZ_a
qezqaz_a
qIb
qil
qIl_Et
qilaqil
qim
qimeN_a
qInata
qInate
qInTat
qInTot
qIr_Eta
qISbet
qISbetawi
qITat
qITfet
qIZet
qIZetam
qonqIwa
qoraTeTe
qoTqWaTa
qoxaxa
qulqulet
qurTet
qusIl
quT_a
qWaT_ere

qWer_eTe
qWeseqW_ese
raqe
rasmItat
reb_exe
rebxa
res_a
rEsa
rIkas_a
rIkax
rIkus
rIr
rITb
sate
seb_ere
SebeNa
seg_a
sek_aram
sel_ebe
seleba
senef
seneTTeqe
sEre
sEreN_a
SeS_ete
sEseN_a
SeSet
Sey_af
seyTan
sId_
sIdb
sIgat
sIhtet
sIkene'akatE
sImEtawi
sImEtawinet
sIn_IkWIl
SInfeNa
SInfeNa
si'ol
sIqay
sIr_Iz
sIrqot
sIs
sus
Taremot
Tase
taw_eke
taz_ebe
Teb
Teb_ab
Teb_ebe
Tebasa
tebed_ere
tebeg_ere
tebelax_e
TebeN_a
tebesaC_e
tebet_ebe
tebkenek_ene
tebtab_a
teCaCane
teCeb_eTe
tedaf_ene
tedbeseb_ese
tedenag_ere
Tef_a
teferekak_ese
tegaC_e
tegaC_e
tegal_eSe
tegan_ene
teged_ebe
tegemed_ele
tegoda
tek_eze
tekade
tekes_ese
Tel_a
Telat
Telefa
telekefe
Temam_a
temenam_ene
temeSad_eqe
temesas_ele
temeseqaq_ele
temeSew_ete
temWaT_eTe
Temzaz_a
tenad_ede
tenade
tenCebarere

tenezaz_a
tenkol
Tenq
tenqIwaq
tentebat_ebe
tenxerat_ete
teqar_ene
teqarani
teqaT_ele
teqaw_eme
teqWeTa
Ter_a
tera
teraqW_ete
tereg_eme
teret
teSarere
tesasate
teseb_ere
teSey_efe
teS'Ino
teS'Ino
teTaTame
teTeq_a
teTeraT_ere
tewenab_ede
teweq_ese
tewes_ene
tewesas_ebe
Tewlag_a
texeb_ere
texen_efe
TEza
tezab_a
tezebab_ete
tezeberar_eqe
tI'bit
tI'biteNa
tIc_It
TIdfIya
TIdfiya
TIfateN_a
TIg_eN_a
TIg_eN_In_et
TIl
TIlac_a
TIlaxet
TIleN_a
TImb
tImkIht
tImkIhteN_a
tIn_Ix
TInb
tInkosa
TIntawi
TIqaq_In
TIqat
TIqerxa
tIrIm_Is
TIrIT_arE
tIrtIr
tIzbIt
TorIn_et
u'uta
wal_ele
walg
waTe
wed_eqe
wefzera
weg_a
wekeba
welaw_ele
weleblaba
welefEnd
welemta
welgad_a
wenfit
wenjel
wenjeleN_a
weq_esa
weq_ese
wer_ad_a
wer_ede
wer_ere
werEN_a
wereNa
weret
wereteN_a
werobela
weT_a
weTeTE
wId_aqi
wIdmet

wIguz
wIgzet
wIrdet
wIrjIb_IN_
wIsbIsb
wIx_a
wIxet
wIxetam
wIZmIbIr
xag_ete
xakara
xefafa
xefT
xegege
xekIm
xelebta
xer_eN_a
xerex_ere
xermuTa
xewede
xex_e
xex_ege
xIb_Ir
xIba
xIbet
xIbrIteNa
xIfta
xInfet
xIngela
xIrdeda
yalteger_a
yeqIl
yIluNta
yIyulIN
zage
zale
zaza
zebet
zefeqede
zeg_a
zegemteN_a
zegen_ene
zegey_e
zel_eqe
zelefa
zelzal_a
zeneg_a
zer_efe
zereN_a
zereNnet
zewet_ere
zIbrIqrIq
zIgmIteNa
zIm_ut
zImteNa
zIngu
zIq_aC
zIq_IteN_a
zIqTet
zIrkIrk

Appendix C: Questionnaire with Sample Responses



ቅድስት ማርያም ዩኒቨርሲቲ

ኢንፎርማቲክስ ፋኩልቲ

የኮምፒውተር ሳይንስ ትምህርት ክፍል

የመጠይቁ አላማ

በቅድስት ማርያም ዩኒቨርሲቲ በኮምፒውተር ሳይንስ ትምህርት ክፍል ለማስተርስ ዲግሪ ምርምር ማሟያ የሚሆን በአማርኛ ቋንቋ ላይ መሰረት ያደረገ ኮምፒዩተራይዝድ የመጽሐፍ አንባቢያን አስተያየት መለያ ሲስተም ለመስራት ግብአት የሚሆኑ አስተያየት ለመሰብሰብ ነውበቅድሚያ ለሚያደርጉልን ትብብር ከልብ እናመሰግናለን።

ጥያቄ: የተሰኘውን መጽሐፍ አንብበውታል ?

እንዳነበቡትስ ያለዎትን ማንኛውንም አስተያየት እጠር ባለ መልኩ ይግለጹልን።

መልስ:

የመጠይቁ አዘጋጅ፡ ሙሉ ሽኩር(በኮምፒውተር ሳይንስ ትምህርት ክፍል የማስተርስ ድግሪ ተማሪ

Appendix D: List of Sample Amharic sentences with negative sentiment

1. meShaffu yelbIweled aS_aSaf meseretawiyanIn ayamW_al_am yewer_ede new
2. tarikIn lemaTelxet teblo yeteSafe bado yeqalat kWakWata
3. beTam yedek_eme ye'arefteneger as_eraru Inkuwan dIrsetun aydel_em amarIN_awn gel_otal
4. ye'artI'ot In_a ye'areftеneger as_ekak gIdfet yebez_ab_et zIrkIrk sIra
5. te'amaninet yel_ewIm yemender quCbelu new yeSafew
6. meShafu kers as_eTaT jem_Iro an_adaj In_a as_afari new
7. meShafu beCIb_ITIm hone be'aSaSaf werdob_IN_al
8. kinawi wIbet yelEl_ew tera yesbIket zeye yalew yeqalat gagata bado qIl yehone geleba meShaf new
9. bebekulE yIhE meShaf altemec_eN_Im mizanawinet yIgodlewal
10. has_aboc_u yeteqed_u mehonac_ew sayans kesaynIs alabawyan meseretawi Iwqet yetefat_u mehonac_ew lIj yabokaw mehonun fIntIw adrIgo yasay_al
11. yetekeb_eru gud_ayoc_In yesne SIhuf gIrmamoges nesto maq_ albIso yaqer_ebe telkaxa meShaf new
12. yIhEn meShaf saneb gIra kemeg_abatE yetenes_a merEt yet Indehonec yITefab_IN_al
13. bez_ihuw meShaf lit_elal_ef yetefel_egew ma'kelawi mel'Ikt mIn Indehone InkWan anbabiw Irasu derasiwm yem_iyawqew aymesleN_Im
14. yeqalat bIzEt lay tedegagami sIhtet yeteser_ab_et yegdElex sIra new
15. keqalat gagata besteqer gITmInetacewn yemiyagola Im_Iq yehas_ab bIslet gomrIto altay_eb_ac_ewIm
16. meShafu wIsT berkat_a yefidel yeqalat In_a yehas_ab alemeTaTam sIhtetoc_ yetemol_a zIrkIrk new
17. balemuya yalhonec sEt IndeqolacIw aynet bun_a hono new yageNhut
18. Sehafiw basay_ew mizanun yesate ye'aSaSaf sIlt sIneShufun aTewlIgotal
19. yaltemec_eN_ neger yeqWanqWaw guramaylEnet lIk_ bengIlizIN_a fidelat amarIN_a Indem_In_ISfew aynet abazE bamarIN_a fidelat ye'gIlizNa qalatIn besfat teT

