AUTOMATIC CLASSIFICATION OF THE EMOTIONAL CONTENT OF WEB DOCUMENTS

By

Alaa Hussainalsaid

B.Sc Mathematics and Science, Taibah University, 2008

A thesis

presented to Ryerson University

in partial fulfilment of the

requirements for the degree of

Master of Science

in the program of

Computer Science

Toronto, Ontario, Canada, 2016

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my thesis may be made electronically available to the public.

ALAA HUSSAINALSAID

Automatic Classification of the Emotional Content of Web Documents

Alaa Hussainalsaid
M.Sc. Computer Science, Ryerson University, 2016

Abstract

This thesis proposes automatic classification of the emotional content of web documents using Natural Language Processing (NLP) algorithms. We used online articles and general documents to verify the performance of the algorithm, such as general web pages and news articles. The experiments used sentiment analysis that extracts sentiment of web documents. We used unigram and bigram approaches that are known as special types of N-gram, where $N=1$ and N=2, respectively. The unigram model analyses the probability to hit each word in the corpus independently; however, the bigram model analyses the probability of a word occurring depending on the previous word. Our results show that the unigram model has a better performance compared to the bigram model in terms of automatic classification of the emotional content of web documents.

Acknowledgments

First, I would like to express my sincere gratitude to my God and to all the people who have supported me during this scientific journey.

This thesis would not have been possible without the help of my supervisor, Dr. Abdolreza Abhari, and his excellent guidance throughout the planning, research, and writing of this thesis. His patience and motivation led me to complete this thesis, and to publish a paper related to my research work in a scientific conference.

A very special thank you to my co-supervisor, Dr. Bahram Zahir, for believing in me from the start of my thesis and for encouraging me to think positively. I am grateful for the support and guidance he provided me during the initial period of my research work.

I would like to take this opportunity to thank the members of the Computer Science Department at Ryerson University for accepting me in to their program. In addition, I would like to express a special thanks to the Ministry of Higher Education in Saudi Arabia, which provided the financial support for me to complete this project.

It would not have been possible to write this thesis without the help and support of kind people around me, my parents Sami Hussainalsaid and Fayza Hakeem, my siblings, and my friends. I am especially thankful to my late father, who passed away during my Master's studies and to whom I am forever grateful.

My appreciation also goes to Ms. Tetyana Pekar for her patience, advice, and technical assistance.

Dedication

To My Family

Contents

List of Tables

List of Figures

List of Abbreviations

ASR Automatic Speech Recognition

CHMM2 Second-order Circular Hidden Markov Model

HMM Hidden Markov Model

HMM1 First-order Hidden Markov Model

HMM2 Second-order Hidden Markov Model

IE Information Extraction

LM Language Model

MLE Maximum Likelihood Estimation

MT Machine Translation

NLP Natural Language Processing

QA Question Answering

SPHMM Suprasegmental Hidden Markov Model

SR Speech Recognition

STT Speech To Text

SVM Support Vector Machine

UGC User Generated Context

URL Uniform Resource Locator

WOM Word of mouth

WSD Word-sense disambiguation

Chapter 1

Introduction

In the recent decades, a huge amount of information has become available in the form of web documents, but not all content that is posted is positive or useful in providing the requested information. In fact, some of the web documents may contain texts that might not be suitable for certain groups of people, because of the negative emotions that they display. Therefore, the automatic detection of emotions in texts has become increasingly important to deal with this task. Natural language processing (NLP) can help automatically classify the emotional content of web documents.

Natural languages are languages that are used by human beings to communicate with each other. NLP is a field in computer science, artificial intelligence and computational linguistics that explores how computers can be used to understand and manipulate natural language text or speech to perform the desired tasks [1].

The foundations of NLP consist of several disciplines such as computer and information sciences, electrical and electronic engineering, linguistics, mathematics, artificial intelligence, robotics, and psychology [1, 2].

NLP aims to build a software that has the ability to understand and analyze human language instead of artificial computer languages such as Java, Python, and C++ [3]. NLP also aims to implant information in text documents through the manipulation of their lexical, syntactic, or semantic linguistic components without altering the meaning [1]. The general interface of NLP is shown in Figure 1 [3].

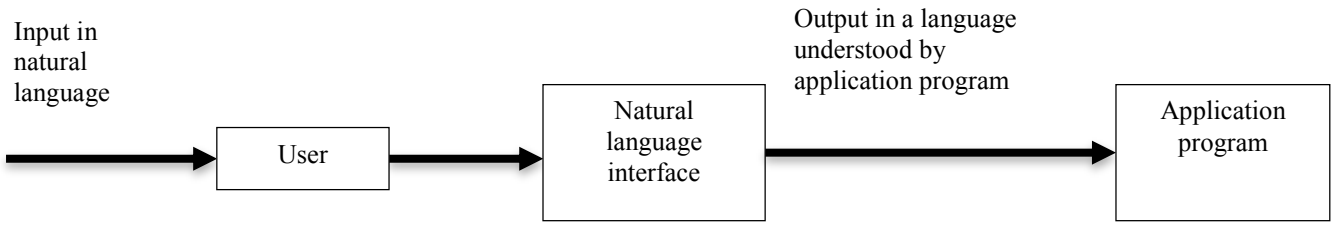

Figure 1: Block diagram of an NLP system [3]

The analysis of NLP can be divided into several levels: phonology, morphology, syntax, lexical semantics, pragmatics and discourse (Figure 2). To understand and extract meaning from natural languages (text or spoken), it is important to be able to recognize the differences among these levels [4, 5]. The knowledge of phonetics and phonology is required to understand the definition of the incoming audio signals, to understand the whole words within the signal, to use that sequence of words to make a new audio signal, and to determine the exact pronunciation of words in speech. The knowledge of morphology is needed to produce and recognize contractions and other differences between individual words, and understand the words in context. The knowledge of syntax is needed to order and group words together. The knowledge of lexical semantics is required to understand the meaning of each component word. Pragmatics is required for the use of indirect language. Finally, in order to structure a conversation perfectly, the knowledge of discourse conventions is required.

There are many different and common applications of NLP, which will be discussed in Section 2.1. In this thesis, we use sentiment analysis, which is regarded as one of the most notable applications of NLP. Sentiment analysis extracts the sentiment of website text documents and determines their sentimental contents. This will be discussed in more detail in Section 2.2.

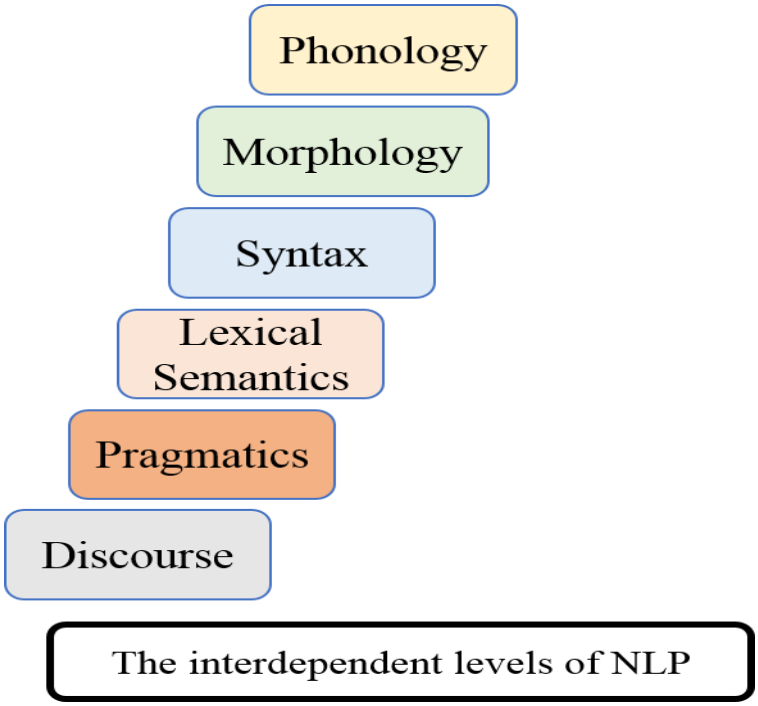

Figure 2: The different and interdependent levels of NLP required to understand natural languages

1.1 Motivation

In this day and age, we are exposed to a variety of information in different formats on a daily basis. There is a need to quantify and qualify the impact of the information on the interlocutor in order to help the reader easily understand the nature of documents, and if possible to minimize any possible negative impact. To achieve this goal, in this thesis we provide a rating for any given piece of information to allow the reader to be able to infer the quantity of the negative content in the document.

The rating provides a means for the reader to decide whether to avoid or read with discretion material that could potentially be harmful for his "reader class." The approach in this thesis is unique in the sense that it uses NLP to dynamically use a computer to determine the rating for any given piece of information. Moreover, this gives us the power to rate a piece of information on many levels and dimensions in a short period of time. The ingenuity of this approach will hinge on the power of NLP algorithms and their strategies to rate pieces of information. For a simple example, we

have a piece of text that tells a story about a violent father and his son in which verbs such as “hitting”, “hurting” and “abusing” are repeated several times. The system could give a rating of 6/10 for violence; hence, the system would warn a user with the child “reader class” that this material is not recommended for him/her.

We believe that analyzing web documents will help identify the nature of the documents, and that rating of this information will appear easily and quickly to inform the reader of the document’s contents and minimize time lost on reading potentially harmful information.

1.2 Research Statement

Presently, information on the Internet is exchanged at a rapid pace, regardless of whether the content has a positive or a negative impact. Quickly determining the nature of the information and webpage content can allow readers to identify the nature of the document and make decisions whether to read it or to avoid it.

People have different tastes, interests, and preferences with respect to the kind of documents or news they want to read. Some people like to read positive information in order to feel good and may avoid negative information or news to avoid feeling depressed. In addition, some parents want to prevent their children from some web pages to protect them from being exposed to aggressive or violent information.

Our research focuses on finding an automatic classifier for detecting web documents and identifying their sentimental content. This classifier explores only well-written documents that are extracted from general web pages and news articles. For the purposes of this thesis, “well-written” is defined as writing that is generally grammatically correct, free of spelling errors, and well-structured (e.g. news articles, scientific articles, academic writing). We analyze the sentiment of these texts using

NLP. By using N-gram classification, sentiment analysis can tell us whether the text we enter expresses positive sentiment, negative sentiment, or any other attributes. The label that the document will receive (positive or negative) will depend on which label has the greatest probability.

1.3 Objectives

The primary goal of this thesis is to perform a sentiment analysis on particular web documents that were mined from general webpages and news articles.

A sentiment analysis aims to understand the nature of the documents. The collected data is classified into two categories based on the presence or absence of an attribute (e.g. violent or not violent). In this study, we consider a few attributes violent/non-violent, love/not love, and hate/not hate. We will then perform an analysis of the classified data to assign two probabilities to the web documents: the probability of belonging to the class with the attribute and the probability of belonging to the class without the attribute. Based on these two probabilities, the system will determine which category the new web document should be assigned to.

1.4 The Proposed Approach

People of all ages like to read information on specific topics or follow the world news by browsing the Internet because it is regarded as one of the easiest and fastest ways of accessing information. However, not all webpages are appropriate to all ages and groups of people. For instance, some webpages discuss bad news or negative situations, and exposure to this content might have a negative influence on people's lives and/or kids' behaviors. The objective of this research is to propose a system for extracting and identifying whether webpage content is positive or negative in order to

help readers know the exact nature of the document without having them read the whole content.

In this thesis, we assume that the user who uses our recommender system is able to read and speak English. It should however be mentioned that the same approach could also work for virtually any other language, if the proper data sets are built and the experiments are repeated. Here, we provide a summary of the proposed solution. First, we select a specific attribute and search for a group of web documents based on this attribute and then search for other groups of web documents that contain other attributes excluding the specific attribute we selected earlier. Second, after labeling each web document based on our attribute, we divide each group into a training set and a test set. Finally, using N-gram models and finding the two probabilities for each web document on the test set, we compare the obtained probabilities for the model that was trained with the attribute versus the one that was trained without the attribute. The web documents on the test set will be labeled based on this comparison. The reader can refer to Chapter 3 for a full explanation of the procedures.

