N-grams for Text Classification Using Supervised Machine Learning Algorithms

Kennedy Odhiambo Ogada

A thesis submitted in fulfillment for the degree of Doctor of Philosophy In Information Technology in the Jomo Kenyatta University Of Agriculture and Technology

2016

DECLARATION

This thesis is my original work and has not been presented for a degree in any other University

Signature: ____________________ Date: ____________________

Kennedy Odhiambo Ogada

This thesis has been submitted for examination with our approval as the University Supervisors

Signature: ____________________ Date: ____________________

Prof. Waweru Mwangi

JKUAT, Kenya

Signature: [illegible] Date: ____________________

Dr Wilson Cheruiyot

JKUAT, Kenya

DEDICATION

To my parents Hulda and late dad Walter, Brothers Harris, late Bill and Philip, Charles, Jerry, Eliakim and sister Justine, Wife Bina and Sons Welsey, Heffy and Jeremy.

ACKNOWLEDGMENT

This PhD thesis has been the fruit of three intense years of research work and experiments. The thesis would have not become what it is without many people who were involved and that I am grateful to.

First of all, I have to thank my supervisors: Professor Waweru Mwangi and Dr Wilson Cheruiyot for counting on me and working closely with me from start of the project to the end. You took time to read, edit and correct journal papers and conference papers. Without them as my co-authors, the articles would not have been possible. I cannot forget Dr Orero who helped me during the initial stages of this project in narrowing my research focus.

Regarding colleagues, first of all I would like to thank Dr George Okeyo, the Chairman Department of Computing. Your comments and advise helped me shape research directions. Dr Rimiru your insight and expertise into this project was great. Thanks to Dr Ann Kibe, Mwalili Tobias, Dr Kimwele, Dr Kimani, Dr Mindila, and Professor Wafula among others for the feedback, comments and guidelines during and after project presentations and all staff of the school of Computing and Information Technology.

Doing PhD was my best decision ever, I have met the most awesome people I could ever meet, fun, passionate and hard working people with innovative ideas and ideals from which I have enriched myself not only professionally but they have made me grow greatly at a personal level.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Sentiment Analysis or Text Classification aims at determining the overall sentiment orientation of a given input text. Most data mining methods assume that the data to be mined is represented in a structured relational database. However, in many applications, available electronic information is in the form of unstructured natural language documents. The bag-of-words (BoW) model has been widely used to represent documents in text classification and many other applications. BoW ignores the relationships between terms, offers a rather poor document representation. N-gram Model is a statistical technique to automatic document classification which involves the determination of certain probability relationships between individual content-bearing words and the subject categories and the use of these relationships to predict the category to which a document containing the words belongs. To capture this discriminative power of words as phrases there is need for a model which can include such N-grams in the vector space model without any additional changes in classifiers based on the vector space models. Furthermore, determining the correct value of n, i.e. the size of the sliding window that is to be used, when using word based n-gram analysis, is an area of experimentation on each particular domain of knowledge. This PhD research analyzed performance of five text classifiers with N-grams of varying sentiment length. The research methodology employed was experiments. The major contribution of this research is a proposed hybrid framework for sentiment classification that includes bag of words, N-grams, Skip-grams, and contextual knowledge into learning and prediction phases of sentiment classification .

Keywords: Text Classification, Natural Language Modeling, Bag of Words, N-grams, Supervised Machine Learning.

Chapter ONE

INTRODUCTION

Language is a powerful tool used in communicating and conveying information about products, people, events and objects. Languages can express information in either written, spoken or expressed graphically. Human beings use text to represent information. Sentiment analysis is the field of study that analyzes people's opinions, sentiments, evaluations, attitudes and emotions towards entities such as products, services, organizations, individuals, issues, events, movies and topics (Liu, 2012).

Nowadays sentiment analysis is an active field of research, used to extract people's opinion about particular product or service. The most useful application of sentiment analysis is sentiment classification (or Text Classification or Text Categorization). The task of sentiment classification is to classify sentiments of users as positive or negative from textual information alone. A modern approach towards sentiment classification is the use of machine learning techniques, which inductively builds a classification model of a given set of categories by training several sets of labeled documents. Popular machine learning methods include Naive Bayes, K-Nearest Neighbor, Support Vector Machines and Neural Network (Dhande & Patnaik, 2014).

1.1 Text Classification

Many researchers have come with different ways of defining or describing text classification. Text Classification is the act of taking a set of labeled text documents, learning a correlation between a document's contents and its corresponding labels and then predicting the labels of a set of unlabeled test documents as best as possible. Text Classification is sometimes referred to as Sentiment Analysis or Opinion Mining or Text Mining. Sentiment analysis aims to identify the orientation (positive or negative) of opinions or emotions expressed in documents (Dulac-Arnold, Denoyer, & Gallinari, 2011).

Sentiment analysis can be utilized to identify the favaourable or unfavourable public opinions expressed in blogs about tribe, race or gender which might cause violence, animosity between people from different tribe, race or gender. The task of sentiment analysis focuses on the detection and extraction of opinions, feelings and emotions in text with respect to a certain subject or object. A subtask of sentiment analysis is the sentiment categorization on the basis of certain polarities. That is, being able to distinguish between positive, neutral or negative expressions or statements of extracted textual or spoken elements. Since positive as well as negative expressions can occur within the same document, this task is challenging. Classical statistical Text Classification approaches are based on well-known machine learning models such as generative models (e.g. Naïve Bayes) or discriminant models such as Support Vector Machines (Dulac-Arnold *et al.*, 2011).

Document classification or document categorization can be stated in simple terms as a task to assign a document to one or more predefined categories. This can be either done manually by intellectuals or automatically by software. The manual classification has mostly been the province of library science, while the automatic classification is mainly performed using machine learning and information retrieval techniques (Yasotha & Charles, 2015).

Text mining is a branch of Data mining. Data Mining refers to extracting informative knowledge from a large amount of data, which could be expressed in different data types, such as transaction data in Electronic Commerce applications or genetic expressions in bioinformatics research domain. The main purpose of data mining is discovering hidden data or unseen knowledge, normally in the form of patterns, from available data repository (Xu, Zhang, & Li, 2011).

Web Mining or Web Data Mining is the means of utilizing data mining methods to induce and extract useful information from Web data information. Web mining may be classified into three categories based on the mining goals, which determine the part of the Web to be mined: Web content mining, web structure mining, and Web usage mining. Web content mining tries to discover valuable information from Web contents. Generally, Web content mining is mainly referred to textual objects, thus, it is also alternatively termed as text mining (Xu *et al*, 2011).

Information retrieval (IR) involves matching the text contained in a query or a document to a set of other documents. Often, the task involves finding the documents from a corpus of documents that are relevant to a user's query. The idea behind information retrieval is that if a user enters a query such as what is the capital of Sri Lanka?, then a good approach to finding the answer is to find a document that contains all (or some) of the words contained in the query (Coppin, 2004).

The widespread and increasing availability of text documents in electronic form increases the importance of using automatic methods to analyze the content of text documents, because the method of using domain experts to identify new text documents and allocate them to well-defined categories is time-consuming and expensive, has limits, and does not improve continuous measure of the degree of confidence with which the allocation was made (Vinodhini & Chandrasekaran, 2012).

Sentiment analysis is a type of natural language processing for tracking the mood of the public about a particular product or topic. Sentiment analysis, which is also called opinion mining, involves in building a system to collect and examine opinions about the product made in blog posts, comments, reviews or tweets. Sentiment analysis can be useful in several ways. For example, in marketing it helps in judging the success of an advertisement campaign or new product launch, determine which versions of a product or service are popular and even identify which demographics like or dislike particular features (Vinodhini & Chandrasekaran, 2012).

1.1.1 History of Text Classification

Text Classification (TC) dates back to the early '60s, but only in the early '90s did it become a major subfield of the information systems discipline, thanks to increased applicative interest and to the availability of more powerful hardware. Until the late '80s, the most popular approach to TC, at least in the "operational" (i.e., real-world applications) community, was knowledge engineering (KE), consisting of manually defining a set of rules encoding expert knowledge on how to classify documents under the given categories (Sebastiani, 2002).

From the '90s, knowledge engineering approach has increasingly lost popularity (especially in the research community) in favor of the machine learning (ML) paradigm, according to which a general inductive process automatically builds an automatic text classifier by learning from a set of preclassified documents the characteristics of the categories of interest. The advantages of this approach are an accuracy comparable to that achieved by human experts, and a considerable savings in terms of expert labor power, since no intervention from either knowledge engineers or domain experts is needed for the construction of the classifier or for its porting to a different set of categories. TC dates back to Maron's seminal work in 1961 on probabilistic text classification (Sebastiani, 2002).

Maron (1961) did an inquiry which examined a technique for automatically classifying (indexing) documents according to their subject content. The task is to have a computing machine read a document and on the basis of the occurrence of selected clue words decide to which of many subject categories the document in question belongs.

The term “automatic indexing” denotes the problem of deciding in a mechanical way to which category (subject or field of knowledge) a given document belongs. It concerns the problem of deciding automatically what a given document is about. The situation is one in which there is a collection of different documents, each containing information on one or several subjects. Also there exists a set of categories, usually not exclusive nor completely independent, but exhaustive in the sense that every document will “fit” into at least one of the given categories. The problem arises because the categories are not defined extensionally (Maron, 1961).

To correctly classify an object or event is a mark of intelligence; a mark of even greater intelligence is to be able to modify and create new and more fruitful categories, i.e., to form new concepts . Statistical technique to automatic document indexing involves the determination of certain probability relationships between individual content-bearing words and the subject categories and the use of these relationships to predict the category to which a document containing the words belongs (Maron, 1961).

1.1.2 Basic Concept of Text Classification

Text categorization is the task of assigning a Boolean value to each pair $(d_j, c_i) \in D \times C$, where D is a domain of documents and $C = \{c_1, ..., c_{|C|}\}$ is a set of predefined categories. A value of T assigned to (d_j, c_i) indicates a decision to file d_j under c_i, while a value of F indicates a decision not to file d_j under c_i. More formally, the task is to approximate the unknown *target function* $\hat{\Phi} : D \times C \rightarrow \{T, F\}$ (that describes how documents ought to be classified) by means of a function $\Phi : D \times C \rightarrow \{T, F\}$ called the *classifier* (aka rule,or hypothesis,or model) such that $\hat{\Phi}$ and Φ “coincide as much as possible”. Classification under $C = \{c_1, ..., c_{|C|}\}$ is viewed as consisting of $|C|$ independent problems of classifying the documents in D under a given category c_i, for $i = 1, ..., |C|$. A classifier for c_i is then a function $\hat{\Phi}_i : D \rightarrow \{T, F\}$ that approximates an unknown target function $\Phi_i : D \rightarrow \{T, F\}$(Sebastiani, 2002).

1.2 Applications of Text Categorization

Since the work of Maron (1961), TC has been applied in a number of different applications. The borders between the different classes of applications are fuzzy and somehow artificial, and some of these may be considered special cases of others (Sebastiani, 2002). With the explosive growth of social media (e.g., reviews, forum discussions, blogs, micro-blogs, Twitter, comments, and postings in social network sites) on the Web, individuals and organizations are increasingly using the content in these media for decision making (Liu, 2007). The following section describes application areas of text categorization.

1.2.1 Automatic Indexing for Boolean Information Retrieval Systems

The application that has spawned most of the early research in the field is that of automatic document indexing for Information retrieval (IR) systems relying on a controlled dictionary, the most prominent example of which is Boolean systems. In Boolean systems, each document is assigned one or more key words or key phrases describing its content, where these key words and key phrases belong to a finite set called controlled dictionary, often consisting of a thematic hierarchical thesaurus (e.g., the NASA thesaurus for the aerospace discipline, or the MESH thesaurus for medicine). Usually, this assignment is done by trained human indexers, and is thus a costly activity (Sebastiani, 2002).

Automatic indexing with controlled dictionaries is closely related to automated metadata generation. In digital libraries, one is usually interested in tagging documents by metadata that describes them under a variety of aspects (e.g., creation date, document type or format, availability, etc.). Some of this metadata is thematic, that is, its role is to describe the semantics of the document by means of bibliographic codes, key words or key phrases. The generation of this metadata may thus be viewed as a problem of document indexing with controlled dictionary, and thus tackled by means of TC techniques (Sebastiani, 2002).

1.2.2 Document Organization

Indexing with a controlled vocabulary is an instance of the general problem of document base organization. In general, many other issues pertaining to document organization and filing, be it for purposes of personal organization or structuring of a corporate document base, may be addressed by TC techniques. For instance, at the offices of a newspaper incoming "classified" advertisements must be, prior to publication, categorized under categories such as Personals, Cars for Sale, Real Estate, etc. Newspapers dealing with a high volume of classified ads would benefit from an automatic system that chooses the most suitable category for a given advertisement. Other possible applications are the organization of patents into categories for making their search easier, the automatic filing of newspaper articles under the appropriate sections (e.g.,Politics,Home News, Lifestyles, etc.), or the automatic grouping of conference papers into sessions (Larkey, 1999).

1.2.3 Text Filtering

Text filtering is the activity of classifying a stream of incoming documents dispatched in an asynchronous way by an information producer to an information consumer (Belkin & Croft, 1992). A typical case is a newsfeed, where the producer is a news agency and the consumer is a newspaper. In this case, the filtering system should block the delivery of the documents the consumer is likely not interested in (e.g., all news not concerning sports, in the case of a sports newspaper). Filtering can be seen as a case of single-label TC, that is, the classification of incoming documents into two disjoint categories, the relevant and the irrelevant. Additionally, a filtering system may also further classify the documents deemed relevant to the consumer into thematic categories; in the example above, all articles about sports should be further classified according to which sport they deal with, so as to allow journalists specialized in individual sports to access only documents of prospective interest for them. Similarly,an e-mail filter might be trained to discard "junk" mail and further classify nonjunk mail into topical categories of interest to the user (Androutsopoulos, Koutsias, Chandrinos, & Spyropulos, 2000).

1.2.4 Word Sense Disambiguation

Word sense disambiguation (WSD) is an enabling technology for applications such as IR, Information extraction, Machine Translation (MT), question answering systems, cross language applications and documents classification. Word Sense Disambiguation is the process of selecting the correct sense for a word in a context. Knowledge-based, supervised and unsupervised approaches are used in WSD. In supervised approach, classifiers are used for assigning the correct sense of each word (Chandra & Dwivedi, 2014).

One of the main differences between natural languages and formal languages like C++ is that a sentence in a natural language can have more than one meaning. This is ambiguity, i.e., the fact that a sentence can be interpreted in different ways depending on who is speaking, the context in which it is spoken, and a number of other factors. The process by which a natural language processing system determines which meaning is intended by an ambiguous utterance is known as disambiguation. Disambiguation can be done in a number of ways. One of the most effective ways to overcome many forms of ambiguity is to use probability. This can be done using prior probabilities or conditional probabilities (Coppin, 2004).

Word sense disambiguation (WSD) is the activity of finding, given the occurrence in a text of an ambiguous (i.e., polysemous or homonymous) word, the sense of this particular word occurrence. For instance, the word “bank” may have (at least) two different senses in English, as in the Bank of England (a financial institution) or the bank of river Thames (a hydraulic engineering artifact). It is thus a WSD task to decide which of the above senses the occurrence of “bank” in “Last week I borrowed some money from the bank” has. WSD is very important for many applications, including natural language processing, and indexing documents by word senses rather than by words for IR purposes. WSD may be seen as a TC task once word occurrence contexts is viewed as documents and word senses as categories. Other examples, which may all be tackled by means of TC techniques along the lines discussed for WSD, are context-sensitive spelling correction, prepositional phrase attachment, part of speech tagging, and word choice selection in machine translation (Sebastiani, 2002).

1.2.5 Hierarchical Categorization of Web Pages

TC aroused a lot of interest for its possible application to automatically classifying Web pages, or sites, under the hierarchical catalogues hosted by popular Internet portals. When Web documents are catalogued in this way, rather than issuing a query to a general purpose Web search engine, a searcher may find it easier to first navigate in the hierarchy of categories and then restrict search to a particular category of interest. Classifying Web pages automatically has obvious advantages, since the manual categorization of a large enough subset of the Web is infeasible (Sebastiani, 2002).

1.3 Problem Statement

Documents representation is one of the pre-processing technique that is used to reduce the complexity of the documents and make them easier to handle. The documents have to be transformed from the full text version to a document vector. The most commonly used document representation is called Vector space Model, where documents are represented by vectors of words. Usually one has a collection of documents which is represented by word by word document matrix (Korde & Mahender, 2012). The bag-of-words (BoW) model has been widely used to represent a document in text classification and many other applications. However, BoW, which ignores the relationships between terms, offers a rather poor document representation. It is well-known that the relationships between words are very important for statistical language modeling (Wu & Wang, 2012). Also the patterns discovered by such models are not interpretable and comprehensive. To capture this discriminative power of words as phrases there is need for a model which can include such N-grams in the vector space model without any additional changes in classifiers based on the vector space models (Deepak & Narsimha, 2012). Furthermore, determining the correct value of n, i.e. the size of the sliding window that is to be used, when using word based n-gram analysis, is an area of experimentation on each particular domain of knowledge (Bouras & Tsogkas, 2016)

1.4 Research Objectives

The broad objective for this research was to analyze the effect of N-gram document modeling technique on performance of machine learning algorithms.

1.4.1 Specific Objectives

1. Analyze Machine Learning Algorithms and their use in text classification.
2. Use Machine learning algorithms to evaluate N-gram document models in comparison with bag of words document modeling technique.
3. Examine different ways for representing word dependence for text classification and evaluate their performance on machine learning algorithms.
4. Identify patterns of accuracy performance of machine learning algorithms with respect to the value of N in N-grams.
5. Compare the behaviour of N-grams when used with different sizes of training data sentiments.
6. Determine the effect of data set quality on the performance of various classifiers.
7. Compare the performance of fixed size N-grams and variable length N-grams on classifier performance.

1.5 Justification

The development of the Internet and mobile devices has increased the popularity of social media. People take advantage of social media as an electronic communication platform to share information, express their opinions, and construct their social networks, and furthermore to hear the voice of other users. Besides, information and opinion, shared and shaped through social media, influence individual views of society and incite on and offline political participations, e.g. South Korea candlelight protest of 2008 and United States Occupy Wall Street of 2011. Thus, now social media has become an open platform for political and social innovations (Suh, 2016).

However, there are problems and challenging issues for social media to grow into a better online place for political and social innovations, as well as for trusty information and opinions sharing. First, the anonymous nature of the Internet makes it difficult to ensure the qualities of users and their posts in social media. If there is no online user feedback on an anonymous user's posts, e.g. like and dislike, there is no way to find whether the anonymous user is good or bad before reading her/his post. This makes users in social media vulnerable to low quality posts and posters, offending and deceiving. Second, manipulations on user reputations arouse suspicion and mistrust on the online feedback system of social media in two ways: good quality posts and their users can be given low reputations maliciously by other users; some users can manipulate their own reputations to be high for certain reasons. However, identity changes, stemming from the anonymity of the Internet, lead to insufficient past reputation records on social media users, and make it hard to detect the manipulations on user reputations by the existing approaches (Suh, 2016).

Web users share information about people, products and events with great ease. Some of these sentiments might give positive and some negative opinions. After the 2007/2008 post election violence in Kenya, the Government has instituted bodies to monitor sentiments in media such as radio, Internet, Mobile phones and Television. Monitoring sentiments on radio and Television is much easier. However, opinions expressed in the social networking sites are many and are difficult to analyze to get the orientation of the blogs. Many business firms are now using online social network platforms for customer support, getting feedback about products and services, employee satisfaction survey, market research amongst other myriad services. Using Machine Learning Algorithms will reduce the burden of reading posts or blogs one by one, thus reducing the time spent analyzing these information (Suh, 2016).

Parts-of-speech pattern selection algorithm has been used in applications that effectively detect cyber bulling. Hate speech is a new category of research and needs an improvement on the existing algorithms for sentiment analysis. Also, this research product shall form a foundation for further research in sentiment analysis. Manual examination of thousands of messages can be an extremely tedious and labor intensive effort when applied across thousands of forum postings. With the increase in computer mediated communication, the need for automated text analysis techniques has grown in recent years (Suh, 2016).

It is known that human analysis of text information is subject to considerable biases, e.g., people often pay attention to opinions that are consistent with their own preferences. People also have difficulty, owing to their mental and physical limitations, producing consistent results when the amount of information to be processed is large. Automated opinion mining and summarization systems are thus, needed, as subjective biases and mental limitations can be overcome with an objective sentiment analysis system (Suh, 2016).

Statistical learning methods have become the absolute mainstream in text categorization. This is because statistical learning methods have less subjective factors compared with knowledge engineering methods. Besides, a lot of statistical learning technologies have a solid theoretical foundation, and there is a definite evaluation standard and good performance in statistical learning (Pei & Wu, 2014).

1.6 Thesis Organization

This thesis is divided into five chapters. Chapter 1 is an introductory chapter which started with a brief history of text classification then described the basic concept of text classification. A description of various applications of text classification followed. This chapter finally concluded by describing the problem statement, research objectives, justification and thesis organization. The next four chapters are organized as follows.

Chapter 2 starts with an introduction to sentiment classification process. A description of feature of sentiment classification and feature selection methods is discussed. This is then followed by an analysis of various text document modeling approaches that have been used in the field of Information Retrieval (IR) and text classification. N-gram document modeling, its application follows. An analysis of various Machine learning algorithms and their applications in text categorization is then discussed. This is followed by methods of evaluating classifiers.
Chapter 3 presents research methodology used. Experiments was used in this research. This chapter start by describing data collection process, research population, sentiment classification steps, and data preprocessing. Finally, it describes how probability for N-grams are estimated. description of N-gram probability estimation.
Chapter 4 is dedicated to Experiments and Results. This starts with training classifiers using bag-of-words and the document modeling technique. The number of feature words are then increased to two, three, four upto 12. The results of prediction are then analyzed and presented in form of tables. The performance of learned models with different N-grams sizes with different classifiers are then compared.
Chapter 5 starts with a description of Knowledge contributions, which is then followed by conclusion and finally describes areas for future work.

Chapter TWO

LITERATURE REVIEW

2.1 Introduction

Literature review is an important element in any research. This section starts by analyzing different text representation models. An evaluation of machine learning algorithms follows text representation.

2.2 Text Classification Steps

The objective of a learning system is to create a mapping between a set of documents and a set of labels. This mapping is then used to determine automatically the class of new (unlabeled) documents. Therefore, the general framework is usually called Supervised learning (also, learning from example, concept learning and includes the following steps (Markov & Larose, 2007).

- Step 1: *Data collection and preprocessing.*
  In this step, documents are collected, cleaned, and properly organized, the terms (features) are identified, and a vector space representation created. In this step, data may be divided into two subsets:
  - Training Set. This part of the data is used to create the model. In some cases, this is then split into two: the actual model construction subset and a model evaluation subset needed to tune the learner parameters.
  - Test Set.This part of the data is used for testing the model.

- Step 2: *Building the model.* This is the actual learning (also called training) step, which includes the use of the learning algorithm. It is usually an iterative and interactive process that may include other steps and may be repeated several times so that the best model is created:
  - Feature Selection.
  - Applying the learning algorithm.
  - Validating the model (using the validation subset to tune some parameters of the learning algorithm).
- Step 3: *Testing and evaluating the model.* In this step, the model is applied to the documents from the test set and their actual class labels are compared to the labels predicted. Note that at this step the document labels are used for evaluation only, which is not the case at the validation step, where the class labels are actually used by the learning algorithm.
- Step 4: In this step, the model is used to classify new documents (with unknown class labels).

2.2.1 Preprocessing

Pre-processing is the initial step where the source and suspicious documents are subjected to certain refinements like stop-word removal, tokenization, lower-casing, sentence segmentation, punctuation removal etc. This helps in reducing the size of actual data by removing the irrelevant information with respect to the approach used (Vani & Gupta, 2014).

Before the documents in a collection are used for classification, some preprocessing tasks are usually performed. For traditional text documents (no HTML tags), the tasks are stopword removal, stemming, and handling of digits, hyphens, punctuations, and cases of letters. For Web pages, additional tasks such as HTML tag removal and identification of main content blocks also require careful considerations (Liu, 2007).

The preprocessing phase converts the original textual data in a data mining ready structure, where the most significant text features that serve to differentiate between text categories are identified. It is the process of incorporating a new document into an information retrieval system. An effective preprocessor represents the document efficiently in terms of both space (for storing the document) and time (for processing retrieval requests) requirements and maintain good retrieval performance (precision and recall). This phase is the most critical and complex process that leads to the representation of each document by a select set of index terms. The main objective of preprocessing is to obtain the key features or key terms from online news text documents and to enhance the relevancy between word and document and the relevancy between word and category (Gaigole, Patil, & Chaudhari, 2013).

Tokenization

Given a character sequence and a defined document unit, tokenization is the task of chopping it up into pieces, called tokens, perhaps at the same time throwing away certain characters, such as punctuation marks. The tokenization process is language dependent (Swamy, Hanumanthappa, & Jyothi, 2014).

Stopword Removal

Stopwords are frequently occurring and insignificant words in a language that help construct sentences but do not represent any content of the documents. Articles, prepositions and conjunctions and some pronouns are natural candidates. Common stopwords in English include: a, about, an, are, as, at, be, by, for, from, how, in, is, of, on, or, that, the, these, this, to, was, what, when, where, who, will, with. Such words should be removed before documents are indexed and stored. Stopwords in the query are also removed before retrieval is performed (Liu, 2007).

Some extremely common words are not informative. These words are called stop words. The strategy used for determining a stop list is to sort the terms by collection frequency (the total number of times each term appears in the document collection), and then to take the most frequent terms (stop words). These words are discarded during indexing (Swamy *et al.*, 2014).

Stemming or Lemmatization

Once tokens are created, the next possible step is to convert each of the tokens to a standard form, a process usually referred to as stemming or lemmatization. The objective of stemming is to reduce the number of distinct types in a text corpus and to increase the frequency of occurrence of some individual types (Swamy *et al.*, 2014).

In many languages, a word has various syntactical forms depending on the contexts that it is used. For example, in English, nouns have plural forms, verbs have gerund forms (by adding “ing”), and verbs used in the past tense are different from the present tense. These are considered as syntactic variations of the same root form. Such variations cause low recall for a retrieval system because a relevant document may contain a variation of a query word but not the exact word itself. This problem can be partially dealt with by stemming. Stemming refers to the process of reducing words to their stems or roots (Liu, 2007).
A stem is the portion of a word that is left after removing its prefixes and suffixes. In English, most variants of a word are generated by the introduction of suffixes (rather than prefixes). Thus, stemming in English usually means suffix removal,or stripping. For example, “computer”, “computing”, and “compute” are reduced to “comput”. “walks”, “walking” and “walker” are reduced to “walk”.

Stemming enables different variations of the word to be considered in retrieval, which improves the recall. There are several stemming algorithms, also known as stemmers (Liu, 2007).

The purpose of performing stemming is to save time and space to make classification efficient and fast. Using this approach, a stemming algorithm is applied, which stems normal words from to their root forms. The most common stemmers are SStemmer, Lovins, Porter, and Paice/Husk Stemmers (Moral *et al.*, 2014, as cited in Rana, Rana, & Akbar, 2014).

- S-Stemmer: S-Stemmer has tendency to conflate both singular and plural forms. Stemmer is only applicable to word that has length greater than three. Purpose of this stemming algorithm is to consider both singular as well as plural forms of news headlines text (Moral *et al.*, 2014, as cited in Rana *et al.*, 2014).
- Lovins Stemmer: Lovins is the first stemmer, which was proposed in 1968 by Lovins. Lovins algorithm comprises of two major steps i.e. removal of suffixes and treating the remaining stem. There are approximately 29 rules any of which can be used for suffix removal. Algorithm works by comparing end of word with the longest suffix (out of total 294 suffixes). Then, double "t" or double "d" endings are performed on remaining stems, which is known as recording phase and it also comprises of 35 rules. At last, partial comparison is applied where all those words are grouped together having nearly same stems. For example, exclaim and exclamation are stemmed to words exclaim and exclam respectively (Ramasubramanian and Ramya, 2013, Moral *et al.*, 2014, as cited in Rana *et al.*, 2014).
- Porter Stemmer: The most common and widely used stemming algorithm is porter stemmer. This algorithm was proposed by Martin Porter in 1980 and this stemmer is now considered the most appropriate one. It is the mostly used stemmer in news headlines classification only because of its

better accuracy results and simple algorithm. It consists of five steps that are easy to implement and are applied to each word in corpus. It plays a vital role in text mining tasks (Ramos, 2000, Ramasubramanian and Ramya, 2013, Moral *et al.*, 2014, as cited in Rana *et al.*, 2014).

- Paice/Husk Stemmer: Paice in 1990 proposed Paice/Husk stemming algorithm, it is an iteration-based algorithm and uses same rules and suffixes for each loop. Each of the rule is sub-divided in 5 further parts and two of them are entirely optional. In Paice/Husk stemmer, rules are divided into five main steps. Three rules are mandatory to apply and remaining two rules are completely optional. For instance, one of the rules of Paice/Husk states that if word terminates with '¿', it means that word still needs stemming and must be passed through next cycle but if word termination contains '.', it means that final stem of the word has been obtained (Ramasubramanian & Ramya, 2013, Moral *et al.*, 2014, as cited in Rana *et al.*, 2014).

2.2.2 Preparing Data for Classification and Prediction

The following preprocessing steps may be applied to the data to help improve the accuracy, efficiency, and scalability of the classification or prediction process (Han & Kamber, 2006):

- Data cleaning: This refers to the preprocessing of data in order to remove or reduce noise (by applying smoothing techniques, for example) and the treatment of missing values (e.g., by replacing a missing value with the most commonly occurring value for that attribute, or with the most probable value based on statistics). Although most classification algorithms have some mechanisms for handling noisy or missing data, this step can help reduce confusion during learning (Han & Kamber, 2006).

- Relevance analysis: Many of the attributes in the data may be redundant.

Correlation analysis can be used to identify whether any two given attributes are statistically related. For example, a strong correlation between attributes A_1 and A_2 would suggest that one of the two could be removed from further analysis. A database may also contain irrelevant attributes. Attribute subset selection can be used in these cases to find a reduced set of attributes such that the resulting probability distribution of the data classes is as close as possible to the original distribution obtained using all attributes. Hence, relevance analysis, in the form of correlation analysis and attribute subset selection, can be used to detect attributes that do not contribute to the classification or prediction task. Including such attributes may otherwise slow down, and possibly mislead, the learning step (Han & Kamber, 2006).

Ideally, the time spent on relevance analysis, when added to the time spent on learning from the resulting "reduced" attribute (or feature) subset, should be less than the time that would have been spent on learning from the original set of attributes. Hence, such analysis can help improve classification efficiency and scalability (Han & Kamber, 2006).

- Data transformation and reduction: The data may be transformed by normalization, particularly when neural networks or methods involving distance measurements are used in the learning step. Normalization involves scaling all values for a given attribute so that they fall within a small specified range, such as 1:0 to 1:0, or 0:0 to 1:0. In methods that use distance measurements, for example, this would prevent attributes with initially large ranges (like, say, income) from outweighing attributes with initially smaller ranges (such as binary attributes). The data can also be transformed by generalizing it to higher-level concepts. Concept hierarchies may be used for this purpose. This is particularly useful for continuous valued attributes (Han & Kamber, 2006).

2.2.3 Text Document Features

Text databases include features (attributes) that contain word descriptions for objects. These features are not simple nominal but hold long sentences or paragraphs of text (Cios *et al*, 2007).

The preprocessing operations that support text mining attempt to leverage many different elements contained in a natural language document in order to transform it from an irregular and implicitly structured representation into an explicitly structured representation. However, given the potentially large number of words, phrases, sentences, typographical elements, and layout artifacts that even a short document may have – not to mention the potentially vast number of different senses that each of these elements may have in various contexts and combinations – an essential task for most text mining systems is the identification of a simplified subset of document features that can be used to represent a particular document as a whole (Feldman & Sanger, 2007).

A set of features are referred to as the presentational model of a document and that individual documents are represented by the set of features that their representational models contain (Feldman & Sanger, 2007). Although many potential features can be employed to represent documents, the following four types are most commonly used.

Characters

The individual component-level letters, numerals, special characters and spaces are the building blocks of higher-level semantic features such as words, terms, and concepts. A character-level representation can include the full set of all characters for a document or some filtered subset (Feldman & Sanger, 2007).

Character-based representations without positional information (i.e., bag-of-characters approaches) are often of very limited utility in text mining applications . Character-based representations that include some level of positional information (e.g., bigrams or trigrams) are somewhat more useful and common. However, character-based representations can often be unwieldy for some types of text processing techniques because the feature space for a document is fairly unoptimized. On the other hand, this feature space can in many ways be viewed as the most complete of any representation of a real-world text document (Feldman & Sanger, 2007).

Words

Specific words selected directly from a "native" document are what might be described as the basic level of semantic richness. For this reason, word level features are sometimes referred to as existing in the native feature space of a document. In general, a single word-level feature should equate with, or have the value of, no more than one linguistic token. Phrases, multiword expressions, or even multiword hyphenates would not constitute single word-level features. It is possible for a word-level representation of a document to include a feature for each word within that document – that is the "full text," where a document is represented by a complete and unabridged set of its word-level features (Feldman & Sanger, 2007).

Terms

Terms are single words and multiword phrases selected directly from the corpus of a native document by means of term-extraction methodologies. Term level features, in the sense of this definition, can only be made up of specific words and expressions found within the native document for which they are meant to be generally representative. Hence, a term-based representation of a document is necessarily composed of a subset of the terms in that document. Several of term-extraction methodologies can convert the raw text of a native document into a series of normalized terms – that is, sequences of one or more tokenized and lemmatized word forms associated with part-of-speech tags (Feldman & Sanger, 2007).

Sometimes an external lexicon is also used to provide a controlled vocabulary for term normalization. Term-extraction methodologies employ various approaches for generating and filtering an abbreviated list of most meaningful candidate terms from among a set of normalized terms for the representation of a document. This culling process results in a smaller but relatively more semantically rich document representation than that found in word-level document representations (Feldman & Sanger, 2007).

Opinion words are words that are commonly used to express positive or negative sentiments. For example, beautiful, wonderful, good, and amazing are positive opinion words, and bad, poor, and terrible are negative opinion words. Although many opinion words are adjectives and adverbs, nouns (e.g., rubbish, junk, and crap) and verbs (e.g., hate and like) can also indicate opinions. Apart from individual words, there are also opinion phrases and idioms, e.g.,cost someone an arm and a leg. Opinion words and phrases are instrumental to sentiment analysis for obvious reasons (Liu & Zhang, 2012).

Shin, Ryu, Ryu, Lee, and Lee (2016) proposed a method that employs a syntactic tree to extract phrases from Open Directory Project (ODP) and applies a phrase selection method to alleviate the high dimensionality problem of bag-of-phrases. Open Directory Project (ODP) is a large scale, high quality and publicly available web directory. They proposed to represent texts from a group of words called "phrase" rather than words alone for better understanding the semantic meaning of text .

Phrases often represent an atomic unit of meaning, so the observation that phrases can help to understand the semantic meaning of text is not new. Many semantically ambiguous words with single word expression can be discriminated by leveraging phrases. Combined word-based classifier with the phrase-based classifier and showed that the combination of classifiers provides a performance improvement for classification task (Shin *et al*, 2016).

Shin *et al.*, (2016) framework augments the text representation by extracting syntactic phrases, which aims to enrich the semantic information of texts. Their strategy was to improve the ODP-based text classification by bringing phrases that can benefit the classification task to gain more information for categories than single words. They then applied phrase selection method to alleviate negative impact of the high dimensionality of phrases from ODP.

Apart from term frequency features, Haddoud, Mokhtari, Lecroq, and Abdeddaim (2016) used term position features. Most of the terms that are related to the main topics of a document occur at its beginning. In order to validate this assumption, they proposed features of the form (t, p), meaning that the first position of t in the document is lower or equal to p. The position being defined as the number of words preceding the term occurrence. As for term frequency features, they generated only features (t, p) when p is a power of 2. For example, if a term t first appears at position 5 in a document of size 100 words, they generated the features $(t, 8)$, $(t, 16)$, $(t, 32)$ and $(t, 64)$, meaning that the first position of t is lower or equal than 8, 16, 32 and 64. The number of position features associated with a term t which appears in a document d at first position p will be $\log_2 |d|$ in the worst case, where $|d|$ is the size of d in number of words.
Chen *et al.* (2012) identified offensive content by using profanities, obscenities, and pejorative terms as features, weighted accordingly based on the associated strength of the term, as well as references to people. They also produced a set of rules to model offensive content, showing an improvement on standard machine learning approaches in terms of a much-reduced false negative rate (as cited in Burnap, P., & Williams, 2016).

Concepts

Concepts are features generated for a document by means of manual, statistical, rule-based, or hybrid categorization methodologies. Concept-level features can be manually generated for documents but are now more commonly extracted from documents using complex preprocessing routines that identify single words, multiword expressions, whole clauses, or even larger syntactical units that are then related to specific concept identifiers (Feldman & Sanger, 2007).