Appendix E: List of Sample Amharic sentences with positive sentiment

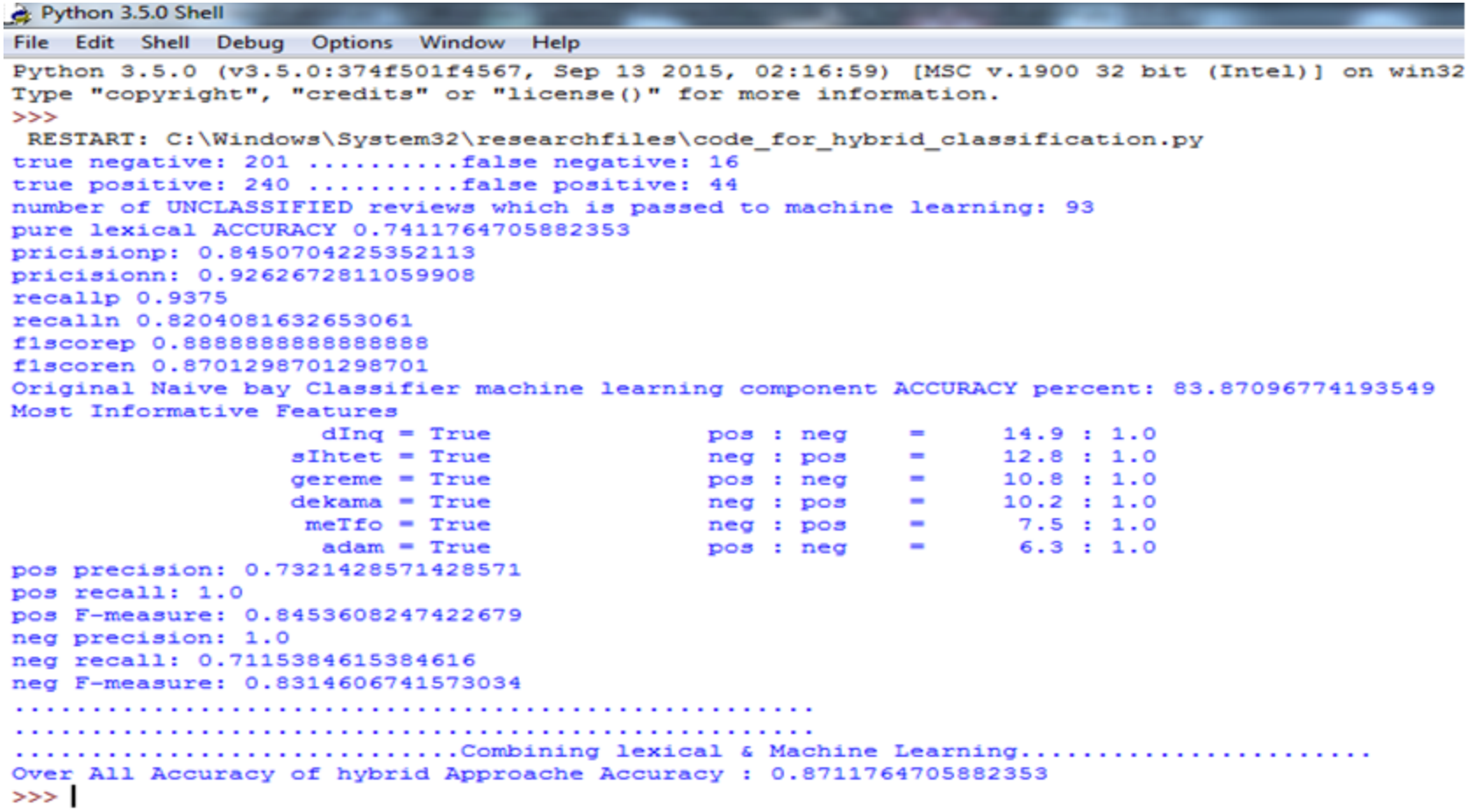
1. derasi dira'az meShafun yaqerebubet yeTbebawi dereja kefta maletIm yeTera yeqWanqWa aTeqaqemacew mel'Iktun lemastelalef yeteketelut yem'Iraf aderejajet yemaydeneqaqef yehasab fIset lIk lIbweled yemnaneb yahIl andE kejemerIn sanCers yemanasqemTew Indihon bIrtu aqIm alabsotal
2. adefrIs yequwanquwa aTeqaqemu geleSaw Indihum yIzetu yeteleye bota yemiyaseTew yemnIgzEm mIrT meShaf new
3. kelbu meShaf be'amarNa lIbweled meSahfIt tarik beTam anegagariw Ina akerakariw Indihum Iske ahun dIres yehzIb fIqIr kaltenefegacew TIqit meSahfIt andum new
4. wIb yehonu berkata yebeleSegu geleSawocIn yetadele gIrum meShafnew
5. yemeShafu CIbT yemigerIm ana astemari bemehonu leljocE asqemcEwalehu
6. beTam agWagi belom lIb anTelTay meShaf new
7. qonjo meShaf new
8. beTam mesaC mIrT meShaf new anbIbut
9. meShafu wede hWala wesdo lIjnetIn sIlemiyastaws des bIloN wedjEw new yanebebkut
10. malefic yehone sIra
11. waw mecEm yemayselec zemen texagari meShaf mecEm ityoPyawi hono fIqIr Iseke meqabrIn yalanebebe aynorIm
12. meShafu geSebahriyatun benebarawiw yehyIwet awed yemnaqacew yahIl Indimeslen tedergo beTru hunEta newe yetesalew
13. besneShuf yetlIm mewaqIr ketarik mIsreta tenesto geSebahriyatIn bemastewaweq gICt kesto wede lIqetu bemadres keziyam bemargeb Ina wede Ifoyta yemiwesd kef yale lIb seqela yemiytaybet asdemami aynet sIra new
14. Inde wIha TeTcE yerekahubet mIrT meShaf
15. yerasun ye'aSaSaf sIlt yIzo yemeTaw adam mereq betebalew mIrT sIne SIhuf afnICah sIr balewu gIn tIkuret nefgehwu abrehwu benorkewu qInTat Iwuneta lay tenterso gudayon baltelemede ateyay yezhon yakIl agzIfo yemislIlh mIrT derasi mehonun asmeskIrWal
16. tarikawi mesrejawocIn be'agbab yeteTeqeme aznaNna astemari meSehaf new