1.5 Research contributions

This study focuses on proposing a machine learning approach for analyzing and detecting the nature of documents using an NLP technique. The major contributions of this study are outlined as follows:

The thesis proposes a novel framework for classifying the web documents by using NLP techniques and generating recommendations for the reader. The framework is built by using existing methods that have not been previously applied for the purpose of detecting the nature of web documents and their sentimental contents. Further, the thesis provides an approach for understanding the sentimental content of the data automatically without requiring the user to look at it in advanced to

know its sentimental content by matching the document's keywords with their labels. In the future, the findings in this thesis can be applied to solve a wide range of problems. For instance, companies and organization can use this application to gain a better understanding of their customers' opinions about their products or services to improve their services. This application can also be used to protect vulnerable populations, such as children and patients with chronic conditions. Parents would be able to use this application to protect their children from potentially harmful Internet data. Patients who want to minimize their exposure to stress could use this application to avoid information that might negatively affect their health.

1.6 Scope and Assumptions

The main focus of this thesis is to find web documents from general webpages and news articles using Language Models (LM) and provide information about their sentimental contents to help the reader easily understand the nature of the document. Hence, the collected information about online articles can either be used for reader awareness or for decision making. For example, for reader awareness, if a document falls under the category of sexually explicit content, parents can prevent their children from reading this information. On the other hand, if a document falls under the category of positive news, such as the launch of a new smartphone, and another document falls under the category of negative news, such as the death of many people by an earthquake in Japan, the reader can decide what type of information they would like to read.

In this thesis, the dataset that has been created from web documents falls under one of these categories: violence attribute, love attribute, and hate attribute. As mentioned above, these web documents include online articles and general documents

that users can find easily by browsing the Internet. Based on these attributes, the entire data corpus is classified as either having or not having the particular attribute.

In addition, we focused on the English language as our target language because English is regarded as one of the most popular languages around the world. This is because the English language is used as an international auxiliary language [6] [7]. Moreover, in our task, we used the sentiment analysis application in NLP and two special kinds of N-gram models (unigram and bigram).

1.7 Organization of Chapters

The remainder of the thesis is organized as follows:

- Chapter 2 presents background information and a series of works related to this research. The first section of this chapter provides background information that is relevant for the rest of the thesis. It introduces the usage of LMs followed by their advantages and disadvantages. The next section provides background information regarding N-gram models, which are the main models used in the proposed system, followed by an introduction on the two types of models used. The final section of this chapter provides a review of published research works in the area of classifying and determining text documents.
- Chapter 3 describes the compositional and functional aspects of our proposed model, followed by the unigram and bigram algorithms for classifying and determining the text documents.
- Chapter 4 presents the implementation and results of this research work. The first section explains the implementation of the proposed model and describes the performance models used in this research work. The following section provides the results of the experiment conducted for detecting web documents and identifying their sentimental contents.

- Chapter 5 concludes the thesis by summarizing this work, giving more in-depth analysis of the results and providing suggestions for future research.

Chapter 2

Background Information and Related Works

In this chapter, basic information regarding the following subjects is discussed:

- The usage of NLP
- Sentiment analysis
- The advantages and disadvantages of NLP
- N-gram, a statistical approach in NLP and
- Types of N-gram models

The background information presented here sets the framework that is necessary for understanding the information presented in the rest of the thesis. The last part of this chapter discusses a number of works relating to this thesis.

2.1 Background

2.1.1 Usage of the Language Model

There are many researched topics in NLP, and these topics consider text or speech as a NLP candidate. While some tasks have real-world applications, others are used to assist in solving larger tasks. Some of the most common topics of research in NLP are the following:

- Text Segmentation refers to separating a written text into significant units, such as words, sentences, or topics. For example, in the English language, word boundaries are expressed by space, while sentence boundaries are expressed through punctuation, particularly the full stop character, and topic boundaries are shown through section titles and paragraphs [8].

- Speech Recognition (SR) refers to the translation of an audio sound into text. The audio sound can contain the sound of one person or several people speaking. SR is also called automatic speech recognition (ASR), computer speech recognition, and speech-to-text (STT). SR has two types: speaker-dependent systems and speaker-independent systems. Speaker-dependent systems use a final stage of per speaker training, where a speaker reads a text or a list of words into the system to fine tune the training of the system, while speaker-independent systems do not have this final stage of fine tuning [9].
- Machine Translation (MT) refers to using software to translate speech or text from one human language to another. MT usually performs very well on translating just words from one language to another language. However, it does not always work that well with complete text translation. This is because there is a need to recognize whole phrases in the target language and that can often become quite a challenging task [10].
- Automatic Summarization is used to produce a summary of a text document and this summary keeps the most salient points of the original document. Automatic summarization is a very important task because of the increasing amounts of data. Search engines, such as Google, are one example of the use of automatic summarization. Automatic summarization has two approaches: extraction and abstraction. In extractive approaches, parts of existing words, phrases, or sentences from the original text are selected to create the summary. While in abstractive approaches, an internal semantic representation is built and then natural language generation techniques are applied to generate a summary [11].

- Text Classification and Sentiment Analysis is used to obtain subjective information from a set of documents and identify their orientations. This is used to detect and review the writer's attitude or opinion regarding a topic, customer service, or product reviews [12]. We will present the sentiment analysis task in more details in Section 2.1.2.
- Information Extraction (IE) refers to extracting structured information, such as the determination of relationships between objects, from unstructured documents. For example, if an article states, "In November, Company Z bought out Company X," the IE algorithm will be able to extract that Company Z merged with Company X. Many recent activities such as automatic annotation and content extraction from images, audio, or video are examples of IE [13].
- Parsing is the grammatical analysis of a string of symbols either in natural language or in computer language. In other words, parsing divides a word or a sentence into parts of speech and depicts them grammatically. Parsing has a different meaning in linguistics and computer science. In linguistics, parsing is used to understand the meaning of a sentence with the use of supporting systems, like sentence diagrams. In computer science, parsing is used to breakdown a sentence into its components, and this leads to the existence of syntactic and semantic relations among words [14].
- Meaning Extraction, also known as word-sense disambiguation (WSD). Many words have more than one meaning, so this task is used to select the meaning that makes the most sense in context. The system provides a list of words from a dictionary or any online resource like WordNet and selects the most suitable word for the sentence [15].

- Question Answering (QA) aims to find answers to questions posed by humans in a natural language. Some of these questions are typical questions that have specific and direct answers, such as “What is the capital city of Canada?” Whereas others are more complicated and have open-ended answers, such as “Why did WWII start?” Thus, QA applications aim to deal with various types of questions [16].
- Completion Prediction is used to predict words in a sentence and give choices on how to complete it. When the system is given an incomplete sentence (e.g. “Please turn off your cell……”), completion prediction can estimate what is being typed and provide options on how to compete the sentence [17].
- Part-of-speech tagging categorizes each word in a sentence into a specific part of speech based on its definition and its context. Part-of-speech tagging is a very important application for some ambiguous languages that have words that can belong to multiple part of a speech, such as in English where "book" can be a noun (e.g. "the book on the desk table") or verb (e.g. "to book a flight") [18].

In addition to the above, NLP is used for other tasks as well, such as natural language generation, natural language understanding, spelling correction, and information retrieval, among others, demonstrating the importance of the NLP algorithm.

2.1.2 Sentiment Analysis

The increasing use of the Internet usage and interchange of public opinions necessitates the existence of sentiment analysis. The Web is a massive storehouse of

unstructured and structured data, which makes the analysis of this data to identify public opinion and sentiment a difficult task.

Sentiment analysis is a task of NLP and IE that intends to acquire the writer's emotions expressed in positive or negative comments by inspecting a large number of documents. In other words, the objective of sentiment analysis is to determine the perspective of the writer or speaker that pertains to a certain topic or to the overall document.

"*Sentiment analysis is the field of study that analyses people's opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards entities such as products, services, organizations, and their attributes*" [19].

The analysis of sentiments can be applied to documents, sentences or phrases; sentiment analysis extracts the sentiment of these texts using NLP, statistics, and machine learning models and then determines whether the texts are positive, negative or neutral. Sentiment analysis is also often referred to as opinion mining, opinion extraction, sentiment mining, and subjectivity analysis [19].

Sentiment analysis can be a simple task when the goal is to determine the nature of a text as objective (facts), positive (a state of happiness) or negative (a state of sadness). However, ranking the attitude of a text based on positive, negative, or objective degree is a more complex task [20].

In this thesis, sentiment analysis detects web documents and labels them as either positive or negative, and determines sentiment in each of the web documents using NLP techniques.

2.1.2.1 Applications of Sentiment Analysis

Sentiment analysis comes from the process of transferring information from person to person, which is called word of mouth (WOM) [20]. This process plays an important

role in commercial businesses, e.g. sharing views or reactions from consumers about products or services with other people. Thus, sentiment analysis has been used in marketing to review products and identify consumers' reactions, social media to find trending topics in town, and movie reviews to preview movies' quality.

In [21], the researchers sort the applications of semantic analysis into different types of applications which are:

a. Review-Related Websites Applications (e.g. product reviews and movie reviews)
b. Sub-Component Technology Applications (e.g. spam detection and context sensitive information detection)
c. Business and Marketing Applications (e.g. consumer attitudes and trends)
d. Different Domains Applications (e.g. public opinions about political leaders, rules or regulations in place

2.1.2.2 Challenges for Sentiment Analysis

Sentiment analysis aims to detect and extract sentimental words and phrases from a text and to categorize the text as positive or negative. If the analysis is unable to identify any sentimental words from the text, it classifies the text as objective. Therefore, sentiment analysis has three categories, which suggests that sentiment analysis is not as difficult as the text classification task, which has many classes due to the existence of various topics. However, this is not the case because there are many general challenges of sentiment analysis. These can be summarized as follows [20]:

A. Implicit Sentiment and Sarcasm

A sentence might have an implicit sentiment of sarcasm even without the existence of any negative sentiment words. Therefore, detecting semantics is more significant in sentiment analysis than in syntax detection. See the following examples:

- How could anyone possibly watch this film?
- We should ask a question about the stability of the mind of the man who does this behavior.

Both of the above sentences are negative even though neither has words with negative sentiment.

B. Domain Dependency

Polarity of the same words can change between different domains. See the following examples:

- The scenario was unpredictable.
- The driver's performance and the steering of his car are unpredictable.

The word "unpredictable" appears in both sentences. However, the first sentence has a positive sentiment, whereas the second sentence has a negative sentiment.

C. Thwarted Expectations

The writer can delay negative semantics until the end of the sentence. See the following example:

- This manager should support his employees and be nice to them. I heard that he is attempting to provide a good and quiet environment for his employees. However, he is not as I expected.

Despite the existence of the positive words in the sentence, the overall sentiment is negative. This is because the criticism was in the last sentence.

D. Pragmatics

Writers use various ways of pragmatism to express their feelings, and these pragmatisms might change the sentiment completely. Consequently, it is crucial to identify the pragmatics of a user's opinion. Sentiment can be denoted using capitalization. See the following examples:

- I just finished watching the match where my favorite team DESTROYED the opposing team.
- The final test destroyed me.

The first sentence indicates a positive sentiment because the capitalization of "destroyed" signifies excitement, while the lack of capitalization in the second sentence indicates a negative sentiment.

E. World Knowledge

In order to detect and identify sentiments, world knowledge needs to be inserted into the system. See the following examples:

- Ali is like Frankenstein.
- I just finished Doctor Zhivago for the first time and all I can say is Russia sucks.

The first example denotes a negative sentiment while the second one denotes a positive sentiment. However, we have to know what Frankenstein and Doctor Zhivago are to detect the sentiment.

F. Subjectivity Detection

Finding the difference between opinionated and non-opinionated text is often very difficult. Subjectivity detection seeks to remove objective facts to identity subjective components in a text; however, this task remains difficult. See the below examples:

- I hate love novels.

- I do not love the book "I hate stories".

The first sentence describes an objective fact while the second sentence describes an opinion about a specific book.

G. Entity Identification

Any text or sentence can have multiple entities, so it is imperative to discover the entity towards which/whom the opinion is directed. See the examples below:

- iPhone is better than Samsung.
- Ali overcame Adam in football.