For instance, a document collection that includes reviews of sports cars may not actually include the specific word "automotive" or the specific phrase "test drives," but the concepts "automotive" and "test drives" might nevertheless be found among the set of concepts used to to identify and represent the collection (Feldman & Sanger, 2007).

Negations

Negation is a very common linguistic construction that affects polarity and therefore, needs to be taken into consideration in sentiment analysis. Negation is not only conveyed by common negation words (not, neither, nor) but also by other lexical units. Research in the field has shown that there are many other words that invert the polarity of an opinion expressed, such as valence shifters, connectives or modals (Vinodhini & Chandrasekaran, 2012).

Negation words are important because their appearances often change the opinion orientation. For example, the sentence "I don't like this camera" is negative. However, negation words must be handled with care because not all occurrences of such words mean negation. For example, "not" in "not only...but also" does not change the orientation direction (Liu & Zhang, 2012).

Syntactic Dependency

Word dependency based features generated from parsing or dependency trees have also been tried by several researchers (Liu & Zhang, 2012). The Semantic orientation approach to Sentiment analysis is “unsupervised learning” because it does not require prior training in order to mine the data. Instead, it measures how far a word is inclined towards positive and negative (Vinodhini & Chandrasekaran, 2012).

Gulin and Frolov (2016) formulated the problem of classification of text documents in view of certain structural features as a problem of machine learning using features of text documents that characterize the relationship in the set of tokens. Their applied models differ from the traditional bag-of-words model, which only reflects unary relations. The efficiency of such an upgrade of the machine learning methods was analyzed by means of computer experiments with the ten biggest classes of the Reuters-21578 set by applying eight common qualifiers. They concluded that it is relevant to consider the structural features of documents in the process of their classification.

Part of Speech

Many researches have found that adjectives are important indicators of opinions. Thus, adjectives have been treated as special features (Liu & Zhang, 2012). Part-of-speech tagging infers the lexical or grammatical category of a specific word inside a phrase or sentence. The drawback of these techniques is that they require language-dependent knowledge, thus having limited effectiveness in multilingual settings (George, George, & Georgios, 2015).

Non-Standard Words (NSWs)

Non-standard words are numbers, dates, time, abbreviations, acronyms, currency, measurement units, etc. Text normalization transforms NSWs in their extended form: the abbreviation /dr./into /doctor/or acronym /SMS/into /Short Message Service/ (Beliga & Martincic-Ipsic, 2014). The motivation for NSWs work rises from an idea that discarded characters and numbers are sufficient for discrimination of classification of the text categories, reducing the feature vector dimensionality at the same time (Manning *et al.*, 2009, as cited in Beliga & Martincic-Ipsic, 2014).

Traditionally, in the field of natural language processing (NLP), normalization is the process of transforming the unit into its normal form (i.e. lemma). Normalization is also the first step in the text preprocessing of Text-to-Speech (TTS) systems. It is performed in the normalization module, which is responsible to identify a NSW token and to transform it into its expanded form (Sproat *et al.*, 2001, Sproat, 2010, as cited in (Beliga & Martincic-Ipsic, 2014).

2.3 Feature Selection Methods

A big challenge in text categorization is learning from high dimensional data. On the other hand, tens and hundreds of thousands of terms in a document may lead to a high computational burden for the learning process. Also, some irrelevant and redundant features may hurt predictive performance of classifiers for text categorization. To avoid the issue of the "curse of dimensionality" and to speed up the learning process, it is necessary to perform feature reduction to reduce feature size. A common feature reduction approach for text categorization is feature selection (Tang, He, Baggenstoss, & Kay, 2015).

Feature selection is a process of finding a subset of features, from the original set of features forming patterns in a given data set, according to the defined criterion of feature selection (a feature goodness criterion) (Cios, Padrycz, Swiniarski, & Kurgan, 2007). Feature selection plays an important role in text categorization. Classic feature selection methods such as document frequency (DF), information gain (IG), mutual information (MI) are commonly applied in text categorization. But usually they only take plain text into account. Knowledge Gain (KG) is a new feature selection method which was proposed by Yan (2014).

Feature selection is done according to some specified set of rules to choose a part of feature words that have bigger contribution from high dimensional feature subspace apply to the subsequent classification process, to improve the text classification system's efficiency of time and space (Wu & Xu, 2015). The following section describes some feature selection methods.

2.3.1 Term Frequency

Weights are assigned to each term in a document that depends on the number of occurrences of the term in the document. By assigning a weight for each term in a document, a document may be viewed as a vector of weights. The simplest approach is to assign the weight to be equal to the number of occurrences of term t in document d. This weighting scheme is referred to as term frequency and is denoted $t_{ft,d}$, with the subscripts denoting the term and the document in order (Swamy *et al.*, 2014).

2.3.2 Document Frequency

Swamy *et al.* (2014) defined document frequency (DF) to be the number of documents in the collection that contain a term t. Yan (2014) explained that document frequency refers to the corpus of the entries appearing in the number of documents. Only terms appearing in more documents are preserved. When DF value is below a certain threshold of term is low-frequency words, such term will be removed from the original feature space. This method can not only reduce the feature space dimension, but also may improve the classification accuracy. DF is a simple word reduction technology; as opposed to corpus size has linear complexity, so it can easily be used for feature selection in large-scale corpus.

Kadmateekarun and Nuanmeesri (2015) used $tf - idf$ (Term Frequency – Inverted Document Frequency) in order to show the relationship between "word" and document. This $tf - idf$ was the tool to create the representation of document in form of vector which was applied to classify document to the designated categories. tf was the frequency in the document and *idf* was the invert of the document or on the other word it was called Inverted Document Frequency. This was found by applying Equation 2.3:

$$idf = 1 + log(N/df) \tag{2.3}$$

where N is the total number of documents in the group and *df* is the number of documents that "word" existed. Term Frequency – Inverted Document Frequency is calculated using Equation 2.3.

$$tf - idf = tf \times idf \tag{2.3}$$

$tf - idf$ was the way how to find the representation of vector which could be used to find out in the document. This was easy and efficient in classify the document (Kadmateekarun & Nuanmeesri, 2015).

TF counts the number of occurrences of a term t in a document, while IDF counts the (inverse) number of documents containing the term t. Traditional unsupervised term weights metrics, as the popular TFIDF, depend only on term frequency in the document and the (inverse) number of training documents containing this term. For the purpose of text classification, supervised alternatives have been developed to take into account the categories of the corpus documents.

The key idea is to build a metric function that discriminates the terms according to the category for which we are testing the document membership (Haddoud *et al.*, 2016).

Such a metric should be highly informative about the relation of a document term t to a category c and discriminative in separating the positive documents from the negative documents of category c. While unsupervised term weights depend only on term frequency in the document and the number of training corpus documents containing this term, supervised term weights are computed for each category and use the number of documents belonging to the category containing this term. Intuitively, supervised term weights measure the degree of correlation between term's presence in a document and membership of this document in the category (Haddoud *et al.*, 2016).

Haddoud *et al.* (2016) proposed an alternative to counting the (inverse) number of documents containing the term with frequency of at least n. By doing this, they integrated the term frequency in a document in the number of documents. The IDF quantity becomes the number of documents containing the term t at least n times. Their intuition was that this new definition of IDF keeps more information for learning. This assumption was confirmed experimentally as it improved the quality of the text classification.

2.3.3 Information Gain

Information gain (IG) is widely used in machine learning. Yan (2014) state that it is an entry by the presence or absence in an article in the amount of information to calculate the contribution of the category value. The information gain of term t for the category c is expressed in Equation 2.3:

$$IG(t) = -\sum_{i=1}^{|c|} P(c_i) log P(c_i) + P(t) \sum_{i=1}^{c} P(c_i|t) log P(c_i|t) + P(\bar{t}) \sum_{i=1}^{c} P(c_i|t) log P(c_i|\bar{t}) \tag{2.3}$$

Information gain (IG) heuristic is adopted due to its reported effectiveness in online text classification. $IG(C, A)$ measures the amount of entropy decrease on a class C when providing a feature A. The decreasing amount of entropy reflects the additional information gained by adding feature A, and higher values between 0 and 1 indicate more information gained by providing certain features (Suh, 2016).

IG is used for feature selection as it is easy to use, computationally efficient and widely used in sentiment classification. Since IG is a filter-based feature selection technique that is based on threshold value, all the features whose information gain value is greater than 0 are taken as prominent features (Agarwal, Mittal, & Cambria, 2013).

Researches show that Information Gain feature selection method is a good feature selection measure. Although the Information Gain feature selection algorithms can significantly improve the classification performance, its performance usually declined sharply when it deal with the situation which there is the imbalanced state of the term and classes (Wu & Xu, 2015).

There are certain drawbacks of Information Gain which have been identified by researchers. Wu and Xu (2015) identified two drawbacks. First, instead of attaching more importance to word frequency, traditional information gain pays more attention to document frequency, which weakens its feature anticipation capacity and makes it hard to choose the most effective feature. For example, if two feature words are distributed in the same document of each category, despite of different frequency, the calculated information gain value will be the same. Another drawback of traditional information gain is that information gain only takes into account the relevance between feature words and categories while neglects the distribution equation degree inside categories.
Some feature selection approaches ignore terms that are too frequent or too rare according to empirically chosen thresholds (Chakrabarti, 2003, as cited in Beliga & Martincic-Ipsic, 2014). As data sets become more complex, heuristics are

usually not sufficient. Therefore, another solution is to use measures such as Information Gain and Chi-Squere, to define relevance of each feature (Beliga, 2013 as cited in Beliga & Martincic-Ipsic, 2014).

2.3.4 CHI Square Statistics

According to Yan (2014), the χ^2 statistics measures the lack of independence between t and c and can be compared to the distribution with one degree of freedom to judge extremeness. Using the two-way contingency table of a term t and a category c, where A is the number of times t and c co-occur, B is the number of times t occurs, and N is the total number of documents, the term-goodness measure is defined by Equation 2.3:

$$\chi^2(t,c) = \frac{N * (AD - CB)^2}{(A+C) * (C+D) * (C+D)} \tag{2.3}$$

$$N = A + B + C + D$$

A, B, C, D represent quantity of document, as shown in Table 2.3

Table 2.3: Definition of A, B, C, D

	type c_i	not type c_i
t included	A	B
t not included	C	D

The χ^2 statistic has a natural value of zero if t and c are independent. For each category, the statistic between each unique term in a training corpus and that category is computed by Equation 2.3, and then combines the category-specific scores of each term into two scores (Yan, 2014):

$$\chi^2_{avg}(t) = \sum_{i=1}^{m} P_r(c_i)\chi^2(t,c_i) \tag{2.3}$$

$$\chi^2_{max}(t) = \max_{i=1}^{m}\{\chi^2(t, c_i)\}$$

The computation of CHI scores has a quadratic complexity, similar to IG, and χ^2 values are comparable across terms for the same category. However, this normalization breaks down (can no longer be accurately compared to the χ^2 distribution) if any cell in the contingency table is lightly populated, which is the case for low frequency terms. Hence, the statistic is known not to be reliable for low-frequency terms (Yan, 2014).

Shin *et al.* (2016) conducted phrase selection to alleviate negative impact of the high dimensionality. High dimensionality often leads to hurt the accuracy due to some noisy data. Their proposed framework also suffers from high dimensionality problem since the number of phrases in bag-of-phrases are 3 times larger than the number of words in bag-of-words. They applied χ^2 statistic (CHI) to reduce the dimensionality of the vector space. The CHI measures the lack of independence between a phrase p and a category c_i, and can be compared to the CHI distribution with one degree of freedom to judge extremeness.

Statistical learning methods rely on effective feature extraction to get a good learning result, so it is important for improving machine learning effect to extract effective features and avoid the noise interference. One effective way to extract features is chi-value which compares the contribution of certain word between one category and others. The method is widely adopted (Pei & Wu, 2014).

2.3.5 Knowledge Gain

Knowledge gain measures the amount of knowledge obtained for category prediction by knowing the presence or absence of a term in a document. Let $m\{C_i\}_{i=1}^{m}$ denote the set of categories in the target space. The knowledge gain of term t is defined by Equation 2.3:

$$KG(t) = KG(C|T) = c \sum_{1 \le i < j \le m} p_i \times p_j - (p(t) \sum_{i \le i < j \le m} p(c_i|t)p(c_j|t) + p(\bar{t}) \sum_{i \le i < j \le m} p(c_i|\bar{t})p(c_j|\bar{t}))c \quad (2.3)$$

Given a training corpus, for each unique term the knowledge gain is computed and those terms whose knowledge gain is less than some predetermined threshold are removed from the feature space (Yan, 2014).

Some researchers suggested that feature selection methods are not appropriate for text categorization domain. According to Mojgan, Yari, and Sayah (2011), throwing away uncommon features is usually not an appropriate strategy in text categorization. Another problem they identified is that feature selection normally uses indirect tests, such as χ^2 or mutual information, which involve setting arbitrary thresholds and conducting a heuristic greedy search to find a good subset of features. Moreover, by treating text categorization as a classical classification problem, standard approaches can ignore the fact that texts are written in natural language, which means that they have much implicit regularity that can be well modeled by specific tools from natural language processing.

2.4 Text Data Representation

Generally, there are two types of data: structured and unstructured. Structured data have keys (attributes, features) associated with each data item that reflect its content, meaning, or usage. A typical example of structured data is a relational table in a database. Given an attribute (column) name and its value, a set of tuples (rows) that include this value can be generated. The problem with using structured data is the cost associated with the process of structuring them (Markov & Larose, 2007).

The information that people use is available primarily in unstructured form. The largest part of it are text documents (books, magazines, newspapers) written in natural language. To have content-based access to these documents, they are organized in libraries, bibitem systems, and by other means. This process takes a lot of time and effort because it is done by people. There are attempts to use computers for this purpose, but the problem is that content-based access assumes understanding the meaning of documents, something that is still a research question, studied in the area of artificial intelligence and natural language processing in particular. One may argue that natural language texts are structured, which is true as long as the language syntax (grammatical structure) is concerned. However, the transition to meaning still requires semantic structuring or understanding (Markov & Larose, 2007).

There exists a solution that avoids the problem of meaning but still provides some types of content-based access to structured data. This is the keyword search approach known from the area of information retrieval (IR). IR approaches are applicable to bibliographic databases, collections of journal and newspaper articles, and other large text document collections that are not well structured (not organized by content), but require content-based access (Markov & Larose, 2007).

To mine text, it needs to be processed into a form that data-mining algorithms can use. The Pre-processing steps are: document standardization, tokenization, removal of stop words, stemming or lemmatization, and vector space table (Swamy, Hanumanthappa, & Jyothi, 2014). The common classifiers and learning algorithms cannot directly process the text documents in their original form. Therefore, during a preprocessing step, the documents are converted into a more manageable representation (Feldman & Sanger, 2007).

2.5 Document Representation Models

Documents need to be represented in a way that is suitable for a general learning process. The most widely used representation is “the bag of words” where a document is represented by a vector of features, each of which corresponds to a term or a phrase in a vocabulary collected from a particular data set. The value of each feature element represents the importance of the term in the document, according to a specific feature measurement (Bo *et al.*, 2016).

IR is the study of helping users to find information that matches their information needs. Technically, IR studies the acquisition, organization, storage, retrieval, and distribution of information. Historically, IR is about document retrieval, emphasizing document as the basic unit. An IR model governs how a document and a query are represented and how the relevance of a document to a user query is defined. There are four main IR models: Boolean Model, vector space model, language model and probabilistic model. The most commonly used models in IR systems and on the Web are the first three models. Although these three models represent documents and queries differently, they use the same framework. They all treat each document or query as a ”bag” of words or terms. Term sequence and position in a sentence or a document are ignored. That is, a document is described by a set of distinctive terms (Liu, 2007).

A term is simply a word whose semantics helps remember the document’s main themes. The term may not be a normal language word in a dictionary. Each term is associated with a weight. Given a collection of documents D, let $V = \{t_1, t_2, ..t_{|v|}\}$ be the set of distinctive terms in the collection, where t_i is a term. The set V is usually called the *vocabulary* of the collection, and $|V|$ is its size, i.e. the number of terms in V. A weight $w_{ij} > 0$ is associated with each term t_i of a document $d_j \in D$. For a term that does not appear in document $d_i, w_{ij} = 0$. Each document d_j is thus represented with a term vector $d_j = (w_{1j}, w_{2j}, ..., w_{|v|j})$, where each weight w_{ij} corresponds to the term $t_i \in V$, and quantifies the level of importance of t_i in document d_j. The sequence of the components (or terms) in the vector is not significant. With this vector representation, a collection of documents is simply represented as a relational table (or matrix). Each term is an attribute, and each weight is an attribute value (Liu, 2007).

2.5.1 Binary Valued or Boolean Model

The Boolean model is one of the earliest and simplest information retrieval models. It uses the notion of exact matching to match documents to the user query. Both query and the retrieval are based on Boolean algebra. In Boolean model, documents and queries are represented as set of terms. That is, each term is only considered present or absent in a document (Liu, 2007).

Using the vector representation of the document $d_j = (w_{1j}, w_{2j}, ..., w_{|v|j})$, the weight $w_{ij} \in \{0, 1\}$ of term t_i in document d_j is 1 if t_i appears in document d_j, and 0 otherwise as expressed in Equation 2.5.

$$w_{ij} = \begin{cases} 1 & if \quad t_i \quad appears \quad d_i \\ 0 & otherwise. \end{cases} \tag{2.5}$$

2.5.2 Vector Space or Bag of Words Model

Vector Space Model (VSM) is an algebraic model representing textual information as a vector. In VSM, after the required initial pre-processing, a dictionary of terms (vocabulary) is extracted from each source document which is compared against all the suspicious documents. VSM represents the importance of a word using term frequencyInverse document frequency (tf-idf) metric. Inverse document frequency, idf(t) is then calculated which emphasizes that a term which is almost present in the entire corpus of documents is not good. Finally their product, i.e. tf-idf is calculated and the similarity between document vectors is calculated using cosine similarity. VSM method is found to give good results when used along with approaches like ranking of documents, Latent Semantic Indexing (Vani & Gupta, 2014).

For document representation, each document is frequently represented by a bag of words (BOW), a feature vector that each dimension corresponds to a feature. BOW also assumes orthogonality of features and ignores word ordering and grammars. Let $C = \{c_1, c_2, ..., c_{|C|}\}$ be a set of classes of interest; $F = \{f_1, f_2, ..., f_{|F|}\}$ be a set of original features that occur at least once in at least one document in the data collection; $Tr = \{d_1, d_2, ...d_{|Tr|}\}$ be a training set of text documents; and $w(f_i, d_j,)$ be the feature weight of feature f_i in document d_i. A document d_i is represented as a feature weight vector $d_j = [w(f_1, d_j), ..., w(f_{|F|}, d_j)]$. This feature weight $w(f_i, d_j,)$ quantifies the importance of feature f_i for describing semantic content of document d_i (Pramokchon & Piamsa-nga, 2014).

Similarity between text documents is defined based on the bag-of-words representation and uses a vector-space model in which each document in the database and the user's query are represented by a multidimensional vector (Cios *et al.*, 2007).

VSM model is perhaps the best known and most widely used IR model. A document in the vector space model is represented as a weight vector, in which each component weight is computed based on some variation of Term Frequency (TF) or Term Frequency Inverse Document Frequency (TF-IDF) scheme. The weight w_{ij} in document d_j is no longer in $\{0, 1\}$ as in the Boolean model, but can be any number. In TF scheme, weight of a term t_i in document d_j is the number of times that t_i appears in document d_j, denoted by f_{ij}. The shortcoming of the TF scheme is that it does not consider the situation where a term appears in many documents of the collection. Such a term may not be discriminative. TF-IDF scheme is the most well known weighting scheme. There are several variations of this scheme (Liu, 2007).

According to Liu (2007), VSM model works as follows. Let N be the total number of documents in the system or the collection and df_i be the number of documents in which term t_i appears at least once. Let f_{ij} be the raw frequency count of the term t_i in document d_j. Then, the normalized term frequency (denoted by tf_{ij} of t_i in d_j is given by Equation 2.5.

$$tf_{ij} = \frac{f_{ij}}{max\{f_{1j}, f_{2j}, ..., f_{|v|j}\}} \tag{2.5}$$

where the maximum is computed over all terms that appear in document d_j. If term t_i does not appear in d_j then $tf_{ij} = 0$. The inverse document frequency (denoted by idf_i) of term t_i is given by

$$idf_i = log\frac{N}{df_i} \tag{2.5}$$

The intuition here is that if a term appears in a large number of documents in the collection, it is probably not important or not discriminative. The final TF-IDF term weight is given by Equation 2.5.

$$w_{ij} = tf_{ij} \times idf_i \tag{2.5}$$

The tf-idf is a bag of words weighting model used to give weights to the terms in a document collection by measuring how often a term is found within a document (term frequency), offset by how often the term is found within the entire collection (inverse document frequency). The "bags of words" assumption is that a document can be considered as a feature vector where each element in the vector indicates the presence (or absence) of a word, so that the information on the position of that word within the document is completely lost (Liu, 2007).

The BOW model is the simplest approach used in NLP and IR. In this model, a text (such as a sentence or a paragraph) is represented as the bag (multiset) of its keywords, not considering grammar and even order in which words are positioned but keeping multiplicity. The bag-of-words model is commonly used in methods of document or text categorization, where the (frequency of) occurrence of each keyword is used as a feature for training a classifier. It is the traditional form of text categorization used by many email clients for spam filtration. It is very simple to implement and useful in the scenarios where frequency of the word doesn't change much and where the same words are repeated again and again. But the use of this technique for giving solution to complex problems is very less as compared to other categorization techniques (Yadav & Parne, 2015).

Binary-valued (Boolean Model) and vector space (Real-valued) feature models are both widely used in TC for classification as well as for feature selection. Under the probabilistic framework of naive Bayes, the Binary-valued feature model is used in Bernouli naive Bayes, and the Real-valued feature model is used in multinomial naive Bayes or poisson naive Bayes. For classification, empirical studies have shown that the Real-valued feature model offers better performance than the Binary-valued feature model. Interestingly, at the stage of feature selection, the Binary-valued feature model is more commonly used than the Real-valued one (Tang *et al.*, 2015).

The vector space model as a technique based on bag-of words has achieved good performance in several tasks related to information retrieval (Marcos, Marcelo, Henrique, Patrick, & Elias, 2014). This model represents each term as a vector in space; this raises a major challenge for text classification which is the high dimensionality of feature space that can be tens of thousands of terms (Zaghoul & Al-Dhaheri, 2013).

Haddoud *et al.* (2016) proposed a model to convert a text document into a vector where each dimension represents a feature of the form (t, n), meaning that the term t appears in the document at least n times. If the term t appears 10 times in the document, they generate all the term frequency features (t, n) with n = 1, 2, 4 and 8 (powers of 2 in order to limit the number of features). Hence, rather than associating the weight TFIDF to a term t, they affect in their model the weight IDF to features (t, n) which depends on the inverse number of documents containing a term t at least n times. They also proposed another type of extended term feature based on the idea that the terms which are more correlated with the subject of the document tend to appear at the beginning. They generated term position features (t, p), meaning that the first position of t in the document is lower or equal to p. Experimental results showed that using these two extended term representations improves significantly the prediction of the text classifier.

A global dictionary is used to represent documents in the Vector Space Model where the dimension of the vectors is the number of words in the dictionary. Documents are represented in an easy and efficient way using the Bag of Words model. But this representation of documents, which is used by many classifiers, clearly ignores the significance of the discriminative power of semantic meaning of two or more words together as a phrase. Also the patterns discovered by such models are not interpretable and comprehensive. To capture this discriminative power of words as phrases there is need for a model which can include such n-grams in the vector space model without any additional changes in classifiers based on the vector space models (Deepak & Narsimha, 2012).

Documents are represented as feature vectors, which are, according to the bag-of words model, unordered list of words with the frequency of their occurrence in a document. Inverse document frequency is used to model the rareness of terms in a document collection (Beliga & Martincic-Ipsic, 2014). Typically, texts are represented with vectors of word's frequencies in the bag-of-words model. The problem with this approach is the dimensionality of the features vectors, which is equal to the number of different words in the collection (Feldman & Sanger, 2007, Manning *et al.*, 2009 as cited in Beliga & Martincic-Ipsic, 2014).

2.5.3 The Combined Model

BoW representation tends to divide text into single words and, hence, causing terms to break down into their constituent words. The model also treats synonymous words as independent features with no semantic association. These issues have been addressed by a number of researchers by representing text as concepts rather than words, using an approach known as Bag-of-Concepts (BOC). In the BOC approach, semantic knowledge bases such as WordNet, Open Directory Project (ODP) and Wikipedia are used to identify the concepts appearing within a document (Alaa, Arash, & Abdulhussain, 2014).

Alaa *et al.* (2014) introduced a set of new approaches for text representation for automatic classification of Arabic textual documents. These approaches were based on combining Bag-of-Words (BOW) and the Bag-of-Concepts (BOC) text representation schemes and utilizing Wikipedia as a knowledge base. By enriching the BOW model with concepts from the Wikipedia using different strategies via a combined model has three benefits: it resolves synonyms, it introduces more general concepts which help to identify related topics, and it resolves the low performance of BOC when used alone.

Adding Concepts (AC)

This approach involved adding concepts identified in the document to its BOW model as extra features. This approach has demonstrated its improving impact on the performance of standard English text classifiers (Alaa *et al.*, 2014). The different approaches of adding concepts are described in the following sections.

Replacing Terms with Concepts (RTC)

This approach is similar to AC, but with the following addition. Words in the document for which a Wikipedia concept has been identified are removed from the combined text representation model. The approach was first proposed for English TC and used for Arabic TC with Arabic WordNet (AWN) for concepts. The main difference between the AC and the RTC is that when using the RTC approach the combined vector will contain more concepts than words (Alaa *et al.*, 2014).

Adding Concepts and Categories (ACC)

In this approach, the parent categories of concepts are added to the combined feature vector of the representation model, along with concepts and words. This approach was first proposed for English TC (Alaa *et al.*, 2014).

Adding Unmapped Concepts (AUC)

This approach simply involve taking the basic BOW and adding to it the concepts whose representative words do not explicitly appear in the BOW model. In this way, the BOW vector is enriched with the general concepts that will be added using this approach. To be able to use this approach, a dictionary needs to be built after extracting features from Training Set (Alaa *et al.*, 2014).

First, all concepts and words are extracted as features from the Training Set. Then, each concept is mapped to its corresponding word(s). By doing this, the algorithm will resolve synonyms and capture different words which refer to the same concept in documents of the Training Set. Next, all words that are considered as features for the BOW model are added to the dictionary. In this way, words which have significant frequency value and are not represented by concepts are kept in the AUC model. Also, each concept that does not appear as a word in the BOW model and not mapped directly to a word(s) is added to the AUC dictionary (Alaa *et al.*, 2014).

Using Concepts for Terms which do not appear in the Document (CTD)

In a similar way to the BOW scheme, this approach works by using words for representing a document. The only difference is in the weighting scheme used, where all the words from the dictionary, which do not appear in the document but have a corresponding concept in the documents BOC vector, would be added to the combined vector with weights equal to those of the corresponding concepts. The weighting scheme works as follows; First, a 'Word-Concept' map or dictionary needs to be created. All concepts from the training documents are counted and mapped to their corresponding words from the documents. In most cases, a concept is mapped to one word (Alaa *et al.*, 2014).

However, depending on the concept, if it is a multi-word or it has synonyms it will be mapped to two or more words. To represent a document by CTD approach, first, each word from the BOW dictionary should be checked. If it appears in the document, the weight of that word is set using Equation 2.5. Otherwise, the corresponding concept for the word is retrieved from the Word-Concept map and the documents BOC vector will be searched for that concept.

$$TFIDF = TF(f_m, d_i)(ID\dot{F}(f_m) \tag{2.5}$$

If a concept exists, the word will be assigned the weight of the concept. If it does not exist, the word will not be represented in the combined vector (Alaa *et al.*, 2014).

2.5.4 Statistical Language Model

Statistical learning methods have become the absolute mainstream in text categorization. This is because statistical learning methods have less subjective factors compared with knowledge engineering methods. Besides, a lot of statistical learning technologies have a solid theoretical foundation, and there is a definite evaluation standard and good performance in statistical learning (Pei & Wu, 2014).

Statistical language models (or simply language models) are based on probability and have foundations in statistical theory. The basic idea of this approach to retrieval is simple. It first estimates a language model for each document, and then ranks documents by the likelihood of the query given the language model (Liu, 2007).

Let the query q be a sequence of terms, and the document collection D be the set of documents, $D = \{d_1, d_2, ..., d_N\}$. In the language modeling approach, the probability of a query q is considered as being "generated" by a probabilistic model based on a document d_j, i.e, $Pr(q \mid d_j)$. To rank documents in retrieval, the posterior probability $Pr(d_j \mid q)$. Using the Bayes rule as expressed in Equation 2.5.

$$Pr(d_j \mid q) = \frac{Pr(q \mid d_j)Pr(d_j)}{Pr(q)} \tag{2.5}$$

The language model used in most existing work is based on unigram, i.e. only individual terms (words) are considered. That is, the model assumes that each term (word) is generated independently, which is essentially a multinomial distribution over words. The general case is the n-gram model, where the nth term is conditioned on the previous $n - 1$ terms (Liu, 2007). Based on the multinomial distribution and the unigram model is expressed in Equation 2.5.

$$Pr(q = q_1 q_2 ... q_m) = \prod_{i=1}^{m} Pr(q_i|d_j) = \prod_{i=1}^{|V|} Pr(t_i|d_j)^{f_{iq}}, \tag{2.5}$$

where f_{iq} is the number of times t_i occurs in q, and $\sum_{i=1}^{V} Pr(t_i|d_j) = 1$. The retrieval problem is reduced to estimating $Pr(t_i|d_j)$, which can be the relative frequency as expressed in Equation 2.5.

$$Pr(t_i|d_j) = \frac{f_{ij}}{|d_j|}. \tag{2.5}$$

f_{ij} is the number of times that term t_i occurs in document d_j. $|d|$ denotes the total number of words in d_j (Liu, 2007).

2.5.5 Latent Dirichl*et al*location

Latent Dirichl*et al*location(LDA) is a Bayesian Model which models each data point (i.e., a document) as a collection of topics from a mixture model, where each mixture component is known as a topic. This is a generative model over a collection of documents which tries to capture the hidden thematic structure of the document. In contrast, given a normalized co-occurrence $w \times m$ matrix where w is the size of the dictionary and m is the number of documents in the corpus, LDA generates two matrices, a topic matrix Φ and a document matrix Θ is a word-topic distribution matrix of dimension $w \times k$, where k is the number of topics. The document matrix Θ is a topic-document distribution matrix. For finding phrases, only the word-topic matrix is needed, in which $\Phi^{(j)} = P(w|z = j)$ is the multinomial distribution of words over the topic j (Deepak & Narsimha, 2012).

Stas, Hladek, and Juhar (2014) used LDA in experimenting with statistical modeling of Slovac language in the step of text document classification. They defined LDA as a form of dimension reduction that uses a probabilistic model based on multinomial and Dirichlet distribution to find the co-occurrence patterns of terms that correspond to semantic topics in a collection of documents.

2.5.6 Context

Considering mere existence of terms (preprocessed words) or their frequencies for categorization, results in inaccurate classification of documents. To improve classification accuracy, context based TC should be used. Context based TC uses unsaid or implicit information, thereby improving relevance of documents to the category. Apart from terms, huge amount of information is hidden within the document itself. This information can be used as context during TC. Every term has different contribution in deciding context of the document. For example, the phrase “Time is money”, is not about ’money’ but about ’time’. Influence or importance of ’time’ is regarded to be more due to its occurrence in the beginning of the sentence. This shows that role of every term towards building context of the document is different. Hence it becomes more important to understand involvement and importance of a term in the context (Anagha, Vrinda & Parag, 2015).

2.6 N-grams

“Bag of words” model doesn’t capture the actual semantic meaning of a word in a particular document. Semantics are better captured by proximity of words and their occurrence in the document. Deepak and Narsimha (2012) proposed a new “Bag of Phrases” model to capture this discriminative power of phrases for text classification. There are varying definitions of n-grams by different researchers.

An N-gram is a token consisting of a series of characters or words. A token is generated by moving a sliding window across a corpus of text where the size of the window depends on the size of the token N and its displacement is done in stages, each stage corresponds to either a word or a character. Based on the different types of displacements, N-grams can be classified into two categories:

character based and word based (Mohan, Baggili, & Rogers, 2010).

An n-gram is defined either as a textual sequence of length n, or similarly, as a sequence of n adjacent 'textual units', in both cases extracted from a particular document. A 'textual unit' can be identified at a byte, character or word level depending on the context of interest. The simplest n-gram is the so-called unigram,where $n = 1$, which falls back to the single minded "bag-of-words" (BOW) representation. Typically, n is a fixed number, highly dependent on the particular corpus of documents and the queries made against that corpus. Each of the n-grams is a coordinate in a vector which represents the text under study and the frequency that this n-gram appears in the text can be the number of this coordinate (Bouras & Tsogkas, 2016).

Jurafsky, Martin, Peter and Stuat (2014) defined an n-gram as a sequence of n words: a 1-gram (unigram), a 2-gram (or bigram) is a two-word sequence, and a 3-word (or trigram) is a three-word sequence of words.

This unigram representation has been called the bag of words model. Unigram model can be thought of as putting the words of the training corpus in a bag and then selecting words one at a time. The notion of order of the words is lost; a unigram model gives the same probability to any permutation of a text. Higher-order n-gram models maintain some local notion of word order (Russel & Norvig, 2010).

N-Grams are the basic method for text categorization. It is also a statistical based approach for classifying text. The N is the number of keywords used for dividing the input text. Based on the number of keywords used, the N-grams are called as 2-grams, 3-grams, etc. It is able to classify the unknown text with highest certainty (Yadav & Parne, 2015).

In n-grams, initially the general pre-processing is carried out. Then the document is divided into N-grams or N-shingles. This refers to a sequence of consecutive words of size 'N', where 'N' is user specified. Both suspicious and source documents are converted to their N-gram profiles and similarity is calculated using Dice's coefficient. This is similar to Jaccard coefficient but it reduces the effect of shared terms between the documents (Vani & Gupta, 2014).

2.6.1 History of N-grams

The concept of N-grams was first introduced in Shannon's seminal paper on Information Theory in 1948. He wanted to extend the general theory of communication. He stated that the fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point. Frequently the messages have meaning; that is they refer to or are correlated according to some system with certain physical or conceptual entities. These semantic aspects of communication are irrelevant to the engineering problem. The significant aspect is that the actual message is one selected from a set of possible messages. The system must be designed to operate for each possible selection, not just the one which will actually be chosen since this is unknown at the time of design. If the number of messages in the set is finite then this number or any monotonic function of this number can be regarded as a measure of the information produced when one message is chosen from the set, all choices being equally likely (Shannon, 1948).

Shannon (1948) was concerned with the effect of statistical knowledge about the source of information in reducing the required capacity of the channel, by the use of proper encoding of the information. In telegraphy, for example, the messages to be transmitted consist of sequences of letters. These sequences, however, are not completely random. In general, they form sentences and have the statistical structure of, say, English. The letter E occurs more frequently than Q, the sequence TH more frequently than XP, etc. The existence of this structure allows one to make a saving in time (or channel capacity) by properly encoding the message sequences into signal sequences. This is already done to a limited extent in telegraphy by using the shortest channel symbol, a dot, for the most common English letter E; while the infrequent letters, Q, X, Z are represented by longer sequences of dots and dashes. This idea is carried still further in certain commercial codes where common words and phrases are represented by four- or five-letter code groups with a considerable saving in average time.

A physical system, or a mathematical model of a system which produces such a sequence of symbols governed by a set of probabilities, is known as a stochastic process. A discrete source may be considered, therefore, to be represented by a stochastic process. Conversely, any stochastic process which produces a discrete sequence of symbols chosen from a finite set may be considered a discrete source. This includes such cases as if successive symbols are not chosen independently but their probabilities depend on preceding letters. In the simplest case of this type a choice depends only on the preceding letter and not on ones before that (Shannon, 1948).

The statistical structure can then be described by a set of transition probabilities $P_i(j)$, the probability that letter i is followed by letter j. The indices i and j range over all the possible symbols. A second equivalent way of specifying the structure is to give the "digram" probabilities, $p(i, j)$, i.e., the relative frequency of the digram ij. Stochastic processes can also be defined which produce a text consisting of a sequence of "words". If all the words are of finite length this process is equivalent to one of the preceding type, but the description may be simpler in terms of the word structure and probabilities (Shannon, 1948).

2.6.2 Applications of N-grams

Shannon (1948) used character based N-grams for analyzing and predicting printed English. Since then, his approach with character based N-grams has been applied to other areas like spelling and error correction, text compression, language identification and text search and retrieval (Mohan, 2010).