17. alweledIm kegizEw yeqedeme sIra yemigerIm geSebahri gena ke'natu mahSen sayweTa IwnetIn yIzo mahberesebun yemimogt yemiweded aSaSafu lIk Inde weraj wIha kulIl bIlo yemifes feta yemiyaderg ayn kefac SIhuf
18. adefrIs beyzet be'aSaSaf sIlt Ijg yeteraraqe gerami meTaTf
19. yesnIbt qelemat sIfran bemsIl nedfo bemsIlocu dengIgo behatit dereja Iyeteraqeqe CImIr sIletemeleketnew lehageracIn sIneShuf IngIda ana malefiya bIlenewal
20. yefqIr Iske meqabIr tarik sinebeb anbabiwn befqIr wejeb alagto yefqIr IsreNa yemiyaderg te'amreNa meShaf yeTnat Ina mIrmIr SIhufoc siserubet yenore meShaf new

Appendix F: stop-words list

	አንዳንድ
ሀሙስ	ኢትዮጵያ
ሁሉ	እሁድ
ሁሉም	እሱ
ሕዝብ	እና
ለመሆኑ	እናንተ
ለምንድን	እኔ
ሌላ	እንኳን
ሌሎች	እንደ
መጽሀፍ	እንዴት
ማን	እኛ
ማን	እዚያ
ማክሰኞ	እግር
ምን	ከመሆን
ሰኞ	ወይንም
ሰው	ወደ
ሲሆን	ዋና
ስንት	ዘንድ
ረቡእ	የሚከተለው
ቅዳሜ	ያ
በዚህ	ያኔ
ብላ	ይህ
ብቻ	ይኼው
ብቻ	ገጽ
ነበር	ጋር
ነው	ግን
ነገር	
ናቸው	
አለ	
አርብ	
አንተ	

Appendix G: Result of Hybrid Approach



Appendix H: Sample Classified Reviews

```
.
.....The First 10 Classified Reviews as a Sample.....
.
1.. [(['aba', 'TEna', 'iyas', 'yIh', 'gereme', 'neger', 'ale', 'nEgativ', 'meShaf', 'ale', 'meShaf'], 'pos')]
2.. [(['beTam', 'misdenq', 'hIywet', 'misTIr', 'aTeqalele', 'arif', 'gITIm'], 'pos')]
3.. [(['InE', 'Imnet', 'yIh', 'fit', 'ityoPya', 'tarik', 'yIh', 'dInq', 'meShaf', 'teSafe', 'aweqe', 'nEgativ'], 'pos')]
4.. [(['haymanot', 'menet', 'migeSbet', 'menged', 'tInx', 'gWerebeTe'], 'neg')]
5.. [(['beTam', 'wededе', 'qonjo', 'Safe', 'telemede', 'nEgativ', 'Cerese'], 'pos')]
6.. [(['hasab', 'teqeda', 'hone', 'anese', 'nEgativ', 'saymIs', 'alabawya', 'meseretawi', 'Iwqet', 'tefata', 'hone', 'lIj', 'aboka', 'hone', 'fIntIw', 'aderege', 'asay
e'], 'pos')]
7.. [(['asreda', 'hasab', 'wIha', 'qWeTere', 'nEgativ'], 'pos')]
8.. [(['yIh', 'meShaf', 'zerfe', 'bIzu', 'wIsbIsb', 'hone', 'sew', 'lIj', 'sIlTanE', 'hidet', 'gIrum', 'hone', 'fesete', 'geleSe', 'tereke'], 'pos')]
9.. [(['teTeqeme', 'TIqs', 'tesasate', 'reba', 'nEgativ', 'asnewari', 'SIhuf'], 'pos')]
10.. [(['tarik', 'maTelxet', 'tebale', 'teSafe', 'bado', 'qalat', 'kIwakwata'], 'neg')]
```