The examples are positive for iPhone and Ali, but they are negative for Samsung and Adam.

H. Negation

Dealing with negation in sentiment analysis is a challenging task. There are many different and subtle ways to express negation without straightforward and explicit usage of any negative word. See the following examples:

- I do not like the movie.
- I do not like reading, but I like writing poems.
- I not only like reading, but I also like writing poems.

In the first example, the negation was used explicitly, reversing the polarity of all the words appearing after "not"; this is easiest to detect. However, this strategy does not work for the second example because the scope of negation extends only until the conjunction "but". In the third example, the polarity is not reversed after "not" because of the existence of "only." These factors should be considered when designing the algorithm.

2.1.3 Advantages and Shortcomings of NLP

As mentioned earlier, NLP plays an important role in analyzing and understanding human language through NLP properties. NLP has the ability to simultaneously describe various entities for any language by providing a set of descriptive tools, such as the usage of noun phrases for determining objects, verb phrases for determining events, and tense and aspect for describing time periods. Moreover, NLP has the ability to understand the nouns that pronouns and other anaphoric expressions are referring to [22]. Indeed, NLP is highly expressive, flexible, and representative of reality systems.

Although NLP is regarded as one of the most effective interface technologies for many problems, offering a number of advantages, it has some shortcomings. Some natural language systems have linguistic and conceptual ambiguities of utterances due to unknown words, synonyms, or ill-formed sentences. In addition, syntactic ambiguities that caused by structure, and semantic ambiguities that caused by homographs are problems with others systems [23]. Thus, even though the users know the system cannot understand every utterance, they do not know what it is able to interpret. Some systems could be error-prone due to multiple attempts that are made to pose a query or command. In addition to this, some NLP systems have difficulties following the alterations in context that exist in dialogue [22]. Therefore, making generic searches in NLP is difficult.

2.1.4 N-gram Models

The N-gram probability, which is regarded as a type of Markov chain model, is the conditional probability of the occurrence of a word provided all the previous words. The simple kind of Markov chain or N-gram is bigram model, which only assesses the immediate occurrence of the previous word. In addition, N-gram models can be

computed by counting the frequency of consecutive words in a corpus. The general equation for N-gram is:

$$P(w_n \mid w_1^{n-1}) \approx P(w_n \mid w_{n-N+1}^{n-1}) \quad (2.1)$$

The above equation shows the probability of a word w_n given only the previous *N* words [2]. Furthermore, N-gram conditional probabilities can be estimated by normalization, which is done by taking the count of N-grams from a training corpus and dividing them by the sum of N-grams that have the same prefix, as follows [2]:

$$P(w_n \mid w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1} w_n)}{C(w_{n-N+1}^{n-1})} \qquad (2.2)$$

This ratio is called a relative frequency, which is regarded as a technique of Maximum Likelihood Estimation (MLE). Hence, N-gram models can be computed by normalizing by the MLE, which is done by dividing the count of the word, *T*, by the total number of word tokens, *M*, as illustrated below [2, 24, 25]:

$$P\,(T|M) \qquad (2.3)$$

There are two important facts about N-gram behavior. First, when the value of N increases, the accuracy of N-gram models increases. Second, when the value of N increases, their dependence on their training corpus increases as well. However, N-gram models impose a challenge, which is that they must be trained from some corpora, and all of these training corpora are finite. Furthermore, the MLE method gives poor estimates when the counts are non-zero but still low. However, some techniques can be used to re-evaluate some of the zero or low probability N-grams and designate them as non-zero values. This task is called smoothing [2] and is a very important part of working with N-grams. Even though the available corpora are large and contain a portion set of the possible N-gram, it is essential to smooth the data in

order to get better estimates of unseen occurrences of N-grams (i.e., words and phrases that do not appear in the text; unigram, bigram, trigram, and so on) [26, 27]. In addition, it is obvious that a document on the training set cannot have all possible words or even the sequence of words in the test set, and vice versa. Thus, it is very important to use smoothing techniques. As explained in [28], there are many different smoothing algorithms, including Additive smoothing, Good-Turing estimate, Jelinek-Mercer smoothing (interpolation), Katz smoothing (backoff), and Witten-Bell smoothing. The simplest smoothing technique and one that is regarded as an introduction and useful baseline for other smoothing algorithms is add-one smoothing. The add-one smoothing technique is known also as Laplace smoothing. In this technique, one is added to all the counts in the matrix of N-gram counts and this is then divided simultaneously by the total number of types (the vocabulary size) as shown in the following equation [2]:

$$P(w_n \mid w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1} w_n) + 1}{C(w_{n-N+1}^{n-1}) + V} \quad (2.4)$$

Where V is the total number of word types in the language, i.e. the size of the vocabulary.

To evaluate corpus-based LMs like N-grams, we divide the corpus into a training set and a test set; training the statistical parameters of the model on the training set, and then using them to compute probabilities on the test set.

To compare between different smoothing algorithms, a large corpus is taken and divided into a training set and a test set. Then it is trained using different N-gram models on the training set to determine which one of the N-grams better fits the test set. More specifically, to evaluate N-gram models, a useful and common metric to examine how well a given test set matches a test corpus is called perplexity.

In addition, the N-gram model can be used for context-sensitive spelling error correction by generating every possible misspelling of each word in a sentence. This is done by typographical modifications or through the inclusion of homophones. Then, the spelling that gives the sentence the highest prior probability is chosen.

There is an important point that should be mentioned when discussing N-gram models. All the gained probabilities from N-gram models are always less than or equal to 1; therefore, when we multiply these probabilities together, the product can get extremely small. Indeed, if we want to compute the probability of the occurrence of words within a paragraph or an entire document, we might face a practical problem that is called the risk of numerical underflow. To solve this problem, we can take the logarithm base 2 for each term in Equation (2.4) [2].

2.1.4.1 Challenges of N-gram Models

N-gram models are an essential element of modern language modeling for NLP, and they have been successfully utilized in many applications, such as part-of-speech tagging, ASR, and syntactic parsing [29, 30, 31]. This characterizes N-gram models as rich lexical knowledge [2]. However, N-gram models do not perform well for some purposes because they depend on the training corpus that will be used [2]. Thus, some of the general challenges of N-gram models can be summarized as:

- Since probabilities are based on computing word, we need to know exactly which words that we are going to count. In other words, we need to find out how many words are in a corpus [2]. Consider the following example:

  "He is going to attend the ceremony party, but he is not going to be at the birthday party tomorrow."

In the above sentence, there are 21 words if we do not count the punctuation marks, but there are 23 words if we count the punctuation marks (period and comma).

- Since sentences and paragraphs contain capitalized and un-capitalized tokens, we need to know whether the system will differentiate between them [2]. Consider the following example:

  "The high school students have not finished their final exams yet, but *the* university students have already finished their final exams."

  In the above example, the words "The" and "the" are the same.

- Inflected form is another challenging issue in N-gram models since these models are based on the word form that appear in the corpus, such as the singular form of the word "cat" and the plural form "cats" or the verbs form like "kill" and "kills". Thus, all of these forms are treated as two separate words, and this is not good for context domains that want to treat "cats" and "cat" as one word [2].

- Another issue that we should consider while using N-gram models is determining how many words are in a corpus. There is a basic way to answer this question, which is based on two terms: types and tokens. If we count the words based on the number of distinct words in a corpus, we are using **types**. If we count all the words in a corpus, we are using **tokens** [2]. Consider the following example:

  "Ali and Adam are going to attend the ceremony party, but they are not going to be at the birthday party tomorrow."

  In the above sentence, we can say that the corpus has 17 types and 22 tokens (without counting the punctuation marks).

2.1.4.2 Some N-gram Types

As mentioned previously, N-gram models are a statistical approach in NLP, and it estimates the probability of each word given prior context. In other words, N-gram model uses only N–1 words of prior context depend on the value of N. The previous word could be one, which known as a unigram, two, which is known as a bigram, or three, which is known as trigram, and so on. The following is an example of finding probabilities for these different types of N-gram models:

"*Please stop this violence*"

Unigram: P(violence)

Bigram: P(violence | this)

Trigram: P(violence | stop this)

In this study, we used only the unigram and bigram among all possible N-gram models.

2.1.4.2.1 Unigram Models

The unigram approach is a special case of N-gram with N=1. It is regarded as the simplest form of N-gram models. The probability to hit each word in the corpus depends on the word itself. The following is the equation for a unigram model:

$$P_{uni}(w_1\, w_2 w_3 \ldots . w_n) = P(w_1)P(w_2)P(w_3)P(w_n) \qquad (2.5)$$

In this model, unigram language models are often smoothed to avoid instances where the probability of the appearance of a word is zero. In order to have a consistent probabilistic model, each sentence is appended with a unique start <s> and end </s> symbols, and these symbols are treated as additional words.

Let us look at the following example in Figure 3, which presents simple sentences, and Figure 4, which presents the appropriate unigram probabilities for these sentences.

To compute the unigram probability of a word occurring within a corpus, we count how many times a word has occurred in the context domain and then divide this number by the total number of words in the domain. For example, if we want to find the probability of a word "I", we will see how many times "I" appears in the context domain and then divide by the total number of words in this domain. For Figure 3, the probability of the occurrence of the word "I" is: 4/24 = 0.16.

I hate violent movies.
I hate Romantic movies.
I like Romantic movies.
I like Romantic stories.
Figure 3: Simple sentences.

<s>	.16	hate	.08	violent	.042
I	.16	like	.08	movies	.125
</s>	.16	Romantic	.125	stories	.042
Figure 4: Unigram probabilities for the sentences in Figure 3.					

Now we can compute the probability of any sentence (e.g. "*I hate violent movies*") by multiplying the unigram probabilities together as follows:

P(<s>, I, hate, violent, movies, </s>) = P(<s>) P(I) P(hate) P(violent) P(movies) P(</s>)

= .16 * .16 * .08 * .042 * .125 * .16 = 0.00000172

As we can see in the above example, all of the probabilities are less than one; consequently, multiplying these probabilities together results in a very small product. We could solve this problem by taking the logarithm for each word and summing them together instead of multiplying them:

Log P(<s>, I, hate ,violent ,movies, </s>)= log P(<s>) + log P(I) + log P(hate) + log P(violent) + log P(movies) + log P(</s>)

= (-0.8) + (-0.8) + (-1) + (-1) + (-0.9) + (-0.8) = -5.3

2.1.5.2 Bigram Models

The bigram approach is a special case of N-gram with N=2, and is known as a **first-order** Markov model. For bigram models, the probability of a word is computed by giving the probability of one previous word as shown in the following equation:

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k \mid w_{k-1}) \quad (2.6)$$

To illustrate the bigram language model, let us compute the bigram probabilities for these sentences shown in Figure 3. To compute the bigram probability of a word occurring within a corpus, we count how many times a word and the previous word occur in the context domain and then divide this number by the total number of words in the domain. For example, if we want to find the probability of a word “I hate”, we will see how many times ”I hate” appears in the context domain and then divide by the total number of words in this domain. For Figure 3, the probability of the occurrence of the word “I hate” is: 2/24 = 0.08.

<s> I	.16	hate violent	.042	violent movies	.042	movies </s>	.042
I hate	.08	hate Romantic	.042	Romantic movies	.125	stories </s>	.042
I like	.08	like Romantic	.08	Romantic stories	.042		

Figure 5: Bigram probabilities for the sentences shown in Figure 3.

Thus, in a bigram (n = 2) language model, the probability of the sentence "*I hate the violent movies*" is approximated as:

P(<s>, I, hate, violent, movies, </s>) = P(I| <s>) P(hate | I) P(violent| hate) P(movies| violent) P(</s> | movies)

= .16 * .08 * .042* .042* .16

= .00000361

Here, we also can take the logarithm for each pair of words, as we have done with the previous example, in order to get a better result.

2.2 Related Works

In this section, we present research that is related to our study. In the last few years, NLP and its approaches have been considered the most significant methods for achieving automatic classification of the emotional content of natural language texts. Several studies have evaluated the performance of standard text categorization techniques such as unigram and bigram models.