There are two obvious bases for the characterisation and manipulation of text, these being either the individual characters that form the basis for the byte-level operations available to computers, or the individual words that are used by people. These basic units can then be assembled into larger text segments such as sentences, paragraphs, chapters, etc. N-grams provide an intermediate level of text characterisation that has advantages in terms of efficiency and/or effectiveness over the conventional character-based or word-based approaches to several important applications in textual computing. Applications that can be implemented efficiently and effectively using sets of ngrams include spelling error detection and correction, query expansion, information retrieval with serial, inverted and signature files, dictionary look-up, text compression, and language identification (Robertson & Willett, 1998).

The use of n-gram probability distribution and n-gram models in Natural Language Processing (NLP) is a relatively simple idea, but it has been found to be effective in many applications. N-grams has been applied in diverse areas, as described in the next sub-sections.

Content Filtering

Content-based filtering is mainly based on the use of the Machine Learning (ML) paradigm. In ML, a classifier is automatically induced by learning from a set of preclassified examples. The feature extraction procedure maps text into a compact representation of its content, which is uniformly applied to training and generalization phases. Bag-of-Words (BoW) approach yields good performance and exists in general over more sophisticated text representation that may have superior semantics but lower statistical quality (Lewis, 1992, as cited in, Thilagavathi & Taarika, 2014).

Machine Learning based text categorization techniques are used to automatically allot each short text message with set of categories based on the content.

Short Text Classifier is built for accurate extraction and set of discriminating feature in the message (Thilagavathi & Taarika, 2014).

Duplicate Detection

Computerized methods for document similarity estimation (or plagiarism detection) in natural languages have evolved. Plagiarism is a common critical problem, which appears almost in all universities and research institutes (Hussein, 2015). There are mainly two classes of methods to detect plagiarism in free texts: Language-independent methods and language sensitive methods. Language-independent methods are based on evaluating text characteristics, which are not inherent to a specific natural language, such as the number of single characters, and the average sentence lengths (Gruner & Naven, 2005, as cited in Hussein, 2015).

On the other hand, language sensitive methods are based on evaluating text characteristics, which are specific to a single language. The methods of detecting external plagiarism in free texts fall into two groups: Stylometry-based methods and content-based methods. Stylometry-based methods generally target the author identification techniques to classify authors‘ styles and identify the possible similarity while content-based methods are concerned with text analysis, in terms of logical structure, to find out the similarity (Alzahrani, Salim, & Abraham, 2012, as cited in Hussein, 2015)

One efficient duplicate detection technique is based on n-grams (also called shingles). An n-gram is simply a consecutive sequence of words of a fixed window size n. For example, the sentence, “John went to school with his brother”, can be represented with five 3-gram phrases “John went to”, “went to school”, “to school with”, “school with his”, and “with his brother” (Liu, 2007).
Let $S_n(d)$ be the set of distinctive n-grams (or shingles) contained in document d. Each n-gram may be coded with a number or a Message-Digest algorithm 5

(MD5) hash (which is usually a 32-digit hexadecimal number). Given the n-gram representations of the two documents d_1 and d_2, $S_n(d_1)$ and $S_n(d_2)$, the Jaccard coefficient, as expressed in Equation 2.6, can be used to compute the similarity of the two documents,

$$sim(d_1, d_2) = \frac{|S_n(d_1) \cap S_n(d_2)|}{|S_n(d_1) \cup S_n(d_2)|}. \tag{2.6}$$

A threshold is used to determine whether d_1 and d_2 are likely to be duplicates of each other. For a particular application, the window size n and the similarity threshold are chosen through experiments (Liu, 2007).

A language is modeled by making use of linguistic and common sense knowledge about the language. Formally, a language model is a probability distribution over word sequences or word N-grams. Specifically, a language model (LM) estimates the probability of next words given preceding words. A word N-gram language model uses the history of $N - 1$ immediately preceding words to compute the occurrence probability P of the current word. The value of N is usually limited to 2 (bigram model) or 3 (trigram model). If the vocabulary size is M words, then to provide complete coverage of all possible N word sequences the language model needs to consist of M^N-grams (i.e., sequences of N words). Typically an N-gram LM lists only the most frequently occurring word pairs, and uses a backoff mechanism to compute the probability when the desired word pair is not found (Majumder, Mitra, & Chaudhuri, 2002).

Vani and Gupta (2014) compared different methods of document categorization in external plagiarism detection. Their primary focus was to explore the unsupervised document categorization/clustering methods using different variations of K-means algorithm and compare it with the general N-gram based method and Vector Space Model based method. From their comparisons and analysis, they observed that K-means method gives promising results when dealing with highly obfuscated data compared to the other two approaches. Further, they concluded that in terms of execution time, their proposed K-means algorithm performed efficiently.

Machine Language Translation

One of the early goals of natural language processing was to build a system that could translate text from one human language to another. Behind this attempt is an implicit assumption that human languages are like codes: in other words, a word in one language is simply a code for a real-world object, emotion, action, place, etc., and can therefore be exchanged for the code in another language for the same thing. In particular, machine translation is not possible simply by using syntactic and lexical analysis: a knowledge of the world that is being discussed is also essential, in order to disambiguate the text that is being translated. It may be, in some cases, that the text can be translated directly, ignoring the ambiguity, and creating a similarly ambiguous sentence in the target language (Coppin, 2004).

Language Identification

Software Language Identification (SLI) is a problem of determining correctly which software language was a source fragment written in. Software language identification techniques are applicable to many situations from universal Integrated Development Environments (IDE) support to legacy code analysis (van Dam & Zaytsev, 2016).

van Dam and Zaytsev (2016) proposed the use of statistical language models from natural language processing field such as n-grams, skip-grams, multinomial naive Bayes and normalised compression distance. Their preliminary results showed that some of these models used as classifiers achieved high precision and recall and can be used to properly identify language families, languages and even deal with embedded code fragments.

A similar, but easier problem to machine translation is that of language identification. There are many thousands of human languages in the world, and several hundred that are widely used today. Many of these are related to each other, and so can be easily confused. For an English speaker who knows no Italian or Spanish, those two languages can sometimes appear similar, for example. A system that can identify which language is being used in a piece of text is thus very useful (Coppin, 2004).

It is also particularly useful in applying textual analysis of all kinds to documents that appear on the Internet. One way to determine the language of a piece of text would be to have a complete lexicon of all words in all languages. This would clearly provide accurate results, but is likely to be impractical to develop for a number of reasons. The lexicon would be enormous, of course, and it would be very difficult to ensure that all words were really included (Coppin, 2004).

The acquaintance algorithm is a commonly used method for language identification that uses n-grams. An n-gram is simply a collection of n letters, but detailed statistics exist that indicate the likelihood of a particular set of letters occurring in any given language. Hence, for example, the trigrams *ing*, and, the, *ent*, and *ant* probably indicate that a document is in English. When the acquaintance algorithm is presented with sufficient text (usually a few hundred to a thousand words is sufficient), it is able to identify the language with a surprisingly high degree of accuracy (Coppin, 2004).

The acquaintance algorithm is trained by being presented with text in each language that it is expected to identify. The system then calculates a vector for each language based on the training data. This vector stores information about how many times each n-gram occurs in that language. When a document in an unknown language is presented to the algorithm, it calculates a similar vector for this document and compares it with the vectors it has calculated for the training data (Coppin, 2004).

The vector that is closest indicates which language is being used in the document. One advantage of this approach is that it is easy to tell how certain the algorithm is about a particular document. A score is calculated for a document for each possible language, and the language with the highest score is selected. If the highest score is very much higher than the second highest score, this indicates a high degree of certainty. Conversely, if the top two or three scores are similar, then the algorithm is less certain, and there are one or more other possibilities that might need to be examined (Coppin, 2004).

Spelling-error Correction

Akshay, Amit, Kunal and Swapnali (2016) explored the use of new software for input on desktops which relied on a dynamic predictive algorithm using n-grams and suffix trees to significantly reduce the effort of typing. In their project, they developed several modules. A module called Autocorrect Engine was responsible for verifying the correctness of a typed word. It builds a set of close words to the mistaken word and searches these words in a dictionary for an exact match. Autocomplete Engine was responsible for predicting remaining part of an incompletely typed word and for guessing which word is likely to continue a given initial text fragment.

Akshay *et al.* (2016) carried out next word predictions by implementing n-gram language models. N-grams were implemented by adding additional nodes to an existing suffix tree and sorting them with respect to their frequencies. N-grams were developed in two phases. In the first phase, predictions were made on the basis of statistical model, consisting of default frequencies of words in a dictionary. In the second phase, frequencies were modified with each iteration according to the user's style of typing.

An effective method for automatic spelling correction must take account of the types of errors which occur in the texts, and thus of the nature and source of the text (Robertson & Willett, 1998). However, all spelling errors can be characterized as arising from the occurrence of the following types of error:

- one extra character inserted (e.g. GRAAM for GRAM);
- one character omitted (e.g. GAM for GRAM);
- an incorrect character substituted for a correct one (e.g. GREM for GRAM), occurs very frequently with OCR text.
- two adjacent characters transposed (e.g. GARM for GRAM), which is characteristic of poor typists and which can alternatively be thought of as two, linked substitution errors .

The differences between the correct spelling of a word and any particular erroneous spelling of it can be described in terms of one or more of the types of errors above. The prevalence of simple errors means that the application of n-gram matching is likely to be effective in many cases, as evidenced by the several studies of n-gram-based correction methods that have been reported (Robertson & Willett, 1998).

Spelling-error detection

N-grams can be used not only for the correction but also for the detection of spelling errors. In spelling-error detection, counts are made of the digrams and trigrams within a corpus of a particular type of text, thus allowing the calculation of the probabilities of occurrence for each of the possible n-grams. Given these probabilities, a score can be computed for a word whose spelling is to be checked, based on the word's constituent n-grams; the word is then flagged as possibly misspelt if it contains one or more improbable n-grams (Robertson & Willett, 1998).

The use of n-grams is exemplified by Salton's description of the TYPO program, an error-detection program in which each word to be checked is first broken down into its constituent trigrams. For each of these, a peculiarity index is computed which measures the probability that a particular trigram was drawn from the same source as the rest of the text. A combined-word peculiarity index is then computed as the root-mean-square of the indexes of the individual trigrams. Specifically, each trigram occurring in a given text is considered together with its constituent digrams: a misspelled word is likely to have a high peculiarity index because acceptable digram frequencies will be combined with unusually low frequencies for the corresponding trigrams. TYPO produces a list of words sorted in decreasing order of their peculiarity index, with the result that questionable word forms will appear near the top of the list, and can thus be inspected for further processing as necessary. Such approaches are elegant in concept but alternative approaches to spelling error detection may be more effective in practice, as demonstrated by the extended experiments at Chemical Abstracts Service which suggest that trigram-based error detection is noticeably inferior to a simpler, dictionary look-up procedure (Robertson & Willett, 1998).

In 2015, Samani, Rahimi, and Rahimi developed a real-word error checker and detector system for Persian language using a predefined candidate set and made content-aware choices based on frequent adjacent n-grams of each potentially real word error. They defined real-word errors as the misspelled words that have been converted wrongly to another lexicon word. Detection of these errors required semantic analysis of the content. Their results indicated that their proposed system had high performance which they attributed to good coverage of Persian language real-words error and using proper algorithms and parameters.

Query Expansion

In a text context, document/query expansion has proven very useful in retrieving objects semantically related to the query. The problem with retrieving documents that are semantically related to the query is that many of these desired documents may not necessarily contain the query term. For example, given the query "airplane", some relevant documents may instead contain the term "aircraft". In query expansion approach to retrieving semantically related documents, semantically related terms are automatically added to the query (Lee & Lee, 2014).

The basic idea for conventional language modeling approach for text is that a query Q and document d are respectively represented as unigram language models θ_Q and θ_d, or term distributions $P(t|\theta_Q)$ and $P(t|\theta_d)$, where t is a term. The relevance score function $S(Q, d)$ used to rank the documents d with respect to the given query Q is the inverse of Kullback-Leibler (KL) divergence between θ_Q and θ_d as expressed in Equation 2.6 (Lee & Lee, 2014).

$$S(Q, d) = -KL(\theta_Q || \theta_d) \tag{2.6}$$

That is, documents whose unigram language models are similar to the query's unigram language model are more likely to be relevant. In this way, the problem of retrieval is reduced to the estimation of the unigram language models for the queries and documents. The term t can be any sequence of n consecutive words or subword units (referred to as word or subword n-gram, or a word or subword unit if $n = 1$). In other words, θ_Q and θ_d are the distributions of such word or subword n-grams (Lee & Lee, 2014).

Freund and Willett (1982) used an inverted file implementation in which each n-gram is associated with a list that contains the identifiers for all of the dictionary words in which it occurs. The best-matching words can then be found by processing just those lists that are associated with the n-grams comprising the query word for which the variants are required. They studied the use of both digrams and trigrams with query words and dictionaries derived from document test collections and found that digrams retrieved a much larger number of non-related terms than did trigrams.

Document Clustering

Collaborative filtering systems typically need to acquire some data about the new user in order to start making personalized suggestions, a situation commonly referred to as the “new user problem”. The new user problem can be addressed via a unique personalized strategy for prompting the user with articles to rate. Information Filtering (IF) focuses on tasks involving classifying streams of new content into categories, such as finding any newly released articles regarding the political situation in Middle East, or any newly released movies without an English-language soundtrack or subtitles (to reject). Collaborative Filtering (CF) focuses on two important questions: which items (from a set or overall) should be proposed to a user, and how appealing these particular items will be for the user (Bouras & Tsogkas, 2016).

Clustering is the task of grouping a set of objects in such a way that objects of the same group are more similar to each other than to those of other groups, plays a central role in data analysis. It can give useful information about the structure of the underlying data. Clustering can be used as a collaborative filtering technique (Bouras & Tsogkas, 2016).

Text Classification

Because of the popularity of social networks, forums, texts with very short length rush into our life, such as, micro-blogs, short messages, comments and interactions on forums. Short texts contain plenty of information throughout all fields like sports, education, science and so on. When faced with so many micro-blogs, confusion arises and it is very hard for us to pick out information we need or are interested in effectively (Zhang & Wu, 2015). Compared with normal texts, short texts have following characteristics (Yan *et al.*, 2009, Su *et al.*, 2006, as cited in Zhang & Wu, 2015):

- Sparseness: Like product description, forum remarks and short messages, short texts all have very short length with no more than 100 words. This leads to the sparseness of features and it is very hard to extract accurate and key features for classification learning.

- Irregular: The grammar of short texts is usually informal. There often exist spelling mistakes and face expression symbols. These irregular words contain a lot of noisy features and thus increase the difficulty of information extraction for computers.

- Real-time: Short texts on the internet are frequently refreshed, so the amount of short texts is very huge. This means that the classification algorithm must have very good time performance.

Because of these characteristics mentioned, many classic classification algorithms failed to achieve a high accuracy when working with short texts. Inspired by the idea of feature extension, Zhang and Wu (2015) presented a method to extend features from original features of short texts on the basis of the N-Gram model.

N-grams are part of large strings generated from larger text. In this approach the input text is divided into substrings with n maximum length, then the frequency of occurrence are counted. After the count of all n-grams, the least frequent are discarded and the rest is written to a profile. The profile of unknown text is then compared to each and every language (category) listed. This approach is useful if small text is to be categorised such as comments in social network sites, conversations in a forum, etc because each sentence is divided into small parts. First the dataset of the language that needs to be classified is prepared. After that the input text is divided into small tokens till it matches with the language dataset. It is used with Bag-of-words model for better result in document classification. The result gives good performance for statistical approach but semantic similarity does not yield an expected output (Yadav & Parne, 2015).

N-Gram is a statistical based approach for classifying text. N is the number of keywords used for dividing the input text. Based on the number of keywords used the N-grams are called 2-grams, 3-grams, etc. It is able to classify the unknown text with highest certainty. This approach is useful when small text is to be categorized such as comments in social network sites, conversations in a forum, etc. because each sentence is divided into small parts (Shashank & Pratik, 2015).

van Dam and Zaytsev (2016) described n-gram language model as a probability model used to predict the probability of a n-gram in a language. More specifically, it predicts the next word after a sequence of words. By using these probabilities, one can calculate the probability that a document d belongs to a certain class c given by Equation 2.6:

$$P(c|d) = \prod_{i=1}^{n} P(w_i|w_{i-n+1}^{i-1})$$

$$P(w_i|w_{i-n+1}^{i-1}) = \frac{freq(w_{i-n+1}^{i-1}w_i)}{freq(w_{i-n+1}^{i-1})} \quad (2.6)$$

where $freq$ gives the frequency of the n-gram in the class.

Since training data is finite, it is very likely that some n-grams will occur in a document are not seen in the training data. These n-grams will cause n-gram model in Equation 2.6 to assign probability of zero to the document. These zero-probability n-grams need to be changed to ensure an accurate probability to be calculated. This is done by smoothing. Smoothing gives probabilities to n-grams of which the smaller n-grams have occurred in the training data (van Dam & Zaytsev, 2016).

2.6.3 N-gram Analysis

An n-gram is defined either as a textual sequence of length n, or similarly, as a sequence of n adjacent ‘textual units’. Another aspect of n-gram analysis that should also be stressed out, is that infrequent n-grams are uninteresting, thus one only needs to keep track of n-grams that occur with frequency above a certain threshold, say at least m times. Furthermore, determining the correct value of n, i.e. the size of the sliding window that is to be used, when using word based n-gram analysis, is an area of experimentation on each particular domain of knowledge (Bouras & Tsogkas, 2016).

For example, on the domain of plagiarism detection, low values for n appear to give the best results with a specific precision-recall tradeoff. Values larger than four diminish the performance of the approach. A similar result is also pointed out that word sequences of length two or three prove to be the most useful since larger sequences reduce classification performance. Also, stemming is not being applied on the extracted n-grams since stemming techniques can be used in word-based systems but not in n-gram-based systems (Bouras & Tsogkas, 2016).

In their experiments, Bouras and Tsogkas (2016) presented a means of utilizing both the classical BOW representation and the n-gram expansion in a tunable fashion, via the n-gram weighting parameters A and B. Their experimentation towards determining the best value for the n parameter revealed similar results with the existing literature, i.e. using 2-gram and 3-gram for the weighting process seems to yield better performance in terms of clustering index (CI) when it comes to clustering news articles from the web. Furthermore, they saw that it is not enough to simply include n-grams into the equation. The weighting, given to the n-grams compared to regular keywords, as conveyed by the parameters A and B, is thus of great importance, not previously explored and an area that is domain-specific and open for further experimentation.

Writing style markers that are known as the most effective discriminators of authorship in social media are lexical, syntactic, structural, and content-specific writing style features. Lexical, syntactic, and structural writing style features are called the content-free writing style features. Among the four writing style features in social media, the content-specific writing style features are expected to outperform the other content-free writing style features because of their two characteristics: first, they consist of important keywords and phrases, so they are more meaningful with high representative ability than the other writing style features. Second, they contain a much larger number of n-grams extracted from the collected social media data, and the large potential feature spaces are known to be effective for online text classification (Suh, 2016).

Deepak and Narsimha (2012) proposed a novel approach to find phrases from the documents using a topic model, Latent Dirichl*et al*location (LDA), without using any explicit knowledge base. The researchers used these phrases together with individual words for Vector Space Model representation of the documents which is exploited in classification. The main advantage of the proposed method was that all established classifiers can be used without any modification with a possibly significant improvement in classification accuracy. Experiments showed that the proposed Bag of Phrases model outperformed the conventional Bag of Words model.

A common aspect of standard machine learning approaches is that they treat text categorization as a standard classification problem, and thereby reduce the learning process to two simple steps: feature engineering, and classification learning over the feature space. Of these two steps, feature engineering is critical to achieving good performance in text categorization problems. Once good features are identified, almost any reasonable technique for learning a classifier seems to perform reasonably well (Farhoodi *et al.*, 2011).

Unfortunately, the standard classification learning methodology has several drawbacks. First, there are enormous number of possible features to consider in text categorization, and standard feature selection approaches do not always cope well in such circumstances. For example, given a sufficiently large number of features, the cumulative effect of uncommon features can still have an important effect on classification accuracy, even though infrequent features contribute less information than common features individually. N-gram model is the simplest and most successful basis for language modeling (Farhoodi *et al.*, 2011).

Unfortunately, using N-grams of length up to n entails estimating the probability of W^n events, where W is the size of the word vocabulary. This quickly overwhelms modern computational and data resources for even modest choices of n. Also, because of the heavy tailed nature of language (i.e. Zipf's law) one is likely to encounter novel N-grams that were never witnessed during training (Farhoodi *et al.*, 2011).

Zipf's Law states that "The n^{th} most common word in a human language text occurs with a frequency inversely proportional to n". That is, $f \propto \frac{1}{r}$, where f is the frequency of the word and r is the rank of the word in the list ordered by the frequency (Cavnar and Trenkle, 1994, as cited in Farhoodi *et al.*, 2011).

2.6.4 N-gram graphs

George *et al.* (2015) argued that the established bag-of-words representation models are inadequate for handling all document types, as they merely extract frequent, yet distinguishing terms from the textual content of the training set. Thus, they suffer from low robustness in the context of noisy or unseen content, unless they are enriched with contextual, application-specific information.

As a result George *et al.* (2015) proposed the use of N-gram graphs, a model that goes beyond the bag-of-words representation, transforming every document into a graph: its nodes correspond to character or word N-grams and the co-occurring ones are connected by weighted edges. Individual document graphs can be combined into class graphs and graph similarities are employed to position and classify documents into the vector space. The researchers identified two advantages of this approach with respect to bag models: First, classification accuracy increases due to the contextual information that is encapsulated in the edges of the N-gram graphs. Second, it reduces the search space to a limited set of robust, endogenous features that depend on the number of classes, rather than the size of the vocabulary.

An n-gram graph is an undirected graph $G = \{V, E, W\}$, where V is the set of its vertices, with each vertex corresponding to a distinct N-gram, E is the set of edges between co-occurring N-grams, and W is a function that determines the weight of each edge according to the co-occurrence frequency of its adjacent vertices. Each vertex is labelled by the corresponding N-gram and each edge by the labels of its adjacent vertices – concatenated in alphabetic order (George *et al.*, 2015).

2.6.5 Skip-grams

A skip-gram is a generalisation of N-grams where words might be skipped and does not need to be consecutive, being an N-gram a sequence of variable characters that stands for a word or string of words in a corpus. In the example “Everyone loves classic movies.”, “everyone loves movies” is an example of 1-skip-3-gram (a 3-gram with 1 skipped word). More specifically, in a k-skip-ngram, n determines the maximum number of adjacent terms involved, and k the maximum number of skipped terms allowed Skip-grams consider the long distance words as a feature. For example consider “You are an idiot” as a comment, if we use 2-skip-gram, count of ’You are’ as one feature and ’an idiot’ as other is added in our feature-matrix. This way, the comments containing co-occurrences of words like “You idiots” which is negative and will be detected using skip-grams (Guthrie, Allison, Liu, Guthrie, & Wilks, 2006, as cited in Gutierrez, Thomas, & Fernandez, 2015).

2.7 Machine Learning

Machine Learning is a subfield of artificial intelligence that deals with the exploration and construction of systems that learn from data. Machine learning trains computers to manage critical situations via examining, self training, inference by observation and previous experience (Prakash, Patric, Jacob, & Radhameena, 2014).

Machine learning is best described as learning from example. Machine learning incorporates a variety of methods such as supervised and unsupervised learning. In supervised learning, a teacher is available to define correct or incorrect responses. Unsupervised learning differs in that no teacher is present. Instead, unsupervised learning learns from the data itself by identifying its relationships (Jones, 2008).

A wide variety of classical machine learning techniques have been used for text classification. Learning methods are frequently used to automatically construct classifiers from labeled documents. The machine learning approach applicable to sentiment analysis mostly belongs to supervised classification in general and text classification techniques in particular. Thus, it is called "supervised learning. In a machine learning based classification, two sets of documents are required: training and a test set. A training set is used by an automatic classifier to learn the differentiating characteristics of documents, and a test set is used to validate the performance of the automatic classifier. A number of machine learning techniques have been adopted to classify sentiments (Liu, 2007).

Supervised learning has been a great success in real-world applications. It is used in almost every domain, including text and Web domains. Supervised learning is also called classification or inductive learning in machine learning.

This type of learning is analogous to human learning from past experiences to gain new knowledge in order to improve our ability to perform real-world tasks. However, since computers do not have "experiences", machine learning learns from data, which are collected in the past and represent past experiences in some real-world applications (Liu, 2007).

Multi-class text classification is a supervised-learning process for assigning a class in a finite set of predefined classes to text documents. Given a supervised training set of labeled text documents, the goal is to induce a classifier based on the assessment of their content to correctly label a class of a new unlabeled document (also known as testing set). In general, multi-class text classification consists of two phases: learning and classifying phases (Pramokchon & Piamsanga, 2014).

2.7.1 Basic Concepts

A data set used in the learning task consists of a set of data records, which are described by a set of attributes $A = \{A_1, A_2, ..., A_{|A|}\}$, where $|A|$ denotes the number of attributes or the size of the set A. The data set also has a special target attribute C, which is called the class attribute. The class attribute C has a set of discrete values, i.e., $\{c_1, c_2, ..., c_{|C|}\}$, where $|C|$ is the number of classes and $|C| \geq 2$. A class value is also called a class label (Liu, 2007).

A data set for learning is simply a relational table. Each data record describes a piece of "past experience". In the machine learning and data mining literature, a data record is also called an example, an instance, a case or a vector. A data set basically consists of examples or instances. Given a data set D, the objective of learning is to produce a classification/prediction function to relate values of attributes in A and classes in C. The function can be used to predict the class values/labels of the future data. The function is also called a classification model, a predictive model or simply a classifier (Liu, 2007).

The data set used for learning is called the training data (or the training set). After a model is learned or built from the training data by a learning algorithm, it is evaluated using a set of test data (or unseen data) to assess the model accuracy. The test data is not used in learning the classification model. The examples in the test data usually also have class labels. In order to learn and also test, the available data (which has classes) for learning is usually split into two disjoint subsets, the training set (for learning) and the test set (for testing). To achieve good accuracy on the test data, training examples must be sufficiently representative of the test data (Liu, 2007).

2.7.2 Training

In this phase, a textual feature is selected as the measurement criterion and then some numeric values extracted from documents with known classes are assigned to the textual feature (these documents are called training set). This phase is referred to training phase. At the testing phase, a classification algorithm after receiving the feature vectors of some documents with unknown classes (test set), predicts the most likely class of them by comparing the test set' s feature vectors with those of documents in training set (Ramezani, Sheydaei, & Kahani, 2013).

In most learning problems, the task is to learn to classify inputs according to a finite (or sometimes infinite) set of classifications. Typically, a learning system is provided with a set of training data, which have been classified by hand. The system then attempts to learn from these training data how to classify the same data (usually a relatively easy task) and also how to classify new data that it has not seen (Coppin, 2004).

Learning to classify unseen data assumes that there is some relationship between the data and the classifications i.e. some function f can be generated such that if a piece of data x belongs in classification y, then $f(x) = y$. For example,

if the equality function were used, the learning task would be relatively simple because each datum would be classified as itself. Most real-world problems are not so simple, and producing a function that approximates the correct mapping is one of the main challenges of machine learning. In most learning problems, the input data consist of more than one variable (Coppin, 2004).
Cios *et al.* (2007) defines classifier as a model of data used for a classification purpose: given a new input, it assigns that input to one of the classes it was designed/trained to recognize. A model can be defined as a description of causal relationships between input and output variables. This description can be formulated in several ways and can take the form of a classifier,neural network, decision tree, production rules, mathematical equations, etc. Many data modeling methods were developed in the form of mathematical equations in the area of statistics.

Rote Learning

The simplest way for a computer to learn from experience is simply to learn by rote. Training involves storing each piece of training data and its classification. Thereafter, a new item of data is classified by looking to see if it is stored in memory. If it is, then the classification that was stored with that item is returned. Otherwise, the method fails. Hence, a rote learner is able to classify only data that it has already seen, and no attempt is made to approximate the mapping function, which is a major weakness (Coppin, 2004).

Learning Concepts

Concept learning involves determining a mapping from a set of input variables to a Boolean value. The methods described here are known as inductive-learning methods. These methods are based on the principle that if a function is found that correctly maps a large set of training data to classifications, then it will also correctly map unseen data. In doing so, a learner is able to generalize from a set of training data (Coppin, 2004).

Version Spaces

Given a set of training examples (positive and negative), the set of hypotheses that correctly map each of the training examples to its classification is called the version space. One method for learning from a set of data is thus to start from a complete version space that contains all hypotheses and systematically remove all the hypotheses that do not correctly classify each training example. Although this method might work well on small problems, for problems of any reasonable size, the task of enumerating all hypotheses would be impractical (Coppin, 2004).

Candidate Elimination

The aim of these methods is to identify a single hypothesis, if possible, that correctly describes the problem. The more training data that are available, the fewer hypotheses are contained in the version space. If all the training data have been used, and the version space contains just a single hypothesis, then this matches all the training data and should also match unseen data (Coppin, 2004).

2.8 Machine Learning Algorithms

Supervised learning is perhaps the most frequently used mining/learning technique in both practical data mining and Web mining. It is also called classification, which aims to learn a classification function (called a classifier) from data that are labeled with pre-defined classes or categories. The resulting classifier is then applied to classify future data instances into these classes. Due to the fact that the data instances used for learning (called the training data) are labeled with pre-defined classes, the method is called supervised learning (Liu, 2007).

Classifiers built by means of ML technology achieve impressive levels of effectiveness, making automatic classification a qualitatively (and not only economically) viable alternative to manual classification (Sebastiani, 2002). Text categorization, which assigns one or more predefined categories to one new document based on the contents of the document, is a very efficient method for managing the vast amount of data. There exist numerous sophisticated algorithms that have been applied to text categorization (Feng, Zuo, & Wang, 2015). The following section describes some common machine learning algorithms.

2.8.1 Decision Tree Induction

A decision tree (DT) is a tree structure, and can also be said to be a flow chart. DT will return an appropriate label based on the value of a given input in this flow chart. The decision tree has decision nodes and leaf nodes. The decision nodes mainly check the features of the input value and the leaf nodes match label for each input value according to its features. To choose a label for an input value, the flowchart begins at the initial decision node known as root node of a decision tree. This node contains a condition that is used to check the one of input value's features and selects a branch according to that feature's value. Following the branch that describes the input value, it arrives at a new decision node with a new condition. Then this flow path goes on until it arrives at a leaf node which will provide a label for the input value (Yaduang, Wenlong, Aina, & Yuqing, 2015).

Decision trees classify the documents based on feature values. Each node in a decision tree represents a feature of a document in an instance to be classified, and each branch represents a value that the node can assume. Instances are classified starting at the root node and sorted based on their feature values of the documents. Each leaf represents a unique classification of documents. The attribute that best divides the training data would be the root node of the tree. There are numerous methods for finding the feature that best divides the training data such as information gain and gini index. The main advantage of the decision tree based classification method is its ability to generate easy to interpret rules (Murty, Murthy, Prasad Reddy, & Satapathy, 2012).

Decision tree learning is one of the most widely used techniques for classification. Its classification accuracy is competitive with other learning methods, and it is very efficient. The learned classification model is represented as a tree, called a decision tree. An interesting and important property of a decision tree and its resulting set of rules is that the tree paths or the rules are mutually exclusive and exhaustive. This means that every data instance is covered by a single rule (a tree path) and a single rule only. By covering a data instance, the instance satisfies the conditions of the rule. Also, a decision tree generalizes the data as a tree is a smaller (more compact) description of the data, i.e. it captures the key regularities in the data (Liu, 2007).

Decision tree induction is the learning of decision trees from class-labeled training tuples. A decision tree is a flowchart-like tree structure, where each internal node (nonleaf node) denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (or terminal node) holds a class label. The topmost node in a tree is the root node. Given a tuple, X, for which the associated class label is unknown, the attribute values of the tuple are tested against the decision tree. A path is traced from the root to a leaf node, which holds the class prediction for that tuple. Decision trees can easily be converted to classification rules (Cios et. al, 2007).

Decision tree is so popular. The construction of decision tree classifiers does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high dimensional data. Their representation of acquired knowledge in tree form is intuitive and generally easy to assimilate by humans. The learning and classification steps of decision tree induction are simple and fast. In general, decision tree classifiers have good accuracy. However, successful use may depend on the data at hand. Decision tree induction algorithms have been used for classification in many application areas, such as medicine, manufacturing and production, financial analysis, astronomy, and molecular biology. Decision trees are the basis of several commercial rule induction systems (Cios et. al, 2007). Below is an algorithm for Decision tree algorithm from Han and Kamber (2006).

Algorithm: Generate_decision_tree. Generate a decision tree from the training tuples of the data partition D.

Input:

- Data partition D which is a set of training tuples and their associated class labels;
- *attribute_list*, the set of candidate attributes;
- *Attribute_selection_method*, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of *splitting_attribute* and, possibly, either a *split point* or *splitting subset.*

Output: A decision tree

Method:

```
1. create a node $N$;
2. **if** tuples in $D$ are of the same class, $C$ **then**
3.       return $N$ as a leaf node labeled with the class $C$;
4. if *attribute_list* is empty **then**
5.       return $N$ as a leaf node labeled with the majority class in $D$; // majority voting
```

```
6. apply **Attribute_selection_ method**($D$, attribute list) to find the "best" *splitting_criterion;*
7. label node $N$ with *splitting_criterion;*
8. if *splitting_attribute* is *discretevalued* **and** multiway splits allowed **then** // not restricted to binary trees
9.     attribute_list ← attribute_list - splitting_attribute; // remove splitting attribute
10. **for each** outcome $j$ of splitting_criterion // partition the tuples and grow subtrees for each partition
11.     let $D_j$ be the set of data tuples in $D$ satisfying outcome $j$; // a partition
12.     **if** $D_j$ is empty **then**
13.         attach a leaf labeled with the majority class in $D$ to node $N$
14.     **else** attach the node returned by Generate decision tree $D_j$, attribute list to node $N$; **endfor**
15. return $N$;
```

The strategy is as follows:

- The algorithm is called with three parameters: D, attribute list, and Attribute selection method. D is referred to as a data partition. Initially, it is the complete set of training tuples and their associated class labels. The parameter *attribute_list* is a list of attributes describing the tuples. *Attribute_selection* method specifies a heuristic procedure for selecting the attribute that "best" discriminates the given tuples according to class. This procedure employs an attribute selection measure, such as information gain or the gini index. Whether the tree is strictly binary is generally driven by the attribute selection measure. Some attribute selection measures, such as the gini index, enforce the resulting tree to be binary. Others, like information gain, do not, therein allowing multiway splits (i.e., two or more branches to be grown from a node).

- The tree starts as a single node, N, representing the training tuples in D (step 1).

- If the tuples in D are all of the same class, then node N becomes a leaf and is labeled with that class (steps 2 and 3). Note that steps 4 and 5 are terminating conditions. All of the terminating conditions are explained at the end of the algorithm.

- Otherwise, the algorithm calls *Attribute_selection_method* to determine the splitting criterion. The splitting criterion tells us which attribute to test at node N by determining the "best" way to separate or partition the tuples in D into individual classes (step 6). The splitting criterion also tells us which branches to grow from node N with respect to the outcomes of the chosen test. More specifically, the splitting criterion indicates the splitting attribute and may also indicate either a split-point or a splitting subset. The splitting criterion is determined so that, ideally, the resulting partitions at each branch are as "pure" as possible. A partition is pure if all of the tuples in it belong to the same class. In other words, splitting up the tuples in D according to the mutually exclusive outcomes of the splitting criterion, should result in partitions that are as pure as possible.

- The node N is labeled with the splitting criterion, which serves as a test at the node (step 7). A branch is grown from node N for each of the outcomes of the splitting criterion. The tuples in D are partitioned accordingly (steps 10 to 11).

- The algorithm uses the same process recursively to form a decision tree for the tuples at each resulting partition, D_j , of D (step 14).

- The recursive partitioning stops only when any one of the following terminating conditions is true:

    1. All of the tuples in partition D (represented at node N) belong to the same class (steps 2 and 3), or
    2. There are no remaining attributes on which the tuples may be further partitioned (step 4). In this case, majority voting is employed (step 5). This involves converting node N into a leaf and labeling it with the most common class in D. Alternatively, the class distribution of the node tuples may be stored.
    3. There are no tuples for a given branch, that is, a partition D_j is empty (step 12).In this case, a leaf is created with the majority class in D (step 13).

- The resulting decision tree is returned (step 15).