Appendix I: Sample Code for Machine Learning

```
import nltk
import re
import csv
import collections
from collections import OrderedDict
import random
import re
import math
from nltk.classify.scikitlearn import SklearnClassifier
import pickle
from sklearn.naive_bayes import MultinomialNB,BernoulliNB
from sklearn.linear_model import LogisticRegression,SGDClassifier
from sklearn.svm import SVC, LinearSVC, NuSVC

from nltk.classify import ClassifierI
from statistics import mode
from sklearn.metrics import precision_score

class VoteClassifier(ClassifierI):
    def __init__(self, *classifiers):
        self._classifiers = classifiers

    def classify(self, features):
        votes = []
        for c in self._classifiers:
            v = c.classify(features)
            votes.append(v)
        return mode(votes)

    def confidence(self, features):
        votes = []
        for c in self._classifiers:
            v = c.classify(features)
            votes.append(v)

        choice_votes = votes.count(mode(votes))
        conf = choice_votes / len(votes)
        return conf

documentsp=[]
documentsn=[]
documents=[]
doc=[([], 'neg'),([], 'pos')]
d={}

##////////////////////////////////////////////////open file "allpos.py"
that contain positive reviews and tokenize each sentence and attach
label(pos)

with open('allpos.py','r',encoding='utf-8') as f:
```

```
    for line in f:
        x=(list(line.split()),"pos")
##////////////////////////////////////////////change the above tokenized
and labeled revew into dictionaty containing tokenized review as key and
label(pos) as value

        documentsp.append(x)
documentsn=[]

##////////////////////////////////////////////////open file "allneg.py"
that contain negative reviews and tokenize each sentence and attach
label(neg)

with open('allneg.py','r',encoding='utf-8') as f:


    for line in f:
        x=(list(line.split()),"neg")

##////////////////////////////////////////////change the above tokenized
and labeled revew into dictionaty containing tokenized review as key and
label(neg) as value
        documentsn.append(x)

documentsn.remove(([], 'neg'))
documentsp.remove(([], 'pos'))

##////////////////////////////////////////////forming a dictionary that
contaon reviews as a key and label(pos or neg) as value
documents=documentsn+documentsp

random.shuffle(documents)

all_words=[]
myfile=open('all_words.py','r',encoding='utf-8')
myfile2=str(myfile.read())
line3 = re.sub('\n', ' ', myfile2)
with open('all_words.py', 'w',encoding='utf-8') as f:
    f.write(line3)


with open('all_words.py','r',encoding='utf-8') as f:


    for line in f:
        x=line.split()
p=open('all_words.py', 'r',encoding='utf-8')
p=p.read()

p=set(p.split())

with open('vvall.py','w',encoding='utf-8') as x:
```

```
    for item in p:
        x.write("%s\n" % item)

num_words = 0

counta= dict()
counta2= dict()
countb= dict()
countb2= dict()
countc= dict()
countc2= dict()
countd= dict()
countd2= dict()

p3=open('vvp.py', 'r',encoding='utf-8')
myfile=p3.read()
line = re.sub('\n',' ',myfile)

with open('vvp2.py','w',encoding='utf-8') as g:
    g.write(line)
f3=open('vvp2.py', 'r',encoding='utf-8')
xp2=f3.read().split()

f3=open('vvall.py','r',encoding='utf-8')
myfile=f3.read()
line = re.sub('\n',' ',myfile)
with open('vvall2.py','w',encoding='utf-8') as g:
    g.write(line)
pall=open('vvall2.py', 'r',encoding='utf-8')
xt2=pall.read().split()

##////////////////////////////////////////////////
ncn=250
##print(ncn)
N=600
##print(N)
import math

z=(ncn/N)*(math.log(ncn/N)) + (ncp/N)*(math.log(ncp/N))

##////////////////////////////////////words and their frequency count to be
used in calculating thier importance based on different feature selection
methods

##y=open('vvall.py', 'r',encoding='utf-8')
##xt=str(y.read().strip())
##xt2=xt.split()
##print(len(xt2))
for word in xn2:
    if word in xn2:
        if word in countb:
            countb[word]+=1
            countd[word]-=1
```

```
        else:
            countb[word]=2
            countd[word]=248

    else:
        countb[word]=1
        countd[word]=249


for wp in xp2:

    if wp in xp2:
         if wp in counta:
            counta[wp]+=1
            countc[wp]-=1

         else:
            counta[wp]=2
            countc[wp]=348
    else:
        counta[wp]=1
        countc[wp]=349

for word in xt2:
    counta2[word]=1
##counta3=dict()
for word in xt2:
    countb2[word]=1
for word in xt2:
    countc2[word]=1
for word in xt2:
    countd2[word]=1
counta3=dict()
countb3=dict()
countc3=dict()
countd3=dict()
##print(counta2)
for k, v in counta2.items():
    counta3[k] = v + counta.get(k, 0)
for k, v in countb2.items():
    countb3[k] = v + countb.get(k, 0)
for k, v in countc2.items():
    countc3[k] = v + countc.get(k, 0)
for k, v in countd2.items():
    countd3[k] = v + countd.get(k, 0)

AD={k : v * counta3[k] for k, v in countd3.items() if k in counta3}
##print(AD)
CB={k : v * countc3[k] for k, v in countb3.items() if k in countc3}
##print(CB)


AC={}
for k, v in counta3.items():
```

```
    AC[k] = v + countc3.get(k, 0)
##print(AC)


BD={}
for k, v in countb3.items():
    BD[k] = v + countd3.get(k, 0)
##print(BD)

AB={}
for k, v in counta3.items():
    AB[k] = v + countb3.get(k, 0)
##print(AB)

CD={}
countavc=0
countavi=0
countavm=0
countavsc=0
for k, v in countc3.items():
    CD[k] = v + countd3.get(k, 0)

##print(CD)

psub={}
for k, v in AD.items():
    psub[k] = abs(v - CB.get(k, 0))

psubsq={k : v * psub [k] for k, v in  psub.items() if k in  psub}

for key in psubsq:
    psubsq[key] *=600

ACBD={k : v * AC[k] for k, v in BD.items() if k in AC}
ABCD={k : v * AB[k] for k, v in CD.items() if k in AB}
##print(ABCD)
ACAB={k : v * AC[k] for k, v in AB.items() if k in AC}
AN={k : 600 * counta3[k] for k, v in counta3.items()}

#////////////////////////////////////////////////////////////////////////c
alculate mutual information gain for each words in review (MI)

m={k : v / ACAB[k] for k, v in AN.items() if k in ACAB}

mi=dict()
for k, v in m.items():
    mi[k] = math.log(v)

ACBDABCD={k : v * ACBD[k] for k, v in ABCD.items() if k in ACBD}
##print(ACBDABCD)

#////////////////////////////////////////////////////////////////////////
calculating chi-square value for each words in review
```

```
posout1 = dict((k, float(psubsq[k]) / ACBDABCD[k]) for k in psubsq)

#////////////////////////////////////////////////////////////////////
calculating GSS value for each words in review

posout2=dict((k, float(psub[k]) / (600*600)) for k in psub)

#////////////////////////////////////////////////////////////////////a
dictionary representing features selected by CHI-square

posout=sorted(sorted(posout1), key=posout1.get, reverse=True)

#////////////////////////////////////////////////////////////////////a
dictionary representing features selected by GSS(simplified chi

posout2=sorted(sorted(posout2), key=posout2.get, reverse=True)

#////////////////////////////////////////////////////////////////////a
dictionary representing features selected by Mutual information gain(MI)

posout3=sorted(sorted(mi), key=mi.get, reverse=True)


x=dict()
c=dict()
g=dict()
infor=dict()
avxx=dict()
avxxt=dict()
avxx2=dict()
av2xxt2=dict()
u=dict()
z=dict()
count=1500
for i, y in enumerate(posout3):
    g[y]=1900-i

for i, y in enumerate(posout):
    c[y]=1900-i

for k, v in c.items():
    avxxt[k] = v + g.get(k, 0)