In [32, 33, 34], Lewis studied how phrases are utilized in text categorization by using part-of-speech parsing on the text and all noun phrases that existed two or more times as features. He found that using phrases led to poorer performance than

using single words due to high dimensionality, lack of redundancy among terms, and high degree of synonymy.

In [35], the authors present the performance of bigrams as well as the use of single words. They applied the information gain metric with term frequency and document frequency in order to select which bigrams should be used. Term frequency refers to the importance of the term in the specific document; document frequency refers to the importance of that term in the entire corpus. They found that bigrams can be good discriminators and improve the categorization performance.

In [36], the authors used Naive Bayes classifier on Yahoo text hierarchy, using different length of word sequences (up to 5). The experimental results illustrated that using word sequences of up to three words in length instead of using only single words enhanced the performance, whereas using longer sequences did not affect the performance.

In [37], the author used the *a priori* algorithm with term frequency and document frequency standards, showing that using sequences of two or three words were most useful; however, the use of longer sequences worsened the performance.

In [38], the authors used dimensionality reduction techniques in order to overcome the problem of document routing, which is a statistical text classification problem. They tried to identify models that minimized the risk of over-fitting, which refers to the situation whereby a model fits to the training data too closely such that it does not generalize to the entire population, and hence perform not that well on the test dataset. They used single words and two-word phrases that were selected according to term frequency. They found that decreasing feature space is effective and useful to resolve document routing.

In [21], the authors classified movie reviews by sentiment polarity using an N-gram algorithm; they found that the unigram approach outperformed the bigram approach. Whereas in [39], the authors reported that bigrams and trigrams performed better in some contexts.

In summary, some research has shown that using bigrams and trigrams improved performance. However, using more than three sequences of words of length N-grams was not useful and might reduce the performance.

In addition to N-gram models, other authors have used different algorithms to extract and identify emotional expressions. For instance, in [40], the authors extracted emotional expressions from blog sentences in English with the Ekman's six basic emotion tags and any of the three intensity types: low, medium and high. A baseline system was used to identify sentential emotion tags, emotional expressions, and intensities. To identify the dependency relations, the sentences were passed through the Stanford Dependency Parser and WordNet affect list was used to identify emotional expressions. In addition, the various types of dependency relations among the parsed sentences offered positional and intensity-related hints regarding the emotional expressions. The SVM-based supervised system was used to detect the emotional expressions and to tag the sentences with intensities and emotions. The system was trained with 2700 sentences, but the best feature set was determined based on the F-Scores obtained from 358 developed sentences. Features, like emotions, emotion words, intensifiers, conjuncts, negations and discourse markers, yielded high information gain on the development of sentences. Words that did not carry any emotion features were filtered out because they gave low information gain. Finding the admissible tag sequences and using a class splitting technique enhanced the system's performance. Emotion pairs and intensity pairs cause a problem in sentential

emotion and intensity tagging. To resolve this problem, a particular feature of emotional composition that determines how closely the two emotion types are was added. In conclusion, the supervised system performed better than the baseline system and better in terms of F-scores for intensities, emotional expressions, and sentential emotion tags.

In the following paragraphs, we would like to mention the works of others in the speech processing field. In [41], the authors focused on text-independent speaker identification when varied emotional states are presented by different recorded speakers. This was implemented by comparing the performance of the text-independent speaker identification system, which trained and tested the recorded speakers in neutral states and in various emotional states. In order to train the speaker models and test the system, Support Vector Machines (SVMs) were employed. The authors used the Berlin Database of Emotional Speech, which contains 10 different speakers who were recorded in different emotional states. When a given emotional state is used for both training and testing, the performances of the speaker recognition systems are very good. However, the system fails when a neutral state is used in training while emotional states are used for testing.

In [42], the authors focused on the performance of speaker identification in neutral and emotional environments. This was implemented by using three different and separate models: Hidden Markov Models (HMMs), Second-Order Circular Hidden Markov Models (CHMM2s) and Suprasegmental Hidden Markov Models (SPHMMs). The speech database used in this study contained 40 nonprofessional speakers (20 males and 20 females). Every speaker pronounced eight sentences and each sentence was pronounced nine times in five emotions (disgust, fear, angry, sad, and happy) in an emotional environment and in a neutral environment. The three

models performed quite well for speaker identification in the neutral environment. However, SPHMMs outperformed HMMs and CHMM2s for speaker identification in emotional environments. In the natural environment, the performance of speaker identification was perfect; however, it significantly deteriorated in emotional environments.

According to [43], under neutral talking conditions, speaker identification systems perform quite well, but they do not perform well under a shouted talking condition based on HMM1s. In this paper, the authors used second-order hidden Markov models for training and testing phases of isolated-word text-dependent speaker identification systems under shouted talking and neutral conditions. This work relied on a speech database that was collected from 40 different speakers and each speaker pronounced the same ten different isolated words under the neutral and shouted talking conditions based on HMM2s and HMM1s. In the first training session, each speaker pronounced each word five times under the neutral talking condition based on HMM1s. In another session, which was the testing session, every speaker said the same word four times under the neutral talking condition and nine times under the shouted talking condition based on HMM1s. The second training and testing sessions were similar to the first training and testing sessions but based on HMM2s. The forward backward algorithm as used in the two training sessions, whereas the Viterbi decoding algorithm as used in the two testing sessions. The authors found that by using HMM2s in both the training and testing phases of isolated word text-dependent speaker identification systems under the shouted talking condition led to better speaker identification performance than using HMM1s. Indeed, the second-order hidden Markov models show improvement of the speaker identification performance over the first-order hidden Markov models.

Our approach differs in several aspects from the above mentioned studies. We explored the performance of unigram models and bigram models in web documents using the sentiment analysis task. The statistical parameters of the model were trained on the training set, and then used to compute probabilities on the test set. We found that the bigram approach did not perform as well as expected. The size of the database as well as the method's accuracy could be the main problems. These findings were also reported in [44].

2.3 Summary

In this chapter, we started by giving a brief introduction to the usage of LMs and discussed sentiment analysis and its applications in more details. Some of the challenges in sentiment analysis, as well as the advantages and disadvantages of NLP were discussed. We provided background information about N-gram models and some challenges of using N-gram models, followed by some types of N-gram models. Finally, related works in the area of classifying and determining text documents were provided at the end of this chapter.

Chapter 3

Proposed Model and Methodology

This chapter covers the creation of the database for this project, and the model design behind it. In addition, it provides an overview of how the data mining, text processing, and machine learning technique were implemented in this project.

3.1 Database Creation

As with any machine learning systems, we need to prepare a database to accommodate our experiments. Due to the novelty of this approach, we could not find any available datasets and had to build our own database for this purpose. The database consists of the following fields (columns):

- A URL where the original webpage had been found
- The date of when the URL had been crawled
- The text that was extracted from the URL on the crawling date (we need to keep this text, as it may change over time or disappear and we need a reliable and constant text for our assessments)
- A manually entered label for each attribute; for instance, for the attribute "violence" we would use one of these values: {"violent", and "nonviolent"}. If we find a web document with a context that cannot be labeled as either violent or not violent, we will not use that web document in the training of the machine-learning algorithm or in the evaluation or test of the final system.

As a sample of our database, we have this entry:

***URL**:"http://news.yahoo.com/protest-chokehold-death-turns-violent-california-065444495.html"*

***Date crawled:** Dec. 8, 2014 at 10:00 am EST*

***Text:** "Chokehold death protest gets violent in California*

By VERENA DOBNIK December 7, 2014 8:21 AM

NEW YORK (AP) — Mostly peaceful protests of a grand jury's decision not to indict a white police officer in the chokehold death of an unarmed black man continued around the country, but authorities said a march in California turned violent when a splinter group smashed windows and threw objects at police. …"

***Label:** "violent"*

In our experiment, we use "violence", "love", and "hate" as our attributes. Naturally, the algorithm can be expanded to other attributes such as "depressive", "sexually implicit", "disturbing", etc. As we only considered English language as our target language, we deleted web documents with largely other language content from our database.

3.2 Model Design

We designed web documents chosen in these experiments in different attributes as violent/nonviolent, love/not love, and hate/not hate. From each attribute, we found a group of web documents, and we then split the data set into the training set and the test set. A block diagram of the system can be seen in Figure 6.

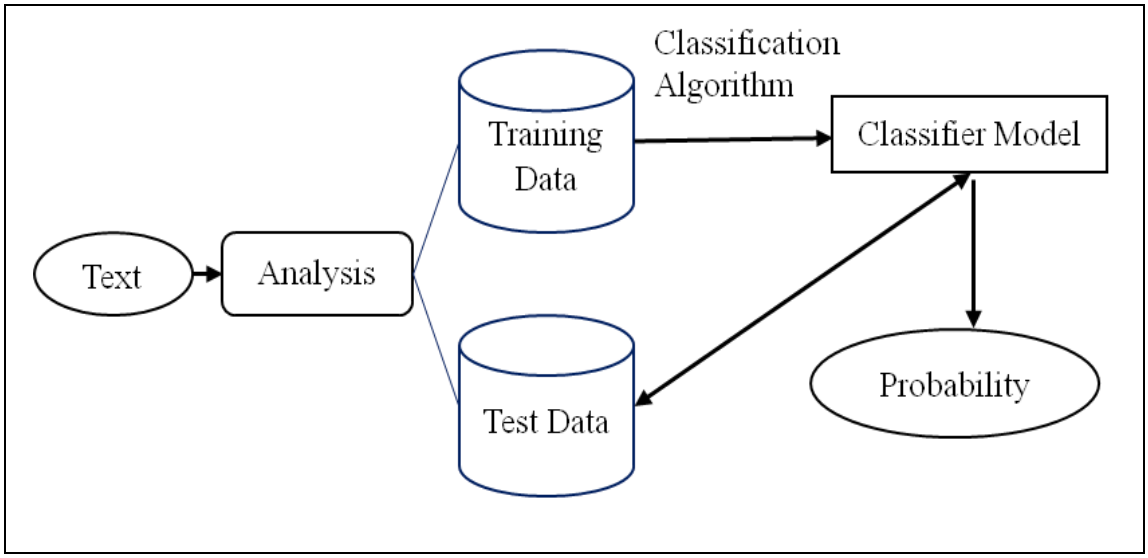

Figure 6: The general block diagram of the proposed system

To carry on the experiment with violent and nonviolent attributes, we created a dataset, as categorized in Table 1. From each group, we assigned some web documents to the training set, some to the validation set and some to the test set. This assignment was done randomly.

	Violent	**Non violent**	**Total**
Training set	167	167	334
Validation set	167	167	334
Test set	166	166	332
Total	500	500	1000

Table 1: Number of web document samples in the "violent" and "nonviolent" sets and each of the training, validation and test sets

For love and not love attributes, we found some web documents that are for love and other web documents that are for not love. What we mean here by "for love" and "not for love" is emotional feelings between two human beings. For each group,

we assigned some web documents to the training set and some to the test set, as shown in Table 2.

	Love	**Not love**	**Total**
Training set	10	10	20
Test set	10	10	20
Total	20	20	40

Table 2: Number of web document samples in the "love" and "not love" sets and each of the training and test sets

The other attribute that we chose is the "hate" attribute. We found some web documents that are for hate and other web documents that are for not hate. An example of a web document related to "hate" is negative and hateful feelings between human beings. For each group, we assigned some web documents to the training set and some to the test set, as categorized in Table 3.

	Hate	**Not hate**	**Total**
Training set	10	10	20
Test set	10	10	20
Total	20	20	40

Table 3: Number of web document samples in the "hate" and "not hate" sets and each of the training and test sets

As shown above, Table 1 differs in some aspects from Tables 2 and 3. Table 1 contains 1000 samples of the dataset including violent and non-violent attributes, and then divided that into three sets: training with 34%, validation 33%, and test with 33% of the data. Whereas, Tables 2 contains 40 samples of the dataset of love/not love attributes, and the same number of samples for hate/ not hate attributes as shown in

Table 3. This is because we concentrated on violent and non-violent category which is our primary objective and then do the simple tests of our system on different attributes such as hate and love and etc. In addition, we have validation set for classifying violence category in order to avoid overfitting. In other words, once we trained our system on the test set and then did some improvements in our algorithm, there was a need to test with more dataset of validation purpose to avoid overfitting between our training set and test set.