According to Yaduang *et al.* (2015), different text data has different characteristics of categorization. So it is particularly critical to choose a decision trump for a text. A decision stump is a decision tree with a single node that decides how to classify inputs based on a single feature. It is better to choose the labels whose features are more obvious, frequent and representative to establish decision stumps. Entropy and information gain are good to determine the most informative features for decision stumps. Entropy is defined as the sum of the probability of each label times the log probability of that same label as expressed in Equation 2.8.

$$H = \sum_{i=labels} P(i) \times log_2 P(i) \tag{2.8}$$

If most input values have the same label, then entropy is low. In particular, labels that have low frequency do not contribute much to the entropy, and labels with high frequency also do not contribute much to the entropy. On the other hand, if the input values have a wide variety of labels, then there are many labels with a "medium" frequency, where neither $P(i)$ nor $log_2 P(i)$ is small, so the entropy is high. Information gain is the change of entropy before and after data partitioning. The higher the Information gain is, the better the decision stump will be. Therefore, traversing all the characteristics, the highest information gain feature will be selected as the feature of decision node (Yaduang *et al.*, 2015).

Decision trees are easy to understand and rules can be easily generated through them. They can solve complex problems very easily. But training through decision trees is very expensive and is very costly. Moreover, a sentiment class can only be connected to one branch only. A single mistake in higher upper level can cause a whole sub tree invalid. Issues of continuous variables as well as over fitting are there in decision trees (Mahmood *et al.*, 2011, Podgorelec *et al*, 2002, Korde and Mahender, 2012, as cited in Rana, Shehzad, & Muhammad, 2014).

2.8.2 Naive Bayesian Classification

The principle of naive Bayes (NB) classification algorithm is: for the given content to be classified, calculate the emergence probability of each label under the conditions of the content features appearing, and then take the label with largest probability as this content's label. For text classification, it is to see the text feature words. The probability whose features word appears relatively large will match the characteristics corresponding label (Yaduang *et al.*, 2015).

Supervised learning can be naturally studied from a probabilistic point of view. The task of classification can be regarded as estimating the class posterior probabilities given a test example d, expressed in Equation 2.8.

$$Pr(C = c_j|d). \tag{2.8}$$

Then the class c_j which is more probable is identified. The class with the highest probability is assigned to the example d (Liu, 2007).

According to (Liu, 2007), formally, let $A_1, A_2, ..., A_{|A|}$ be the set of attributes with discrete values in the data set D. Let C be the class attribute with $|C|$ values, $c_1, c_2, ..., c_{|C|}$. Given a test example d with observed attribute values a_1 through $a_{|A|}$, where a_i is a possible value of A_i, i.e.,

$$d =< A_1 = a_1, ..., A_{|A|} = a_{|A|} > .$$

The prediction is the class c_j such that $Pr(C = c_j|A_1 = a_1, ..., A_{|A|} = a_{|A|})$ is maximal. c_j is called a maximum a posteriori (MAP) hypothesis.
By Bayes' rule, Equation 2.8 can be expanded to Equation 2.8.

$$\begin{aligned} &Pr(C = c_j|A_1 = a_1, ..., A_{|A|} = a_{|A|}) \\ &\quad = \frac{Pr(A_1 = a_1, ..., A_{|A|} = a_{|A|})Pr(C = c_j)}{Pr(A_1 = a_1, ..., A_{|A|} = a_{|A|})} \\ &\quad = \frac{Pr(A_1 = a_1, ..., A_{|A|} = a_{|A|}|C = c_j)Pr(C = c_j)}{\sum_{k=1}^{|C|} Pr(A_1 = a_1, ..., A_{|A|} = a_{|A|}|C = c_k)Pr(C = c_k)}. \end{aligned} \tag{2.8}$$

$Pr(C = c_j)$ is the class prior probability of c_j, which can be estimated from the training data. It is simply a fraction of the data in D with c_j. If the only interest is in making a classification, $Pr(A_1 = a_1, ..., A_{|A|} = a_{|A|})$ is irrelevant for decision making because it is the same for every class. Thus, only $Pr(A_1 = a_1 \wedge ... \wedge A_{|A|} = a_{|A|}|C = c_j)$ needs to be computed, which can be written as Equation 2.8.

$$\begin{aligned} &Pr(A_1 = a_1, ..., A_{|A|} = a_{|A|}|C = c_j) \\ &= Pr(A_1 = a_1|A_2 = a_2, ..., A_{|A|} = a_{|A|}, C = c_j) \times Pr(A_2 = a_2, ..., A_{|A|} = a_{|A|}|C = c_j). \end{aligned} \tag{2.8}$$

Recursively, the second term above (i.e., $Pr(A_2 = a_2, ..., A_{|A|} = a_{|A|}|C = c_j)$)

can be written in the same way (i.e. $Pr(A_2 = a_2|A_3 = a_3, ..., A_{|A|} = a_{|A|}, C = c_j) \times Pr(A_3 = a_3), ..., A_{|A|} = a_{|A|}|C = c_j)$, and so on. However, to further the derivation, conditional independence assumption is made. The assumption is that all attributes are conditionally independent given the class $C = c_j$. Formally, it is assumed,

$$Pr(A_1 = a_1|A_2 = a_2, ..., A_{|A|} = a_{|A|}, C = c_j) = Pr(A_1 = a_1|C = c_j) \qquad (2.8)$$

and similarly for A_2 through $A_{|A|}$. We obtain

$$Pr(A_1 = a_1, ..., A_{|A|} = a_{|A|}|C = c_j) = \prod_{i=1}^{|A|} Pr(A_i = a_i|C = c_j) \qquad (2.8)$$

$$Pr(C = c_j|A_1 = a_1, ..., A_{|A|} = a_{|A|}) = \frac{Pr(C = c_j) \prod_{i=1}^{|A|} Pr(A_i = a_i|C = c_j)}{\sum_{k=1}^{|A|} Pr(A_i = a_i|C = c_k)}. \qquad (2.8)$$

Next, the *prior* probabilities $Pr(C = c_j)$ and the conditional probabilities $Pr(A_i = a_i|C = c_j)$ need to be estimated from the training data,

$$Pr(C = c_j) = \frac{number \quad of \quad examples \quad of \quad class \quad c_j}{total \quad number \quad of \quad examples \quad in \quad the \quad data \quad set} \qquad (2.8)$$

$$Pr(A_i = a_i|C = c_j) = \frac{number \quad of \quad examples \quad with \quad A_i = a_i \quad and \quad class \quad c_j}{number \quad of \quad examles \quad of \quad class \quad c_j} \qquad (2.8)$$

If only a decision on the most probable class for each test instance is needed, only the numerator of Equation 2.8 is needed since the denominator is the same for every class. Thus, given a test case, the Equation 2.8 is used to decide the most probable class for the test case (Liu, 2007).

$$c = \arg\max_{c_j} Pr(C = c_j) \prod_{i=1}^{|A|} Pr(A_i = a_i | C = c_j) \tag{2.8}$$

Owing to scarcity in training data, unseen terms in the document can cause "zero probability problems" in Naïve Bayes assumption. To avoid this, probability mass of existing terms is redistributed and unseen terms take extra probability mass. As a result, this process is called smoothing (Kilimci & Ganiz, 2015).

Bayes classifier is the most powerful technique for discrete text classification. Bayes classifier works on the probability principle. As discrete text has less semantic relation, therefore it focuses on the occurrences of the keywords. It uses statistical method for classifying the text (Yadav & Parne, 2015).

There are two first order probabilistic models for Naive Bayes classification, both of which make the assumption that features are independent of each other. One is the Bernoulli model and other is the Multinomial model. The Bernoulli model is a Bayesian Network with no word dependencies and binary word features whereas the Multinomial model is a unigram language model with integer word counts. Multinomial model is used when the frequency of a word occurring in a document counts. Hence, a binarized version of the Multinomial version is used which only takes in to account the presence of a word and not its frequency (Sagar, Avit, Uchit, & Meera, 2014).

The Bernoulli model on the other hand generates a Boolean indicator for each term of the vocabulary depending upon its presence or absence. Thus, the Bernoulli model also takes words that do not appear in the document into account. It is found that the multivariate Bernoulli performs well with small vocabulary sizes, but the multinomial model usually performs even better at larger vocabulary sizes, providing on an average 27% reduction in error over the multivariate Bernoulli model at any vocabulary size (Sagar *et al*, 2014).

Multinominal Naive Bayes (MNB) is a well-known classifier, a simple and easy to implement method, and still giving very good results (Baldwin & Lui, 2010, as cited in van Dam & Zaytsev, 2016). MNB finds the probability of a document d belonging to a certain class c by using Equation 2.8 and Equation 2.8.

$$P(c|d) = P(c) \prod_{i=1}^{n} P(x_i|c) \tag{2.8}$$

where

$$P(x_i|c) = \frac{freq(x_i, c) + 1}{\sum\limits_{x \in V} freq(w_{i-n+1}^{i-1})} \tag{2.8}$$

where n is the number of all the features in the document; $freq$ gives the frequency of x in class c; V is the vocabulary of all classes (van Dam & Zaytsev, 2016).

Some advantages of Naive Bayes are, NB requires a small amount of training data to estimate the parameters necessary for classification. Also, NB algorithm affords fast, highly scalable model building and scoring, and can be used for both binary and multiclass classification problems (Al-Ghuribi & Alshomrani, 2013).

Unigram Naive Bayes

For unigram Naive Bayes, the probability of a term belonging to a class is given as the empirical counts of that term in messages with same class. In multinomial model, the probability is given by Equation 2.8:

$$P(t_k|C) = \frac{T_{ct_k}}{|V_c|} \tag{2.8}$$

Where T_{ct_k} is the number of times the term is associated with the class and V_c is the total number of terms seen for the class. In contrast to the multinomial model, the Bernoulli multivariate model deals with the number of documents containing the term for that class divided by the total number of documents for the class. The binarized variation of the multinomial model clips the word count in each document as one (Sagar *et al.*, 2014).

Bigram Naive Bayes

The bigram Naive Bayes classifier calculates the probability that a document belongs to a class on the basis of the number of times word pairs are seen for the class. But since the training set becomes sparse, linear interpolation as well as back off model can be used. The linear interpolation weighs the unigram as well as bigram probabilities to calculate the overall probability of the document using Equation 2.8.

$$P(c|d) = wP_{unigram}(c|d) + (1-w)P_{bigram}(c|d) \tag{2.8}$$

The back off model uses the bigram probability if its seen with the class or else backs off to the unigram probability (Sagar *et al.*, 2014).

One of the main reasons that make NB model works well for text domain is because the evidences are "vocabularies" or "words" appearing in texts and the size of the vocabularies is typically in the range of thousands. The large size of evidences (or vocabularies) makes NB model work well for text classification problem (Mitchell, 1996, Makoto & Takenobu, 1994, & Yang & Liu, 1999, as cited in Swamy *et al.*, 2014).

Good thing about Naive bayes is that it works best with textual as well as numeric data formats and is easy to implement and to compute. But on the other hand, it shows bad performance when features are correlated like short texts or news headlines classifications. Moreover, in real world data, conditional independence assumption is poorly violated (Andreas, 2005, Ting and Tsang, 2011, Korde and Mahender, 2012 as cited in Rana *et al.*, 2014).

2.8.3 Support Vector Machines

Support vector machines (SVM) is another type of learning system, which has many desirable qualities that make it one of the most popular algorithms. It not only has a solid theoretical foundation, but also performs classification more accurately than most other algorithms in many applications, especially those applications involving very high dimensional data. It has been shown by several researchers that SVM is perhaps the most accurate algorithm for text classification. It is also widely used in Web page classification and bioinformatics applications (Liu, 2007).

Lui (2007) described SVM as a linear learning system that builds two-class classifiers. Let the set of training examples D be

$$\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\},$$

where $x_i = (x_{i1}, x_{i2}, ..., x_{ir})$ is a $r-$dimensional input vector in a real-valued space $X \subseteq \Re^r$, y_i is its class label (output value) and $y_i \in \{1, -1\}$. 1 denotes the positive class and -1 denotes the negative class.
To build a classifier, SVM finds a linear hyperplane of the form in Equation 2.8

$$f(x) = \langle w \cdot x \rangle + b \tag{2.8}$$

so that an input vector x_i is assigned to the positive class if $f(x_i) \geq 0$, and to the negative class otherwise as expressed in Equation 2.8.

$$y_i = \begin{cases} 1 \quad if \langle w \cdot x_i \rangle + b \geq 0 \\ -1 \quad if \langle w \cdot x_i \rangle + b < 0 \end{cases} \tag{2.8}$$

Hence, $f(x)$ is a real-valued function $f : X \subseteq \Re^r \to \Re$. $w = (w_1, w_2, ..., w_r) \in \Re^r$ is called the weight vector. $b \in \Re$ is called the bias. $\langle w \cdot x \rangle$ is the dot product of w and x (or Euclidean inner product). Without using vector notation, Equation

2.8 can be written as:

$$f(x_1, x_2, ..., x_r) = w_1x_1 + w_2x_2 + ... + w_rx_r + b,$$

where x_i is the variable representing the i^{th} coordinate of the vector $\mathbf{x}$. In essence, SVM finds a hyperplane represented in Equation 2.8.

$$\langle w \cdot x \rangle + b = 0 \tag{2.8}$$

that separates positive and negative training examples. This hyperplane is called the decision boundary or decision space. Geometrically, the hyperplane $\langle w \cdot x \rangle + b = 0$ divides the input space into two half spaces: one half for positive examples and the other half for negative examples. A hyperplane is commonly called a line in a 2-dimensional space and a plane in a 3-dimensional space (Liu, 2007).

The support vector machine (SVM) is a highly accurate classification method. However, SVM classifiers suffer from slow processing when training with a large set of data tuples (Han & Kamber, 2006).

Support vector machines are high margin classifiers. The main idea underlying SVM for sentiment classification is to find a hyper plane which divide the documents, and the margin between the classes being as high as possible. SVMs are based on the Structural Risk Minimization principle. The objective is to find a hypothesis h for which the true error is the lowest. The true error can be defined as the probability that the h will make an error in categorizing unseen data or a randomly selected test sample. Feature selection is an important task in machine learning techniques. There are many features that have to be taken into account for text classification, to avoid over fitting and to increase general accuracy (Sagar *et al.*, 2014).

SVMs have the potential to handle large feature spaces with high number of dimensions. Also, the learning ability of a SVM is independent of the dimension of the feature space. SVMs measure the complexity of the hypothesis with which they separate the documents and not the number of features. As long as the text classification problem is linearly separable, the number of features in the feature space is not one of the issues (Sagar *et al.*, 2014).

According to Sagar *et al.* (2014), in order to deal with a large number of features, traditional text categorization methods assume that some of the features are irrelevant. But even the lowest ranked features according to feature selection methods contain considerable information. Considering these features as irrelevant often result in a loss of information. Since SVMs do not require such an assumption, information loss can be reduced. Although SVM outperforms all the traditional methods for sentiment classification, it is a black box method. It is difficult to investigate the nature of classification and to identify which words are more important for classification. This is one of the few disadvantages of using SVM as a method for document classification.

Support Vector Machines have been applied successfully in many text classification tasks because of their principle advantages as follows: robust in high dimensional spaces, in which over fitting does not affect so much the computation of the final decision margin; any feature is important even some features that could be considered as irrelevant ones have been found to be good when calculating the margin; robust when there is a sparsely of samples and most text categorization problems are linearly separable. Additionally, SVM method is flexible and can easily be combined with interactive user feedback methods (Al-Ghuribi & Alshomrani, 2013).

SVM can handle millions of inputs with good performance. SVM is a highly robust technique that has provided powerful classification capabilities for online authorship analysis. In head-to-head comparisons, SVM significantly outperformed other supervised learning methods such as NN and C4.5 (Suh, 2016).

Since SVM has the strong learning ability and is able to capture the inherent characteristics of the data, high classification efficiency results. Therefore less training samples can be used to get a trained classifier with high performance, so choice of SVM classifier for text classification is ideal (Wu & Xu, 2015).

The support vector machine is a statistical classification method proposed by Vapnik . Based on the structural risk minimization principle from the computational learning theory, SVM seeks a decision surface to separate the training data points into two classes and makes decisions based on the support vectors that are selected as the only effective elements in the training set. Multiple variants of SVM have been developed in which Multi class SVM is used for Sentiment classification (Kaiquan Xu, 2011, as cited in Vinodhini & Chandrasekaran, 2012).

2.8.4 K-Nearest Neighbor Learning

The main idea of K-Nearest Neighbor (KNN) classification algorithm is: there is a data set that has been categorized. When adding some new data into it, K nearest data from the new data should be found in the data set; and then observing which category most of these k data belong to, then putting the new data in that category (Yaduang *et al.*, 2015).

The k-nearest-neighbor method was first described in the early 1950s. The method is labor intensive when given large training sets, and did not gain popularity until the 1960s when increased computing power became available. It has since been widely used in the area of pattern recognition (Han & Kamber, 2006).

Decision Trees, Naive Bayes, Support Vector Machines learning methods learn some kinds of models from the data, e.g., decision trees, sets of rules, posterior probabilities, and hyperplanes. These learning methods are often called eager learning methods as they learn from models of the data before testing. In contrast, k-nearest neighbor (kNN) is a lazy learning method in the sense that no model is learned from the training data. Learning only occurs when a test example needs to be classified. The idea of kNN is extremely simple and yet quite effective in many applications, e.g., text classification (Liu, 2007).

According to Lui (2007), KNN works as follows: Let D be the training data set. Nothing will be done on the training examples. When a test instance d is presented, the algorithm compares d with every training example in D to compute the similarity or distance between them. The k most similar (closest) examples in D are then selected. This set of examples is called the k nearest neighbors of d. d then takes the most frequent class among the k nearest neighbors. $k = 1$ is usually not sufficient for determining the class of d due to noise and outliers in the data. A set of nearest neighbors is needed to accurately decide the class. The general kNN algorithm is given in Figure 2.8.

```
Algorithm $k$NN$(D, d, k)$
   1. Compute the distance between $d$ and every example in $D$;
   2. Choose the $k$ examples in $D$ that are nearest to $d$, denote the set by $P(\subseteq D)$;
   3. Assign $d$ the class that is the most frequent class in $P$ (or the majority
      class).
```

Figure 2.8: The k-nearest neighbor algorithm

The key component of a kNN algorithm is the distance/similarity function, which is chosen based on applications and the nature of the data. For relational data, the Euclidean distance is commonly used. For text documents, cosine similarity is a popular choice. The number of nearest neighbors k is usually determined by using a validation set, or through cross validation on the training data. That is, a range of k values are tried, and the k value that gives the best accuracy on the validation set (or cross validation) is selected (Liu, 2007).

Nearest-neighbor classifiers use distance-based comparisons that intrinsically assign equal weight to each attribute. They therefore can suffer from poor accuracy when given noisy or irrelevant attributes. The method, however, has been modified to incorporate attribute weighting and the pruning of noisy data tuples.

Nearest-neighbor classifiers can be extremely slow when classifying test tuples (Han & Kamber, 2006).

The K-nearest neighbour classifications are based on learning by correlation. The KNN technique starts by selecting first arbitrary data points (text documents) as initial seed clusters from the vector space model. Then, it enters a learning phase when training documents are iteratively assigned to a cluster whose center is located at the nearest distance based on Euclidean distance, Cosine similarity, KullbackLiebler-distance measure. The training samples are described by n-dimensional numeric attributes. Each document sample represents a pair in an n-dimensional space. In the document vector the numeric values indicates word frequencies. Cluster centers are repeatedly adjusted to the mean of their currently acquired data points (Murty *et al.*, 2012).

When given unknown samples that are close to the centroid. Similarity is defined in terms of Euclidean distance between two documents P and Q. The classification algorithm tries to find the K-nearest neighbour of a test documents and uses a majority vote to determine its class label. The presentation of KNN classifier is mainly resolute by an appropriate selection of K which can be quite complicated if either the data is non-uniformly distributed or if there is noisy data, and depends on the distance metric applied. The value of K may need to be tuned for a given application. The major potency of the KNN algorithm is that it presents excellent generalized accuracy on many areas and the learning phase is fast. But it is slow during instance classification because all the training instances have to be visited. The accuracy of the KNN algorithm degrades with increase of noise in training data (Murty *et al.*, 2012).

The K-nearest neighbor (KNN) is a typical example based classifier that does not build an explicit, declarative representation of the category, but relies on the category labels attached to the training documents similar to the test document. Given a test document d, the system finds the k nearest neighbors among training documents. The similarity score of each nearest neighbor document to the test document is used as the weight of the classes of the neighbor document (Songho, 2008, as cited in Vinodhini & Chandrasekaran, 2012).

In Figure 2.8, supposing the small dot in the middle need to be classified. If selecting 3 nearest neighbors, that is $k = 3$, there are two triangles and a square can be found in the 3 nearest shapes from the dot. So the dot should be classified in the triangles according to the definition of K-Nearest Neighbor classification. Similarly, if $k = 5$, the dot should be classified in the squares. So the value of k determines the category of the dot. There are other factors such as the position of the dot, the way to calculating the distance and so on. When this classification algorithm is used for text categorization, the best parameters should be tested in the classifier to make the effect of categorization best (Yaduang *et al.*, 2015).

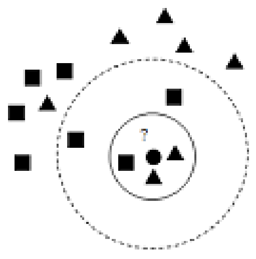


Figure 2.8: Example of KNN Model

Sagar *et al.* (2014) found that KNN works best among the conventional methods for text classification and SVMs are a better option independent of the choice of parameters.

Al-Ghuribi and Alshomrani (2013) emphasized that the main thesis in KNN is that documents which belong to the same class are likely to be close to one another based on similarity measures such as the dot product or the cosine metric. KNN assumes that the data is in a feature space, since the points are in feature space, they have a notion of distance. Also it assumes that each of the training data consists of a set of vectors and class label associated with each vector. A single number “k” is given; this number decides how many neighbors influence the classification.

Although, KNN classifier is a very simple classifier that works well on basic recognition problems, it has a set of drawbacks. KNN is a lazy learning example-based method that does not have an off-line training phase. The main computation is the on-line scoring of training documents given a test document in order to find the k nearest neighbors, this makes KNN not efficient because nearly all computation takes place at classification time rather than when the training examples are first encountered. Moreover, KNN classifier has a major drawback of selecting the value of k; the success of classification is very much dependent on this value (Al-Ghuribi & Alshomrani, 2013).

K Nearest neighbor method has been applied widely in the existing text classification methods. It is a nonparametric classification techniques. It is very effective on pattern recognition based on the statistics. It can achieve higher classification accuracy for the unknown and non-normal distribution. It’s advantages are its robust, and its concept is very clear (Aizhang & Tao, 2015).

But KNN also has some obvious disadvantages: First, when it identifies the category of text which is to be classified, it needs to calculate the similarity of all the samples in the training sample sets. So with the increase of training samples, classification performance will drop soon. Secondly, KNN algorithm must specify the k value. Currently, there are no better methods to determine the neighbors number for text classification. How to select k value is important to category identification. If k is too big or too small, this will affect the accuracy of text classification. Third, When KNN algorithm calculate text similarity, it doesn’t consider the relationship between characteristic words in texts (Aizhang & Tao, 2015).

Because time complexity and k value of the traditional KNN text classification method is hard to determine, in order to solve this question, Aizhang

& Tao (2015) proposed a KNN text classification algorithm based on rough set and correlation analysis. Their experiments showed that rough sets can improve the efficiency and accuracy of text classification. When KNN text classification algorithm is to determine category of text which is to be classified, it needs to calculate similarity of all training sample sets. Then, they choose k samples whose similarity is higher of the text to be necessary to calculate the similarity with the training sample set of all samples. When dealing with high-dimensional text vector or large text data, with increase of the number of training samples, the time and space complexity will also increase, and text classification efficiency will decrease (Aizhang & Tao, 2015).

Aizhang and Tao (2015) used Web crawler to get the finance, technology, education, cars and real estate in five categories of news from sina news. They were used as samples for their experiment, each category were grabbed 2500 news. A total of 12,500 documents were used, where each class took 2000, a total of 10,000 as training set; the remaining 500 in each category, a total of 2500 is as a test set. From their results, they found that KNN text classification algorithm based on rough sets and association analysis and KNN text classification algorithm based on association analysis have higher accuracy in science and technology, real estate and economics, comparing with two other algorithms. Data observed indicated that the classification efficiency and improving the efficiency of KNN text classification algorithm based on rough sets and associated analysis is superior to the rough-based KNN text classification algorithm text classification algorithm based association analysis, and it has the absolute advantage effectively.

2.8.5 Neural Networks

Neural networks (NN) can be built to perform text categorization. Usually, the input nodes of the network receive the future values, the output nodes produce the categorization status values, and the link weights represent dependence relations. For classifying a document, its feature weights are loaded into the input nodes; the activation of the nodes is propagated forward through the network, and the final values on output nodes determine the categorization decisions (Feldman & Sanger, 2007).

The neural networks are trained by *back propagation*, where by the training documents are loaded into the input nodes. If a misclassification occurs, the error is propagated back through the network, modifying the link weights in order to minimize the error. The simplest kind of a neural network is a perceptron. It has only two layers - the input and the output nodes. Such network is equivalent to a linear classifier. More complex networks contain one or more *hidden* layers between the input and output layers. However, experiments have shown very small or no improvement of nonlinear networks over their linear counterparts in the text categorization task (Feldman & Sanger, 2007).

Zaghoul and Al-Dhaheri (2013) experimented on an Arabic corpus. Results demonstrated that the Artificial Neural Networks (ANN) model is effective in representing and classifying Arabic documents. The results indicated that ANN model using features reduction methods is more able to capture the non-linear relationships between the input document vectors and the document categories than that of basic ANN model. Also their results showed that back propagation learning in neural networks was able to give good categorization performance as measured by precision and recall. However, comparing performance of ANN model with other learning algorithms would have given a better performance evaluation.

NN has been popular because of its unique learning capability and has achieved good performance in many different applications (Suh, 2016). Artificial neural networks produce good results on complex domains and allow researchers to perform fast testing. But on the other hand, training through neural networks is very slow. Learned results of neural networks are highly complex for users to translate or interpret as compared to learned rules (Andreas, 2005, Korde and

Mahender, 2012 as cited in Rana *et al.*, 2014).

2.8.6 Association Rule Analysis

Frequent patterns and their corresponding association or correlation rules characterize interesting relationships between attribute conditions and class labels, and thus have been recently used for effective classification. Association rules show strong associations between attribute-value pairs (or items) that occur frequently in a given data set. Association rules are commonly used to analyze the purchasing patterns of customers in a store. Such analysis is useful in many decision-making processes, such as product placement, catalog design, and cross-marketing (Han & Kamber, 2006).

The discovery of association rules is based on frequent itemset mining. The general idea is that strong associations between frequent patterns (conjunctions of attribute-value pairs) and class labels are searched. Because association rules explore highly confident associations among multiple attributes, this approach may overcome some constraints introduced by decision-tree induction, which considers only one attribute at a time (Han & Kamber, 2006).

2.8.7 Case Based Reasoning

Case-based reasoning (CBR) classifiers use a database of problem solutions to solve new problems. Unlike nearest-neighbor classifiers, which store training tuples as points in Euclidean space, CBR stores the tuples or “cases” for problem solving as complex symbolic descriptions. Business applications of CBR include problem resolution for customer service help desks, where cases describe product-related diagnostic problems. CBR has also been applied to areas such as engineering and law, where cases are either technical designs or legal rulings, respectively. Medical education is another area for CBR, where patient case histories and treatments are used to help diagnose and treat new patients (Han & Kamber, 2006).

When given a new case to classify, a case-based reasoner will first check if an identical training case exists. If one is found, then the accompanying solution to that case is returned. If no identical case is found, then the case-based reasoner will search for training cases having components that are similar to those of the new case. Conceptually, these training cases may be considered as neighbors of the new case. If cases are represented as graphs, this involves searching for subgraphs that are similar to subgraphs within the new case (Han & Kamber, 2006).

The case-based reasoner tries to combine the solutions of the neighboring training cases in order to propose a solution for the new case. If incompatibilities arise with the individual solutions, then backtracking to search for other solutions may be necessary. The case-based reasoner may employ background knowledge and problem-solving strategies in order to propose a feasible combined solution (Han & Kamber, 2006).

2.8.8 Maximum entropy classification

Maximum entropy classification classifies the uncategorized content based on some known content classification information. The principle of the maximum entropy is that when some pieces of information are known about the unknown content, the probability distributions that meet the information content and have maximum entropy should be chosen (Wu, Fu, Sui, & Ding, 2015).

Entropy represents the randomness of thing happened. The randomness and uncertainty of events is positively correlated with the change of entropy. Maximum Entropy classification analyzes unknown content based on the categorization of known information content uncertainly and randomly. So it can make content classification more equitable (Wu *et al.*, 2015).

Text categorization in natural language processing is to match text data with labels depending on the features of the text. Text categorization of Maximum Entropy is the same. First, the maximum entropy classifier should be trained based on the text content categorized by a human. During the training, the classifier must constantly adjust parameters to make the classification effect best. When training on text corpus, the model of maximum entropy classifier can be described in Equation 2.8, f stands for features and l means label (Wu *et al.*, 2015).

$$P(f) = \sum_{x \in corpus} P(l(x)|f(x)) \tag{2.8}$$

Where $P(l|f)$, the probability that an input whose features are features will have class label, is calculated using Equation 2.8.

$$P(l|f) = \frac{P(l, f)}{\sum_l P(l, f)} \tag{2.8}$$

Wu *et al.* (2015) explained that there is no way to calculate the model parameters directly and improve the likelihood of the training set best because there is potential impact of the complex interactions between the relevant features. This is why Maximum Entropy classifiers choose the model parameters using iterative optimization techniques, which initialize the model's parameters to random values, and then repeatedly refine those parameters to make them closer to the best practices.

2.8.9 Random Forests

It is well known that basic categorizer has some limitations and hence cannot be efficiently applied. To improve these limitations, the Integrated Learning (IL) has been developed and used. By combining multiple basic categorizers and analyzing the result of each basic categorizer, IL is able to determine the category of target sample with improved accuracy and can effectively improve the generalization ability of basic categorizer. However, not any combination of more than one basic categorizers can be called integrated learning. Two requirements need to be satisfied for IL: 1) each basic categorizer has to be valid; 2) employed basic categorizers are different from each other (Dashen & Fengxin, 2015).

Random Forests (RF) is an integrated learning model proposed by the American scientist Leo Breiman. It is based on the K decision tree $\{h(X, \theta_k,$ k = 1,2,...,K$\}$ as a basic categorizer. The categorization result of the RF is decided by each decision tree through a simple voting way. The $\{h(X, \theta_k,$ k = 1,2,...,K$\}$ is a random sequence and determined by the following two random ideas : Bagging ideas and Feature subspace ideas (Dong & Huang, 2013, as cited in Dashen & Fengxin, 2015).

Bagging Ideas

Assuming that the original sample set has a capacity of M, Bagging ideas is to randomly select samples from the original sample set as the single decision tree training set $\{T_m, m = 1, 2, ..., M\}$.

Feature Subspace ideas

In the process of constructing a decision tree, when a node is split, an attribute subset has to be first extracted from the all the nodes attributes. Then, an optimal attribute is selected from the subset to split the node. In the process of constructing a decision tree, both training set and attribute subset are selected randomly. Hence $\{\theta_k, k = 1, 2, ..., K\}$ is a random sequence and makes RF have very strong generalization ability (Dong & Huang, 2013, as cited in Dashen & Fengxin, 2015). Figure 2.8 is an illustration of the construction process of a decision tree.

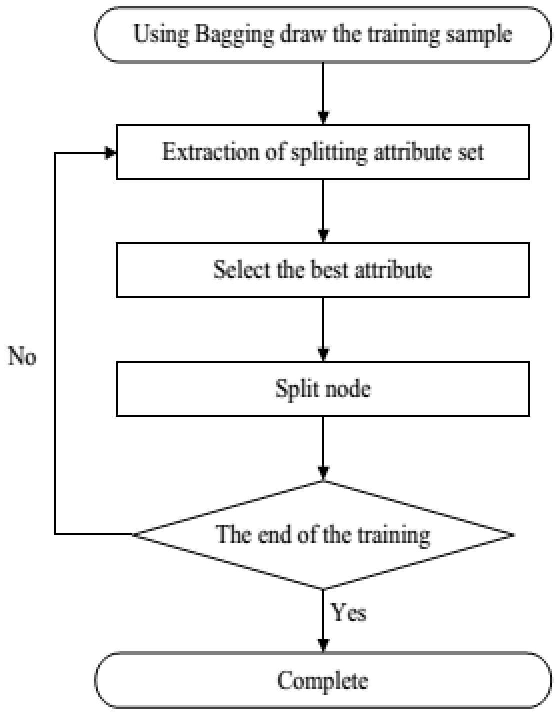

Using Bagging draw the training sample
Extraction of splitting attribute set
Select the best attribute
Split node
The end of the training
No
Yes
Complete

Figure 2.8: The Construction Process of a Decision Tree

Language Models as Text Classifiers

Text classifiers attempt to identify attributes which distinguish documents in different categories. Such attributes may include vocabulary terms, word average length, local N-grams, or global syntactic and semantic properties. Language models also attempt capture such regularities, and hence provide another natural avenue to constructing text classifiers (Farhoodi *et al.*, 2011).

An N-gram language model can be applied to text classification in a similar manner to a naïve Bayes model. In this case Equation 2.8, Equation 2.8 and Equation 2.10 are used.

$$c^* = \underset{c \in C}{\operatorname{argmax}}\{P(c|d)\} = \underset{c \in C}{\operatorname{argmax}}\{P(d|c)P(c)\} \tag{2.8}$$

$$= \underset{c \in C}{\operatorname{argmax}} \{P(d|c)\} \tag{2.8}$$

$$= \underset{c \in C}{\operatorname{argmax}} \{\prod_{i=1}^{T} P(w_i|w_{i-n+1}...w_{i-1})\} \tag{2.8}$$

Using random variables D and C to denote the document and category values respectively. A probabilistic text classifier is formulated as the following decision problem: given a document d, determine the class label c^* that yields the highest posterior probability that yields the highest posterior probability $P(C = c|D = d)$, written $P(c|d)$ for simplicity, using Equation 2.8 (Farhoodi *et al.*, 2011).

$$c^* = \underset{c \in C}{\operatorname{argmax}} \{P(c|d)\} \tag{2.8}$$

The principle for using an N-gram language model as a text classifier is to determine the category that makes a given document most likely to have been generated by the category model. Thus, a separate language model is trained for each category, and a new document is classified by evaluating its likelihood under each category, choosing the category according to Equation 2.8 (Farhoodi *et al.*, 2011).

A pure multinomial naïve Bayes is a special case of N-gram based text classifier where $N = 1$ and add-one smoothing is used. However, n-gram modeling based approach considers more context information and deals with unobserved attributes with back-off approach, coupled with better smoothing techniques than add-one smoothing (Farhoodi *et al.*, 2011).

2.9 Evaluating Performance of Text Classifiers

After a classifier is constructed, it needs to be evaluated for its performance. Appropriate evaluation is crucial because without knowing the approximate accuracy of a classifier, it cannot be used in real-world tasks. There are many ways to evaluate a classifier, and there are also many measures. The main measure is the classification accuracy, which is the number of correctly classified instances in the test set divided by the total number of instances in the test set (Liu, 2007).

Because the text categorization problem is not sufficiently well-defined, the performance of classifiers can be evaluated only experimentally. Any TC experiment requires a document collection labeled with a set of categories. This collection is divided into two parts: the training and test document sets. When there is a need to optimize some classifier parameters experimentally, the training set is further divided into two parts – the training set proper and a validation set, which is used for the parameter optimizations (Feldman & Sanger, 2007).

A commonly used method to smooth out the variations in the corpus is the n-fold cross-validation. In this method, the whole document collection is divided into n equal parts, and then the training-and-testing process is run n times, each time using a different part of the collection as the test set. Then the results for n folds are averaged. The most common performance measures are the classic IR measures of recall and precision (Feldman & Sanger, 2007).

2.9.1 Accuracy

The accuracy of a classifier on a given test set is the percentage of test set tuples that are correctly classified by the classifier. The associated class label of each test tuple is compared with the learned classifier's class prediction for that tuple. If the accuracy of the classifier is considered acceptable, the classifier can be used to classify future data tuples for which the class label is not known. Such data are also referred to in the machine learning literature as "unknown" or "previously unseen" data (Cios *et al.*, 2007).

If the training set is used to measure the accuracy of the classifier, this estimate would likely be optimistic, because the classifier tends to overfit the data i.e., during learning it may incorporate some particular anomalies of the training data that are not present in the general data set overall. The accuracy of a classifier refers to the ability of a given classifier to correctly predict the class label of new or previously unseen data (i.e., tuples without class label information). Similarly, the accuracy of a predictor refers to how well a given predictor can guess the value of the predicted attribute for new or previously unseen data (Cios *et al.*, 2007).

2.9.2 Recall

A recall for a category is defined as the percentage of correctly classified documents among all documents belonging to that category i.e. measure of completeness (Swamy *et al.*, 2014).