#////////////////////////////////////////////////////////////////////a
dictionary representing features selected by combination of(MI & chi-
square)

posoutav=sorted(sorted(avxxt), key=avxxt.get, reverse=True)

av1=dict()
av2=dict()
av3=dict()
avt1=dict()
avt2=dict()
```

```
avt3=dict()
posoutsm=dict((k, float(psub[k]) / (600*600)) for k in psub)


with open('all_words.py','r',encoding='utf-8') as f:

    for line in f:
        x=line.split()

all_words = nltk.FreqDist(x)

word_features2 = list(all_words.keys())[:1899]

word_features = posout[:1500]
word_features4 = posoutav[:1500]
word_features5 = posout2[:1500]
word_features6 = posout3[:1500]


def find_features(document):
    words = set(document)
    features = {}
    for w in word_features:
        features[w] = (w in words)
    return features
def find_features2(document):
    words = set(document)
    features2 = {}
    for w in word_features2:
        features2[w] = (w in words)
    return features2
def find_features3(document):
    words = set(document)
    features3 = {}
    for w in word_features3:
        features3[w] = (w in words)
    return features3
def find_features4(document):
    words = set(document)
    features4 = {}
    for w in word_features4:
        features4[w] = (w in words)
    return features4

def find_features5(document):
    words = set(document)
    features5 = {}
    for w in word_features5:
        features5[w] = (w in words)
    return features5
def find_features6(document):
    words = set(document)
    features6 = {}
    for w in word_features6:
```

```
        features6[w] = (w in words)
    return features6


myfile=open('all_words.py','r',encoding='utf-8')
myfile2=str(myfile.read())
line3 = re.sub('\n', ' ', myfile2)
with open('all_words.py', 'w',encoding='utf-8') as f:
    f.write(line3)


with open('all_words.py','r',encoding='utf-8') as w:
    for line in w:
        y=line.split()

##//////feature set for chi-square
featuresets = [(find_features(rev), category) for (rev, category) in
documents]

##//////feature set for bag of words
featuresets2 = [(find_features2(rev), category) for (rev, category) in
documents]

##//////feature set for combinition of chi-square and MI

featuresets4 = [(find_features4(rev), category) for (rev, category) in
documents]

##//////feature set for GSS(simple-chi
featuresets5 = [(find_features5(rev), category) for (rev, category) in
documents]

##//////feature set for MI
featuresets6 = [(find_features6(rev), category) for (rev, category) in
documents]



##////////////////////////Start of Naive bayes/////////////////


num_folds=10

subset_size=60
accurc=0
accurbag=0
accurav1=0
accursim=0
accurmi=0

precisionnaivechi=0
precisionnaivebag=0
precisionnaiveav=0
precisionnaivesim=0
```

```
precisionnaivemi=0

precisionnaivechin=0
precisionnaivebagn=0
precisionnaiveavn=0
precisionnaivesimn=0
precisionnaivemin=0

recallchi=0
recallbag=0

recallav=0
recallsim=0
recallmi=0

recallchin=0
recallbagn=0

recallavn=0
recallsimn=0
recallmin=0

f_measurechi=0
f_measurebag=0

f_measureav=0
f_measuresim=0
f_measuremi=0

f_measurechin=0
f_measurebagn=0

f_measureavn=0
f_measuresimn=0
f_measuremin=0

////////////////////////////all with k-forl validation

##/////////////////naive bayes with Chi-square

for a in range(num_folds):
    refsets = collections.defaultdict(set)
    testsets = collections.defaultdict(set)

    testing_set = featuresets[a*subset_size:][:subset_size]
    training_set= featuresets[:a*subset_size] +
featuresets[(a+1)*subset_size:]
    classifier = nltk.NaiveBayesClassifier.train(training_set)
    accurc+=(nltk.classify.accuracy(classifier, testing_set)*100)
    for i, (feats, label) in enumerate(testing_set):
        refsets[label].add(i)
        observed = classifier.classify(feats)
        testsets[observed].add(i)
    precisionnaivechi+=nltk.precision(refsets['pos'], testsets['pos'])
```

```
    recallchi+=nltk.recall(refsets['pos'], testsets['pos'])
    f_measurechi+=nltk.f_measure(refsets['pos'], testsets['pos'])
    precisionnaivechin+=nltk.precision(refsets['neg'], testsets['neg'])
    recallchin+=nltk.recall(refsets['neg'], testsets['neg'])
    f_measurechin+=nltk.f_measure(refsets['neg'], testsets['neg'])

##/////////////////naive bayes with bag of words

for a in range(num_folds):
    refsets = collections.defaultdict(set)
    testsets = collections.defaultdict(set)
    testing_set2 = featuresets2[a*subset_size:][:subset_size]
    training_set2= featuresets2[:a*subset_size] +
featuresets2[(a+1)*subset_size:]
    classifier2 = nltk.NaiveBayesClassifier.train(training_set2)
    accurbag+=(nltk.classify.accuracy(classifier2, testing_set2)*100)
    for i, (feats, label) in enumerate(testing_set2):
        refsets[label].add(i)
        observed = classifier2.classify(feats)
        testsets[observed].add(i)
    precisionnaivebag+=nltk.precision(refsets['pos'], testsets['pos'])
    recallbag+=nltk.recall(refsets['pos'], testsets['pos'])
    f_measurebag+=nltk.f_measure(refsets['pos'], testsets['pos'])
    precisionnaivebagn+=nltk.precision(refsets['neg'], testsets['neg'])
    recallbagn+=nltk.recall(refsets['neg'], testsets['neg'])
    f_measurebagn+=nltk.f_measure(refsets['neg'], testsets['neg'])

##/////////////////naive bayes with combination of Chi-square and MI

for a in range(num_folds):
    refsets = collections.defaultdict(set)
    testsets = collections.defaultdict(set)
    testing_set4 = featuresets4[a*subset_size:][:subset_size]
    training_set4= featuresets4[:a*subset_size] +
featuresets4[(a+1)*subset_size:]
    classifier4 = nltk.NaiveBayesClassifier.train(training_set4)
    accurav1+=(nltk.classify.accuracy(classifier4, testing_set4)*100)
    for i, (feats, label) in enumerate(testing_set4):
        refsets[label].add(i)
        observed = classifier4.classify(feats)
        testsets[observed].add(i)
    precisionnaiveav+=nltk.precision(refsets['pos'], testsets['pos'])
    recallav+=nltk.recall(refsets['pos'], testsets['pos'])
    f_measureav+=nltk.f_measure(refsets['pos'], testsets['pos'])
    precisionnaiveavn+=nltk.precision(refsets['neg'], testsets['neg'])
    recallavn+=nltk.recall(refsets['neg'], testsets['neg'])
    f_measureavn+=nltk.f_measure(refsets['neg'], testsets['neg'])

 ##/////////////////naive bayes with simple Chi-square (GSS)

for a in range(num_folds):
    refsets = collections.defaultdict(set)
    testsets = collections.defaultdict(set)
    testing_set5 = featuresets5[a*subset_size:][:subset_size]
```

```
    training_set5= featuresets5[:a*subset_size] +
featuresets5[(a+1)*subset_size:]
    classifier5 = nltk.NaiveBayesClassifier.train(training_set5)
    accursim+=(nltk.classify.accuracy(classifier5, testing_set5)*100)
    for i, (feats, label) in enumerate(testing_set5):
        refsets[label].add(i)
        observed = classifier5.classify(feats)
        testsets[observed].add(i)
    precisionnaivesim+=nltk.precision(refsets['pos'], testsets['pos'])
    recallsim+=nltk.recall(refsets['pos'], testsets['pos'])
    f_measuresim+=nltk.f_measure(refsets['pos'], testsets['pos'])
    precisionnaivesimn+=nltk.precision(refsets['neg'], testsets['neg'])
    recallsimn+=nltk.recall(refsets['neg'], testsets['neg'])
    f_measuresimn+=nltk.f_measure(refsets['neg'], testsets['neg'])


##/////////////////naive bayes with MI

for a in range(num_folds):
    refsets = collections.defaultdict(set)
    testsets = collections.defaultdict(set)