3.3 Methodology

We used the N-gram-based unigram and bigram models for our experiments. For both models, to evaluate corpus-based language models like N-grams, we divided the corpus into a training set and a test set; training the statistical parameters of the model on the training set, and then using them to compute probabilities on the test set.

The machine-learning algorithm will go through the body of the text for each web document, for instance, whether it is labeled as violent or nonviolent, and find the frequencies of each word based on unigram model or the pairs based on bigram model in each category. When a new web document is presented to the system, the machine will assign a probability of the new web document being part of the violent class and another probability of that web document belonging to the nonviolent class based on the frequencies obtained in the training phase. Based on these two probabilities and using the MLE method, the system will determine to which category the web document should be labeled.

In addition to that, after finding the probability of each word or the pairs, we found the log2 for each probability then sum them instead of multiplying them in order to get larger number magnitudes once probabilities are all small numbers. In addition, we used add-one smoothing technique by adding one to all the counts in the

matrix of N-gram counts and then divided simultaneously by the total number of vocabulary size [2]. Indeed, we used this technique to avoid zero or low probability of N-grams and to designate them as non-zero values. Next, we implemented this methodology in two different types in N-grams models, the unigram model (Section 3.4) and the bigram model (Section 3.5).

3.4 Unigram Algorithm

The goal of our algorithm is to find the list of unigrams that appear in all documents for each category and use them as a model to assess the testing set. Our unigram machine-learning algorithm goes through the body of the text for each web document, whether it is labeled as violent or nonviolent, and finds the frequencies of each word in each category. The general steps for implementing the unigram algorithm in both phases is shown in Figures 7 and 8.

1. Open web documents as a CSV file
2. Extract each word in the training data for violent and nonviolent separately and for each word do the following:
    a. Get unigram frequencies in each category
    b. Count all words in the text
    c. Count the total number of words in the text
3. Get the probability for both violent and nonviolent data according to this formula:

$$P_{uni}(w_1\, w_2 w_3 \ldots . w_n) = P(w_1)P(w_2)P(w_3)P(w_n)$$

4. Set total final probability equal to one
5. Add one over the total number of words to each probability
6. Get $\log_2$ probability for each probability
7. Final probabilities for violent and non-violent is equal to $\log_2$ of the probability plus the final probability

Figure 7: General steps of the unigram algorithm for the training phase

1. Open a new document as a CSV file either the one labeled as violent or nonviolent
2. Extract each word in the new web document to assign two probabilities based on the frequencies obtained in the training set as being part of the violent class and as being part of the nonviolent class and for each word do the following:
    a. Get unigram frequencies in each category
    b. Count all words in the text
    c. Count the total number of words in the text
3. Get the probability for both violent and nonviolent data according to this formula:

$$P_{uni}(w_1\, w_2 w_3 \ldots . w_n) = P(w_1)P(w_2)P(w_3)P(w_n)$$

4. Set total final probability equal to one
5. Add one over the total number of words to each probability
6. Get $\log_2$ probability for each probability
7. Final probabilities for violent and non-violent is equal to $\log_2$ of the probability plus the final probability
8. Return two probabilities P_1 and P_2, respectively
9. If P_1 (violent probability) is greater than P_2 (nonviolent probability), then declare the web document as “violent”; otherwise, declare it as "nonviolent"

Figure 8: General steps of the unigram algorithm for testing phase

The rest of the methodology is as explained above in Section 3.3. The pseudo-code of our unigram algorithm is explained in Algorithm 1. This algorithm calculates the probability of the occurrence of the words that have appeared on the database based on the unigram model.

Algorithm 1: **Probability Calculation Algorithm based on the Unigram Model**

Input: Text data to be examined TEXT, dictionary for unigram counts UC, and an integer count of the unique words C
Output: Probability of TEXT based on UC and C

1. Function CalculateProbabilityOfText
2. UNIGRAMS = Extracted all unigrams from TEXT
3. RESULT_PROBABILITY = initialize to 1
4. for each unigram U in UNIGRAMS do
5. P = probability of U given UC and C

$$P_{uni}(w_1\, w_2 w_3 \ldots . w_n) = P(w_1)P(w_2)P(w_3)P(w_n)$$

6. if P is zero then
7. P = 1/C
8. X = Log2(P)
9. RESULT_PROBABILITY = RESULT_PROBABILITY + X
10. return RESULT_PROBABILITY

Algorithm 2 provides a score for a given text based on the trained model. The algorithm can be used with either a unigram or a bigram model.

Algorithm 2: Text Analysis and Labeling Algorithm

Input: violent_training_file, non-violent_training_file, testing_file
Output: list of "violent" or "non_violent" labels and probabilities corresponding to each record in testing_file

```
    # Initialize variables
1. RESULT_LIST = empty list
    # Create Data Structures based on training files
2. VC = Dictionary with frequency counts of all bigrams from violent_training_file
3. NVC = Dictionary with frequency counts of all bigrams from non-violent_training_file
4. VWORDS = integer count of all unique words in VC
5. NVWORDS = integer count of all unique words in NVC
   # Calculate Probabilities For Testing Data Records
6. for each record in testing_file do
7.     LABEL = expected label of this record, either "violent" or "non-violent"
8.     TEXT = raw textual data of the record which was examined
9.     PV = calculated probability of TEXT being violent given VC and VWords
       using CalculateProbabilityOfText function
10.    PNV = calculated probability of TEXT being non-violent given NVC and
       NVWords using CalculateProbabilityOfText function
11.    if PV > PNV then
12.        append ("violent", PV, PNV) to the RESULT_LIST
13.    else
14.        append ("non-violent", PV, PNV) to the RESULT_LIST
15. return RESULT_LIST
```

3.5 Bigram Algorithm

In this experiment, the goal of our algorithm was to find bigrams in all the training documents that were scanned. In other words, the machine-learning algorithm will go through the body of the text for each web document, whether it is labeled as violent or nonviolent, and finds the frequencies of the pair in each category. The general steps for implementing the bigram algorithm in both phases is shown in Figures 9 and 10.

1. Open web documents as a CSV file
2. Extract the pairs in the training data for violent and nonviolent separately and for each pair do the following:
    a. Get bigram frequencies in each category
    b. Count all pairs in the text
    c. Count the total number of words in the text
3. Get the probability for both violent and nonviolent data according to this formula:

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k \mid w_{k-1})$$

4. Set total final probability equal to one
5. Add one over the total number of words to each probability
6. Get $\log_2$ probability for each probability
7. Final probabilities for violent and non-violent is equal to $\log_2$ of the probability plus the final probability

Figure 9: General steps of the bigram algorithm for the training phase

1. Open a new web document as a CSV file, whether labeled as violent or nonviolent
2. Extract the pairs in the new web document to assign two probabilities based on the frequencies obtained in the training set: one being part of the violent class and another as being part of the nonviolent class and for each pair do the following:
    a. Get bigram frequencies in each category
    b. Count all pairs in the text
    c. Count the total number of words in the text
3. Get the probability for both violent and nonviolent data according to this formula:

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k \mid w_{k-1})$$

4. Set total final probability equal to one
5. Add one over the total number of words to each probability
6. Get $\log_2$ probability for each probability
7. Final probabilities for violent and non-violent is equal to $\log_2$ of the probability plus the final probability
8. Return two probabilities P_1 and P_2, respectively
9. If P_1 (violent probability) is greater than P_2 (nonviolent probability), then declare the web document as “violent”; otherwise, declare it as "nonviolent."

Figure 10: General steps of the bigram algorithm for testing phase

The rest of the methods are as explained in Section 3.3. The pseudo-code of our bigram algorithm is explained in Algorithm 3.

Algorithm 3: Probability Calculation Algorithm based on the Bigram Model

Input: Text data to be examined TEXT, dictionary for bigram counts BC, and an integer count of the unique words C
Output: Probability of TEXT based on BC and C

1. Function CalculateProbabilityOfText
2. BIGRAMS = Extracted all bigrams from TEXT
3. RESULT_PROBABILITY = initialize to 1
4. for each bigram B in BIGRAMS do
5. P = probability of B given BC and C ($P(w_1^n) = \prod_{k=1}^{n} P(w_k \mid w_{k-1})$)
6. if P is zero then
   P = 1/C
7. X = Log2(P)
8. RESULT_PROBABILITY = RESULT_PROBABILITY + X
9. return RESULT_PROBABILITY

The test part of the experiment can be carried out using the same algorithm that we used in the unigram, namely in Algorithm 2.

3.6 Summary

In this chapter, we explained in-depth the methodology used in this thesis. First, we described the compositional and functional aspects of our model. Then, the algorithms of the unigram and the bigram that were used to classify and determine the text documents were detailed.

Chapter 4

Implementation and Results

This chapter discusses the implementation of the proposed machine-learning algorithm explained in Chapter 3, followed by the results of the experiments conducted for web documents to accomplish classification of the emotional content.

4.1 Implementation

The software and hardware used for this thesis implementation is discussed in this section. For the purpose of this project, Python programming language was used as it is an easy-to-learn, interactive, interpreted, object-oriented and high-level programming language [45, 46]. Python was created by Guido van Rossum in the late eighties and early nineties. There are extensive open source libraries available in Python, which are used for the implementation of the system discussed in Chapter 3. Table 4 lists the major libraries used in this research work.

Python Library	Description
TextBlob	Used for processing textual data by providing a simple API that could deal with common NLP tasks. TextBlob also depends on NLTK, which is a Python library for NLP [47] [48]. We used the following TextBlob features: • Tokenization • N-grams • Get Word and Noun Phrase Frequencies
CSV Module	Applied to read and write tabular data, which was generated by Excel in CSV format without knowing the precise details [49].
Operator Module	Provides efficient functions that implement practically logical operations, comparison operations, sequence operations, and mathematical operations [50].
Math Module	Gives access to a variety of mathematical functions determined by the C standard [51]. The function that we used from this module is logarithmic function.

Table 4: Main Python libraries used for system implementation

4.2 Results

The performance algorithms used in this thesis for classifying the emotional contents of documents are discussed in this section, followed by the results of the experiments conducted.

4.2.1 Unigram Results

In the unigram approach, when we trained both the violent model, which includes 167 web documents, and the non-violent model, which includes 167 web documents, and used them to compute the log probability base2 on the test set for new web documents that were labeled as violent (166 web documents), the results showed high accuracy. The algorithm identified 150 web documents as violent that have been labeled previously as violent (90.36% accuracy), but missed 16 web documents (9.64%), labeling them as non-violent. In addition, when we trained the violent and non-violent model on a different test set that was labeled as non-violent (166 web documents) to assign their probabilities, 165 web documents were non-violent as have been labeled and only one web document was labeled violent by the system. Indeed, the results were extremely good with 0.6% false positive (but 9.64% false negative) as shown in Table 5.

	Violent Model	**Non-violent Model**
Web documents labeled as violent	90.36 %	9.64 %
Web documents labeled as non-violent	0.6 %	99.4 %

Table 5: Training both models to compute the probability on the test set for new web documents that are labeled as either violent or non-violent

To see these results visually, we plotted the results in two-dimensional graph. The x-axis presents violent log probabilities and y-axis represents violent log probabilities minus non-violent log probabilities for the violent model (red) and on the same graph, we also plotted violent log probabilities on the x-axis and violent log

probabilities minus non-violent log probabilities in y-axis for the non-violent model (green). As shown in Figure 11, there is good separation between the two labels.