In the context of IR, Witten and Frank (2005) defined recall for a category as the percentage of correctly classified documents among all documents belonging to that category. Accuracy is calculated using Equation 2.9.

$$Recall = \frac{number \quad of \quad documents \quad retrieved \quad that \quad are \quad relevant}{total \quad number \quad of \quad documents \quad that \quad are \quad relevant} \tag{2.9}$$

2.9.3 Precision

Precision is the percentage of correctly classified documents among all documents that were assigned to the category by the classifier (Witten & Frank, 2005). According to Swamy *et al.* (2014) precision is the percentage of correctly classified documents among all documents that were assigned to the category by the classifier i.e. measure of exactness. Precision is calculated using Equation

2.9.

$$Precision = \frac{number \quad of \quad documents \quad retrieved \quad that \quad are \quad relevant}{total \quad number \quad of \quad documents \quad that \quad are \quad retrieved} \tag{2.9}$$

2.9.4 F-Measure

This is calculated as the harmonic mean of recall/sensitivity and precision (Cios *et al*, 2007) as expressed in Equation 2.9.

$$F - Measure = \frac{(2 \times Precision \times Recall)}{(Precision + Recall)} \tag{2.9}$$

The harmonic mean discourages classifiers that sacrifice one measure for another too drastically (Chakrabarti, 2003). Among the four standard measures, accuracy assesses the overall classification correctness, while the others evaluate the correctness regarding each class (Suh, 2016).

2.10 Machine Learning Algorithms and Text Classification

Much research has been done in sentiment classification using machine learning. Algorithms used in Machine Learning are commonly used in data mining for classification, prediction, association rule mining, and detection (Anahita, Jamilu, & Azuraliza, 2014).

Kristin and Richard (2012) conducted research on one of the central problems in the new field of social media analytics: deciding whether a given document, such as a blog post or forum thread, expresses positive or negative opinion toward a particular topic. They presented two new methods of inferring sentiment orientation of social media content which addresses these challenges. Each method formulated the task as one of text classification, models the data as bipartite graph of documents and words, and assumed that only limited prior information is available regarding the sentiment orientation of any of the documents or words of interest. This procedure enabled accurate sentiment classifiers to be learned.

Kaiser and Bodendorf (2012) carried out research to mine and analyze opinion formation on the basis of consumer dialogs in online forums. They found that combining text mining and social network analysis enables the study of opinion formation and yields encouraging results. Naïve Bayes (NB), K-Nearest Neighbour (k-NN), Decision trees (C4.5) and support Vector Machines (SVM) text mining algorithms were used in the research. SVM yielded the best results for all three tasks, followed by decision trees and Naïve Bayes, and lastly K-nearest neighbours.

Hamouda and Akaichi (2013) conducted a comparative experimental procedure between the Naïve Bayes and the SVM algorithms by combining different feature extractors performed on Tunisian users statuses on Facebook posts during the Arabic Spring era. Their conclusion was that those algorithms can achieve high accuracy for classifying sentiment when combining different features.

Dalal and Zaveri (2013) conducted an experiment on automatic classification of unstructured blog posts using a semi-supervised machine learning approach. Empirical studies indicated that the multi-step classification strategy outlined can classify blog text with good accuracy. They confirmed that the combination of Term Frequency and Inverse Document Frequency (TF-IDF) and multi-word heuristics is an effective statistical feature-set extractor for blog entries. Their empirical results indicated that the naïve Bayesian classification model clearly outperforms the basic Artificial Neural Networks (ANN) based classification model for highly domain-dependent unstructured blog text classification especially when a restricted feature-set is available.

Caropeso *et al.* (2001) experimented with N-grams for text categorization on the Reuters data set. They defined an N-gram as an alphabetically ordered sequence of n stems of consecutive words in a sentence (after stop words were removed). The authors use both unigrams and bigrams as document features. They extracted the top-scored features using various feature selection methods including Mutual Information. Their results indicated that in general bigrams can better predict categories than unigrams. However, despite the fact that bigrams represent the majority of the top-scored features, the use of bigrams does not yield significant improvement of the categorization results while using the Rocchio classifier.

Pelemans *et al.* (2014) examined several combinations of classical N-gram language models with more advanced and well known techniques based on word similarity such as cache models and Latent Semantic Analysis. They compared the efficiency of these combined models to a model that combines N-grams with the recently proposed, state-of-the-art neural network-based continuous skip-gram. They discussed the strengths and weaknesses of each of these models, based on their predictive power of Dutch language and found that a linear interpolation of a 3-gram, a cache model and a continuous skip-gram is capable of reducing perplexity by up to 18.63%, compared to a 3-gram baseline. This was three times the reduction achieved with a 5-gram.

Warmer *et al.* (2012) analyzed hate speech sentiments used in United States of America. They described pilot classification experiments in which they classified anti-semitic speech reaching accuracy of 94%, precision of 68% and recall at 60%, for an F1 measure of 0.6375 using SVM. Their experiments showed that that unigram feature sets out performed the full set, with the smallest feature set, comprised of only positive unigrams, performing the best. In terms of f-measure, their best classifier equalled the performance of their volunteer annotators. However, bigram and trigram templates degraded the performance of the classifier. The concluded that the learning phase of the classifier is sensitive to features that ought to cancel each other out.

Khreisat (2006) analyzed the results of classifying Arabic text documents using the N-gram frequency statistics technique employing a dissimilarity measure called the Manhattan distance, and Dices measure of similarity. The Dice measure was used for comparison purposes. Results showed that N-gram text classification using the Dice measure outperforms classification using the Manhattan measure.

Kulkarni, Stranieri, Kulkarni, Ugon and Mittal (2014) carried out research on visual character N-grams for classification and retrieval of radiological images. Their experiment showed that character N-grams were useful in classifying in classifying regions into normal and abnormal categories. Promising classification accuracies were observed (83.33%) for fatty background.

Bouras and Tsogkas (2016) used a corpus consisting of 10,000 news articles obtained from major news portals like BBC, CNN, etc. over a period of 5 months. Those articles were evenly shared among the eight base categories featured by our system in order to avoid any bias towards a specific class. Firstly, they tried to determine the best value for n, i.e. up to what size should the words window be when capturing grams. For this experiment, they arbitrarily set $A = B = 0.5$ (i.e. giving the same weighting significance to both keywords and N-grams) and tried different values of n where $2 \leq n \leq 6$. For each n value, they repeated the clustering process 10 times with different starting cluster centers. The results depicted showed that when $n = 3$, i.e. they kept both 2- and 3-g for weighting the sentences, the W-kmeans algorithm's performance is increased by an average of 0.3 regarding the clustering index of the generated clusters. Increasing even further the window size seemed to have a negative impact on the overall clustering index results.

Bouras and Tsogkas (2016) explained that larger window sizes mean that n-grams that randomly appear together in larger sequences are weighted much more than they should, a situation that probably has a negative impact on the overall weighting, given their random nature. Another interesting finding of this experiment was that for $n = 2$, the results get slightly worse than when not using n-grams at all. They explained that a possible cause for that might be statistical anomalies in the underlying data, or even poor extraction for n-grams given the library that they used. Extracting, thus, in some cases 2-g that are not really good candidates has less to the observed result, which was later repaired when using both 2 and 3-grams for the weighting process.

Yasotha and Charles (2015) proposed a new approach to automatically categorize text documents. Instead of the common practice of string matching approach used for automatic text document categorization this paper proposes a Latent Dirichl*et al*location (LDA) based approach. The clusters identified by LDA are labeled based on the underlying natural clusters on the domain in concern. Proposed model was able to categorize unseen documents with an accuracy of 66.66%. The lack of accuracy is due to the limitation of classification on one level.

Kadmateekarun and Nuanmeesri (2015) used Naïve Bayes text classification model to classify the 150 video samples. Their research used Recall, Precision, and F-measure to assess the efficiency of computing and accuracy of the analysis processing. When running the test in classifying the 150 video samples, they applied Naive Bayes classifier on different video sample sizes; 40, 50, and 60 respectively. The result of sentiment analysis was satisfactory because the average results had Recall value at 0.833, Precision value at 0.877 and F-measure value at 0.841. Table 2.10 shows the results they obtained.

Table 2.10: Kadmateekarun and Nuanmeesri Experimental Results

Model Types	**Recall**	**Precision**	**F-Measure**
40 video samples per group	0.82	0.85	0.827
50 video samples per group	0.83	0.88	0.836
60 video samples per group	0.85	0.90	0.859
Average	0.833	0.877	0.841

The results of the comparison sample of video in each group are different; the higher the number of video samples, the better the direction (Kadmateekarun and Nuanmeesri, 2015). However, apart from the increase in the number of video samples, there is no explanation on the increase in classification accuracy. Also, a performance comparison with other machine learning algorithms was not evaluated with respect to the same dataset.

Vispute and Potey (2013) categorized Marathi text documents using Lingo Clustering algorithm based on Vector space Model (VSM). The data set consisted of 107 Marathi documents of 3 different categories- Tourism, Health Programmes and Maharashtra festivals. Recall, Precision and F-measure were calculated for each category. Tourism data set was tested for queries such as "Shivaji","Kolhapur", "Sant Sai Baba", "Shani Shingnapur" and "S. S.Deshmukh", the average precision obtained for such queries was 75.53% and average Recall was 90%.

Vispute and Potey (2013) also tested Health programme data set for queries such as "Aarogya", "Aarogya niyatran", "Niyantran", "Rashtriya Karyakram", average precision obtained is 87.68% and average recall is 100%. Maharashtra Festival data set is tested for queries such as "Utsav", "Dasara", "Diwali", "Ramayana", "Raksha bandhan" and average precision obtained was 96.52% and average recall was 99%. They concluded that LINGO algorithm has given total average of 91.10% for the Marathi documents. Thus it shows that the LINGO clustering algorithm for categorizing Marathi documents works well and its performance is better in categorizing the Marathi documents. However, other classification algorithms were not used with the data set to give an objective comparative analysis on which algorithm is better.

Suh (2016) proposed an automatic approach that adopts the writing styles of users as objective features for estimating user reputations in social media. Supervised techniques were adopted because they have been extensively studied due to their predominant classification performance. Four base learners, i.e. C4.5, NN, SVM, and NB were adopted for automatic classification. Web forum was used for data collection because it is a major type of social media with a balanced nature of discussions among participants and a relatively broader range of topics.

Suh (2016) carried out experiments to compare writing syle feature-based classification methods for estimating user reputations in social media. Test bed users were segmented into Good or Bad reputation classes by proposed four approaches: like, dislike, sum, and portfolio. Moreover, four writing style features were extracted, i.e. lexical (denoted by F1), syntactic (denoted by F2), semantic (denoted by F3), and content specific (denoted by F4), to represent the reputations of the test bed users.

Next, Suh (2016) evaluated the classification performance of 32 configurations, resulted by combining four feature sets, i.e. F1, F1 + F2, F1 + F2 + F3, and F1 + F2 + F3 + F4. Eight classification techniques, i.e. four base learners (C4.5, NN, SVM, and NB) and four Random Subspace (RS) ensemble methods were used based on the four base learners, with respect to accuracy for a given way of defining the classes of the test bed user reputations. In addition, classification performances of different feature sets and different classification techniques were statistically compared by conducting pairwise t tests.

For the experiment, randomly 100 users for each reputation class were selected from the collected data, and a tenfold validation was performed to train a classifier and evaluate it (Suh, 2016).

Wu *et al.* (2015) carried out experiments to compare the performance of Decision Tree Classifier, Naive Bayes Classifier, Maximum Entropy Classifier and K- nearest neighbor classifiers on 3000 text of movie reviews. Two thirds of the reviews was taken as training set and the rest as test set. These words were relatively more representative and characteristic. In the experiments, another important part was to make a feature extractor to get the features of the text. The classifiers were then trained, applied and evaluated.

The accuracy of Naïve Bayes classifier on movie reviews was the best of the four. The average accuracy of NB was more than 80% and this classifier was relatively stable. The K-Nearest neighbor classifier was better and its average accuracy was about 73%. But its stability was not as good as Naïve Bayes classifier. The stability of Decision Tree classifier was the best, however, its average accuracy is about 65%. In this experiment on movie reviews, the accuracy and stability of the Maximum Entropy classifier were the worst (Wu *et al.*, 2015).

Aldy and Ayu (2015) conducted experiments on emotion classification of Indonesian Twitter text. To conduct the experiments, the researchers built a corpus of labeled Twitter data with size of 7622 Twitter text taken from 69 Twitter accounts, manually labeled by 5 native speakers. Researchers used 6 basic emotion labels (angry, disgust, fear, joy, sad, surprise) and added one label of neutral emotion class. The researchers compared a rule based method with a statistical based method. There were several experimental scenarios in statistical based method. Every part of the experiment was completed with Support Vector Machine (SVM) algorithm. First, the experiment was conducted to determine the type of features. There were three types of features: unigram, bigram and trigram. The best result was gained by unigram feature which reached an accuracy of 63.47% for SVM. This showed that unigram feature greatly affected emotion classification. The research concluded that Imbalanced data is a problem that decreases classification accuracy.

Deepak and Narsimha (2012) used 20 newsgroup dataset, LingSpam dataset and cade12 dataset to evaluate the performance of their proposed method. LingSpam dataset is a two class data in which the class labels are Spam and Genuine. The researchers used only the main body of the mail in their corpus without using any header. The researchers used the sorted by-date version of the 20 newsgroup data, where training (60%) and testing (40%) data are separated in time. This gave the researchers more realistic scenario for classification. Cade12 is a 12 class text data. Since the researchers wanted to extract phrases from the documents, they did not stem the words. The researchers used only training data to find phrases and used these phrases for classification of the test data. Naive Bayes Classifier(NBC) and Support Vector Machine(SVM) were selected for experimentation. LibSVM with a linear kernel was used in the experiments.

Experimentation results in Table 2.10 show that for well separated classes like Spam v/s Genuine, Graphics v/s Hockey, classifiers are performing well even with the Bag of words model, the proposed Bag of phrases model was able to achieve a significant relative improvement. But when classes are not well separated like Atheism v/s Christian or Internet v/s Informatica, classifiers were not performing very well with the Bag of words model, but when using the Bag of phrases model, inclusion of phrases outperformed the conventional Bag of words model. For some datasets, researchers were getting lesser accuracies with Bag of phrases model for NBC, because the number of occurrence of a phrase in a document is very less than that of a word. When such phrases are added to the vector space, in NBC their contribution towards the classification would be noisy (Deepak & Narsimha, 2012).

Table 2.10: Classification accuracies(%) of the Bag of Phrases(BOP) and the Bag of words (BOW) models

Data set	**Classifier**	**BOW**	**BOP**
LingSpam	SVM	98.15	98.5
	NBC	97.92	98.3
Athiesm-Christian	SVM	70.29	77.4059
	NBC	72.52	73.9
Graphics-Hockey	SVM	96.19	96.7005
	NBC	99.74	99.74
Internet-Informatica	SVM	78.8165	79.4597
	NBC	80.79	80.57

Deepak and Narsimha (2012) also conducted experiments for the comparison between the vector space containing all bi-grams which appeared in the corpus and vector space generated by the Bag of Phrases model. Researchers presented the idea of using all 1-gram and 2-grams into the Bag of words model. It was observed that considering all possible bi-grams consumes a lot of space because it almost doubles the dimension and does not give better accuracy (73.9% for Atheism-Christian) than the proposed Bag of phrases model (77.4%).

Agarwal *et al.* (2013) proposed to enhance the performance of unigram features by incorporating more sentiment-rich information in the form of bi-tagged phrases with the unigrams. These bi-tagged phrases were subjective in nature, which is very important for sentiment classification. In their proposed method, sentiment-rich phrases were initially extracted using fixed patterns based on part of speech (POS). The vectors generated using such features were quite large and included features that were not useful for classification. These noisy and irrelevant features were eliminated using an information gain (IG) feature selection method. Features extracted using IG method are called prominent features. Next, the optimized feature vector was constructed using prominent unigrams and bi-tagged phrases. Finally, the optimized feature vector was exploited by machine learning methods for sentiment classification.

Prominent features extracted from unigrams, bi-grams and bi-tagged phrase were named as PromUni (prominent unigrams), PromBi (prominent bi-grams) and PromBiTa (prominent bi-tagged) features respectively. Literature showed that the performance of unigram features increased when combined with bi-grams, but at higher computational cost. Unigrams with additional information from bi-grams and bi-tagged features performed better than individual unigram features (Agarwal *et al.*, 2013).

Composite features were created using unigram with bi-grams and unigrams with bi-tagged features namely ComUniBi and ComUniBiTa, respectively. Finally, PromUniBi feature set was created by combining the prominent unigrams (PromUni) and prominent bi-grams (PromBi). Similarly, PromUniBiTa feature set was created by combining prominent unigrams (PromUni) and prominent bitagged features (PromBiTa). To evaluate the performance of supervised sentiment analysis using sentiment-rich and unigram features, the Cornell Movie Review Dataset was used. This is a popular publicly available movie review dataset contains 1000 positive reviews and 1000 negative reviews about movies (Agarwal *et al.*, 2013). The results of their experiments are shown in Table 2.10.

Table 2.10: F-Measure for various features with SVM and NB

Features	**SVM**	**NB**	**Feature Size**
Unigram	84.2%	79.4%	9045
Bi-gram	78.8%	73.5%	6050
Bi-tagged Phrases	75.3%	71.8%	4841
ComUniBi	86.7%	81.1%	15095
ComUniBiTa	87.6%	82.3%	13886
PromUni	85.8%	85.4%	1130
PromBiTa	86.5%	73.7%	1114
PromUniBiTa	89.4%	86.2%	2244

Unigram features without any feature selection method was considered as baseline performance. Experimental results showed that unigram features individually perform better than other features like bi-gram and bi-tagged features for both SVM and NB. For example, unigram produces F-measure 84.2%, which is significantly higher than 78.8% and 75.3% for bi-grams and bi-tagged features for the SVM classifier, respectively. Composite features ComUniBi features perform better than unigram features, but with increased computational cost. This was due to the fact that bi-grams features incorporate some contextual information, which is important for sentiment classification (Agarwal *et al.*, 2013).

Total bi-gram features were very large in number; therefore, document frequency was used for initial pruning of unimportant features. Sometimes important sentiment–rich bi-grams are eliminated due to low document frequency. Even then, bi-gram generally contain a very large number of noisy features. In contrast, bi-tagged phrases are sentiment-rich and, hence, very important for sentiment classification. Although bi-tagged features were a subset of bi-gram features, they achieved better performance as compared to bigram features (Agarwal *et al.*, 2013).

Agarwal *et al.* (2013), explained that the main reason for this was that, by using rules for extracting features, most of the irrelevant and noisy features are discarded. In addition, while sentiment-rich bi-gram with low frequency were removed with simple bi-grams, with bi-tagged phrases, all the important features are selected. The proposed bi-tagged phrases extraction method can be considered as a feature selection method that extracts subjective phrases from bi-grams by eliminating noisy and irrelevant bi-gram features.

Swamy *et al.* (2015) used and evaluated k Nearest Neighbor (KNN), Naïve bayes and Decision tree C4.5(J48) algorithms to classify south Indian languages such as Kannada, Tamil and Telugu. The researchers used a corpus of their own; the corpus consisted of 300 documents that belonged to 3 categories. All the documents were preprocessed by removing stop words and light stemming all the tokens. The documents were represented using the vector space model. For measuring the effectiveness of classification algorithm, the researchers used the traditional recall and precision measures. The results illustrated that kNN had 93% accuracy, Decision tree C4.5 had 97.33% and Naïve Bayes had 97.66% accuracy.

Swamy *et al.* (2015) concluded that very satisfactory results had been achieved. The researchers confirmed that text mining algorithm can also applicable for Indian language for categorization and Naïve Bayes is efficient algorithm for Indian language text categorization.

Dashen and Fengxin (2015) carried out text categorization experiment on Sogou laboratory text categorization corpus as benchmark corpus including five categories: IT, economics, health, tourism and sports. The size of training corpus was 2636, and the size of testing corpus was 6224. Bigram method was used as the word segmentation method. This the researchers explained was because the language of Sogou laboratory text classification corpus is simplified Chinese, and about 70% is a two-character words in Chinese according to relevant statistics. So by using Bigram method, the complexity of the algorithm can be reduced and good categorization effect can be achieved.

Zhang and Wu (2015) trained NB classifier and evaluated the classifier using the test set. The researchers got their data set from Sina Weibo, which contained 84896 micro-blogs in total, and covers 11 categories, which are labeled as "Cartoon", "Fashion", "Food", "Finance", "Health", "Movie", "Music", "Pet", "Science". "Sports", and "Travel". The data set was then split into two parts, the train set, which accounted for about 80%, and the test set which accounted for the rest 20%.

Research results of Zhang and Wu (2015) showed that their method for feature extension indeed was able to bring about a striking effect to short text classification. In some categories, their method showed an amazing improvement, and the F1 score increases 0.15 in these categories. And in general, the macro F1 score reaches 0.85, more than 0.1 higher than the macro F1 score without feature extension. Feature extension using word co-occurrence also improves the F1 score in some categories, but the efficiency was not as obvious as their method. However, the researchers did not use trigrams or 4-grams to compare the performance with the bigrams.

Pei and Wu (2014) performed text categorization research on Sogou Lab Data which has a large number of Chinese news data as their dataset. It has 9 labels: ‘auto’, ’house’, ’sport’, ’entertainment’, ’finance’, ’fashion’, ’technology’, ’military’, ’education’. The researchers randomly picked 5000 articles in each category as training data, and 3000 articles in rest data set as testing data. The researchers then built a training set by first segmenting all documents into words, thus they got a whole word-bag of training data, then they extracted keywords from raw training data that have highest chi-value and finally representing each document as a TF-IDF weighted feature vector, each dimension of the vector represented a keyword. The testing set was built by represent each document as a TF-IDF weighted feature vector.

After training set and testing set had been built successfully, Pei and Wu (2014) used KNN classification algorithm and SMO classification algorithm to classify data. They compared classification performance of two algorithms. To measure classifier performance, the researchers used six metrics: classification precision, classification recall, classification F1-measure, mean absolute error, mean root error and time cost. The performance results are presented in Table 2.10.

Table 2.10: Comparing Results of Different Classifiers

	Algorithm	
	KNN	SMO
Dimension	500	500
Precision	84.7%	87.4%
Recall	84.7%	87.4%
F1-measure	85.5%	88.1%
Mean absolute error	0.0343	0.1745
Mean root error	0.1824	0.2840
Time cost	66s	180s

Genereux *et al.* (2011) extracted single words as unigram and used some feature selection methods and two features weighting methods (binary and term frequency) and applied a pool of features from financial and health metaphor to obtain the best result of 67.6% (as cited in Foroozan, Murad, Sharef, & Latiff, 2015).

Another study was performed by Alvim *et al.* (2010), which focused on the combination of various feature extraction methods (especially bigram) along with SVM to achieve a high accuracy of 86.09% (as cited in Foroozan *et al.*, 2015).

Koppel *et al.* (2006) were one of the first to use machine leaning methods to determine the sentiment orientation of financial news as good or bad. The extracted feature set used by them was very simple. They extracted single words that appeared only 60 times in the corpus and delivered an accuracy of 70.3%. The researchers also showed that classifiers such as SVM, Naïve Bayes (NB) and Decision Tree (DT) and their kernels had the same result. Therefore, it shows that only a proper feature extraction can support a high accuracy (as cited in Foroozan *et al.*, 2015).

Existing studies on financial news classification relied on simple features, such as unigram or bigram model. However, unigram or bigram features alone cannot solely carry the meaning of desired text due to creation of sparse matrix of terms. Most of prior researches solely relied on bag-of-word approach while, this method hardly capture any semantic of the context (Foroozan *et al.*, 2015). The researchers therefore performed experiments to measure the classification accuracy of support vector machines (SVM) with two kernel methods of linear and Radial Basis Function (RBF). Results showed that an efficient feature extraction increased classification accuracy when it was used as a combination of unigram and bigram. Moreover, they also found that DF can be applied as a dimension reduction method to reduce the feature space without loss of accuracy.

Foroozan *et al.* (2015) employed experiments using unigram, bigram, and the combination of unigram and bigram extracted from their content as feature extraction. Unigram covered some adverbs, adjectives, verbs, and nouns that appeared in the text. In addition, three types of feature weighting methods (binary, TF, and TF-IDF) with document frequency (DF) approach were used to represent each news file in the form of vector. Their experiments set covered multiple feature spaces with different imbalanced dataset.

For their experiments, Foroozan *et al.* (2015) extracted Lexical features from 1917 sentiment labeled financial news files, which included 757 negative news files and 1160 positive news files. There were 28,668 unigrams without the stopwords and 20,480 unigrams with the applied stem method in order to remove the stopwords. Due to the limitation of memory and generally computing resources, only the most frequent features with sparsity between 95% and 99% (frequency range) were considered. Randomly, 33% (639) of test set and 67% (1278) of train set were assigned. Results were obtained by running the SVM with linear and RBF kernels and both are compared against each other.
Figure 2.10 and Figure 2.10 illustrate the classification accuracy using linear SVM and RBF SVM with different feature types. Both figures showed the best performance is obtained when using the combination of unigram and bigram along

with TF-IDF

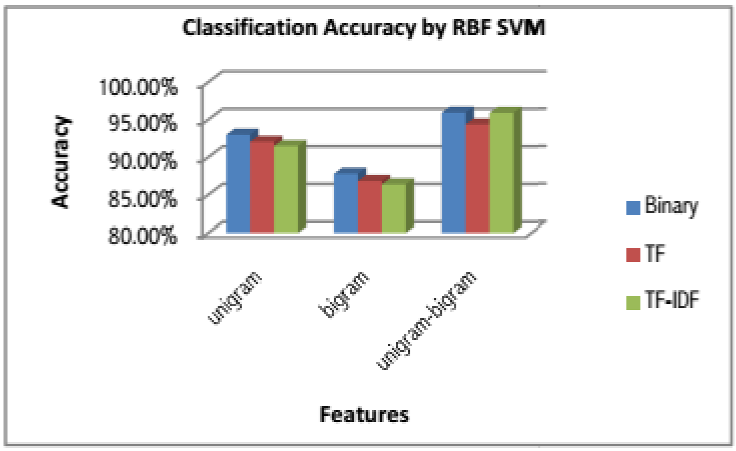


Figure 2.10: Average Classification accuracy by RBF SVM for different feature types

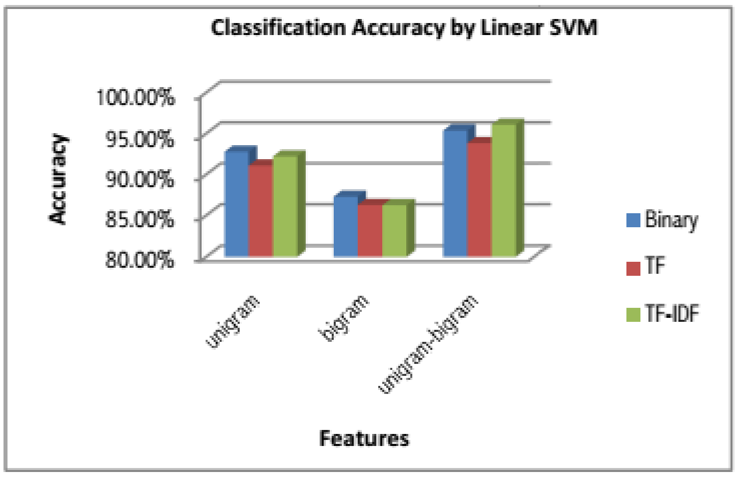


Figure 2.10: Average Classification accuracy by Linear SVM for different feature types

Foroozan *et al.* (2015) focused on classification accuracy where accuracy was measured as a percentage of correctly classified financial news. They found several remarkable results from their experiments. First, the combination feature extraction methods (unigram and bigram) boosts classification accuracy (97.03%).

Second, document frequency (DF) method can be used as a dimensional reduction method because it reduces the number of features while improving the classificatin accuracy.

Beliga and Martincic-Ipsic (2014) conducted experiments on categorization of Croatian texts using Non-Standard Words (NSW) as features. NonStandard Words are: numbers, dates, acronyms, abbreviations, currency, etc. NSWs in Croatian language are determined according to Croatian NSW taxonomy. For the purpose of this research, 390 text documents were collected and formed the SKIPEZ collection with 6 classes: official, literary, informative, popular, educational and scientific. Text categorization experiment was conducted on three different representations of the SKIPEZ collection: in the first representation, the frequencies of NSWs are used as features; in the second representation, the statistic measures of NSWs (variance, coefficient of variation, standard deviation, etc.) were used as features; while the third representation combines the first two feature sets.

Beliga and Martincic-Ipsic (2014) used Naive Bayes, CN2, C4.5, kNN, Classification Trees and Random Forest algorithms in text categorization experiments. The best categorization results are achieved using the first feature set (NSW frequencies) with the categorization accuracy of 87%. They suggested that the NSWs should be considered as features in highly inflectional languages, such as Croatian. NSW based features reduce the dimensionality of the feature space without standard lemmatization procedures, and therefore the bag-of-NSWs should be considered for further Croatian texts categorization experiments.

Beliga and Martincic-Ipsic (2014) conducted experimets on categorization of Croatian texts using Non-Standard Words (NSW) as features. For the purposes of normalization and categorization tasks, the researchers prepared the collection of texts in Croatian language – SKIPEZ (Sluzbeno, Knjizevno, Informativo, Popularno, Edukativno, Znanstveno). SKIPEZ contains 390 texts organized in 6 predefined classes: official, literature, informative, popular, educational and scientific. The distribution of text over classes was balanced so each class contained exactly 65 texts. Texts were selected according to their relevance to the class and according to the % of contained NSWs, because the researchers needed a balance between both aspects in order to train a classifier.

The content of texts in each subclass was carefully selected, in order to capture as many different and representative text in each class used and contain representative set of NSWs, in order to compile the exhausted list of Croatian NSWs. The first representation of SKIPEZ collection was a feature set composed from NSW type frequencies. For each document the features vector contained 85 values (Beliga & Martincic-Ipsic, 2014).

The second representation of text, based on the NSW frequencies, represented the base from which the values of the features are derived. In this case, the statistics served as a tool for obtaining metadata from the previously calculated frequency of NSW occurrence. For each element of the population, therefore each vector, the researchers calculated some of its numerical characteristics called statistical features (Beliga *et al.*, 2013, Sarapa *et al.*, 1993, as cited in Beliga & Martincic-Ipsic, 2014). The main idea when choosing a feature vector was utilizing the basic statistical features, which can describe the dispersion of identified patterns of NSW in a particular text (Beliga & Martincic-Ipsic, 2014).

The third representation of texts was formed as a union from the features of the first and second representations, therefore included all their values. Through the combination of features, the researchers tried to take advantage of the knowledge that was previously extracted from all the texts, on the principle ”the more, the better” (Beliga & Martincic-Ipsic, 2014).

The results of trained classifiers (Naïve Bayes, Classification Tree, kNN, CN2, C4.5 and Random Fortest) were obtained for three different feature representations using 5 fold cross-validation. The text categorization results using the first feature set (NSW frequencies) are shown in Table 2.10; using statistical features shown in Table 2.10 and for combination of features in Table 2.10

Table 2.10: Results of Text Categorization where texts are represented by frequencies of NSWs

Classifier	**#tokens**	**Accuracy (%)**
Naïve Bayes	263	67.44
Classification Tree	305	78.21
kNN	296	75.90
CN2	306	78.46
C4.5	309	79.23
Random Forest	335	86.15
The mean score of all classifiers:		77.56

Table 2.10: Results of Text Categorization where texts are represented by Statistical features

Classifier	**#tokens**	**Accuracy (%)**
Naïve Bayes	176	45.13
Classification Tree	223	57.18
kNN	220	56.41
CN2	212	54.36
C4.5	229	58.72
Random Forest	260	67.95
The mean score of all classifiers:		56.62

Table 2.10: Results of Categorization where texts are represented by union of NSWs frequencies and Statistical features

Classifier	**#tokens**	**Accuracy (%)**
Naïve Bayes	176	62.05
Classification Tree	223	76.92
kNN	220	70.00
CN2	212	74.62
C4.5	229	78.97
Random Forest	260	86.92
The mean score of all classifiers:		74.91

Results of Beliga and Martincic-Ipsic (2014) showed that Random Forest achieved the highest accuracy and Naïve Bayes the lowest in all representation of the collection. The accuracy of all classifiers with statistical features was generally worse than the frequency features. With statistical features, Random Forest achieved a slightly higher accuracy (increase less than 1%). Numerous techniques of selection of features improve the performance of the classifier only marginally. This was confirmed also with their experiment, although the selection of features used non-standard features of the vector (Chakrabarti, Yang & Pedersen, 1997, as cited in Beliga & Martincic-Ipsic, 2014).

Farhoodi *et al.* (2011) studied the behavior of the N-Gram Frequency Statistics technique for classifying Persian text documents. The researchers used Hamshahri dataset. All documents, whether training documents or documents to be classified went through a preprocessing phase removing punctuation marks, stop words, diacritics, and non letters. For the training documents, the N-gram ($N = 1 - 4$) frequency profile was generated for each document and saved in text files. Then for each document to be classified, the N-gram frequency profile was generated and compared against the N-gram frequency profiles of all the training classes. Results showed that 3-gram text classification gives better classification results compared to the other n-grams.

Farhoodi *et al* (2011) employed language models classifier based on word level n-grams for Persian text classification. The dataset consisted of 9000 Persian documents of different lengths that belong to 9 categories. The categories were: Literature and Art, Social, Science and Culture, Miscellaneous, Politics, Sport, Natural Environment, Economy, Tourism. For each category, the researchers randomly selected about 1000 sample documents. By using an XML parser, the researchers saved each document of news in a separate text file. The researchers experimented to determine the influence of linguistic preprocessing, influence of the n-gram Order, and influence of smoothing techniques.

In order to investigating the effect of lingual preprocessing in classification performance, Farhoodi *et al.* (2011) got the accuracy without and with considering the preprocessing. The intention of linguistic preprocessing was normalizing, eliminating the stop words, tokenizing and stemming. Table 2.10 shows the results of classification accuracy without considering linguistic preprocessing and Table 2.10 shows classification accuracy results with considering linguistic preprocessing.

Table 2.10: Classification Accuracy without considering Linguistic Preprocessing

Number of Dataset Samples	**1-gram**	**2-gram**	**3-gram**	**4-gram**
1000	0.47	0.81	0.84	0.846
3000	0.49	0.84	0.85	0.85
5000	0.51	0.87	0.87	0.88
9000	0.61	0.90	0.92	0.925

Table 2.10: Classification Accuracy with considering Linguistic Preprocessing

Number of Dataset Samples	**1-gram**	**2-gram**	**3-gram**	**4-gram**
1000	0.42	0.87	0.91	0.92
3000	0.43	0.87	0.92	0.93
5000	0.54	0.89	0.96	0.96
9000	0.67	0.96	0.98	0.984

From these results, it shows that with linguistic preprocessing, there is an increase in classification accuracy performance as the number of samples in the dataset increases and increasing classification performance with the increase in n.

Farhoodi *et al.* (2011) stated that the order n is a key factor in n-gram language modeling. An order n that is too small will not capture sufficient information to accurately model word dependencies. On the other hand, a context n that is too large will create sparse data problems in training. In their experiments, they did not observe significant improvement when using higher order n-gram models ($n > 3$). They observed an immediate decrease in performance for the word level model, due to the early onset of sparse data problems. For solving this problem, they used smoothing methods, hence, the accuracy of results which are presented in Table 2.10 and Table 2.10 were calculated by applying the back-off smoothing. Also if more training data were available, the higher order models may begin to show an advantage. For example, in the larger dataset (average 1000 documents per class for training) they observed an obvious increase in classification performance with higher order models as presented in Table 2.10. However, the researchers emphasised that it is valuable to mention when n becomes too large, overfitting will begin to occur.

Hu, Manna and Rruong (2014) studied different classifiers for aspect identification from unlabeled free-form textual customer reviews about smart phones. Since their N-gram collection was processed in sentence level, they changed the contextual window size W to adjust the focus on position of N-grams. This value was set to 3 for their experiment after several repititive trials with the values. For the N-grams collected with such restriction, they called them Local N-grams. Once the contextual window size W was set to infinite, practically all the N-grams within the sentence was collected, which they called Global N-gram.

To evaluate the performance of the proposed approach, Hu *et al.* (2014) manually tagged 733 sentences and classified each sentence into either one of the selected aspects (camera, screen and battery). They concluded that feature selection is one of the important factors which affects the performance of a supervised machine learning algorithm to a great extent. For their analysis, they used N-gram information as features for the classifiers used. In order to achieve good performance of the classifiers, they investigated different N-gram information at different level i.e., unigram, bigram and trigram for both local and global N-grams.

From feature selection perspective, among different levels of N-gram, local unigram on average achieved the best performance, with highest accuracy of 79% for SVM and then 77% for RF; followed by global unigram which achieved 76% accuracy for SVM and 75% for RF. As for bigrams (for both local and global) the performances are relatively lower than local and global unigrams. However, for RF, local bigrams achieved better accuracy (66%) than global bigrams (60%). Local trigrams and global trigrams are apparently the worst choices for feature selection. None of them achieved 50% accuracy with any classifier (Hu *et al.*, 2014).