    testing_set6 = featuresets6[a*subset_size:][:subset_size]
    training_set6= featuresets6[:a*subset_size] +
featuresets6[(a+1)*subset_size:]
    classifier6 = nltk.NaiveBayesClassifier.train(training_set6)
    accurmi+=(nltk.classify.accuracy(classifier6, testing_set6)*100)
    for i, (feats, label) in enumerate(testing_set6):
        refsets[label].add(i)
        observed = classifier6.classify(feats)
        testsets[observed].add(i)
    precisionnaivemi+=nltk.precision(refsets['pos'], testsets['pos'])
    recallmi+=nltk.recall(refsets['pos'], testsets['pos'])
    f_measuremi+=nltk.f_measure(refsets['pos'], testsets['pos'])
    precisionnaivemin+=nltk.precision(refsets['neg'], testsets['neg'])
    recallmin+=nltk.recall(refsets['neg'], testsets['neg'])
    f_measuremin+=nltk.f_measure(refsets['neg'], testsets['neg'])


print("Original Naive bay Classifier accuracy percent Chi-
Square:",accurc/10)

print("Original Naive bay Classifier accuracy percent bag of
words:",accurbag/10)
##print("Original Naive bay Classifier accuracy percent Info
Gain:",accuri/10)
print("Original Naive bay Classifier accuracy percent AV:",accurav1/10)
print("Original Naive bay Classifier accuracy percent simplified
chi:",accursim/10)
print("Original Naive bay Classifier accuracy percent MI:",accurmi/10)



##.............
```

```
print("......................naive-bayes with
CHI.....................................")

print ('pos precision:',precisionnaivechi/10)
print ('pos recall:',recallchi/10)

print ('pos F-measure:', f_measurechi/10)
print ('neg precision:', precisionnaivechin/10)
print ('neg recall:', recallchin/10)
print ('neg F-measure:', f_measurechin/10)
print("......................naive-bayes with bag of
words.....................................")

print ('pos precision:',precisionnaivebag/10)
print ('pos recall:',recallbag/10)
print ('pos F-measure:', f_measurebag/10)
print ('neg precision:', precisionnaivebagn/10)
print ('neg recall:', recallbagn/10)
print ('neg F-measure:', f_measurebagn/10)
##print(".............................naive-bayes with Information
Gain.................................")
##
##print ('pos precision:',precisionnaiveinfo/10)
##print ('pos recall:', recallinfo/10)
##print ('pos F-measure:', f_measureinfo/10)
##print ('neg precision:', precisionnaiveinfon/10)
##print ('neg recall:', recallinfon/10)
##print ('neg F-measure:', f_measureinfon/10)
print(".............................naive-bayes with (Combination if
CHI&MI.................................")

print ('pos precision:',precisionnaiveav/10)
print ('pos recall:', recallav/10)
print ('pos F-measure:', f_measureav/10)
print ('neg precision:', precisionnaiveavn/10)
print ('neg recall:', recallavn/10)
print ('neg F-measure:', f_measureavn/10)
print("........................naive-bayes with SIMPLE-
CHI.....................................")

print ('pos precision:',precisionnaivesim/10)
print ('pos recall:', recallsim/10)
print ('pos F-measure:', f_measuresim/10)
print ('neg precision:', precisionnaivesim/10)
print ('neg recall:', recallsimn/10)
print ('neg F-measure:', f_measuresimn/10)
print("..............................naive-bayes with
MI................................")

print ('pos precision:',precisionnaivemi/10)
print ('pos recall:', recallmi/10)
print ('pos F-measure:', f_measuremi/10)
print ('neg precision:', precisionnaivemi/10)
print ('neg recall:', recallmin/10)
```

```
print ('neg F-measure:', f_measuremin/10)
print(".....................................................................")

####////////////////////////////////// end of NAIVE
NAYES//////////////////


##////begining of logistic regeretion

accurc=0
accurbag=0
accuri=0
accurav2=0
accursim=0
accurmi=0
precisionnaivechi=0
precisionnaivebag=0
precisionnaiveinfo=0
precisionnaiveav2=0
precisionnaivesim=0
precisionnaivemi=0

precisionnaivechin=0
precisionnaivebagn=0
precisionnaiveinfon=0
precisionnaiveavn2=0
precisionnaivesimn=0
precisionnaivemin=0

recallchi=0
recallbag=0
recallinfo=0
recallav2=0
recallsim=0
recallmi=0

recallchin=0
recallbagn=0
recallinfon=0
recallavn2=0
recallsimn=0
recallmin=0

f_measurechi=0
f_measurebag=0
f_measureinfo=0
f_measureav2=0
f_measuresim=0
f_measuremi=0

f_measurechin=0
f_measurebagn=0
f_measureinfon=0
f_measureavn2=0
```

```
f_measuresimn=0
f_measuremin=0

LogisticRegression_classifier = SklearnClassifier(LogisticRegression())
LogisticRegression_classifier2 = SklearnClassifier(LogisticRegression())
LogisticRegression_classifier3 = SklearnClassifier(LogisticRegression())
LogisticRegression_classifier4 = SklearnClassifier(LogisticRegression())
LogisticRegression_classifier5 = SklearnClassifier(LogisticRegression())
LogisticRegression_classifier6 = SklearnClassifier(LogisticRegression())

##/////////////////logistic  with MI

for a in range(num_folds):
    refsets = collections.defaultdict(set)
    testsets = collections.defaultdict(set)
    testing_set6 = featuresets6[a*subset_size:][:subset_size]
    training_set6= featuresets6[:a*subset_size] +
featuresets6[(a+1)*subset_size:]
    LogisticRegression_classifier6.train(training_set6)
    accurmi+=(nltk.classify.accuracy(LogisticRegression_classifier6,
testing_set6)*100)
    for i, (feats, label) in enumerate(testing_set6):
        refsets[label].add(i)
        observed = LogisticRegression_classifier6.classify(feats)
        testsets[observed].add(i)
    precisionnaivemi+=nltk.precision(refsets['pos'], testsets['pos'])
    recallmi+=nltk.recall(refsets['pos'], testsets['pos'])
    f_measuremi+=nltk.f_measure(refsets['pos'], testsets['pos'])
    precisionnaivemin+=nltk.precision(refsets['neg'], testsets['neg'])
    recallmin+=nltk.recall(refsets['neg'], testsets['neg'])
    f_measuremin+=nltk.f_measure(refsets['neg'], testsets['neg'])

##/////////////////logistic  with simple Chi-square (GSS)

for a in range(num_folds):
    refsets = collections.defaultdict(set)
    testsets = collections.defaultdict(set)
    testing_set5 = featuresets6[a*subset_size:][:subset_size]
    training_set5= featuresets5[:a*subset_size] +
featuresets5[(a+1)*subset_size:]
    LogisticRegression_classifier5.train(training_set5)
    accursim+=(nltk.classify.accuracy(LogisticRegression_classifier5,
testing_set5)*100)
    for i, (feats, label) in enumerate(testing_set5):
        refsets[label].add(i)
        observed = LogisticRegression_classifier5.classify(feats)
        testsets[observed].add(i)
    precisionnaivesim+=nltk.precision(refsets['pos'], testsets['pos'])
    recallsim+=nltk.recall(refsets['pos'], testsets['pos'])
    f_measuresim+=nltk.f_measure(refsets['pos'], testsets['pos'])
    precisionnaivesimn+=nltk.precision(refsets['neg'], testsets['neg'])
    recallsimn+=nltk.recall(refsets['neg'], testsets['neg'])
    f_measuresimn+=nltk.f_measure(refsets['neg'], testsets['neg'])
```

```
  ##/////////////////logistic  with combination of Chi-square and MI

for a in range(num_folds):
    refsets = collections.defaultdict(set)
    testsets = collections.defaultdict(set)

    testing_set4 = featuresets4[a*subset_size:][:subset_size]
    training_set4= featuresets4[:a*subset_size] +
featuresets4[(a+1)*subset_size:]
    LogisticRegression_classifier4.train(training_set4)
    accurav2+=(nltk.classify.accuracy(LogisticRegression_classifier4,
testing_set4)*100)
    for i, (feats, label) in enumerate(testing_set4):
        refsets[label].add(i)