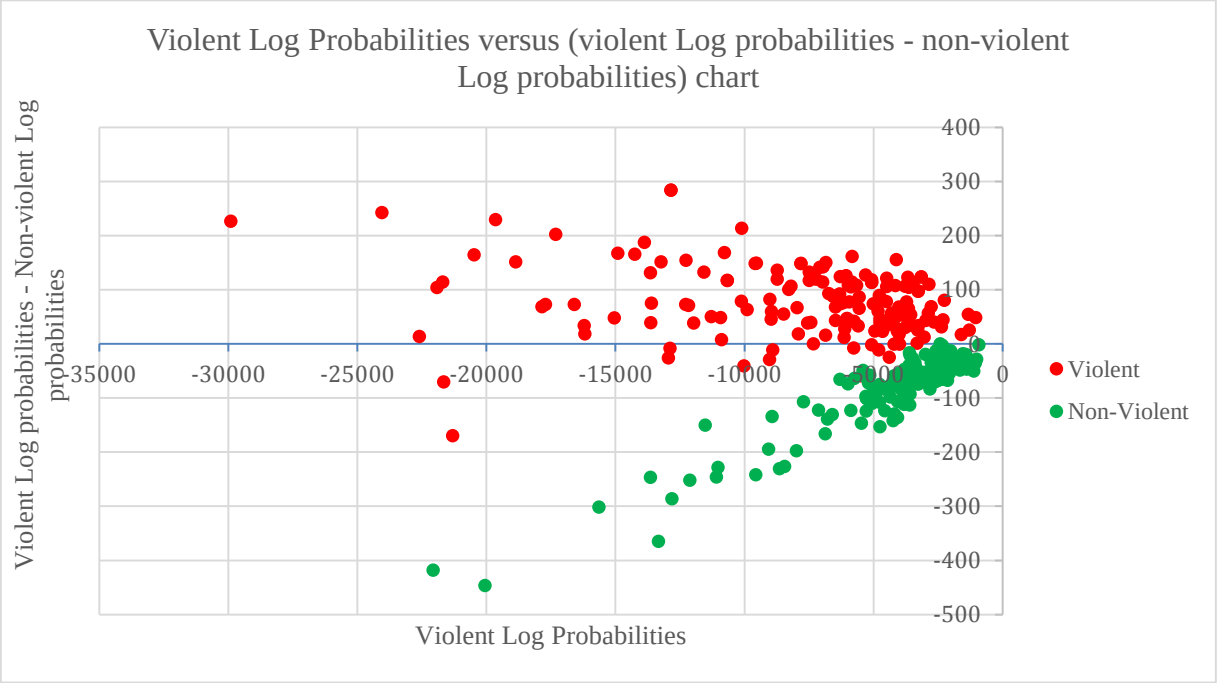

Figure 11: There is good separation between the violent and non-violent clouds when running the unigram models. The x-axis represents the violent log probabilities and the y-axis represents the violent log probabilities minus the non-violent log probabilities.

In the validation set, when we trained both models to compute the probability of new web documents being labeled as either violent or non-violent, the results were good with 1.2% false positive (but 1.8% false negative), as shown in Table 6.

	Violent Model	**Non-violent Model**
Web documents labeled as violent	98.2%	1.8%
Web documents labeled as non-violent	1.2%	98.8%

Table 6: Training both models to compute the probability on the validation set for a new web documents that are labeled as either violent or non-violent

To test our system furthermore, we designed a few new tests where we used some other attributes other than violent / non-violent. To achieve this goal, we created new datasets for a couple of other attributes. We used these datasets on only the unigram model not on the bigram model. This is because the bigram model requires a huge size of datasets to reach satisfactory results based on the unsatisfactory results that have been found from our pervious experiment with bigrams.

The first new attribute that we experimented with is "love". After finding some web documents are for love and others are for not love, we assigned some of them to the training set and others to the test set. Then, we trained both the love model and the not love model and used them to compute the probability on the test set for new web documents that were being labeled as either love or not love. The preliminary results showed only 10% false positive and zero false negative, as shown in Table 7.

	Love Model	**Not love Model**
Web documents labeled as love	90 %	10 %
Web documents labeled as not love	0 %	100 %

Table 7: Results of the test set for new web documents that are labeled either love or not love in the unigram model

The other new attribute that we chose is about "hate". We found some web documents that are for hate and others are for not hate. For each group, we assigned some web documents to the training set and some to the test set. Then, we trained the hate model and the not hate model and used them to compute the probability of new

web documents being labeled as either hate or not hate. The preliminary results were 20 % false positive (but zero false negative), as shown in Table 8.

	Hate Model	**Not hate Model**
Web documents labeled as hate	80 %	20 %
Web documents labeled as not hate	0 %	100 %

Table 8: Results of the test set for new web documents that are labeled either hate or not hate in the unigram model

4.2.2 Bigram Results

In the bigram approach, when we trained both the violent model and non-violent model and used them to compute the log probability base2 on the test set for new web documents that were labeled as non-violent, the results were good as zero were classified as violent and 100% were classified as non-violent. However, when we trained again the violent and non-violent models on the test set to assign the probability for web documents that were labeled as violent, the results were poor, with a zero percent probability that the web documents were classified as violent and 100% probability that they were classified as non-violent. Therefore, the bigram approach gives us unsatisfactory results. It seems that the classifier has labeled everything as non-violent (almost 100% false negative and zero false positive), as shown in Table 9.

	Violent Model	**Non-violent Model**
Web documents labeled as violent	0.60%	99.4%
Web documents labeled as non-violent	0%	100 %

Table 9: Training both models to compute the probability on the test set and validation set for new web documents that are labeled either violent or non-violent in the bigram model

However, when we plotted the results, we observed a relatively good degree of separation between the two labels, as shown in Figure 12.

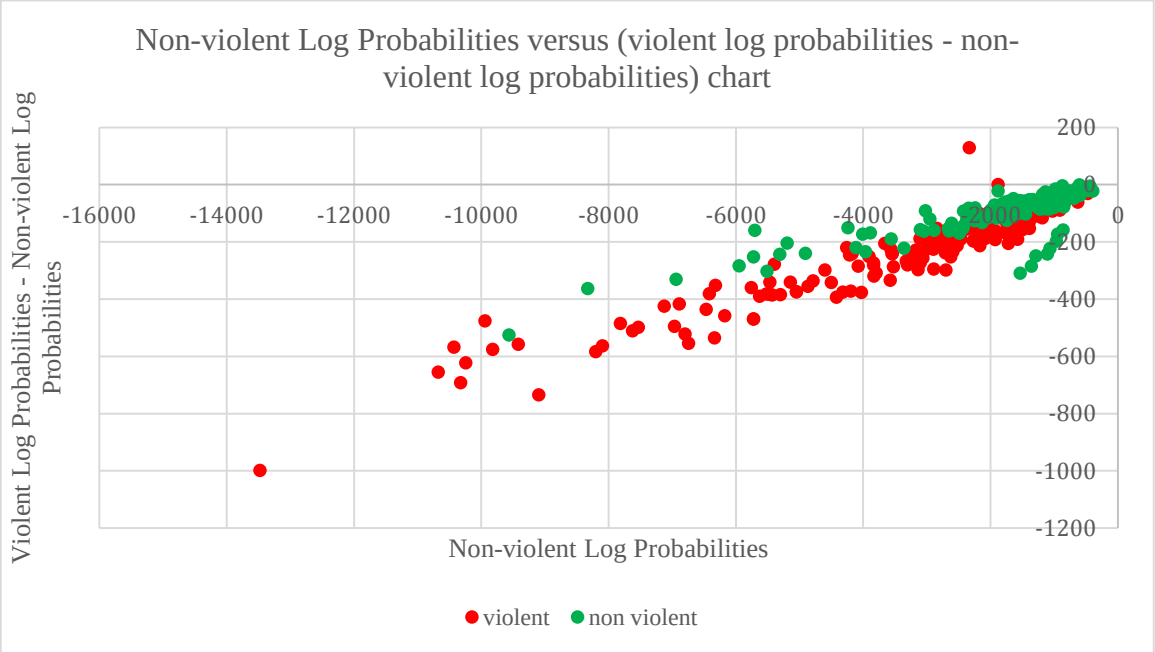

Figure 12: Plot of the violent and non-violent probabilities shows a good degree of separation between violent and non-violent when running the bigram model. The x-axis represents the non-violent log probabilities and the y-axis represents the violent log probabilities minus the non-violent log probabilities.

Although the simple equation P(violent)>P(non-violent) does not solve our problem the way that we wanted, we can see in Figure 12 that the two sub-spaces are almost separated. The separation is of course not a 100%; however, there is a good separation between them. These results suggest that instead of using the logic that if the violent probabilities are bigger than the non-violent probabilities, then the results would be violent, and vice versa, we can use more complicated logic to find the separation that is suggested by Figure 12.

Visual classification Solution

From Figure 12, we found a visual classification solution that we can use when we get a point (x, y) from a new web document. Then, based on this logic, we will label the web document as violent or non-violent. To apply this solution, we draw lines that should fairly well separate the violent and non-violent sup-spaces of the data points. Using the data shown in Figure 12, we drew two lines that distinguished the violent and non-violent labels and represented this in Figure 13. Next, we chose two points on each line to find the equations of these two lines. The points that we chose are shown below:

Line 1
- a (-8000, -400)
- b (-800, -60)

Line 2
- c (-8000, -800)
- d (-1000, -100)

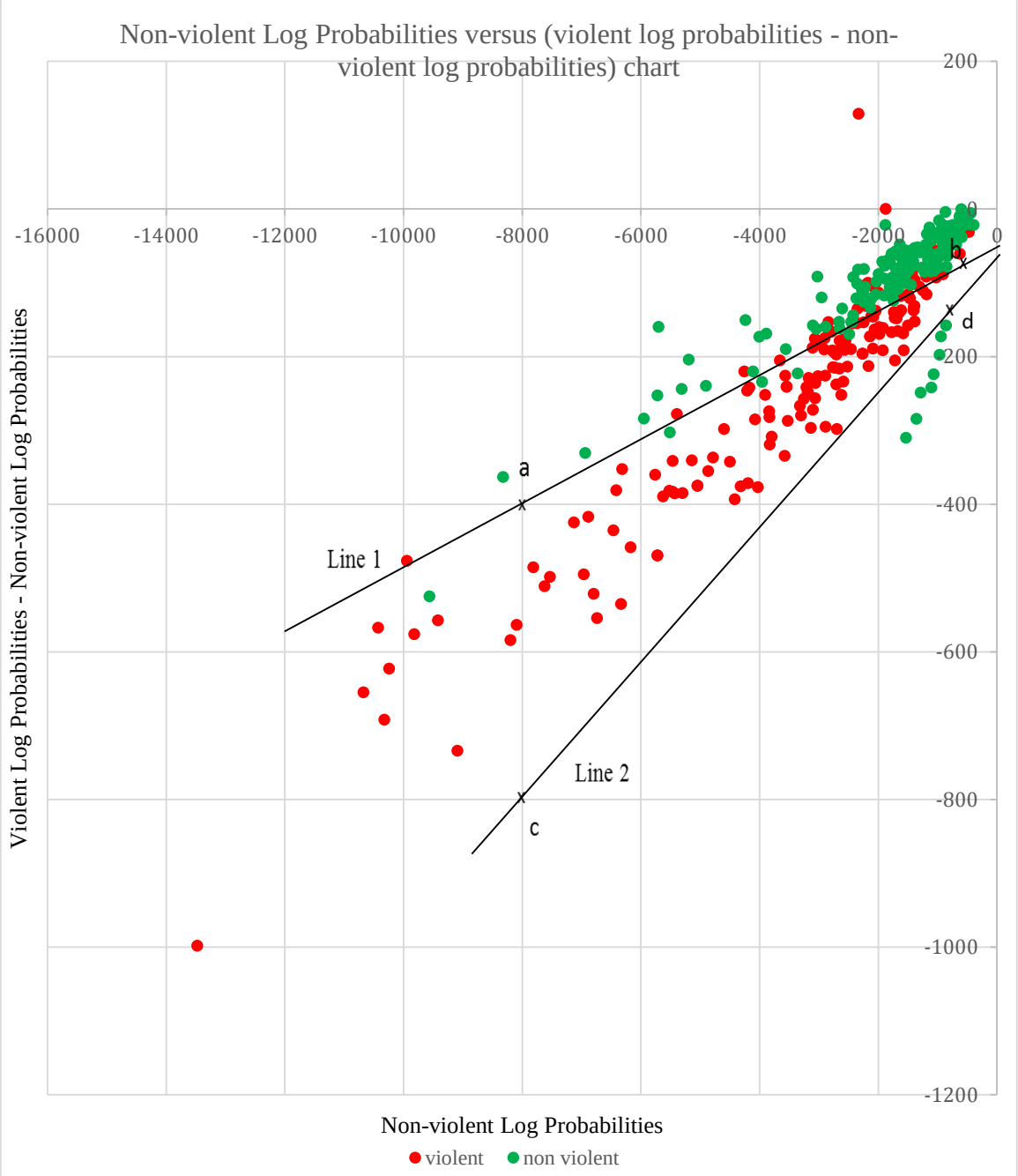

Figure 13: The implementation of our visual classification solution to separate the two labels (violent and non-violent) for the bigram model. The x-axis represents the non-violent log probabilities and the y-axis represents the violent log probabilities minus the non-violent log probabilities.