Hu *et al.* (2014) listed the following as their findings. They concluded that their result implied two things: first, people tend to use aspect related phrases close to the aspect words. It means when people explicitly mention aspect word in their sentences, they have high chance to use some specific descriptive or attribute phrases surrounding it. This explains why local unigrams and local N-gram could outperform global N-gram. Secondly, the length of the descriptive phrases normally ranges from 1 to 2. Three consecutive words as a single descriptive or attribute phrase seldom happens, therefore it is hard for local trigram to get decent performance.

From classifier perspective, Hu *et al.* (2014) noted that SVM achieved the highest accuracies 79% and 76% with local unigram and global unigram, respectively. However, in all other levels of N-gram, RF topped in 3 out of 4 cases: 66% with local bigram, 60% with global bigram and 39% with local trigram. The two Naive Bayes classifiers have similar performance but comparatively lesser than other two classifiers. However, it was noticeable that the improvement of one classifier comparing with other classifiers is not quite significant, especially comparing with the impact from selecting different level of N-grams, suggesting feature selection is more important than choosing classifier in text classification.

In their analysis, Hu *et al.* (2014) concluded that SVM outperformed other methods' performance for majority cases. RF also performed very close to SVMs results and relatively stable in cases for battery and screen aspects. While Naive Bayes classifiers performance fluctuated the most, suggesting the performance of pure statistical method might be quite domain specific.

2.11 Model Selection

One should choose a parsimonious model, one that uses the fewest number of parameters, among several acceptably well-performing models. There is always a model error associated with it. Model error is calculated as the difference between the observed/true value and the model output value, and is expressed either as an absolute or squared error between the observed and model output values. When a model of the data is generated, it is said that the model is fit to the data. However, in addition to fitting the model to the data, the model is also used for prediction. Thus, several models have to be generated and the "best" model selected, it is validated, not only for its goodness of fit(fit error), but also for its goodness of prediction (prediction error). In the neural network and machine learning literature, goodness of prediction is often referred to as the generalization error (Cios *et al*, 2007).

The generalization error ties goodness of prediction into the concepts of overfitting, or underfitting, the data. Overfitting means an unnecessary increase in model complexity. For example, increasing the number of parameters and the model degrees of freedom, beyond what is necessary increases the model's variance. Underfitting is the opposite notion to overfitting, i.e., too simple a model will not fit the data well. As a corollary/rule of thumb of these definitions, when the available data set is small, we should fit to it a simple model, not a complex one (Cios *et al*, 2007).

2.12 Classifiers Performance and Accuracy

According to Sagar *et al.* (2014), statistical techniques have a probabilistic background and focus on words and their categories. An important advantage of statistical techniques over the syntactic techniques is that these techniques are independent of language. As per their survey, the performance and accuracy of classifiers depends on a number of factors.

2.12.1 Features Used

The features used influence the classifier accuracy. For example, while using N-gram framework, using a high value for n degrades the performance. For sentiment analysis, the value of n should be at the most 2 or 3. Also, the number of occurrences of a word does not matter much, the mere fact that it does is enough. Apart from N-gram features, one can even use emoticons for labeling.

It is seen that the accuracy of learning algorithms increases when trained using emoticons (Sagar *et al.*, 2014).

In the study of microblog, user's posted contents and social activities are mined in a period of time to forecast their activities. These features are also valid to classify users. Useful information are extracted (both text features and non-text features), which is called multi-type features to represent each users. Multi-type Feature sets combination shows that only using content feature for classification leads to a bad result. After integrating user social network and behavior network features, the classifiers have better performance (Yan, 2014).

Farhoodi and Yari (2010) examined two efficient machine learning algorithm for Persian text document. The data set they used consisted of 800 Persian documents of different lengths that belong to 8 categories. The categories are: Literature and Art, Social, Science and Culture, Miscellaneous, Politics, Sport, Natural Environment, Economy. In their experiments, although both algorithms show acceptable results for Persian text classification, the performance of KNN was better in comparison to SVM. Their experiments showed that more number of features makes the classifier efficiency increase. However, when the number of features is more than 4000, error rises and precision value started reducing.

Fowaz and Sami (2013) studied and empirically tested the categorization effectiveness and feasibility of Arabic text categorization model based on artificial neural network. Features dimensionality reduction and selection techniques were applied to reduce the high dimension feature space, which is common for textual data based on vector space model. The introducing of features reduction methods improved the categorization performance, the reduced size of the vectors also decreased the computational time in the back-propagation neural network.

A BOW approach uses words within a corpus as predictive features and ignores word sequence as well as any syntactic or semantic content. This approach can lead to misclassification due to word use in different contexts and, if words are used as a primary features for classification, it has been shown that combining sequential words into N-grams (list of words occurring in sequence from 1–n) improves classifier performance by incorporating some degree of context into the features (Pendar, 2007, as cited in Burnap & Williams, 2016). However, an n-gram approach can suffer from the problem of high levels of distance between related words - for example, if related words appear near the start and near the end of a sentence (Chen *et al.*, 2012, as cited in Burnap & Williams, 2016).

Burnap *et al.* (2013) developed a rule-based approach to classifying antagonistic content on Twitter and, similarly to Chen *et al.*, (2012), they used associational terms as features. They also included accusational and attributional terms targeted at a person or persons following a socially disruptive event as features, in an effort to capture the context of the term use. Their results demonstrated an improvement on standard learning techniques (as cited in Burnap & Williams, 2016).

Burnap *et al.*(2013) also discovered that 'othering' language was a useful feature for classifying cyber hate based on religious beliefs - specifically for identifying anti-muslim sentiment. Othering is an established construct in rhetorical narrative surrounding hate speech (Meddaugh, 2009), and the 'we-they' dichotomy has long been identified in racist discourse (Wodak, 1999). Burnap and Williams (2016) listed examples of language that distanced particular social groups geographically (e.g.'send them home'), attempted to justify an expectation of malicious behaviour from the group (e.g.'told you so'), and was openly derogatory (e.g.'muslim savages') were reported on Twitter following the murder of Lee Rigby by Islamist extremists in London, 2013 (Marneffe, MacCartney, & Manning, 2006, as cited in Burnap & Williams, 2016).

Following the effectiveness of identifying othering terms and their success as features in a machine classifier for cyber hate targeted at specific religious groups, the present research aimed to test the effectiveness of the 'us and them' model on other types of hate speech to develop evidence for the generalizability of this method. To extract potential othering terms the Stanford Lexical Parser was implemented, along with a context-free lexical parsing model, to extract typed dependencies within the tweet text (Marneffe, MacCartney, & Manning, 2006, as cited in Burnap & Williams, 2016).

Typed dependencies provide a representation of syntactic grammatical relationships in a sentence that can be used as features for classification, and have the potential to capture othering language. For example, if the plain text is "Totally fed up with the way this country has turned into a haven for terrorists. Send them all back home". The typed dependency parser returns the following output: [root(ROOT-0, Send-1), nsubj(home-5, them-2), det(home-5, all-3), amod(home-5, back-4), xcomp(Send-1, home-5)]. The second instance (nsubj(home-5, them-2)) identifies a relationship between 'home' and 'them', with 'home' being the fifth word in the sentence and 'them' appearing before 'home' as the second word. Word order within a sentence is preserved in the type dependency and provides a feature for classification as well as the syntactic relationship between words (Burnap & Williams, 2016).

Word order within a sentence is preserved in the type dependency and provides a feature for classification as well as the syntactic relationship between words. The relationship identified by the parser in this case is nsubj, which is an abbreviation of nominal subject. This will include a noun phrase ('them'), which is the syntactic subject in the sentence, and an associated relational term ('home'). Linguistically therefore, the term 'them' is associated with 'home' in a relational sense (Burnap & Williams, 2016).

Sociologically, this is an othering phrase, which essentially distances 'them' from 'us' through the relational action of removing 'them' to their 'home', as perceived by the author of the tweet. Similarly, the third typed dependency (det(home-5, all-3)) identifies a det relationship, which is short for determiner, where a link is established between a noun phrase and its determiner. The noun phrase here being 'home' (as in a place) and the determiner being 'all'. Again, this falls into an othering behaviour, suggesting that the entire social group should have a relationship with ‘home’, which we can assume means the perceived ‘home’ of the social group by the author of the tweet (i.e., ‘not my country’). This combination of linguistics and sociology potentially provides a very interesting set of features for the more nuanced classification of cyber hate, beyond the BOW approach that utilizes expletives and derogatory terms. It allows a more common-sense reasoning approach to classifying cyber hate by considering the integration of othering terms and calls for retribution action into the classification features (Burnap & Williams, 2016).

Dashen and Fengxin (2015) confirmed that that feature dimension has an impact on the performance of classification model. From the dimension of value from 400 to 900, it can be observed that with the increase of the dimension, the performance of the model was enhanced. Between the dimension of value 900 and 1000, when the feature dimension reached a critical value; with the increase of the dimension, model performance was slightly reduced. When size reached 500, the macro average and the average values were stable. When the size was 900, best results were achieved. The reseachers also further investigated the impact of the size of decision tree on the performance of random forests, feature size is set as 900.

The size of decision tree was ranging from 100 to 400. The research showed that the number of decision trees has a certain impact on the performance of the model. From the number of decision trees from 100 to 250, it was known that increasing number of decision tree model improved the performance. From the number of decision trees from 250 to 400, it was observed that when arriving at a critical value, even though the number of decision tree was increased, the model performance remains unchanged. When the size was increased to 250, the macro average and the average values were stable. The research showed that random forests have a good performance in text categorization because Bigram can only count up to 70% Chinese vocabulary (Dashen and Fengxin, 2015).

Quan, Wei, and Ren (2014) compared SVM and semantic similarity method for sentiment classification. According to their results, they made the following observations: First, supervised method has higher accuracy, but the accuracy is dependent on the quantity and quality of training sets. Supervised method has two disadvantages: classification totally dependent on the quality of the training sets, when the quality of training sets is not good enough, classification effect is not obvious. Second, when the training set is very small, there will be over-fitting phenomenon, which affects classifier performance. The solution is increase the number of training samples, but data tagging is a very difficult thing to do, it requires a lot of manpower and resources to carry out a standard.

Yadav and Manwatkar (2015) proposed and tested a computer virus classifier using n-grams and Hidden Markov Model (HMM). They explained that selection of N-grams that represent the intent of the file is very important and affects classification of malware files. They used N-grams to represent n consecutive opcodes in a disassembled executable file and used them as features. Size of the vocabulary i.e. number of distinct N-grams depends on the value of n. Identifying the right value of n in N-gram extraction is critical and is a subject of great research interest.

For text classification, Yadav and Manwatkar (2015) recommended values for n range from 2 to 12. For a value of n set to 2, a large number of N-grams are generated and if n value is chosen to be high, too few number of N-grams are generated. Further, a large value for n (7, 8, 9) will generate more sparse and unique features for each file. The actual correlation between documents cannot be achieved and this degrades the performance of a classifier. Choosing a moderate value of n i.e. 3 obtained a reasonable sized feature set.

2.12.2 Feature Selection Method Used

Not all features that are returned by the tokenization algorithm must be used, because the list contains a lot of irrelevant features. Hence, the feature selection method used also determines the accuracy of a classifier. Generally for sentiment analysis, the Chi-square test and the Mutual Information method is used. Each algorithm evaluates the keywords in a different manner and thus leads to different selections. Moreover, each algorithm requires different configuration such as the level of statistical significance, the number of selected features, and so on (Sagar *et al.*, 2014).

Many researchers have proved that DF, IG and CHI are almost the top feature selection methods beyond different corpus. All of the feature selection methods can improve classifier's performance. In multi-type feature space, when feature number was decreased from 15000 to 400, the classification effect curve continues to increase. But once feature number reaches 200, the effect starts to turn worse instead of better. IG is the best in higher feature dimensional space and superior to the other three methods with obvious gap. But along with dimension reducing, KG approach gradually shows its advantages. KG has better performance when the dimension decrease to 5000. In low feature dimension, KG get best performances in Precision, Recall, F-Measure (Yan, 2014).

Machine learning is a tool used in Artificial Intelligence to provide computers with capabilities for automatically improving themselves in the recognition of patterns. Machine learning algorithms rely on selected features and training data to infer the commonalities that a group of searched items share and that discriminate them from the rest of the universe. The success of these algorithms therefore depend on the relevance of the features for discrimating between the group of searched items and the rest, and on the quality of training data for being unbiased and representative of the universe of items (Allix, Bissyande, Jerome, Klein, State, & Le Traon, 2014).

2.12.3 Learning Method

In general, supervised techniques for classification consist of two steps: first, the extraction of features from training data and their conversion to feature vectors; second, training of the classifier on the feature vectors and application of the classifier to unseen instances. Hence, feature construction and learning method selection are crucial for accurate classification (Suh, 2016).

C4.5, an extension of the ID3 algorithm, is a decision-tree building algorithm adopts a divide-and-conquer strategy and an entropy measure for object classification. Its goal is to classify mixed objects into their associated classes based on the objects' attribute values. Second, NN has been popular because of its unique learning capability, and has achieved good performance in many different applications. Third, SVM is a novel learning machine and is based on the structural risk minimization principle from computational learning theory. Because SVM can handle millions of inputs with good performance Fourth, based on Bayes Theorem, NB is a fairly simple probabilistic classification algorithm that uses strong independence assumptions regarding various features. NB assumes that the presence of any feature is entirely independent of the presence of the other features, and allows building classification models efficiently (Suh, 2016).

2.12.4 Domain in Question

Apart from the features and the feature selection method, the classifier accuracy depends a lot on the domain in question. Different classifiers yield different results when applied to different domains. For twitter, the binarized version of Naive Bayes along with Mutual Information feature selection method is known to outperform even the Support Vector Machines (Sagar *et al.*, 2014).

2.12.5 Amount of Historic Information Needed

Some researchers have also investigated the effect of amount of historic information that is needed on classifier performance and accuracy. Engelstad, Hammer, Yazidi, and Bai (2015) analyzed the effect of time in automatic securicy classification. Engelstad *et al.* (2015) recommended that before applying machine learning for automatic security classification, the organization needs to build up a certain amount of information that is used to train the machine learner, so that it performs sufficiently well. How much historic information is needed depends on the specific corpus and the type of information objects. They started machine training on only 40% of the corpus and measured the classification accuracy on the consecutive 200 documents.

Engelstad *et al.* (2015) then they added another 0.1% of the corpus to the training set, and trained the machine learner again from the start and measured the classification accuracy on the next 200 documents. They continued until the training set reached 90% of the entire corpus. Their experiments showed that when the size of the training set is small, the classification accuracy suffers. However, as the training set reached around 55% of the corpus, which corresponded to around 1100 - 1200 documents in the corpus, they reached a state where the classification accuracy stabilized and they observed little performance gain by increasing the training set further.

Chapter THREE

RESEARCH METHODOLOGY

3.1 Introduction

Any successful project must have valid and approved methodologies to be followed in carrying out various research activities. Sentiment analysis is a very popular and actively explored research area, and therefore, there exists a range of different methods and algorithms for applying sentiment classification on texts. This chapter starts by describing how data collection was carried out. This is then followed by a description of research population, data preprocessing, N-gram sentiment modeling, and finally estimation of probabilities using N-grams.

3.2 Data Collection

Text data was collected from online social networks pages of human rights activists, politicians, media houses, bloggers and prominent people in the country and outside. Blogs of media houses was also useful as a data source. Spoken sentiments from people who were already having hate speech court cases was also converted to English text as used as data. Print media also was used for data collection. Many cases of hate sentiment were documented and report in the newspapers. These were then extracted and used as data set. Since 2007/2008 general elections, there has also been many researchers interested in studying hate speech. In some of the research papers and journals by these researchers, they also highlighted some of the sentiments which were used. Some of the words or phrases which have been used to prosecute some hate speech suspects in Kenya and other countries was used as samples to indicate hate speech (yes) or not hate speech (no).

This research used guidelines given by National Cohesion and Integration Commission (NCIC). Hate speech is defined as any form of expression that is hostile to an entire community and is aimed at encouraging either contempt or denigration or defamation or exclusion or victimization of individuals belonging to that particular community. It is any communication that contains expression of hatred of some group, especially in circumstances in which it is likely to provoke violence. It is an incitement to hatred primarily against a group of persons defined in terms of race, ethnicity, national origin, gender, religion, sexual orientation, among others. It can be any form of expression regarded as offensive to racial, ethnic and religious groups and other discrete minorities (National Cohesion and Integration Commission -NCIC, 2010).

Speech is used to mean oral, written, displayed and or acted out action. The following are indicators of Hate Speech (Guidelines for Monitoring Hate Speech -NCIC, 2010):

- Speech that causes hatred: The speech must be such that will solicit disdain against a person or group because of their ethnicity.
- Speeches or utterances that encourage ethnic, religious or group violence: The Speech must encourage the audience into some negative action.
- Utterances that depict others as inherently inferior: The speech must infer superiority and inferiority to parallel groups.
- Utterances that degrade others: The utterances must state or infer that the other person is a lesser human.
- Utterances that dehumanizes: The utterance must state or infer that the other person is not human e.g. calling them a weed.

- Use of Cultural stereotypes: The generalization of a group while depicting them in a negative way.

- Utterances that promote discrimination on the basis of tribe, colour, ethnic group, and religious group.

- Use of abusive, negative and insulting language.

- Use of inciting and/or provocative language.

- Use of stories that profile people and communities negatively.

- Use of imagery, poems, metaphor, proverbs which could stir up ethnic hatred.

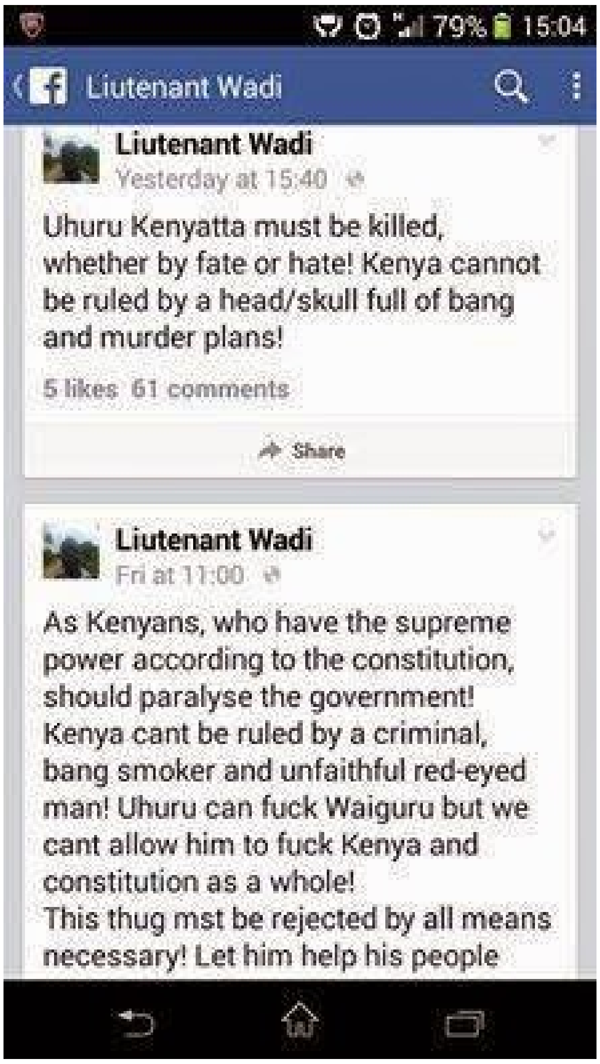
- Pictures published in media which could lead to ethnic, religious, or racial discrimination.

- Stories or essays used by the media houses to depict others less inferior or which could be used to propagate hatred.

- Ridiculing of another on basis of ethnicity, race or religious belief.

- Use of alarming language.

Figure 3.2, 3.2, 3.2 and 3.2 are some extracts from blogs of people who have been charged with hate speech.



Liutenant Wadi
Yesterday at 15:40

Uhuru Kenyatta must be killed, whether by fate or hate! Kenya cannot be ruled by a head/skull full of bang and murder plans!

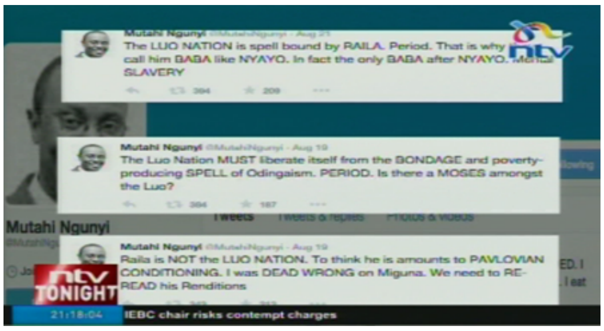
5 likes 61 comments

Share

Liutenant Wadi
Fri at 11:00

As Kenyans, who have the supreme power according to the constitution, should paralyse the government! Kenya cant be ruled by a criminal, bang smoker and unfaithful red-eyed man! Uhuru can fuck Waiguru but we cant allow him to fuck Kenya and constitution as a whole!
This thug mst be rejected by all means necessary! Let him help his people

Figure 3.2: Sample Online Hate Speech

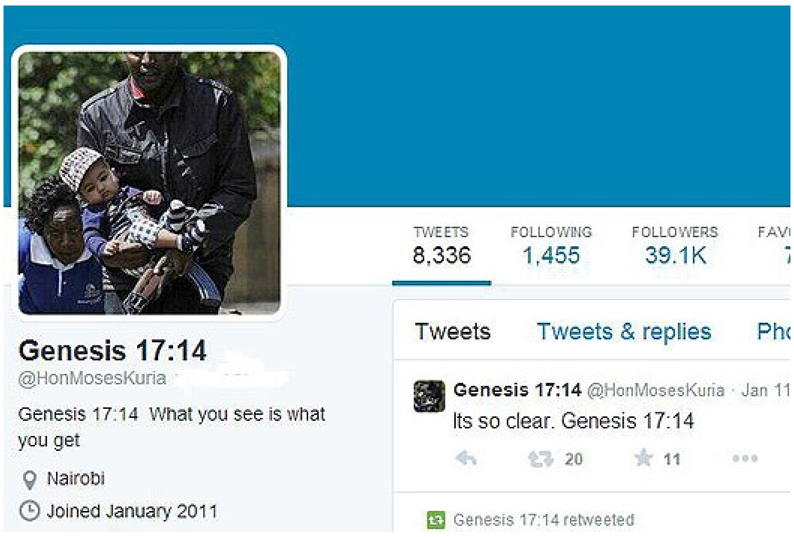


Mutahi Ngunyi · Aug 21
The LUO NATION is spell bound by RAILA. Period. That is why call him BABA like NYAYO. In fact the only BABA after NYAYO. SLAVERY
394 209

Mutahi Ngunyi · Aug 19
The Luo Nation MUST liberate itself from the BONDAGE and poverty-producing SPELL of Odingaism. PERIOD. Is there a MOSES amongst the Luo?
394 187

Mutahi Ngunyi · Aug 19
Raila is NOT the LUO NATION. To think he is amounts to PAVLOVIAN CONDITIONING. I was DEAD WRONG on Miguna. We need to RE-READ his Renditions

Figure 3.2: Sample Online Hate Speech



TWEETS 8,336 FOLLOWING 1,455 FOLLOWERS 39.1K

Tweets Tweets & replies

Genesis 17:14
@HonMosesKuria

Genesis 17:14 What you see is what you get

Nairobi

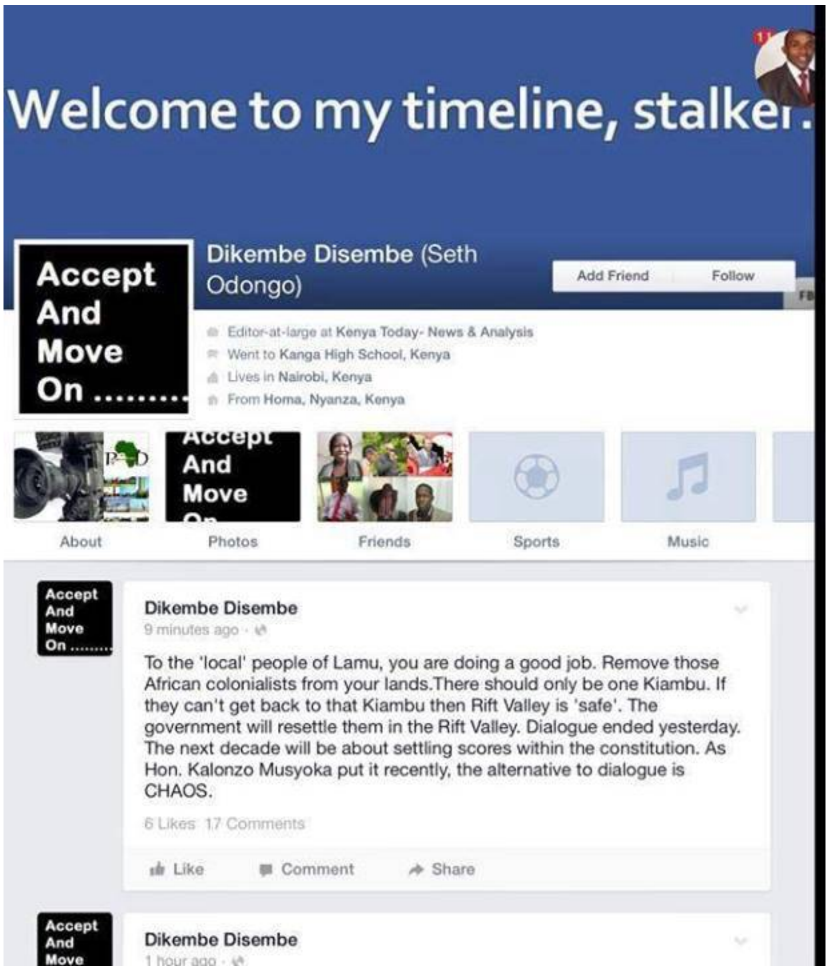
Joined January 2011

Genesis 17:14 @HonMosesKuria · Jan 11
Its so clear. Genesis 17:14

20 11

Genesis 17:14 retweeted

Figure 3.2: Sample Genesis Online Hate Speech



Welcome to my timeline, stalker.

Accept And Move On ....

Dikembe Disembe (Seth Odongo)

Add Friend Follow

Editor-at-large at Kenya Today- News & Analysis
Went to Kanga High School, Kenya
Lives in Nairobi, Kenya
From Homa, Nyanza, Kenya

About Photos Friends Sports Music

Dikembe Disembe
9 minutes ago

To the 'local' people of Lamu, you are doing a good job. Remove those African colonialists from your lands.There should only be one Kiambu. If they can't get back to that Kiambu then Rift Valley is 'safe'. The government will resettle them in the Rift Valley. Dialogue ended yesterday. The next decade will be about settling scores within the constitution. As Hon. Kalonzo Musyoka put it recently, the alternative to dialogue is CHAOS.

6 Likes 17 Comments

Like Comment Share

Dikembe Disembe
1 hour ago

Figure 3.2: Sample Online Hate Speech

3.3 Research Population

Purposive sampling technique was used for this research. Judgment (purposive) sampling is a non-probability sampling technique in which the researcher selected the sample based on his judgment and knowledge about some appropriate characteristics of online sentiments required of the sample member. This was done by selecting words or word phrases that suggest hate or violence against a certain tribe, race, gender, and religion. In the context of text classification, features or attributes usually mean significant words, multi-words or frequently occurring phrases indicative of the text category.

A total of one thousand seven hundred and sixty two (1,762) sentiments were used for this experiment. These sentiments were divided further into five data sets; namely training, testing, testing20, testing50 and Combined. These data sets differed in terms of length and number of the sentiments. Majority of training and testing data sets consisted of short sentiments of a minimum of one word to ten words. Most of the Testing20 data set had longer sentiments than training and testing but not longer than testing50. Testing50 had longest sentiments compared to training, testing and testing20. Training, testing20, and testing50 were used as training data sets as well as testing data sets in different experiments conducted in this research.

Training data set has a total of 392 data instances; 196 belonging to "yes" class and 196 sentiments belonging to "no" class. Testing has 119 sentiments ; 68 belonging to "yes" class and 51 sentiments belonging to "no" class. Testing20 data set has 261 sentiments; 119 belonging to "yes" class and 142 belonging to "no" class. Testing50 data set has 144 sentiments; 86 belonging to "yes" class and 58 sentiments belonging to "no" class. Combined data set has 846 sentiments; 429 sentiments belonging to "yes" class and 417 sentiments belonging to "no" class.

In Experiment 1, training data set was used to learn classifiers in NB, NBM, SVM, KNN, and DT algorithms and the learned model was used to predict the classifier accuracy using testing, testing20, testing50, and combined as testing data sets.

3.4 Sentiment Classification Process

Figure 3.4 is a diagrammatic representation of the sentiment classification processes. It illustrates the steps that were involved in this research from data collection to assigning class labels to sentiments. The process starts by identifying and collecting online sentiments from blogs and news forums. The selected sentiments formed the data sets. These sentiments were then preprocessed. An arff file was then created, which served as the input file to machine learning algorithms (classifiers). These classifiers gave class labels (yes or no) of sentiments as the outputs.

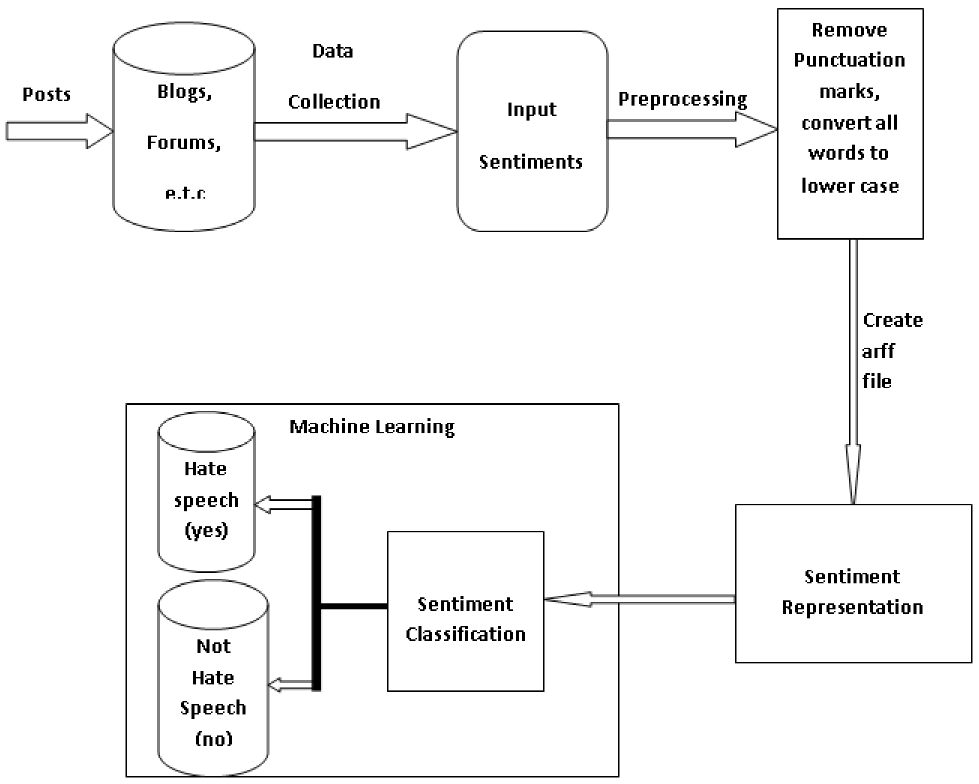
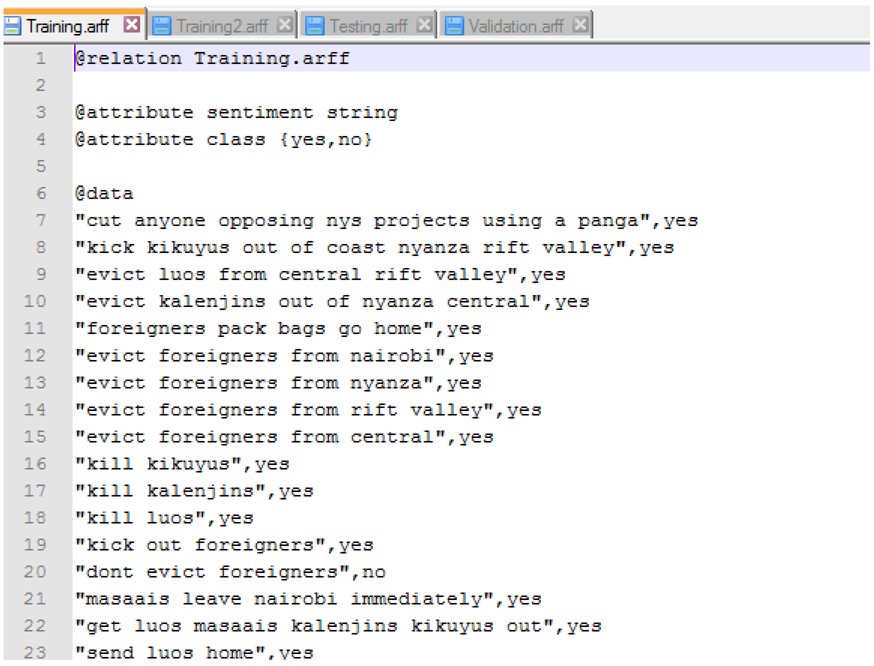


Figure 3.4: Sentiment Classification Process

For learning classifiers, the input N-gram information is vectorized by generating a vector V for each sentence s in which $v_i \in V$ correspond to the presence of N-gram phrase g_i from the sentence s. Once the classifier has learned, the model created will be used to determine which class the sentence from test set is describing.

3.5 Data Preprocessing

Sentiments collected were not in a format suitable for machine learning. They contained punctuation marks, spelling mistakes, symbols and numbers, abbreviations, and sentiments were both in lower and upper cases. For the sentiments to be in a suitable data format for machine learning, they had to be preprocessed. Sentiments were preprocessed and properly organized. This involved removing full stops, semicolons, symbols, quotation marks, writing abbreviations in full,converting all sentiments to lower case. The sentiments were then represented in an Attribute Relation File Format (arff) file format. This format is suitable as input file format for WEKA machine learning environment. Figure 3.5 shows sample training data in arff file format.



```
@relation Training.arff

@attribute sentiment string
@attribute class {yes,no}

@data
"cut anyone opposing nys projects using a panga",yes
"kick kikuyus out of coast nyanza rift valley",yes
"evict luos from central rift valley",yes
"evict kalenjins out of nyanza central",yes
"foreigners pack bags go home",yes
"evict foreigners from nairobi",yes
"evict foreigners from nyanza",yes
"evict foreigners from rift valley",yes
"evict foreigners from central",yes
"kill kikuyus",yes
"kill kalenjins",yes
"kill luos",yes
"kick out foreigners",yes
"dont evict foreigners",no
"masaais leave nairobi immediately",yes
"get luos masaais kalenjins kikuyus out",yes
"send luos home",yes
```

Figure 3.5: Training Data in ARFF File Format

3.6 N-gram Sentiment Modeling

A document model, sometimes also called generic document, aims at the description of the structuring rules of a document class. It defines in a generic way of how specific documents can be structured. The physical structure groups hierarchically the appearance of physical entities in a document. N-grams document modeling and bag-of-words modeling techniques were used. N-grams are a commonly used natural language model, which assumes, that only the previsous $n - 1$ words in a sentence have any effect on the probabilities, for the next word.

Fixed-length N-grams were generated as well as variable-length N-grams.

Unigram text model is the same as bag-of-words text model. In unigram, this research used one word as a feature. In a bigram model, used two words, trigram models used three words as features. The number of words were increased by one until 12gram. For example, the sentiment "evict foreigners from rift valley" if modeled using unigrams would have "evict", "foreigners", "from", "rift", "valley" as the sentiment features. Bigrams would model the sentiment with "evict foreigners", "foreigners from", "from rift", "rift valley" as the sentiemnt features. Trigram would model the sentiment as "evict foreigners from", "foreigners from rift", "from rift valley" as sentiment features.

Bigram and trigram models the sentiment with dependence or relationship between words which make up the sentiment. This was used to restrict the context within which the words are used. N-gram modeling was performed by Ngram Tokenizer, a tokenizer available in WEKA environment by choosing the minimum and maximum ngram size. For fixed-length n-grams, the minimum and maximum n-gram number value was equal, while in variable-length, the minimum n-gram size value was less than maximum n-gram size value. If $N =$ the number of words in a given sentence K, the number of n-grams for sentence K would be $Ngrams_K = N - (n - 1)$.

3.7 Probability Estimations for N-grams

N-gram approach computes the occurrence probability on word sequence in training data. By extracting the word sequence in test data, it can predict the highest probability for related class to given news text . An advantage of N-gram approach is that the language modeling approach does not discard low frequency features during classification, as is commonly done in traditional classification learning approaches. Also, the language modeling approach uses n-gram models to capture more contextual information than standard bag-of-words approaches, and employs better smoothing techniques than standard classification learning (Farhoodi *et al.*, 2011).