        observed = LogisticRegression_classifier4.classify(feats)
        testsets[observed].add(i)
    precisionnaiveav2+=nltk.precision(refsets['pos'], testsets['pos'])
    recallav2+=nltk.recall(refsets['pos'], testsets['pos'])
    f_measureav2+=nltk.f_measure(refsets['pos'], testsets['pos'])
    precisionnaiveavn2+=nltk.precision(refsets['neg'], testsets['neg'])
    recallavn2+=nltk.recall(refsets['neg'], testsets['neg'])
    f_measureavn2+=nltk.f_measure(refsets['neg'], testsets['neg'])


    ##/////////////////logistic  with bag of words


for a in range(num_folds):
    refsets = collections.defaultdict(set)
    testsets = collections.defaultdict(set)

    testing_set2 = featuresets2[a*subset_size:][:subset_size]
    training_set2= featuresets2[:a*subset_size] +
featuresets2[(a+1)*subset_size:]
    LogisticRegression_classifier2.train(training_set2)
    accurbag+=(nltk.classify.accuracy(LogisticRegression_classifier2,
testing_set2)*100)
    for i, (feats, label) in enumerate(testing_set2):
        refsets[label].add(i)
        observed = LogisticRegression_classifier2.classify(feats)
        testsets[observed].add(i)
    precisionnaivebag+=nltk.precision(refsets['pos'], testsets['pos'])
    recallbag+=nltk.recall(refsets['pos'], testsets['pos'])
    f_measurebag+=nltk.f_measure(refsets['pos'], testsets['pos'])
    precisionnaivebagn+=nltk.precision(refsets['neg'], testsets['neg'])
    recallbagn+=nltk.recall(refsets['neg'], testsets['neg'])
    f_measurebagn+=nltk.f_measure(refsets['neg'], testsets['neg'])



    ##/////////////////logistic  with Chi-square
for a in range(num_folds):
    refsets = collections.defaultdict(set)
    testsets = collections.defaultdict(set)
```

```
    testing_set = featuresets[a*subset_size:][:subset_size]
    training_set= featuresets[:a*subset_size] +
featuresets[(a+1)*subset_size:]
    LogisticRegression_classifier.train(training_set)
    accurc+=(nltk.classify.accuracy(LogisticRegression_classifier,
testing_set)*100)
    for i, (feats, label) in enumerate(testing_set):
        refsets[label].add(i)
        observed = LogisticRegression_classifier.classify(feats)
        testsets[observed].add(i)
    precisionnaivechi+=nltk.precision(refsets['pos'], testsets['pos'])
    recallchi+=nltk.recall(refsets['pos'], testsets['pos'])
    f_measurechi+=nltk.f_measure(refsets['pos'], testsets['pos'])
    precisionnaivechin+=nltk.precision(refsets['neg'], testsets['neg'])
    recallchin+=nltk.recall(refsets['neg'], testsets['neg'])
    f_measurechin+=nltk.f_measure(refsets['neg'], testsets['neg'])

print("LogisticRegression_classifier accuracy percent Chi-Square:",
accurc/10)
print("LogisticRegression_classifier accuracy percent bag of words:",
accurbag/10)
##print("LogisticRegression_classifier accuracy percent Info Gain:",
accuri/10)
print("LogisticRegression_classifier accuracy percent AV:", accurav2/10)
print("LogisticRegression_classifier accuracy percent simplified CHI:",
accursim/10)
print("LogisticRegression_classifier accuracy percent MI:", accurmi/10)
print("..............................................................
......")
##


print(".....................LG with
CHI.........................................")

print ('pos precision:',precisionnaivechi/10)
print ('pos recall:',recallchi/10)

print ('pos F-measure:', f_measurechi/10)
print ('neg precision:', precisionnaivechin/10)
print ('neg recall:', recallchin/10)
print ('neg F-measure:', f_measurechin/10)
print(".....................LG with bag of
words.....................................")

print ('pos precision:',precisionnaivebag/10)
print ('pos recall:',recallbag/10)
print ('pos F-measure:', f_measurebag/10)
print ('neg precision:', precisionnaivebagn/10)
print ('neg recall:', recallbagn/10)
print ('neg F-measure:', f_measurebagn/10)
```

```
print(".........................LG with (Combination if
CHI&MI.....................................")

print ('pos precision:',precisionnaiveav2/10)
print ('pos recall:', recallav2/10)
print ('pos F-measure:', f_measureav2/10)
print ('neg precision:', precisionnaiveavn2/10)
print ('neg recall:', recallavn2/10)
print ('neg F-measure:', f_measureavn2/10)
print(".............................LG with SIMPLE-
CHI.............................")

print ('pos precision:',precisionnaivesim/10)
print ('pos recall:', recallsim/10)
print ('pos F-measure:', f_measuresim/10)
print ('neg precision:', precisionnaivesim/10)
print ('neg recall:', recallsimn/10)
print ('neg F-measure:', f_measuresimn/10)
print(".........................LG with
MI.................................")

print ('pos precision:',precisionnaivemi/10)
print ('pos recall:', recallmi/10)
print ('pos F-measure:', f_measuremi/10)
print ('neg precision:', precisionnaivemi/10)
print ('neg recall:', recallmin/10)
print ('neg F-measure:', f_measuremin/10)
print(".................................................................")


##////////////////////////////logistic-end



##////////////////////////////////start of SVM

accurc=0
accurbag=0
accuri=0
accurav3=0
accursim=0
accurmi=0

precisionnaivechi=0
precisionnaivebag=0
precisionnaiveinfo=0
precisionnaiveav3=0
precisionnaivesim=0
precisionnaivemi=0

precisionnaivechin=0
precisionnaivebagn=0
precisionnaiveinfon=0
precisionnaiveavn3=0
```

```
precisionnaivesimn=0
precisionnaivemin=0

recallchi=0
recallbag=0
recallinfo=0
recallav3=0
recallsim=0
recallmi=0

recallchin=0
recallbagn=0
recallinfon=0
recallavn3=0
recallsimn=0
recallmin=0

f_measurechi=0
f_measurebag=0
f_measureinfo=0
f_measureav3=0
f_measuresim=0
f_measuremi=0

f_measurechin=0
f_measurebagn=0
f_measureinfon=0
f_measureavn3=0
f_measuresimn=0
f_measuremin=0

SGDClassifier_classifier6 = SklearnClassifier(SGDClassifier())
for i in range(num_folds):
    refsets = collections.defaultdict(set)
    testsets = collections.defaultdict(set)

    testing_set6 = featuresets6[i*subset_size:][:subset_size]
    training_set6= featuresets6[:i*subset_size] +
featuresets6[(i+1)*subset_size:]
    SGDClassifier_classifier6.train(training_set6)
    accurmi+=(nltk.classify.accuracy(SGDClassifier_classifier6,
testing_set6)*100)
    for i, (feats, label) in enumerate(testing_set6):
        refsets[label].add(i)
        observed = SGDClassifier_classifier6.classify(feats)
        testsets[observed].add(i)
    precisionnaivemi+=nltk.precision(refsets['pos'], testsets['pos'])
    recallmi+=nltk.recall(refsets['pos'], testsets['pos'])
    f_measuremi+=nltk.f_measure(refsets['pos'], testsets['pos'])
    precisionnaivemin+=nltk.precision(refsets['neg'], testsets['neg'])
    recallmin+=nltk.recall(refsets['neg'], testsets['neg'])
    f_measuremin+=nltk.f_measure(refsets['neg'], testsets['neg'])
print("..............................................................
......")
```

```
SGDClassifier_classifier5 = SklearnClassifier(SGDClassifier())
for i in range(num_folds):
    refsets = collections.defaultdict(set)
    testsets = collections.defaultdict(set)

    testing_set5 = featuresets5[i*subset_size:][:subset_size]
    training_set5= featuresets5[:i*subset_size] +
featuresets5[(i+1)*subset_size:]
    SGDClassifier_classifier5.train(training_set5)
    accursim+=(nltk.classify.accuracy(SGDClassifier_classifier5,
testing_set5)*100)
    for i, (feats, label) in enumerate(testing_set5):
        refsets[label].add(i)
        observed = SGDClassifier_classifier5.classify(feats)
        testsets[observed].add(i)