Now, we applied these points to the following equation to find out the values of α and β for both lines:

$$y = \alpha x + \beta \qquad (4.1)$$

Thus, the equation for line 1 is:

$$y = .0472`x - 22.2` \qquad (4.2)$$

And the equation for line 2 is:

$$y = .1\,x \qquad (4.3)$$

In the next step, we found new web documents on which to run our system to assign their violent and non-violent probabilities; one web document was labeled as violent, and the other one was labeled as non-violent, shown in Table 10.

	Violent Model	**Non-Violent Model**	**PV – PNV**	**Is below or above line 1**	**Is below or above line 2**	**Based on the logic violent or non-violent**
Violent Web document	- 920.3	- 858.9	- 61.4	Below	Above	Violent
Non-violent Web document	- 1189.8	- 1140.7	- 49.1	Above	Above	Non-Violent

Table 10: Testing our visual classification solution on two new web documents (violent and non-violent)

After we got (x, y) from these new web documents, we wanted to label them as violent or non-violent by substituting the values of x and y in the equations 4.2 (representing line 1) and 4.3 (representing line 2). The values of x and y that were assigned from one violent web document were -920.3 and -61.4, respectively. The values of x and y that were assigned from one non-violent web document were -1189.8 and -49.1, respectively.

Next, we inserted the obtained results from both equations to our dataset result, as shown in Figure 14.

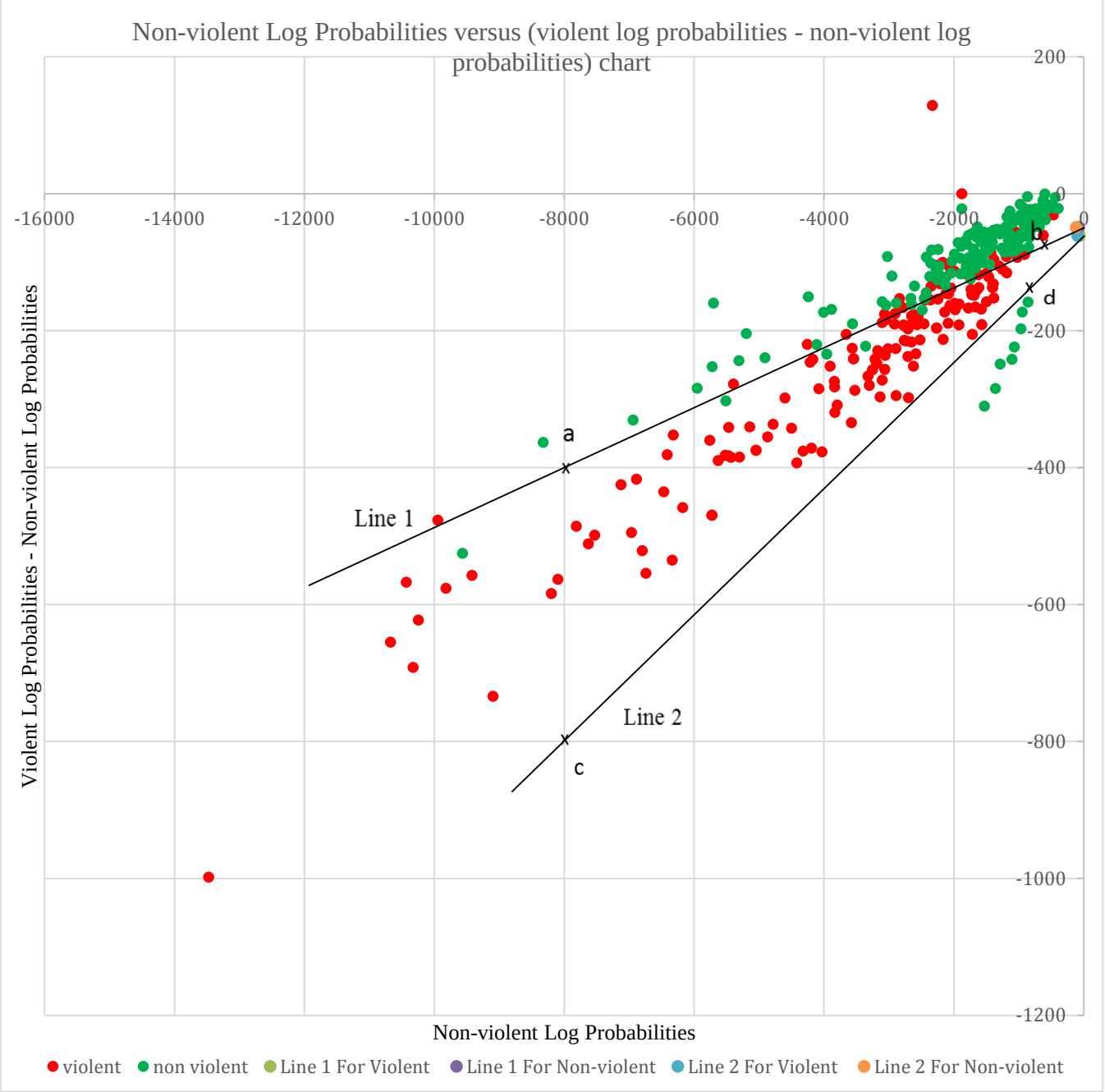

Figure 14: New points (x, y) of the violent and non-violent web documents were applied to the results for the bigram model shown in Figure 13. The x-axis represents the non-violent log probabilities and the y-axis represents the violent log probabilities minus the non-violent log probabilities.

From Figure 14, we can observe that points that exist between line 1 and line 2 have been labeled as a violent web document, while others have been labeled as a non-violent web document. In addition, the obtained result of the violent web document appeared below line 1, whereas the obtained result of the non-violent web

document appeared above line 1; on the other hand, both obtained results of violent web document and non-violent web document appeared above line 2, suggesting separation between the two labels. According to these results, instead of stating that if violent probabilities greater than non-violent probabilities, we can use the above statements to identify the web document as violent or non-violent. Thus, we found new logical assumptions as follows:

- If any point is shown between the two lines (line 1 and line 2), it will be labeled as violent.
- Any point shown outside of these two lines will be labeled as non-violent.

4.3 Evaluations and Discussions

The results of our experiments suggest that the unigram approach is better than the bigram approach at identifying the nature of the web documents, considering all circumstantial factors, such as the size of the training datasets. Similar to the previous research findings, although going on larger values of *N* in an N-gram, would increase the accuracy of a system, it often reduces the performance [36] [37]. According to our findings, it is obvious that the unigram model provides a stronger assumption than the bigram model. In the unigram model, the occurrence of each word is independent and consequent. Conversely, in the bigram model, the performance of a system becomes intractable and low because the probability of a new word depends on the probability of one previous word [20]. In addition, the unigram model is preferable to the bigram model because of the sequence of words. Any document in the training set can include the same word sequence or even all of the words that are included in the test set, and vice versa. Thus, increasing the size of our dataset might improve the performance of the bigram model.

Moreover, the running time of the unigram models on violent test set was 13 seconds to find the similarity of each word between the training and the test set, on an Intel® Core™ i5-4300U CPU @ 2.50 GHz, using Python 3.4.3 compiler. While, the running time of the bigram models on violent test set on the same machine and using the same compiler as well, was approximately 40 minutes to find the similarity of all pairs of words between the training and the test set. Indeed, in addition to that the result was much better in unigram approach as compared to the bigram models, the complexity of running the unigram model is much lower than running the bigram model.

4.4 Comparison between Unigram Models and Bigram Models

The selection of classified models can influence the experimental outcomes. Therefore, we experimented with two different classification models of N-gram to find which model performs better for detecting web documents and determining their sentimental content. The two well-known models that were used on the experimental dataset are unigram model and bigram model. We implemented the algorithm with unigram model and bigram model on the same database of violence attribute. The unigram model identified the web documents as violent that have been labeled previously as violent (90.36% accuracy) and the web documents as non-violent that have been labeled as non-violent with 99.40 % accuracy. In contrast, the bigram algorithm identified all web documents as non-violent either have been labeled previously as violent or non-violent, by 99.60 %. Thus, the true positive (TP) of the unigram algorithm is greater than the true negative (TN). Whereas, the true positive (TP) of the bigram algorithm is smaller than the true negative (TN), as illustrated on the following formula.

$$TP_{Unigram} > TP_{Bigram} \qquad (4.4)$$

Figure 15 shows the obtained results of both unigram algorithm and bigram algorithm based on false positive, false negative, and how good is the accuracy of both classifiers. To find the accuracy of our classifiers, we used the following formula:

$$Accuracy = \frac{correct\ number\ of\ web\ documents's\ labelling}{Total\ number\ of\ web\ documents} \times 100 \qquad (4.5)$$

In addition, we have considered as well the obtained results of the visual classification solution with the bigram models in this comparison. The visual classification algorithm has been implemented by plotting the results in two-dimensional graph in order to find a good separation between different labels (violent and non-violent). As a result, there was a relatively good separation between these two labels for the bigram model (with almost 70%).

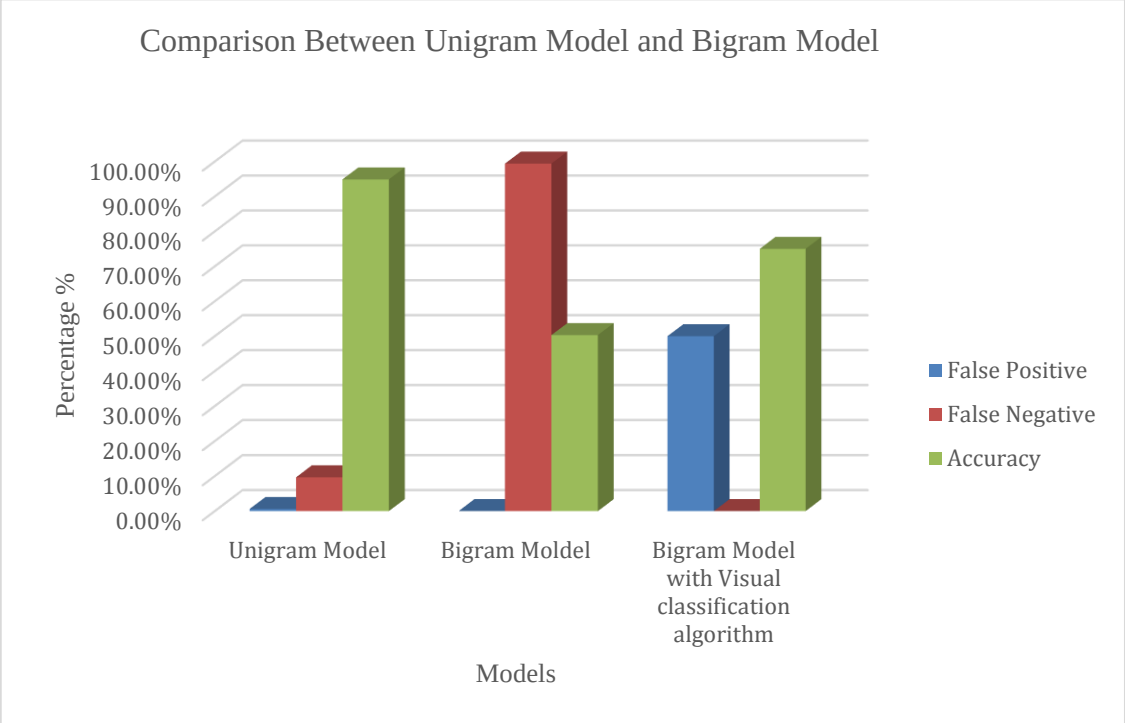

Figure 15: Performance differences between the unigram model, the bigram model, and the bigram model with visual classification algorithm based on false positive, false negative, and accuracy

Indeed, the results of our experiments illustrate that the unigram model overcomes the bigram model at detecting the sentimental content of web documents. This could be caused by the following reasons:

- The occurrence of words in the bigram approach is dependent, while the occurrence of each word is independent in the unigram approach.
- The sequence of words is important and essential in the bigram approach, while this is not important in the unigram approach.
- The bigram approach might need a huge size of dataset to perform well not as much as unigram approach needs.