Statistical text categorization uses machine learning methods to learn automatic classification rules based on labeled documents. A label states the category of a text document which was manually identified by a human expert (Yasotha & Charles, 2015).

Given a word sequence $W = w_1, w_2, ..., w_T$, the probability of W can be calculated. By the chain rule of probability, the probability of any sequence (Peng and Huang, 2006 as cited in Farhoodi *et al.*, 2011) is expressed in Equation 3.7.

$$P(W) = P(w_1 w_2 ... w_T) = \prod_{i=1}^{T} P(w_i|w_1...w_{i-1}) \tag{3.7}$$

An N-gram model approximates this probability by assuming that the only words relevant to predicting $P(w_i|w_1...w_{i-1})$ are the previous n-1 words; that is, it assumes the Markov N-gram independence assumption and is expressed in Equation 3.7

$$P(w_i|w_1...w_{i-1}) = P(w_i|w_{i-n+1}...w_{i-1}) \tag{3.7}$$

A straightforward maximum likelihood estimate of N-gram probabilities from a corpus is given by the observed frequency as expressed in Equation 3.7

$$P(w_i|w_{i-n+1}...w_{i-1}) = \frac{\#(w_{i-n+}...w_i)}{\#(w_{i-n+}...w_{i-1})} \tag{3.7}$$

where $\#(\bullet)$ is the number of occurrences of a specified gram in the training corpus (Yasotha & Charles, 2015).

The conditional probability is then used find the probability of a sentence being in a certain aspect given a sentence broken down in a series N-gram features. The classifier uses the N-gram training data in order to find the frequency count of an N-gram according to a specific aspect. For example, once the Naive Bayes classifier has been trained using the N-gram training data, the probabilities of a sentence being in a certain class are then compared. The classifier can then predict which class a sentence belongs to by picking the class with a biggest probability given the N-gram set of the sentence, formally defined in Equation 2.8 (Yasotha & Charles, 2015).

Chapter FOUR

EXPERIMENTS AND RESULTS

4.1 Introduction

In order to assess the performance and behaviour of the researchers approach, bag of words and N-grams document modeling techniques were evaluated with respect to machine learning algorithms. The objectives of the experiments were to assess whether the size of the sentiments of the training data set has an effect on classifier and N-gram performance, identify the behaviour of N-grams when used with different sizes of sentiments, evaluate the performance trend of N-grams with different classifiers, compare the performance of using fixed N-grams and variable size N-grams, and examine the effect of quality of data used for training data set on the performance of various classifiers.

This chapter starts by describing the experimental setup for this research. Next sections detail experiments and results with different text classifiers, different training and testing data sets.

4.2 Experimental Setup

This research used classifiers provided by Weka. The data sets were used as input to five machine learning algorithms; Naive Bayes (NB), Naive Bayes Multinomial (NBM), Support Vector Machines (SVM), K-Nearest Neighbour (KNN) and Decision Trees (DT). The experiments started with $N = 1$ (unigram), then bigram, then steadily increasing the size of N by 1 until the value of N reached 12. A range of values for variable size of N were also used by setting the minimum N value and maximum N value. For example, Min1Max2 means that the minimum N value is 1 and maximum N value is 2. The minimum value was then kept constant and the maximum value increased to 3 and 4 respectively. Then a minimum value of 2 and a maximum value of 3 was the last to be used in this experiment. These algorithms were used to create learning models and then the learned models used to predict class of testing data sentiments. Experiment results were then presented in tables, analyzed and interpreted as detailed in Section 4.3 to Section 4.6.

4.3 Training Using Short Sentiments

A classifier describing predetermined set of data classes was built. This was done using training data with short sentiments (Training) and the performance of classifiers recorded in tables. and creating models from machine learning algorithms. The performance measure Accuracy was used for performance evaluation and recorded in tables. Testing, Testing20, Testing50 and Combined data sets were used as testing data sets for various machine learning algorithms as detailed from subsection 4.3.1 to subsection 4.3.5.

4.3.1 Experiment 1: Prediction Using NB

The NB results of the prediction are presented in Table 4.3.

Table 4.3: Prediction Results Using Naive Bayes

N-gram Size	**Testing**	**Testing20**	**Testing50**	**Combined**
Unigram	57.63%	61.30%	59.72%	60.12%
Bigram	44.07%	60.92%	49.31%	54.56%
Trigram	44.07%	62.45%	45.14%	54.67%
4gram	44.92%	62.84%	43.75%	54.56%
5gram	44.07%	62.45%	40.97%	53.25%
6gram	44.07%	62.45%	40.97%	53.25%
7gram	44.07%	62.45%	40.97%	53.25%
8gram	43.22%	62.45%	40.97%	53.14%
9gram	43.22%	62.45%	40.97%	53.14%
10gram	43.22%	62.45%	40.97%	53.14%
11gram	43.22%	59.77%	40.97%	51.48%
12gram	42.37%	56.31%	40.97%	49.70%
Min1Max2	50%	59.39%	62.5%	59.05%
Min1Max3	45.76%	61.30%	58.33%	57.99%
Min1Max4	44.07%	59.39%	58.33%	56.57%
Min2Max3	44.54%	61.69%	47.22%	54.49%

Naive Bayes use presence or absence of a feature (in this research a word or phrase) as a model for representing text data. For unigram, Testing20 produced the highest accuracy value of 61.30%. The performance of N-grams reduced for Testing, Testing50, and combined data sets. However, for Testing20, the accuracy reduced with bigrams and then increased to a constant value of 62.45% and only starts reducing from 11-grams. This increase is as a result of the presence of more representative bigrams in training data set and Testing20 data set. This constant accuracy values is as a result of increased feature sparsity from trigrams in the Training data set and Testing20 data set. This data set is made up of few words (i.e. short sentiments) and thus increasing n beyond the maximum sentiment length does not significantly affect the size N of the training feature set.

4.3.2 Experiment 2: Prediction Using NBM

Table 4.3: Prediction Results Using NBM

N-gram Size	Testing	Testing20	Testing50	Combined
Unigram	59.32%	55.17%	63.19%	58.11%
Bigram	62.71%	57.47%	49.31%	55.15%
Trigram	61.02%	52.87%	56.94%	54.79%
4gram	57.63%	47.89%	57.64%	51.36%
5gram	57.63%	46.36%	58.33%	50.77%
6gram	57.63%	46.36%	59.03%	51.01%
7gram	57.63%	46.36%	59.03%	51.01%
8gram	57.63%	46.36%	59.72%	51.24%
9gram	57.63%	46.36%	59.72%	51.24%
10gram	57.63%	46.36%	59.72%	51.24%
11gram	57.63%	46.36%	59.72%	51.24%
12gram	57.63%	46.36%	59.72%	51.24%
Min1Max2	61.86%	60.15%	55.56%	58.22%
Min1Max3	61.86%	59.77%	58.33%	59.03%
Min1Max4	56.78%	59.39%	58.33%	57.99%
Min2Max3	61.34%	55.56%	55.56%	55.20%

NBM results showed increase in accuracy with an increase in N in Testing and Testing20 data sets. With unigram, Testing data set had an accuracy of 59.32%, which increased to 62.71% with bigram, and 61.02% with trigram. In Testing20 data set, there is an increase from 55.17% to 57.47% between unigram and bigram. Testing50 data set had the highest accuracy value at 63.19% with unigram. The accuracy then decreased to 49.31% before steadily increasing to a maximum accuracy value of 59.72%. This increase is as a result of term frequency and inverse document frequency term weighting function used by NBM. Since

Testing50 is made up of long sentiments compared to other data sets, the terms and phrases are more frequent than in other data sets, resulting into higher weighting value used in the computation of probability values of a sentiment being in a class. This causes higher accuracy values.

4.3.3 Experiment 3: Prediction Using SVM

Prediction results for SVM are presented in Table 4.3.

Table 4.3: Prediction Results Using SVM

N-gram Size	**Testing**	**Testing20**	**Testing50**	**Combined**
Unigram	61.02%	61.30%	64.58%	61.30%
Bigram	59.32%	64.75%	54.86%	60.24%
Trigram	60.17%	50.96%	54.17%	52.66%
4gram	57.63%	46.74%	57.64%	50.65%
5gram	57.63%	46.36%	58.33%	50.77%
6gram	44.07%	62.45%	40.97%	53.25%
7gram	44.07%	62.45%	40.97%	53.25%
8gram	43.22%	62.45%	40.97%	53.14%
9gram	43.22%	62.45%	40.97%	53.14%
10gram	43.22%	59.77%	40.97%	51.48%
11gram	43.22%	59.77%	40.97%	51.48%
12gram	57.63%	46.36%	40.97%	51.24%
Min1Max2	60.17%	62.52%	63.89%	63.91%
Min1Max3	61.86%	63.98%	54.17%	60.47%
Min1Max4	58.47%	55.94%	34.72%	49.59%
Min2Max3	58.82%	65.13%	56.25%	61.11%

With unigram, Testing50 had the highest accuracy value of 64.58%, followed by Combined and Testing20 data sets at 61.30%, and lastly Testing data set at

61.02%. SVM is good at handling large features set and is robust when there is a sparse set of examples. All data sets except Testing20 shows a decrease in accuracy with the increase in N, which starts to increase when $N = 6$ to a constant value of 62.45%. With N range between 1 and 2, Testing20 also had the highest accuracy of 65.13%. This is because SVM has the strong learning ability and is able to capture the inherent characteristics of training and testing data sets. Testing20 data set disputes the common result that ngram performance always decreases from trigram. The possible number of observed n-grams at each value of n grows exponentially. As n grows, the frequency of n-gram permutations more closely matches the expected linear relationship as expected of a hyperbolic distribution.

4.3.4 Experiment 4: Prediction Using KNN

Prediction results are presented in Table 4.3.

Table 4.3: Prediction Results Using KNN

N-gram Size	**Testing**	**Testing20**	**Testing50**	**Combined**
Unigram	61.02%	62.84%	60.42%	61.30%
Bigram	54.24%	68.20%	49.31%	60.83%
Trigram	57.63%	46.36%	59.03%	51.01%
4gram	57.63%	46.36%	59.03%	51.01%
5gram	57.63%	46.36%	59.03%	51.01%
6gram	57.63%	46.36%	59.03%	51.01%
7gram	57.63%	46.36%	59.03%	51.01%
8gram	57.63%	46.36%	59.72%	51.24%
9gram	43.22%	57.09%	40.28%	49.59%
10gram	43.22%	57.09%	40.28%	49.59%
11gram	43.22%	57.09%	40.28%	49.59%
12gram	42.37%	56.71%	40.28%	49.47%
Min1Max2	53.39%	67.05%	47.92%	59.53%
Min1Max3	57.63%	47.13%	58.33%	51.12%
Min1Max4	57.63%	46.73%	59.72%	51.36%
Min2Max3	57.14%	46.74%	59.03%	51.06%

Testing20 data set had the highest accuracy with unigram, at an accuracy value of 62.84%, and is the only data set that showed an increase in accuracy with bigram (68.20%). This means that Testing20 data set has accurate sentiment similarity values with bigrams, since no model is learned from the training data. Learning only occurs when a test example needs to be classified. The accuracy value started being constant for all the data sets with trigram. With 9gram, Testing20 data set accuracy value increased from the constant value of 46.36% to 57.09%. With variable N-grams, a minimum value of 1 and a maximum value of 2 displayed the highest accuracy of 67.05%.

4.3.5 Experiment 5: Prediction Using DT

Results of prediction using DT are presented in Table 4.3.

Table 4.3: Prediction Results Using DT

N-gram Size	**Testing**	**Testing20**	**Testing50**	**Combined**
Unigram	55.08%	55.17%	59.03%	56.33%
Bigram	42.37%	54.02%	45.83%	50.77%
Trigram	57.63%	45.59%	59.72%	50.77%
4gram	57.63%	45.59%	59.72%	50.77%
5gram	57.63%	45.59%	59.72%	50.77%
6gram	57.63%	45.59%	59.72%	50.77%
7gram	57.63%	45.59%	59.72%	50.77%
8gram	57.63%	45.59%	59.72%	50.77%
9gram	57.63%	45.59%	59.72%	49.23%
10gram	57.63%	45.59%	59.72%	50.77%
11gram	57.63%	45.59%	59.72%	50.77%
12gram	57.63%	45.59%	59.72%	50.77%
Min1Max2	57.63%	54.02%	68.06%	58.93%
Min1Max3	55.93%	55.17%	68.75%	59.88%
Min1Max4	62.71%	50.96%	63.19%	56.45%
Min2Max3	42.86%	54.02%	45.83%	50.83%

With unigram, Testing50 registered the highest accuracy value of 59.03%, followed by 56.33%, 55.17%, and 55.08% for Combined, Testing20, and Testing respectively. This is because Decision trees classify the sentiments based on feature values. Unigram has features that are more obvious, frequent and representative than other ngram features. This explains the increase in accuracy value on the Testing, Testing50 data set since, Testing data set and Training data set are similar in terms of length of the sentiments. Testing50 has longer sentiments thus resulting into representative classification features. With increased N-gram size, only Testing and Testing50 data sets showed increase in accuracy. Testing accuracy value increased from bigram (42.37%) to 57.63% with trigram, which remains constant to 12gram. Testing50 data set accuracy value increased from 45.83% with bigram to a constant value of 59.72%. The highest accuracy value with variable length was with a minimum value of 1 and a maximum value of 3 with Testing50 data set at 68.75%.

4.4 Training Using Medium Length Sentiments

In this section, Testing20 data set, which consist of longer sentiments than Training was used for training. The results of predictions of different classifiers are presented in tables.

4.4.1 Experiment 6: Prediction Using NB

Table 4.4: Prediction Results Using NB

N-gram Size	Testing50	Combined	Testing	Training
Unigram	43.06%	64.26%	42.37%	52.81%
Bigram	41.67%	63.08%	42.37%	50.77%
Trigram	40.28%	62.25%	42.37%	50.51%
4gram	40.28%	62.25%	42.37%	50.51%
5gram	40.28%	63.67%	42.37%	50.51%
6gram	40.28%	66.50%	42.37%	51.02%
7gram	40.28%	67.45%	42.37%	51.28%
8gram	40.28%	69.35%	42.37%	51.53%
9gram	40.28%	70.41%	42.37%	51.53%
10gram	40.28%	70.88%	42.37%	51.53%
11gram	40.28%	71.24%	42.37%	51.53%
12gram	40.28%	70.89%	42.37%	51.53%
Min1Max2	41.67%	63.78%	42.37%	51.79%
Min1Max3	40.28%	62.49%	42.37%	51.28%
Min1Max4	40.28%	62.49%	42.37%	50.77%
Min2Max3	40.28%	62.49%	42.37%	50.77%

Table 4.4 shows prediction results using NB. With unigram, Combined data set had the highest accuracy value of 64.26%. This high performance is because Combined data set has many sentiments and this results in high presence of terms or words in both Testing20 and Combined data sets. This accuracy value then decreased to 63.08% for bigram, 62.25% for trigram and 4gram, before increasing steadily from 5gram (63.67%) to 11gram (71.24%). Similar trend was realised with Training data set. This increase from 5gram can be attributed to overfitting resulting from larger values for N resulting into complete sentiments being used

as features in the training and prediction phase.

4.4.2 Experiment 7: Prediction Using NBM

Table 4.4: Prediction Results Using NBM

N-gram Size	**Testing50**	**Combined**	**Testing**	**Training**
Unigram	45.14%	76.69%	56.78%	54.85%
Bigram	51.39%	78.11%	52.54%	57.14%
Trigram	51.39%	77.16%	43.22%	53.94%
4gram	40.97%	74.20%	43.22%	54.34%
5gram	41.67%	74.44%	42.37%	52.04%
6gram	40.28%	74.96%	42.37%	52.04%
7gram	40.28%	73.96%	42.37%	52.04%
8gram	40.28%	73.96%	42.37%	52.04%
9gram	40.28%	73.96%	42.37%	52.04%
10gram	40.28%	73.96%	42.37%	52.04%
11gram	40.28%	73.73%	42.37%	52.04%
12gram	40.28%	73.49%	42.37%	51.79%
Min1Max2	42.36%	76.08%	61.02%	59.18%
Min1Max3	41.67%	76.21%	58.47%	59.44%
Min1Max4	41.67%	75.86%	55.93%	56.12%
Min2Max3	50.69%	77.75%	51.69%	56.38%

Table 4.4 presents the results of prediction using NBM. With unigram, Combined data set had the highest accuracy value (76.69%). This is due to higher weighting values resulting from term frequency and inverse document frequency calculations. Combined data set has more sentiments and therefore results into higher weights being used in the training phases. This results in high accuracy for unigrams. The accuracy values then decreased before becoming constant with

7gram at 73.96%. Only Testing50 and Training data set showed increase in accuracy from unigram to bigram, 45.14% to 51.39% for Testing50 and from 54.85% to 57.14% for Training data set. This increase is due to common bigram features in the data sets. However, with trigrams and more ngrams, phrase feature sparsity results in low feature weights and thus results in more errors.

4.4.3 Experiment 8: Prediction Using SVM

Table 4.4: Prediction Results Using SVM

N-gram Size	**Testing50**	**Combined**	**Testing**	**Training**
Unigram	45.83%	76.80%	52.54%	53.83%
Bigram	44.44%	75.86%	46.61%	56.38%
Trigram	40.28%	74.08%	42.37%	47.96%
4gram	40.28%	74.08%	42.37%	52.04%
5gram	40.28%	74.08%	42.37%	52.04%
6gram	40.28%	73.96%	42.37%	52.04%
7gram	40.28%	73.96%	42.37%	52.04%
8gram	40.28%	73.96%	42.37%	52.04%
9gram	40.28%	73.96%	42.37%	52.04%
10gram	40.28%	73.96%	42.37%	52.04%
11gram	40.28%	73.73%	42.37%	52.04%
12gram	40.28%	73.49%	42.37%	51.79%
Min1Max2	48.61%	77.28%	50%	59.44%
Min1Max3	47.22%	76.80%	48.31%	55.61%
Min1Max4	41.67%	74.91%	44.92%	53.06%
Min2Max3	41.67%	74.56%	43.22%	52.04%

The prediction results using SVM are shown in Table 4.4. There was an increase in accuracy between unigram and bigrams for Training data set only from 53.83% to 56.38%. This means that the data used for training generates an accurate hyperplane that is much more appropriate to the Testing data set. All the other data sets showed decreased accuracy value with increase in the value of N. Accuracy for Testing50 data set remained constant from trigram at 40.28%, Testing data set at 42.37% from trigram, and Training at 52.04% from 4gram onwards. Combined data set had two constant accuracy values; 74.08% between trigram and 5gram and 73.96% between 6gram and 10gram.

4.4.4 Experiment 9: Prediction Using KNN

Table 4.4: Prediction Results Using KNN

N-gram Size	**Testing50**	**Combined**	**Testing**	**Training**
Unigram	54.86%	79.88%	55.08%	50.26%
Bigram	59.72%	80.95%	57.63%	51.02%
Trigram	59.72%	80.95%	57.63%	50.51%
4gram	59.72%	80.95%	57.63%	50.51%
5gram	59.72%	80.95%	57.63%	50.51%
6gram	59.72%	80.95%	57.63%	50.51%
7gram	59.72%	80.95%	57.63%	50.51%
8gram	59.72%	80.71%	57.63%	50.51%
9gram	59.72%	80.71%	57.63%	50.51%
10gram	59.72%	80.71%	57.63%	50.51%
11gram	40.28%	68.99%	42.37%	52.08%
12gram	40.28%	68.76%	42.37%	51.79%
Min1Max2	61.81%	81.66%	57.63%	51.02%
Min1Max3	60.42%	81.18%	57.63%	51.02%
Min1Max4	59.72%	80.95%	57.63%	50.51%
Min2Max3	59.72%	80.95%	57.63%	50.51%

Table 4.4 shows the predicion results using KNN. All data sets showed increased accuracy with increase in N between unigram and bigram; 54.86% to 59.72% for Testing50, 79.88% to 80.95% for Combined, 55.08% to 57.63% for Testing, and 50.26% to 51.02% for Training data sets. This means that bigram features are much more appropriate for predicting sentiment neighbours as compared to unigrams. From bigram to 7gram, all data sets had a constant accuracy value similar to that with bigram, except for Training data set which had 50.51%. This constant means that increase in N-gram size did not affect document similarity function.

4.4.5 Experiment 10: Prediction Using DT

Table 4.4: Prediction Results Using DT

N-gram Size	Testing50	Combined	Testing	Training
Unigram	40.28%	68.17%	49.15%	56.89%
Bigram	40.28%	62.13%	44.92%	50.26%
Trigram	40.28%	58.22%	42.37%	50.51%
4gram	40.28%	58.22%	42.37%	50.51%
5gram	40.28%	58.22%	42.37%	50.51%
6gram	40.28%	58.22%	42.37%	50.51%
7gram	40.28%	58.22%	42.37%	50.51%
8gram	40.28%	57.99%	42.37%	50.51%
9gram	40.28%	52.78%	42.37%	50.51%
10gram	40.28%	58.22%	42.37%	50.51%
11gram	40.28%	49.23%	42.37%	50%
12gram	40.28%	49.23%	42.37%	50%
Min1Max2	39.58%	67.69%	50.85%	57.14%
Min1Max3	39.58%	67.54%	50%	57.14%
Min1Max4	39.58%	67.34%	48.31%	57.14%
Min2Max3	39.58%	62.13%	44.92%	50.26%

Table 4.4 shows accuracy results using DT. With DT, there was no increase in accuracy with increase in N value for all the data sets. DT works well with frequent and representative features, and this is why increase in N results in decreased accuracy values. All the data sets, except Testing50, registered constant accuracy values from trigram. Testing50 had the same accuracy value from bigram to 12gram. This decrease is attributed to increased data sparsity with increase in N value.

4.5 Training Using Long Sentiments

In this section, training data set with long sentiments was used for training. The performance of the testing phase on different classifiers and different data sets were then presented in tables.

4.5.1 Experiment 11: Prediction Using NB

Table 4.5: Prediction Results Using NB

N-gram Size	**Testing20**	**Combined**	**Testing**	**Training**
Unigram	56.71%	62.25%	42.37%	56.12%
Bigram	54.02%	59.41%	42.37%	50%
Trigram	54.41%	58.34%	42.37%	50%
4gram	54.41%	58.82%	42.37%	50%
5gram	54.41%	59.05%	42.37%	50%
6gram	54.41%	59.29%	42.37%	50%
7gram	54.41%	59.52%	42.37%	50%
8gram	54.41%	59.73%	42.37%	50%
9gram	54.41%	59.73%	42.37%	50%
10gram	54.41%	60%	42.37%	50%
11gram	54.41%	60.24%	42.37%	50%
12gram	54.41%	60.71%	42.37%	50%
Min1Max2	53.64%	59.41%	42.37%	50%
Min1Max3	54.41%	56.92%	42.37%	50%
Min1Max4	54.41%	56.69%	42.37%	50%
Min2Max3	54.41%	59.17%	42.37%	50%

Table 4.5 shows the prediction results using NB. Combined data set had the highest accuracy value for unigram at 62.25%. This is due to reduced feature

sparsity in the data set since the data set had the highest amount of sentiments formed by combining other data sets. There was decreased accuracy value from bigram to 12gram. This is attributed to increased data sparsity as a result of increased value of N. However, Combined data set had increased performance from trigram to 12gram. Training and Testing data sest started showing constant accuracy value from bigram while the Testing20 data set started having constant accuracy values from trigram.

4.5.2 Experiment 12: Prediction Using NBM

Table 4.5: Prediction Results Using NBM

N-gram Size	Testing20	Combined	Testing	Training
Unigram	57.09%	69.37%	52.54%	53.83%
Bigram	54.41%	67.57%	54.24%	51.53%
Trigram	52.11%	67.57%	57.63%	51.79%
4gram	46.36%	64.14%	57.63%	49.74%
5gram	45.59%	63.67%	57.63%	50%
6gram	45.59%	63.67%	57.63%	50%
7gram	45.59%	63.67%	57.63%	50%
8gram	45.59%	63.67%	57.63%	50%
9gram	45.59%	63.67%	57.63%	50%
10gram	45.59%	63.67%	57.63%	50%
11gram	45.59%	63.91%	57.63%	50%
12gram	45.59%	63.91%	57.63%	50%
Min1Max2	57.47%	64.02%	53.39%	54.85%
Min1Max3	47.51%	69.70%	51.69%	54.34%
Min1Max4	45.98%	64.02%	56.78%	51.28%
Min2Max3	54.79%	68.05%	54.24%	54.08%

The prediction results of NBM as represented in Table 4.5 show that only Testing data set showed increased accuracy value with increase in ngrams until trigrams, after which it became constant. This is as a result of larger vocabulary sizes in Testing50 data set similar to vocabulary in Testing data set. However, Combined data set had the highest accuracy value compared to the other data sets with similar value for N. Combined data set has more sentiments and longer sentiments and thus results in higher term frequency and inverse document frequency values. From unigram to bigram, accuracy value increased from 52.54% to 54.24%, and to 57.63% for trigrams. The remaining data sets had decreased accuracy value with increase in N from unigram to 12gram. Testing20, Combined, and Training data sets all started having a constant accuracy value from 5gram.

4.5.3 Experiment 13: Prediction Using SVM

Table 4.5: Prediction Results Using SVM

N-gram Size	**Testing20**	**Combined**	**Testing**	**Training**
Unigram	56.71%	71.01%	58.47%	56.38%
Bigram	53.64%	65.80%	39.83%	49.74%
Trigram	51.34%	67.46%	58.47%	51.53%
4gram	46.36%	64.38%	57.63%	49.74%
5gram	45.59%	63.91%	57.63%	50%
6gram	45.59%	63.91%	57.63%	50%
7gram	45.59%	63.91%	57.63%	50%
8gram	45.59%	63.91%	57.63%	50%
9gram	45.59%	63.91%	57.63%	50%
10gram	45.59%	63.91%	57.63%	50%
11gram	45.59%	63.91%	57.63%	50%
12gram	45.59%	63.91%	57.63%	50%
Min1Max2	49.43%	66.51%	58.47%	57.14%
Min1Max3	45.59%	63.91%	57.63%	50%
Min1Max4	49.43%	63.91%	57.63%	50%
Min2Max3	43.64%	66.04%	40.68%	49.23%

The prediction results for SVM, shown in Table 4.5, indicate that there was decreased accuracy performance with increase in N. However, from bigram to trigram, Combined, Testing, and Training data sets showed increased accuracy performance; from 65.80% to 67.46% for Combined data set, from 39.83% to 58.47% for Testing data set, and 49.74% to 51.53% for Training data set. This means that trigrams features are better at generating an accurate hyperplane compared to bigram since a similar trend cuts across all data sets. From 5gram, Testing20, Combined, and Training data sets all started showing constant accuracy values of 45.59%, 63.91%, and 50% respectively. Testing data set started having a constant value of 57.63% from 4gram.

4.5.4 Experiment 14: Prediction Using KNN

Table 4.5: Prediction Results Using KNN

N-gram Size	**Testing20**	**Combined**	**Testing**	**Training**
Unigram	60.54%	71.83%	56.78%	51.79%
Bigram	51.72%	65.56%	44.07%	50.26%
Trigram	54.79%	67.22%	42.37%	50%
4gram	54.79%	67.22%	42.37%	50%
5gram	54.79%	67.22%	42.37%	50%
6gram	54.41%	67.10%	42.37%	50%
7gram	54.41%	67.10%	42.37%	50%
8gram	54.41%	67.10%	42.37%	50%
9gram	54.41%	67.10%	42.37%	50%
10gram	54.41%	67.10%	42.37%	50%
11gram	54.41%	67.10%	42.37%	50%
12gram	54.41%	67.10%	42.37%	50%
Min1Max2	57.85%	70.77%	57.63%	52.55%
Min1Max3	45.59%	63.91%	57.63%	50%
Min1Max4	45.59%	63.83%	57.14%	50%
Min2Max3	52.11%	63.91%	57.63%	50%

KNN prediction results as represented in Table 4.5 indicate that with unigram, Combined data set had the highest accuracy value of 71.83%. From unigram to bigram, all data sets decreased in accuracy value, 60.54% to 51.72% for Testing20, 71.83% to 65.56% for Combined, from 56.78% to 44.07% for Testing, and 51.79% to 50.26% for Training data set. However, from bigram to trigram, Testing20 and Combined data sets showed increased accuracy values. This is as a result of increased N-gram lengths that make it appropriate to calculate similarity measures since Testing50, Combined, and Testing20 all consisted of longer sentiments.

4.5.5 Experiment 15: Prediction Using DT

Table 4.5: Prediction Results Using DT

N-gram Size	**Testing20**	**Combined**	**Testing**	**Training**
Unigram	59.39%	68.64%	43.22%	54.08%
Bigram	51.34%	59.53%	42.37%	53.32%
Trigram	45.59%	50.77%	57.63%	50%
4gram	45.59%	50.77%	57.63%	50%
5gram	45.59%	50.77%	57.63%	50%
6gram	45.59%	50.77%	57.63%	50%
7gram	45.59%	50.77%	57.63%	50%
8gram	45.59%	50.77%	57.63%	50%
9gram	45.59%	50.77%	57.63%	50%
10gram	45.59%	50.77%	57.63%	50%
11gram	45.59%	50.77%	57.63%	50%
12gram	45.59%	50.77%	57.63%	50%
Min1Max2	45.21%	61.07%	53.39%	52.30%
Min1Max3	44.06%	59.41%	53.39%	51.79%
Min1Max4	55.94%	65.56%	37.29%	51.02%
Min2Max3	51.34%	59.53%	42.37%	53.32%

The performance of DT, as presented in Table 4.5, show that all data sets had decreased accuracy performance value from unigram to bigram. This is as a result of decrease in frequent phrases due to increase in N. However, accuracy value for Testing data set increased from bigram to trigram, which then remained constant at 57.63% from trigram to 12gram. All data sets started having a constant accuracy value from trigram.

4.6 Training Using Combined Length Sentiments

In this section, training data set with a mixture of short, medium, and long sentiments was used as training data and the performance results recorded in tables.

4.6.1 Experiment 16: Prediction Using NB

Table 4.6: Prediction Results Using NB

N-gram Size	**Testing20**	**Testing50**	**Testing**	**Training**
Unigram	79.69%	74.31%	46.61%	61.73%
Bigram	76.25%	84.03%	42.37%	50%
Trigram	76.25%	81.94%	42.37%	50.51%
4gram	77.39%	84.03%	42.37%	50.51%
5gram	80.84%	84.03%	42.37%	50.51%
6gram	79.69%	85.42%	42.37%	50.51%
7gram	82.76%	85.42%	42.37%	50.51%
8gram	83.91%	85.42%	42.37%	50.51%
9gram	86.21%	85.42%	42.37%	50.51%
10gram	86.59%	85.42%	42.37%	50.51%
11gram	86.59%	85.42%	42.37%	50.51%
12gram	86.21%	85.42%	42.37%	50.51%
Min1Max2	76.25%	71.53%	41.53%	52.55%
Min1Max3	73.18%	67.36%	41.53%	51.79%
Min1Max4	72.41%	64.58%	42.37%	51.27%
Min2Max3	75.10%	65.97%	42.37%	50.51%

All prediction accuracy from unigram to bigram reduced for all data sets except for Testing50 as shown in Table 4.6; 79.69% to 76.25% for Testing20, 46.61% to 42.37%, and 61.73% to 50% for Training data set. Testing50 data set had an increased accuracy value from 74.31% to 84.03%. However, from 4gram, Testing20 and Testing50 data sets showed increased accuracy value. This can be attributed to the fact that when the size of the ngram increases, only data sets with longer sentiments are likely to have longer sentiments occurring as feature sets.

4.6.2 Experiment 17: Prediction Using NBM

Table 4.6: Prediction Results Using NBM

N-gram Size	**Testing20**	**Testing50**	**Testing**	**Training**
Unigram	94.25%	93.75%	70.34%	64.29%
Bigram	96.93%	96.53%	68.64%	59.95%
Trigram	90.80%	92.35%	64.41%	54.59%
4gram	89.27%	95.14%	59.32%	51.02%
5gram	89.27%	95.14%	57.63%	50.51%
6gram	89.27%	95.83%	57.63%	50.51%
7gram	88.89%	95.83%	57.63%	50.51%
8gram	88.51%	95.83%	57.63%	50.51%
9gram	88.51%	95.83%	57.63%	50.51%
10gram	88.51%	95.83%	57.63%	50.51%
11gram	88.51%	96.53%	57.63%	50.51%
12gram	88.12%	96.53%	57.14%	50.51%
Min1Max2	82.38%	72.22%	57.63%	58.67%
Min1Max3	75.86%	61.81%	48.31%	57.63%
Min1Max4	72.03%	61.81%	49.15%	57.40%
Min2Max3	95.79%	97.22%	65.25%	55.87%

Results for prediction using NBM are presented in Table 4.6. Only Testing20 and Testing50 data sets had increased accuracy value from unigram to bigram; 94.25% to 96.93% for Testing and 93.75% to 96.53% for Testing50. From trigram to 12gram Testing20 showed decreased accuracy value while Testing50 showed increased accuracy values. Again, this is attributed to the characteristic of the data set used for training and Testing20 and Testing50 data sets. With the increase in N, the term frequency and inverse document frequency reduces resulting in decreased accuracy. They are made up of long sentiments as compared to Testing and Training data sets.

4.6.3 Experiment 18: Prediction Using SVM

Table 4.6: Prediction Results Using SVM

N-gram Size	**Testing20**	**Testing50**	**Testing**	**Training**
Unigram	99.23%	99.31%	95.76%	65.05%
Bigram	98.85%	100%	84.75%	51.53%
Trigram	92.92%	99.31%	70.34%	52.55%
4gram	89.27%	96.53%	60.17%	50.51%
5gram	89.27%	96.53%	57.63%	50.51%
6gram	89.27%	96.53%	57.63%	50.51%
7gram	88.89%	96.53%	57.63%	50.51%
8gram	88.51%	96.53%	57.63%	50.51%
9gram	88.51%	96.53%	57.63%	50.51%
10gram	88.51%	96.53%	57.63%	50.51%
11gram	88.51%	96.53%	57.63%	50.51%
12gram	88.12%	96.53%	57.63%	50.51%
Min1Max2	99.23%	99.31%	94.92%	63.01%
Min1Max3	99.23%	99.31%	91.53%	59.13%
Min1Max4	99.23%	99.31%	91.53%	56.12%
Min2Max3	99.23%	100%	84.75%	45.49%

Predictions using SVM showed better accuracy values compared to NBM as shown in Table 4.6. These are very good accuracy results and is because support vector machines takes into account only the data points close to the dividing hyperplane for predictions (support vectors), and are expected to be in much smaller number than the entire data set, thus providing an algorithm with good performance during execution time. Increasing the value of N increases the sparsity of support vectors thus reducing the accuracy. Only Testing50 data set had increased accuracy value from unigram to bigram; from 99.31% to 100%. Testing50 showed a slight decrease; from 99.23% to 98.85%. All the data sets started displaying constant accuracy values from 4gram, except for Testing data set.

4.6.4 Experiment 19: Prediction Using KNN

Table 4.6: Prediction Results Using KNN

N-gram Size	**Testing20**	**Testing50**	**Testing**	**Training**
Unigram	98.85%	100%	100%	53.55%
Bigram	99.23%	100%	88.98%	51.28%
Trigram	92.72%	99.31%	70.34%	51.53%
4gram	89.27%	96.53%	60.17%	50.51%
5gram	89.27%	96.53%	58.47%	50.51%
6gram	89.27%	96.53%	57.63%	50.51%
7gram	88.89%	96.53%	57.63%	50.51%
8gram	88.51%	96.53%	57.63%	50.51%
9gram	88.51%	96.53%	57.63%	50.51%
10gram	88.51%	96.53%	57.63%	50.51%
11gram	88.51%	96.53%	57.63%	50.51%
12gram	88.12%	96.53%	57.63%	50.51%
Min1Max2	98.47%	100%	97.46%	55.61%
Min1Max3	98.85%	100%	94.92%	52.55%
Min1Max4	98.85%	100%	91.53%	46.68%
Min2Max3	99.23%	100%	97.29%	51.02%

Table 4.6 describes prediction results using KNN. Only Testing20 data set showed increased accuracy value from unigram to bigram; from 98.85% to 99.23%.

Testing50 had the same accuracy value of 100% for unigram and for bigram, from which it started decreasing. Testing20, Testing50, and Training data sets all started having a constant accuracy value from 4gram, while Testing data set started showing a constant value from 6gram. Increasing the value of N reduces the accuracy since k closest instances in the data set to a new occurrence that needs to be classified are reduced, and making a prediction based on what classes the minority of the k neighbours belong to.