    precisionnaivesim+=nltk.precision(refsets['pos'], testsets['pos'])
    recallsim+=nltk.recall(refsets['pos'], testsets['pos'])
    f_measuresim+=nltk.f_measure(refsets['pos'], testsets['pos'])
    precisionnaivesimn+=nltk.precision(refsets['neg'], testsets['neg'])
    recallsimn+=nltk.recall(refsets['neg'], testsets['neg'])
    f_measuresimn+=nltk.f_measure(refsets['neg'], testsets['neg'])
print("..............................................................
......")
SGDClassifier_classifier4 = SklearnClassifier(SGDClassifier())
for i in range(num_folds):
    refsets = collections.defaultdict(set)
    testsets = collections.defaultdict(set)

    testing_set4 = featuresets4[i*subset_size:][:subset_size]
    training_set4= featuresets4[:i*subset_size] +
featuresets4[(i+1)*subset_size:]
    SGDClassifier_classifier4.train(training_set4)
    accurav3+=(nltk.classify.accuracy(SGDClassifier_classifier4,
testing_set4)*100)
    for i, (feats, label) in enumerate(testing_set4):
        refsets[label].add(i)
        observed = SGDClassifier_classifier4.classify(feats)
        testsets[observed].add(i)
    precisionnaiveav3+=nltk.precision(refsets['pos'], testsets['pos'])
    recallav3+=nltk.recall(refsets['pos'], testsets['pos'])
    f_measureav3+=nltk.f_measure(refsets['pos'], testsets['pos'])
    precisionnaiveavn3+=nltk.precision(refsets['neg'], testsets['neg'])
    recallavn3+=nltk.recall(refsets['neg'], testsets['neg'])
    f_measureavn3+=nltk.f_measure(refsets['neg'], testsets['neg'])

SGDClassifier_classifier3= SklearnClassifier(SGDClassifier())
for i in range(num_folds):
    refsets = collections.defaultdict(set)
    testsets = collections.defaultdict(set)

    testing_set3 = featuresets3[i*subset_size:][:subset_size]
    training_set3= featuresets3[:i*subset_size] +
featuresets3[(i+1)*subset_size:]
    SGDClassifier_classifier3.train(training_set3)
```

```
    accuri+=(nltk.classify.accuracy(SGDClassifier_classifier3,
testing_set3)*100)
    for i, (feats, label) in enumerate(testing_set3):
        refsets[label].add(i)
        observed = SGDClassifier_classifier3.classify(feats)
        testsets[observed].add(i)
    precisionnaiveinfo+=nltk.precision(refsets['pos'], testsets['pos'])
    recallinfo+=nltk.recall(refsets['pos'], testsets['pos'])
    f_measureinfo+=nltk.f_measure(refsets['pos'], testsets['pos'])
    precisionnaiveinfon+=nltk.precision(refsets['neg'], testsets['neg'])
    recallinfon+=nltk.recall(refsets['neg'], testsets['neg'])
    f_measureinfon+=nltk.f_measure(refsets['neg'], testsets['neg'])

SGDClassifier_classifier2 = SklearnClassifier(SGDClassifier())
for i in range(num_folds):
    refsets = collections.defaultdict(set)
    testsets = collections.defaultdict(set)

    testing_set2 = featuresets2[i*subset_size:][:subset_size]
    training_set2= featuresets2[:i*subset_size] +
featuresets2[(i+1)*subset_size:]
    SGDClassifier_classifier2.train(training_set2)
    accurbag+=(nltk.classify.accuracy(SGDClassifier_classifier2,
testing_set2)*100)
    for i, (feats, label) in enumerate(testing_set2):
        refsets[label].add(i)
        observed = SGDClassifier_classifier2.classify(feats)
        testsets[observed].add(i)
    precisionnaivebag+=nltk.precision(refsets['pos'], testsets['pos'])
    recallbag+=nltk.recall(refsets['pos'], testsets['pos'])
    f_measurebag+=nltk.f_measure(refsets['pos'], testsets['pos'])
    precisionnaivebagn+=nltk.precision(refsets['neg'], testsets['neg'])
    recallbagn+=nltk.recall(refsets['neg'], testsets['neg'])
    f_measurebagn+=nltk.f_measure(refsets['neg'], testsets['neg'])

SGDClassifier_classifier = SklearnClassifier(SGDClassifier())
for i in range(num_folds):
    refsets = collections.defaultdict(set)
    testsets = collections.defaultdict(set)

    testing_set = featuresets[i*subset_size:][:subset_size]
    training_set= featuresets[:i*subset_size] +
featuresets[(i+1)*subset_size:]
    SGDClassifier_classifier.train(training_set)
    accurc+=(nltk.classify.accuracy(SGDClassifier_classifier,
testing_set)*100)
    for i, (feats, label) in enumerate(testing_set):
        refsets[label].add(i)
        observed = SGDClassifier_classifier.classify(feats)
        testsets[observed].add(i)
    precisionnaivechi+=nltk.precision(refsets['pos'], testsets['pos'])
    recallchi+=nltk.recall(refsets['pos'], testsets['pos'])
    f_measurechi+=nltk.f_measure(refsets['pos'], testsets['pos'])
    precisionnaivechin+=nltk.precision(refsets['neg'], testsets['neg'])
```

```
    recallchin+=nltk.recall(refsets['neg'], testsets['neg'])
    f_measurechin+=nltk.f_measure(refsets['neg'], testsets['neg'])
##
##

print("SGDClassifier_classifier accuracy percent Chi-Square:", accurc/10)
print(".....................................................................
......")
print("SGDClassifier_classifier accuracy percent bag of words:",
accurbag/10)
print(".....................................................................
......")

##print(".....................................................................
........")
print("SGDClassifier_classifier accuracy percent AV:", accurav3/10)
print(".....................................................................
......")
print("SGDClassifier_classifier accuracy percent simplified CHI:",
accursim/10)
print(".....................................................................
......")
print("SGDClassifier_classifier accuracy percent MI:", accurmi/10)
print(".....................................................................
......")

####////////////////////
####/////////////////////


print(".....................SVM with
CHI.........................................")

print ('pos precision:',precisionnaivechi/10)
print ('pos recall:',recallchi/10)

print ('pos F-measure:', f_measurechi/10)
print ('neg precision:', precisionnaivechin/10)
print ('neg recall:', recallchin/10)
print ('neg F-measure:', f_measurechin/10)
print(".....................SVM with bag of
words.........................................")

print ('pos precision:',precisionnaivebag/10)
print ('pos recall:',recallbag/10)
print ('pos F-measure:', f_measurebag/10)
print ('neg precision:', precisionnaivebagn/10)
print ('neg recall:', recallbagn/10)
print ('neg F-measure:', f_measurebagn/10)

print(".........................SVM with (Combination if
CHI&MI.....................................")

print ('pos precision:',precisionnaiveav3/10)
```

```
print ('pos recall:', recallav3/10)
print ('pos F-measure:', f_measureav3/10)
print ('neg precision:', precisionnaiveavn3/10)
print ('neg recall:', recallavn3/10)
print ('neg F-measure:', f_measureavn3/10)
print(".............................SVM with SIMPLE-
CHI.............................")

print ('pos precision:',precisionnaivesim/10)
print ('pos recall:', recallsim/10)
print ('pos F-measure:', f_measuresim/10)
print ('neg precision:', precisionnaivesim/10)
print ('neg recall:', recallsimn/10)
print ('neg F-measure:', f_measuresimn/10)
print(".........................SVM with
MI.................................")

print ('pos precision:',precisionnaivemi/10)
print ('pos recall:', recallmi/10)
print ('pos F-measure:', f_measuremi/10)
print ('neg precision:', precisionnaivemi/10)
print ('neg recall:', recallmin/10)
print ('neg F-measure:', f_measuremin/10)
print(".............................................................")


##//////////////////////////////////end-of-sgd
```