4.6.5 Experiment 20: Prediction Using DT

Table 4.6: Prediction Results Using DT

N-gram Size	**Testing20**	**Testing50**	**Testing**	**Training**
Unigram	96.93%	97.92%	80.51%	48.72%
Bigram	82.38%	79.17%	47.46%	50.51%
Trigram	68.97%	40.28%	42.37%	50.51%
4gram	68.97%	40.28%	42.37%	50.51%
5gram	68.97%	40.28%	42.37%	50.51%
6gram	60.92%	40.28%	42.37%	50.51%
7gram	68.97%	40.28%	42.37%	50.51%
8gram	60.92%	40.28%	42.37%	50.51%
9gram	60.92%	40.28%	42.37%	50.51%
10gram	60.15%	40.28%	42.37%	50.51%
11gram	45.59%	59.72%	57.63%	50%
12gram	45.59%	59.72%	57.63%	50%
Min1Max2	97.79%	96.53%	75.42%	57.40%
Min1Max3	95.40%	96.53%	72.88%	57.65%
Min1Max4	95.40%	96.53%	72.03%	59.44%
Min2Max3	82.38%	79.17%	47.46%	50.51%

In this experiment, DT also demonstrated increased performance as indicated in Table 4.6. Testing50 had the highest accuracy value for unigram at 97.92%, followed by Testing20 at 96.93%, Testing, and Training respectively. All data sets had decreased accuracy performance from unigram to bigram; 96.93% to 82.38% for Testing20, 97.92% to 79.17% for Testing50, 80.51% to 47.46% for Testing, and 48.72% for Training data set. This decrease is as a result of decreased frequency in phrases with increase in the value of N.

Table 4.6: NB Average Performance Results with Varying Lengths

N-gram size	**Size of Training Data**				**Average**
	Short	Medium	Long	Combined	
Unigram	59.69%	50.63%	54.36%	65.59%	57.57%
Bigram	52.22%	49.47%	51.47%	63.91%	55.02%
Trigram	51.58%	48.85%	51.28%	62.77%	53.62%
4gram	51.52%	48.85%	51.40%	63.58%	53.84%
5gram	50.19%	49.21%	51.46%	64.44%	53.83%
6gram	50.19%	50.04%	51.52%	64.50%	54.06%
7gram	50.19%	50.35%	51.58%	65.27%	54.35%
8gram	49.95%	50.88%	51.63%	65.55%	54.50%
9gram	49.95%	51.15%	51.63%	66.13%	54.71%
10gram	49.95%	51.27%	51.70%	66.22%	54.79%
11gram	48.86%	51.36%	51.76%	66.22%	54.55%
12gram	48.86%	51.36%	51.76%	66.22%	54.55%
Average	**50.94%**	**50.28%**	**51.81%**	**65.03%**	
Min1Max2	57.74%	59.90%	51.36%	60.47%	57.37%
Min1Max3	55.85%	49.11%	50.93%	58.47%	53.59%
Min1Max4	54.59%	48.98%	50.87%	57.66%	53.03%
Min2Max3	51.99%	48.98%	51.49%	58.49%	52.74%
Average	**55.04%**	**51.74%**	**51.16%**	**58.77%**	

Naive Bayes uses presence or absence of features in creating classification model. When there is scarcity in training data, absent terms causes zero probability problem. With boolean indicators, it takes words that do not appear in the sentiments into account. Naive Bayes performed well with unigram in short and long training data sets and as well as average performance for all the sentiment lengths. However, medium and combined data sets had high accuracy values at 11gram and 10gram respectively. Naive Bayes performs well with small vocabulary sources and this explains comparative high accuracy values on short sentiments compared to medium sentiments for low N values. The increase of classification performance between medium, long and combined data sets is as a result of large size of vocabulary that arises from increase sentiment length. There is a general trend of increasing accuracy from 7gram with increasing sentiment length.

Table 4.6: NBM Average Performance Results with Varying Training data sets

N-gram size	**Size of Training Data**				**Average**
	Short	Medium	Long	Combined	
Unigram	58.95%	58.37%	58.21%	80.66%	64.05%
Bigram	56.16%	59.80%	56.94%	80.51%	63.35%
Trigram	56.41%	56.43%	57.28%	75.54%	61.42%
4gram	53.63%	53.18%	54.47%	73.69%	58.74%
5gram	53.27%	52.63%	54.22%	73.14%	58.22%
6gram	53.51%	52.41%	54.22%	73.31%	58.36%
7gram	53.31%	52.16%	54.22%	73.22%	58.23%
8gram	53.74%	52.16%	54.22%	73.12%	58.31%
9gram	53.74%	52.16%	54.22%	73.12%	58.31%
10gram	53.74%	52.16%	54.22%	73.12%	58.31%
11gram	53.74%	52.16%	54.28%	73.30%	58.36%
12gram	53.74%	51.98%	54.28%	73.08%	58.27%
Average	**53.66%**	**53.80%**	**55.07%**	**74.65%**	
Min1Max2	58.95%	59.66%	57.43%	67.73%	60.94%
Min1Max3	59.75%	58.95%	55.81%	60.90%	58.85%
Min1Max4	58.12%	57.40%	54.52%	60.10%	57.54%
Min2Max3	56.92%	59.13%	57.79%	78.53%	63.09%
Average	**58.44%**	**58.79%**	**56.39%**	**66.82%**	

Unlike NB which uses presence or absence, NBM uses weights or frequency in vector representation of features calculated using TFIDF function. Increasing the size of N reduces the frequency of phrases and thus results in low weights and frequency values. This reduces the accuracy performance as N increases. Combined data set has many sentiments with varying sizes and this provides a representative set of features for learning. This is evident with its high accuracy performance. There is an increasing trend with medium, long and combined data sets for all N sizes.

Table 4.6: SVM Average Performance Results with Varying Training data sets

N-gram size	**Size of Training Data**				**Average**
	Short	Medium	Long	Combined	
Unigram	62.05%	57.25%	60.64%	89.84%	67.45%
Bigram	59.79%	55.82%	52.25%	83.78%	62.91%
4gram	53.17%	52.19%	54.53%	74.12%	58.50%
5gram	53.27%	52.19%	54.28%	73.49%	58.31%
6gram	50.19%	52.16%	54.28%	73.49%	57.53%
7gram	50.19%	52.16%	54.28%	73.39%	57.51%
8gram	49.95%	52.16%	54.28%	73.30%	57.42%
9gram	49.95%	52.16%	54.28%	73.30%	57.42%
10gram	48.86%	52.16%	54.28%	73.30%	57.15%
11gram	48.86%	52.11%	54.28%	73.30%	57.14%
12gram	49.05%	51.98%	54.28%	73.20%	57.13%
Average	**52.49%**	**52.79%**	**54.91%**	**76.11%**	
Min1Max2	62.62%	58.83%	57.89%	89.12%	67.12%
Min1Max3	60.12%	56.99%	54.28%	87.3%	64.67%
Min1Max4	49.68%	53.64%	55.24%	86.55%	61.28%
Min2Max3	60.33%	52.87%	49.90%	82.37%	61.37%
Average	**58.19%**	**55.58%**	**54.33%**	**86.34%**	

SVM showed high classification accuracy with unigrams. SVM performed more accurately with high dimensional data in combined data set. SVM does not require the assumption that some features are irrelevant and even the lowest ranked features according feature selection methods contain considerable information. There is a general upward trend in classification performance from short to combined data sets as is support by accuracy by sentiment size averages. SVM is a statistical classification method and seeks a decision surface to separate training data points into two classes and makes decisions based on support vectors that are selected as the only effective elements in the training set. Unigrams produced the best vectors for classification. Increasing N reduces the effectiveness of support vectors. Min1Max2 also has the highest classification accuracy.

Table 4.6: KNN Average Performance Results with Varying Training data sets

N-gram size	**Size of Training Data**				**Average**
	Short	Medium	Long	Combined	
Unigram	61.40%	60.02%	60.24%	88.10%	67.44%
Bigram	58.15%	62.33%	52.90%	84.87%	64.56%
Trigram	53.51%	62.20%	53.60%	78.48%	61.95%
4gram	53.51%	62.20%	53.60%	72.12%	60.86%
5gram	53.51%	62.20%	53.60%	73.70%	60.75%
6gram	53.51%	62.20%	53.47%	73.49%	60.67%
7gram	53.51%	62.20%	53.47%	73.39%	60.64%
8gram	53.74%	62.14%	53.47%	73.30%	60.66%
9gram	47.55%	62.14%	53.47%	73.30%	59.12%
10gram	47.55%	62.14%	53.47%	73.30%	59.12%
11gram	47.55%	53.43%	53.47%	73.30%	56.94%
12gram	47.21%	50.80%	53.47%	73.20%	56.17%
Average	**52.56%**	**60.33%**	**54.02%**	**76.05%**	
Min1Max2	56.97%	63.03%	59.70%	87.89%	66.90%
Min1Max3	53.55%	62.56%	54.28%	86.58%	64.24%
Min1Max4	53.86%	62.20%	54.14%	84.27%	63.62%
Min2Max3	53.49%	62.20%	55.91%	86.89%	64.62%
Average	**54.47%**	**62.50%**	**56.01%**	**86.41%**	

KNN classification accuracy was highest with unigram for all the sentiments length except for medium which had bigram having the highest accuracy. KNN calculates similarity of all samples in training sample sets, then chooses K samples whose similarity is higher or the text to be necessary to calculate the similarity with training sample set of all samples. From the results in Table 4.6, unigram has the text that gives the highest similarity value. Accuracy reduced with increase in N. When there is an increase in N, time and space complexity increases and text classification accuracy is reduced.

Table 4.6: DT Average Performance Results with Varying Training data sets

N-gram size	**Size of Training Data**				**Average**
	Short	Medium	Long	Combined	
Unigram	56.40%	53.62%	56.33%	80.95%	61.83%
Bigram	48.25%	49.40%	51.64%	64.88%	53.54%
Trigram	53.43%	47.85%	51.00%	50.33%	50.70%
4gram	53.43%	47.85%	51.00%	50.53%	50.70%
5gram	53.43%	47.85%	51.00%	50.33%	50.70%
6gram	53.43%	47.85%	51.00%	48.52%	50.20%
7gram	53.43%	47.85%	51.00%	50.53%	50.70%
8gram	53.43%	47.79%	51.00%	48.52%	50.19%
9gram	53.04%	46.49%	51.00%	48.52%	49.76%
10gram	53.42%	47.85%	51.00%	50.53%	50.70%
11gram	53.42%	45.47%	51.00%	53.24%	50.78%
12gram	53.72%	45.47%	51.00%	53.24%	50.78%
Average	**53.21%**	**47.95%**	**51.50%**	**54.24%**	
Min1Max2	59.66%	53.82%	53.00%	81.79%	62.07%
Min1Max3	59.93%	53.57%	52.16%	80.62%	61.57%
Min1Max4	58.33%	53.09%	52.45%	80.85%	61.18%
Min2Max3	48.39%	49.22%	51.64%	64.88%	53.53%
Average	**56.58%**	**52.43%**	**52.31%**	**77.04%**	

Decision Trees had the highest accuracy with unigrams as shown in Table 4.6. Every sentiment instance is covered by a single rule or tree path and a single rule only and also covers key regularities in the training data set. In decision trees, features that are more obvious, frequent and representative to establish decision trumps results in higher accuracy values. As the value of N increased, the phrases frequency reduced and therefore did not contribute much to the entropy. This also increased error chances since a single mistake in a higher upper level in the decision tree causes a whole subtree to be invalid, and thus results into wrong classifications of sentiments. Similar trend is evidence in varying the value of N, where Min1Max2 produced the highest accuracy values for all the sentiment lengths except in short sentiments.

4.7 Summary of Results by Average Values

This section discusses the results of various experiments. It starts by discussing the effect of size of the training data sentiments on classifier performance. It will then describe the performance trend for classifiers used against the size of the sentiments. The effect of quality of training data set on classifier performance will then be examined. Finally, the difference between fixed size and variable size N-grams will be compared. Based on results of this research, the following conclusions were made.

The size of sentiments used for training has an effect on the classifier performance. Machine learning algorithms rely on selected features from training data to infer the similarities or commonalities that a group of sentiments share and that discriminate them from the rest of the sentiments. The success of classifiers therefore rely on the relevance of the features for discriminating between class labels. The longer the sentiments, the more the features used in constructing classifier and the better the model. This is supported by the high levels of classification accuracy exhibited when short, medium, long, and Combined data sets showed that there is increased classification accuracy as the length of the sentiments used for training increased.

The method used for classification vary with the size of the sentiments used for training classifiers. Not all classifiers gave the same accuracy value even if the same training and testing data were used. Some gave low accuracy values and other showed high accuracy values. For example, looking at the results presented in Table 4.3, Table 4.4, Table 4.5, and Table 4.6 for SVM showed that classification accuracy increased with increase in size of sentiments for the same N-gram size. Similar results are supported by results from DT (Table 4.3, Table 4.4, Table 4.5, and Table 4.6), NBM (Table 4.3, Table 4.4, Table 4.5, and Table 4.6), SVM (Table 4.3, Table 4.4, Table 4.5, and Table 4.6), and NB (Table 4.3, Table 4.4, Table 4.5, and Table 4.6) classifiers. This is because different classifiers develop models in different ways, some of which depend on the amount of features used in learning and classification. There are also classifiers which work well with high dimensional data e.g. SVM while performance of KNN will reduce when dealing with high dimensional data.

There is a clear behaviour of N-grams when used with different sizes of training sentiments. For the same N value, longer sentiments had higher accuracy compared to medium and short for different classifiers. There were cases where short sentiments had higher accuracy values than medium, but only in minority cases. Size of training sentiments helps develop a better model. In cases where classification is based on weights and frequency, longer sentiments provide more frequent and representative features which results in higher accuracy.

There is a consistent performance of N-grams with different classifiers. Most classifiers had high accuracy values with unigram and bigram. Bigram features incorporate some contextual information which is important for sentiment classification and also generally contain a large number of noisy features and sparse matrix of terms. The length of descriptive phases normally ranges from unigram to bigram and therefor three consecutive or more phrase rarely happens. It is therefore hard for trigram or more to get higher accuracy performance. Large values for N generates more sparse and unique features for each data set. The actual correlation between sentiment cannot be achieved and this decreases performance of a classifier.

Quality of data used for training affect performance of various classifiers. Machine learning classifiers rely on selected features and training data to infer the similarities that a group of sentiments share and that differentiate them from the rest of the sentiments. The success of classifiers therefore depends on the relevance of features between training data and other sentiments. Accuracy of classifiers is dependent on quantity and quality of training data sets. When the quality of training data set is not good, classification effect is not obvious. When the training set is small, there will be over-fitting phenomenon, which eventually affects classifier performance and performance suffers. The quality of training data for being unbiased and representative of the sentiments will give higher classifier performance. Some classifiers like KNN do not build an explicit, declarative model of sentiment class, but relies on the category labels attached the training sentiments similar to the test sentiment. This means that if training data is not properly labeled, classification performance will decrease.

There is a small difference in classification accuracy when using fixed N-gram and variable N-gram size. For this experiment, variable N-gram size was Min1Max2, Min1Max3, Min1Max4, and Min2Max3, and this can be compared to fixed size N-gram for unigram, bigram, trigram, and 4gram respectively. Varying N-gram size produces features that are similar or almost similar to unigram, bigram, trigram and 4gram features. As a result, they are likely to produce the same effect on the classifier.

Chapter FIVE

CONCLUSION AND FUTURE WORK

5.1 Knowledge Contributions

This researched proposed that the use of N-grams outperforms bag of words model for sentiment classification. Also, this research proposed that longer length training data sets for sentiments improves accuracy performance of classifiers. This PhD thesis provided a set of contributions that can be summarized while considering different points of view. On the more theoretical, modelling side, word N-gram modeling for sentiment analysis was proposed.

On the implementation side, this research improved results on accuracy with increase in sentiment length. This is supported by the high levels of classification accuracy exhibited when short, medium, long, and Combined data sets showed that there is increased classification accuracy as the length of the sentiments used for training increased.

One of the research objectives was to compare the behaviour of N-grams when used with different sizes of training data sentiments. On this objective, this research contributed by proposing the use of longer sentiments for N-gram sentiment classification. The performance of bag-of-words model and N-grams model was analyzed using different characteristics of data sets, machine learning algorithms, and classification performance evaluated using accuracy. This research proposed N-gram modeling technique for sentiment classification using longer sentiments. Quality of data used for training affect performance of various classifiers.

Another objective was to determine the effect of data set quality on the performance of various classifiers. Machine learning classifiers rely on selected features and training data to infer the similarities that a group of sentiments share and that differentiate them from the rest of the sentiments. The success of classifiers therefore depends on the relevance of features between training data and testing sentiments. This research proposed use of more sentiments with mixed length sentiments for training classifiers for better models.

Another research objective was to identify patterns of accuracy performance of machine learning algorithms with respect to the value of N in N-grams. This researched proposed use of different N-gram sizes for different classifiers depending on the methods classifiers use in constructing models. Some methods perform better with unigrams, bigrams, trigrams, and others with mixed N-grams. Most of the classifiers used had high accuracy values with unigram, bigrams, and trigram and decreasing accuracy from 4grams onwards.

Another objective was to compare the behaviour of N-grams when used with different sizes of training data sentiments.This research proposed the use of longer sentiments for better accuracy performance with increasing N-grams.

5.2 Conclusion

The main research objective for this research was to analyze the effect of N-gram document modeling technique on performance of machine learning algorithms. Different sentiments of different sizes were used for training and prediction and results showed improved prediction results with increased sentiment length. Current trends show that tracking and monitoring people is becoming an integral part of everyday life. Data-driven approaches (Naive Bayes, Support Vector Machines, Nearest Neighbours, and decision trees) are described in many literature. N-gram feature such as characters, words have been used in sentiment classification. This research experimented with word N-grams with considers sequence of two or more words as features. The accuracy of different N values were experiment with different classifiers. Also, different N-gram sizes were also experimented with different lengths of sentiment data sets. The results of experiments were presented in tables and analyzed.

5.3 Future Work

N-gram is language independent and can be used for many natural languages. However, there is need for more research to be done on contextual knowledge being incorporated as part of feature selection and model creation for specific domains where precise context, which does not depend on word ordering needs to be used in learning and prediction is required. Also, there is need to experiment N-gram models with real live streaming text data. This research can also be enhanced by experiment with more sentiments in training and testing data sets. A hybrid framework for sentiment classification that includes bag of words, N-grams, Skip-grams, and contextual knowledge into learning and prediction phases of sentiment classification should also be explored.

REFERENCES

Agarwal, B., Mittal, N., & Cambria, E. (2013). Enhancing Sentiment Classification Performance Using Bi-Tagged Phrases. In *Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on* (pp. 892-895). IEEE.

Akshay, B., Amit, B., Kunal, S., & Swapnali, K. (2016). Predictive and Corrective Text Input for desktop editor using n-grams and suffix trees. *International Conference on Advances in Human Machine Interaction,* Bangalore, India.

Alaa, A., Arash, J., & Abdulhussain, E., M. (2014). Combining Bag-of-Words and Bag-of-Concepts Representations for Arabic Text Classification. *ISSC 2014*, Limerick.

Aldy R., A., & Ayu, P. (2015). Comparison on the Rule based Method and Statistical based Method on Emotion Classification for Indonesian Twitter Text. *2015 International Conference on Information Technology Systems and Innovation (ICITSI)*, Bandung.

Aizhang, G., & Tao, Y. (2015). Based on Rough Sets and the associated analysis of KNN Text Classification Research. In *2015 14th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)* (pp. 485-488). IEEE.

Al-Ghuribi, S. M., & Alshomrani, S. (2013). A Simple Study of Webpage Text Classification Algorithms for Arabic and English Language. in *IT Convergence and Security (ICITCS), 2013 International Conference on* (pp. 1-5). IEEE.

Allix, K., Bissyande, T. F., Jerome, Q., Klein, J., State, R., & Le Traon, Y. (2014). Empirical assessment of machine learning-based malware detectors for Android. *Empirical Software Engineering*, 1-29.

Anagha, K., Vrinda, T., & Parag, K. (2015). Term Weighting using Contextual Information for Categorization of Unstructured Text Documents. *IEEE INDICON.*

Anahita, G., A., Jamilu, A., & Azuraliza, B., A. (2014). Comparative Analysis of Algorithms used in Supervised Classification: A Case of Bank Notes Data Sets. *International Journal of Computer Trends and Technology, 7*, 39-43.

Androutsopoulos, I., Koutsias, J., Chandrinos, K., V., & Spyropoulos, C., D. (2000). An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages. In *Proceedings of the 23rd annual International Conference on Research and development in Information Retrieval* (pp. 160-165). ACM.

Beliga, S., & Martincic-Ipsic, S. (2014). Non-Standard Words as Features for Text Categorization. In *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on* pp. 1165-1169). IEEE.

Bhuta, S., Doshi, A., Doshi, U., & Narvekar, M. (2014). A Review of Techniques for Sentiment Analysis of Twitter Data. In *Issues and Challenges in Intelligent Computing Techniques (ICICT), 2014 International Conference on*, pp. 583-591. IEEE.

Bouras, C., & Tsogkas, V. (2016). Assisting Cluster Coherency via n-grams and clustering as a tool to deal with the new user problem. *International Journal of Machine Learning and Cybernetics,* 1-14.

Burnap, P., & Williams, M. L. (2016). Us and them: identifying cyber hate on Twitter across multiple protected characteristics. *EPJ Data Science, 5*(1), 1.

Caropeso, M., F., Matwin, S., & Sabastiani, F. (2001). A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization, *Text databases and document management: Theory and practice,* 78-102.

Chandra, G., & Dwivedi, S. K. (2014). A Literature Survey on Various Approaches of Word Sense Disambiguation. In *Computational and Business Intelligence (ISCBI), 2014 2nd International Symposium on* (pp. 106-109). IEEE.

Cios, K., J., Pedrycz, W., Swiniarski, R., W., & Kurgan, L., A. (2007). *Data Mining: A Knowledge Discovery Approach*, New York: Springer Science and Business Media.

Coppin, B. (2004). *Artificial Intelligence Illuminated,* Boston: Jones and Bartlett.

Dalal, M. K. & Zaveri, M. A. (2013). Automatic Classification of Unstructured Blog Text. *Journal of Intelligent Learning Systems and Applications, 5*(2), 108.

Dashen, X., & Fengxin, L. (2015). Research of Text Categorization Model based on Random Forests. *2015 IEEE International Conference on Computational Intelligence and Communication Technology*, 173-176. DOI 10.1109/CICT.2015.101.

Deepak G., & Narsimha, M. (2012). Efficient classification using phrases generated by topic models. *21st International Conference on Pattern Recognition (ICPR 2012)*,2331 - 2334, Tsukuba, Japan.

Dhande, L., & Patnaik, G. (2014). Analyzing Sentiment of Movie Review Data using Naive Bayes Neural Classifier. *International Journal of Emerging Trends and Technology in Computer Science, 3*, 313-320.

Dulac-Arnold, G., Ludovic D., L., & Gallinari, P. (2011). Text Classification: A Sequential Reading Approach. *Advances in Information Retrieval*, 411–423.

Engelstad, P. E., Hammer, H., Yazidi, A., & Bai, A. (2015). Analysis of time-dependencies in automatic security classification. In *Cyber-Enabled Distributed Computing and Knowledge Discovery (Cyberc), 2015 International Conference on* (pp. 54-61). IEEE.

Farhoodi, M., & Yari, A. (2010). Applying Machine Learning Learning Algorithms for Automatic Persian Text Classification. In *2010* 6th *Advanced Information Management and Service (IMS), 2010 6th International Conference on (pp. 318-323).* IEEE.

Farhoodi, M., Yari, A., & Sayah, A. (2011). N-Gram Based Text Classification for Persian Newspaper Corpus. In *Digital Content, Multimedia Technology and its Applications (IDCTA), 2011 7th International Conference on* (pp. 55-59). IEEE.

Feldman, R., & Sanger, J. (2007). *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data.* Cambridge, Cambridge University Press.

Feng, L., Zuo, W., & Wang Y. (2015). Improved Comprehensive Measurement Feature Selection Method for Text Categorization . In *Network and Information Systems for Comuters (ICNISC), 2015 International Conference on* (pp. 125-128). IEEE.

Foroozan, S., Murad, M., A., A., Sharef, N., M., & Latiff, A., R., A. (2015). Improving Sentiment Classification Accuracy of Financial News using N-gram Approach and Feature Weighting Methods, In *Information Science and Security (ICISS), 2015 2nd International Conference on* (pp. 1-4). IEEE.

Freund, G. E., & Willett P. (1982). Online identification of word variants and arbitrary truncation searching using a string similarity measure. *Information Technology; Research and development, 1*(3), 177-187.

Gaigole, P., C., Patil, L., H., & Chaudhari, P., M. (2013). Preprocessing Techniques in Text Categorization. *International Journal of Computer Applications*, 1-3.

George, P, George, G., & Georgios, P. (2015). Graph vs. bag representation models for the topic classification of web documents. Springer. DOI 10.1007/s11280-015-0365-x

Gutierrez, Y., Thomas, D., & Fernandez, J. (2015). Benefits of using Ranking Skip-Gram Techniques for Opinion Mining Approaches. In *eChallenges e-2015 Conference* (pp. 1-10). IEEE.

Gulin, V. V., & Frolov, A. B. (2016). Categorization of Text Documents Taking into Account Some Structural Features. *Journal of Computer and Systems Sciences International, 55*(1), 96–105.

Haddoud, M., Mokhtari, A., Lecroq, T.,& Abdeddaim, S. (2016). Combining Supervised term-weighting metrics for SVM text Classification with extended term representation. *Knowledge and Information Systems*, 1-23.

Hamouda, S. B., & Akaichi, J. (2013). Social Networks Text Mining for Sentiment Classification: The case of Facebook statuses updates in the Arabic Spring Era, *International Journal of Application or Innovation in Engineering and Management, 2*(5), 470-478.

Han, J. & Kamber, M. (Ed.) (2006). *Data Mining: Concepts and Techniques*, San Francisco: Morgan Kaufmann.

Hu, X., Manna, S., & Rruong, B. N. (2014). Product Aspect Identification: Analyzing Role of Different Classifiers. In *Computational Intelligence and Data Mining (CIDM), 2014 IEEE Symposium on* (pp. 202-209). IEEE.

Hussein, A. S. (2015). Arabic Document Similarity Analysis using N-grams and Singular Value decomposition. In *Research Challenges in Informa-*

tion Science (RCIS), 2015 IEEE 9th International Conference on (pp. 445-455). IEEE.

Jones, M. T. (2008). Artificial Intelligence: A Systems Approach, New Delhi: Infinity Science Press.

Jurafsky, D. & Martin, J., M., Peter, N., & Stuart, R. (2014). Speech and Language Processing. Springer Science and Business Media

Kadmateekarun, P. & Nuanmeesri, S.(2015). Automatic Sentiment Analysis from Opinion of Thais Speech Audio. In *Science and Technology (TICST), 2015 International Conference on* (pp. 288-291). IEEE.

Kaiser, C., & Bodendorf, F. (2012). Mining consumer dialog in online forums, *Internet Research, 22*(3), 275-297.

Korde, V., & Mahender, C. N. (2012). Text Classification and Classifiers: A Survey. *International Journal of Artificial Intelligence and Applications, 3*(2), 85.

Khreisat, L. (2006). Arabic Text Classification Using N-Gram Frequency Statistics: A Comparative Study. *DMIN*, 78-82.

Kilimci, Z. H., & Ganiz, M. C. (2015). Evaluation of Classification Models for Language Processing. In *Innovations in Intelligent SysTems and Applications (INISTA), 2015 International Symposium on* (pp. 1-8). IEEE.

Kristin, G. & Richard, C. (2012). Estimating the sentiment of social media content for security informatics applications. *Journal of Security Informatics, 1*(1), 1-16.

Kulkarni, P., Stranieri, A., Kulkarni, S., Ugon, J., and Mittal. (2014). Visual Character n-grams for classification and retrieval of radiological images. *International Journal of Multimedia and its applications, 6*, 35-49.

Larkey, L., S. (1999). A patent search and classification system. *Proceedings of the 4th ACM Conference on Digital Libraries* (pp. 90-95). ACM.

Lee, H. Y., & Lee, L. (2014). Improved Semantic Retrieval of Spoken Content by Document/Query Expansion with Random Walk Over Acoustic Similarity Graphs. *Audio, Speech, and Language Processing, IEEE/ACM Transactions on, 22*(1), 80-94.

Liu, B. (2007). *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*, New Jersey: Prentice Hall.

Liu, B. (2012). *Sentiment Analysis and Opinion Mining*, San Rafael: Morgan and Claypool Publishers.

Majumder, P., Mitra, M., & Chaudhuri, B., B. (2002). N-gram: A language independent approach to IR and NLP. In *International Conference on universal knowledge and language.*

Mohan, A., Baggili, I. M., & Rogers, M. K. (2010). Authorship attribution of SMS messages using an n-grams approach. *Proceedings of CERIAS Tech Report 2010-11, Center for Education and Research Information Assurance and Security Purdue University*, USA, 1-12.

Marcos, R., S., Marcelo, M., S., Henrique, G., B., Patrick, M., C., & Elias, O. (2014). Automatic Moderation of a Large Data Set of Users Comments. *XL Latin America Computing Conference*, 1-7, DOI: 10.1109/CLEI.2014.6965181

Markov, Z., & Larose, D., T. (2007). *Data Mining the Web: Uncovering Patterns in Web Content, Structure, and Usage.* New Jersey: John Wiley & Sons Inc.

Maron, M. (1961). Automatic indexing: An experimental inquiry. *Journal of the ACM (JACM)* 8(3), 404-417.

Murty, M. R., Murthy, J. V. R., Prasad Reddy, P. V. G. D., & Satapathy, S. C. (2012). A Survey of Cross-Domain Text Categorization Techniques. In *Recent Advances in Information Technology (RAIT), 2012 1st International Conference on* (pp. 499-504). IEEE.

National Cohesion and Integration Commission. (2010). Guidelines for Monitoring Hate Speech in the Electronic Media in Kenya. Nairobi: Government Press.

Pei, M, & Wu, X (2014). Text classification based on SMO and fuzzy model. In *Information Technology and Artificial Intelligence Conference (ITAIC), 2014 7th Joint International* (pp. 306-310). IEEE.

Pelemans, J., Demuynck, K., Hamme, H., & Wambacq, P. (2014). The effect of word similarity on N-gram language models in Northern and Southern Dutch. *Computational Linguistics in the Netherlands Journal, 4*, 91-104.

Prakash, G., Patric, B., Jacob, S. G., & Radhameena, S. (2014). Mining Semantic Representation From Medical Text: A Bayesian Approach. In *Recent Trends in Information Technology (ICRTIT), 2014 International Conference on* (pp. 1-4). IEEE.

Pramokchon, P., & Piamsa-nga, P. (2014). A Feature Score for Classifying Class-Imbalanced Data. In *Computer Science and Engineering Conference (ICSEC), 2014 International* (pp. 409 - 414). IEEE

Quan, C., Wei, X., & Ren, F. (2014). Comparison of SVM Classification Method and Semantic Similarity Method for Sentiment Classification. IN *Cloud Computing and Intelligence Systems (CCIS), 2014 IEEE 3rd International Conference on* (pp. 23-28). IEEE.

Ramezani, R., Sheydaei, N., & Kahani, M.(2013). Evaluating the Effects of Textual Features on Authorship Attribution Accuracy. In *Computer and Knowledge Engineering (ICCKE), 2013, 3rd International Conference on eConference on* (pp. 108-113). IEEE.

Rana, M. I., Khalid, S., & Akbar, M. U. (2014). News Classification Based On Their Headlines: A Review. In *Multi-Topic Conference (INMIC), 2014 IEEE 17th International* (pp. 211-216). IEEE.

Robertson, A., M. & Willet, P. (1998). Applications of N-Grams In Textual Information Systems, *Journal of Documentation, 54*(1), 48-67.

Russel, S., & Norvig, P. (2010). Artificial Intelligence: A Modern Approach (3^{rd} ed.). New Jersey: Pearson Education.

Samani, M. H., Rahimi, Z. & Rahimi, S. (2015). A Content-based Method for Persian Real-Word Spell checking. In *Information and Knowledge Technology (IKT), 2015 7th Conference on* (pp. 1-5). IEEE.

Sebastiani, F. (2002). Machine Learning in Automated Text Categorization. *ACM Computing Surveys, 34*(1), 1–47.

Shannon, C., E. (1948). A Mathematical Theory of Communication, *Bell System Technical Journal, 27*, 379–423.

Shin, H., Ryu, B. G., Ryu, W. J., Lee, G., & Lee, S. (2016). Bringing Bag-of-Phrases to ODP-based Text Classification. In *2016 International Conference on Big Data and Smart Computing (BigComp)* (pp. 485-488). IEEE.

Stas, J., Hladek, D., & Juhar, J. (2014). Recent Advances in the Statistical Modeling of the Slovak Language. In *ELMAR (ELMAR), 2014 56th International Symposium* (pp. 1-4). IEEE.

Suh, J. H. (2016). Comparing writing style feature-based classification methods for estimating user reputations in social media. *SpringerPLus*, 5(1), 1.

Swamy, M., N., Hanumanthappa, M., & Jyothi, N., M. (2014). Indian Language Text Representation and Categorization using Supervised Learning Algorithm. In *Intelligent Computing Applications (ICICA), 2014 International Conference on* (pp. 406-410). IEEE.

Tang, B., He, H., Baggenstoss, P. M., & Kay, S. (2015). A Bayesian Classification Approach Using Class-specific Feature for Text Categorization. *IEEE Transactions on Knowledge and Data Engineering 28*(6), 1602-1606.

Tang, B., Kay, S., & He, H. (2016). Toward Optimal Feature Selection in Naive Bayes for Text Categorization. *IEEE Transactions on Knowledge and Data Engineering.*

Thilagavathi, N. & Taarika, R. (2014). Content Based Filtering in Online Social Network using Inference Algorithm. In *Circuit, Power and Computing Technologies (ICCPCT), 2014 International Conference on* (pp. 1416 - 1420). IEEE.

Thunga, S. P., & Neelisetti, R. K. (2015). Identifying Metamorphic Virus Using n-grams and Hidden Markov Model. In *Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on* (pp. 2016-2022). IEEE.

van Dam, J. K., & Zaytsev, V. (2016). Software Language Identification with Natural Language Classifiers. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, (vol. 1, pp. 624-628). IEEE.

Vani, K., & Gupta, D. (2014). Using K-means Cluster Based Techniques in External Plagiarism Detection. In *Contemporary Computing and Informatics (IC3I), 2014 International Conference on* (pp. 1268-1273). IEEE.

Vinodhini, G., & Chandrasekaran, R., M. (2012). Sentiment Analysis and Opinion Mining: A Survey. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(6).

Vispute, S. R., & Potey, M. A. (2013). Automatic Text Categorization of Morathi Documents using Clustering Technique. In *Advanced Computing Technologies (ICACT), 2013 15th International Conference on* (pp. 1-5). IEEE.

Wang, Y., Fu, W., Sui, A., & Ding, Y. (2015). Comparison of Four Text Classifiers on Movie Reviews. In *Applied Computing and Information Technology/2nd International Conference on Computational Science and Intelligence (ACIT-CSI), 2015 3rd International Conference on* (pp. 495-498). IEEE.

Witten, I., H. & Frank, E. (Ed.) (2005). *Data Mining: Practical Machine Learning Tools and Techniques.* Morgan Kaufmann.

Wu, M. S., & Wang, H. M. (2012). A Term Association Translation Model for Naive Bayes Text Classification. In *Advances in Knowledge Discovery and Data Mining* (pp. 243–253). Springer Berlin Heidelberg.

Xu, G., Zhang, Y., & Li, L. (2011). *Web Mining and Social Networking: Techniques and Applications.* Verified OK.

Wu, G., & Xu, J. (2015). Optimized Approach of Feature Selection based on Information Gain. In *Computer Science and Mechanical Automation (CSMA), 2015 International Conference on* (pp. 157-161). IEEE.

Yadav, S. H., & Parne, B. L. (2015, January). A Survey of Different Text Categorization Techniques for Text Filtration. In *Intelligent Systems and Control (ISCO), 2015 IEEE 9th International Conference on* (pp. 1-5). IEEE.

Yadav, S. H., & Manwatkar, P. M. (2015). An approach for offensive text detection and prevention in Social Networks. In *Innovations in Information Embedded and Communication Systems (ICIIECS), 2015 International Conference on* (pp. 1-4). IEEE.

Yan, X. (2014). Application of Knowledge Gain on Multi-Type Feature Space in Microblog User Classification.In *Granular Computing (GrG), 2014 IEEE International Conference on* (pp. 340-345). IEEE.

Yasotha, R., & Charles, E. Y. A. (2015). Automated Text Document Categorization. In *2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS)* (pp. 522-528). IEEE.

Zaghoul, F. A., & Al-Dhaheri, S. (2013). Arabic Text Classification Based on Features Reduction Using Artificial Neural Networks. In*Computer Modeling and Simulation (UKSim), 2013 UKSim 15th International Conference on* (pp. 485-490). IEEE.

Zhang, X., & Wu, B. (2015). Short Text Classification Based on Feature Extension Using The N-Gram Model. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2015 12th International Conference on* (pp. 710-716). IEEE.