4.5 Summary

In this chapter, we provided a brief introduction to the software and hardware that were used in this thesis. Following this, we explained the implementation and the performance of the proposed model in detail. Then, we discussed the results of these experiments.

Chapter 5

Conclusions and Future Works

This chapter of the thesis presents the conclusions of our research and closes with presenting future works.

5.1 Conclusions

The automatic labeling of web documents for emotional categories using NLP is done in the thesis. In our experiment, we use “violence” as our only attribute and the English language as our target language. After designating some web documents as violent and some as nonviolent, we divide the data randomly into the training set, the validation set and the test set. Then, we trained the statistical parameters of the model on the training set, and then used them to compute probabilities on the test set. We used unigrams and bigrams as our features, which are regarded as the most broadly useful practical tools in the new paradigm of statistical analysis in language processing environments. In unigram approach, the result was much better than when using the bigram models. This could be in part because of the fact that the occurrence of each word is independent in the unigram model, while the occurrence of each word is dependent on the previous one in the bigram model. For instance, when writing, it is unlikely that two different writers, even if using the same vocabulary, will place the words in exactly the same sequence. The findings in this thesis suggest a way of extracting emotions in texts, which can be very useful for avoiding material that could be harmful or undesirable for readers.

5.2 Future Works

We would like to continue working on our system and improve the performance on both unigram and bigram approaches. Some of the tasks that we can undertake for this purpose are listed below:

- Increasing the size of our database to achieve more satisfactory results with both unigram and bigram models. In fact, if we had extremely large data sets, we could even go to higher values on *N* in our N-grams, like N=3 or 4.
- Only web documents in English language were considered in this thesis. Without loss of generality, the same methodology can be applied to other languages, such as French and Arabic.
- Increasing the number of attributes, such as funny, hate, depressed, and so on. In these cases, people who would like to read about specific sentimental content of a document could easily choose the appropriate document. In addition, vulnerable populations (e.g., young children, individuals with chronic health conditions) who need to avoid reading potentially harmful information can utilize this approach to avoid reading negative documents.
- Applying the methodology to User Generated Context (UGC), such as social media comments, tweets, etc. Additionally, we spend a large amount of time in direct contact with smart devices. However, these devices currently do not understand the emotional status of our SMS message. Applying the technique in this thesis could allow smart devices to identify the sentiment of human messages and respond appropriately. For example, if someone feels depressed, the device could turn music to cheer him/her up or could send a message to his/her favourite contact list to invite them to talk to the individual. Moreover,

if a smart decide can understand the sentiment of a human message, it would be able to respond in emergency situations, such as automatically calling 911.

Appendix I

The following Figures A-1.1 and A-1.2 show snapshots for the results of running unigram model on violent and non-violent web documents, respectively.

```
violent -1290.1698686696845 -1315.3572183317121
non_violent -4357.559605198068 -4382.134594243371
violent -4511.143836818965 -4545.054807264063
violent -3302.719545119579 -3303.8300496432125
non_violent -2140.3819285046593 -2154.474903617328
violent -6570.813482532718 -6658.760157001378
violent -5308.341396064344 -5435.439306165392
violent -11290.634336775232 -11341.16795562672
violent -11573.195078166988 -11705.779043720304
violent -6854.4450811843 -6870.371686448053
violent -6028.537376993691 -6075.02614000272
violent -5069.627700031945 -5183.075568103934
violent -13875.427116309225 -14062.658006951944
violent -19649.183012302226 -19878.844803739506
violent -7812.586041259038 -7961.260018979664
violent -4749.938883473756 -4796.3158143274995
non_violent -5398.909093695843 -5457.964993483734
violent -3632.123346027186 -3736.567440725397
non_violent -4203.203934690867 -4203.68063479023
violent -3272.774288051149 -3370.025576358221
violent -5756.515375217607 -5797.993522658825
violent -10670.207318295006 -10787.279248131663
violent -7812.586041259038 -7961.260018979664
violent -6737.011833960202 -6829.755560904506
violent -13227.410439984016 -13378.822246898613
violent -4497.159154126815 -4574.992164720537
violent -4157.439029092883 -4265.32491910944
violent -3143.3675176100555 -3267.256939635401
violent -6961.433308634795 -7104.119069366342
violent -3668.3267936805883 -3791.586379533232
violent -12850.760084980036 -13134.909561501578
violent -2301.8667095710734 -2346.0415972007177
violent -1328.5490840076677 -1383.031669540397
violent -12169.504413649605 -12240.837451997797
non_violent -12873.040941182238 -12881.165051894846
violent -8732.394361304376 -8851.832624123881
violent -5935.984432749312 -6013.319560890598
non_violent -5761.00429595192 -5768.375273815112
violent -7966.014733988171 -8032.729689429649
violent -20479.21000664789 -20643.47111893394
violent -7489.299559709493 -7606.234643562097
violent -22601.871045046933 -22615.657329999827
non_violent -4785.387845927912 -4796.732888396619
violent -4195.107285794063 -4242.3790620617365
violent -3714.0140224409156 -3791.296174279066
violent -10774.31398349038 -10942.942458573985
violent -14900.81343417774 -15068.22020073708
violent -5880.135342171313 -5985.349078851182
violent -7085.736951633114 -7227.123135873476
violent -1598.5787388645983 -1616.1451494302398
violent -16210.748084480418 -16244.733984517638
violent -12850.760084980036 -13134.909561501578
non_violent -7310.039439178049 -7328.706859778687
violent -16596.160630000137 -16669.026394013144
violent -4950.338455902588 -4974.020489405412
```

Figure A-1.1: Snapshot for the results of running unigram model on violent web documents

```
non_violent -3842.088915349227 -3925.410692977415
non_violent -1889.878474113872 -1927.059873053706
non_violent -3930.0259525056767 -4066.102939892555
non_violent -3310.2678000921446 -3353.023461784429
non_violent -3482.560605669161 -3595.174747874821
non_violent -1353.9061357440921 -1382.359971083593
non_violent -2047.1374605878036 -2081.0445156557244
non_violent -6703.132324590668 -6869.427322802123
non_violent -3500.071007011393 -3539.983386483106
non_violent -1403.52049609642 -1434.7860539879182
non_violent -1667.2679399467747 -1698.6886445200053
non_violent -2481.135562986686 -2540.520971017818
non_violent -3828.6269513436027 -3912.332110113384
non_violent -2724.7424703292227 -2807.838019427406
non_violent -9319.65195642066 -9561.598700308343
non_violent -4934.39339154743 -5040.901927133011
non_violent -4440.9790319891235 -4564.716172399565
non_violent -12964.53353284583 -13329.530330144764
non_violent -5080.612542009763 -5138.251738120377
non_violent -3909.6552405563993 -3962.795558592438
non_violent -2466.210299439548 -2534.8854993353248
non_violent -4684.468175797949 -4759.532483798875
non_violent -5169.034728681636 -5269.794887141497
non_violent -2557.6483823425647 -2587.5223218149667
non_violent -2058.916406084911 -2126.5258010282528
non_violent -15335.753802244459 -15637.3481072967
non_violent -3241.593644507768 -3291.185303309944
non_violent -2338.8849675209926 -2402.09655933877
non_violent -4778.638371476461 -4867.463160222722
non_violent -1480.035851480656 -1523.98591823195
non_violent -3512.3522255245225 -3560.293809041535
non_violent -1063.6432591388796 -1113.4665078427183
non_violent -3548.155529824567 -3596.966423611886
non_violent -4946.6144510067115 -5056.812968233572
non_violent -2075.425976975946 -2104.0315239425145
non_violent -977.3883018236486 -1005.6315104496149
non_violent -8868.049918183568 -9062.684072779763
non_violent -4256.139191682877 -4331.458438006612
non_violent -10795.106826025984 -11023.11206464988
non_violent -4329.775055125921 -4417.69308798193
non_violent -3751.6734193814164 -3799.005996099552
non_violent -3197.7672203436205 -3272.0596926471057
non_violent -2395.986031884553 -2455.653702763126
non_violent -1342.4180253980833 -1389.314608291003
non_violent -3653.904432976034 -3732.1875757331322
non_violent -3484.0245368508026 -3549.377836431904
non_violent -8406.732044832828 -8637.44367471001
non_violent -3960.56799058427 -4067.03717490964
non_violent -5150.530045555942 -5214.233045742806
non_violent -1270.2437473521968 -1310.9269328426988
non_violent -5673.923591422483 -5737.757993702702
non_violent -2841.4017057085807 -2904.7642007135046
non_violent -908.317307763634 -910.1690572452537
non_violent -2004.3985267020257 -2046.8706569058395
non_violent -2528.8915746364387 -2574.9653915814765
non_violent -2246.1396041512708 -2302.4588791064057
non_violent -5750.228277716405 -5873.38881401067
non_violent -2007.1881785042308 -2046.9668423523906
```

Figure A-1.2: Snapshot for the results of running unigram model on non-violent web documents

Appendix II

Figures A-2.1 and A-2.2 provide snapshots for the results of running bigram model on violent and non-violent web documents, respectively.

```
non_violent -575.668011925464 -604.0119752875529
non_violent -2001.6092963266497 -2114.845177016419
non_violent -2060.44756943994 -2223.437315567377
non_violent -1583.3778724360163 -1674.2171119926084
non_violent -1055.5278534132817 -1115.775912603829
non_violent -3105.7078528999386 -3293.77350453972
non_violent -2440.4204662155203 -2595.4508015834626
non_violent -5142.419615873877 -5482.944090504086
non_violent -5465.432129155296 -5806.61560856604
non_violent -3058.39846650149 -3294.304896884675
non_violent -2776.2023301319073 -2967.8571072143704
non_violent -2266.632262673685 -2462.4574262253686
non_violent -6335.972024726754 -6871.21030760613
non_violent -9418.12892439389 -9975.484439958962
non_violent -3542.841310479327 -3783.6423753179597
non_violent -2140.345230647371 -2312.800353281333
non_violent -2352.3230894407425 -2487.481898871199
non_violent -1653.1510177883983 -1793.7868770076052
non_violent -1777.6598651158495 -1944.5281561765275
non_violent -1402.2244652037919 -1534.339456645914
non_violent -2708.6363588134022 -2946.207382012139
non_violent -5045.960420714356 -5420.67813207059
non_violent -3542.841310479327 -3783.6423753179597
non_violent -2890.399238560183 -3185.3426462345915
non_violent -6173.084105297569 -6631.237087917726
non_violent -1877.4794592479052 -2081.6598756911153
non_violent -1978.8626214030558 -2147.9623697360075
non_violent -1405.2035349738544 -1542.0591609153055
non_violent -3188.181669555024 -3436.209402303351
non_violent -1623.8544877017796 -1741.9337164607118
non_violent -5723.804369220064 -6193.135080640284
non_violent -1026.2105851914455 -1119.0049405939646
non_violent -625.5966811769697 -686.2889904089427
non_violent -5469.909102603926 -5852.742285271803
non_violent -5759.269482659077 -6119.43715468346
non_violent -4076.9499652720774 -4361.776869373056
non_violent -2765.6004337659083 -2979.823827632852
non_violent -2690.321724925918 -2883.136992560849
non_violent -3522.8023400055536 -3809.9343664036746
non_violent -9094.264238643415 -9828.230062733874
non_violent -3172.601350391628 -3401.535530805344
non_violent -10324.746784975843 -11016.500313860364
violent -2203.0337488632567 -2332.026195179460 5
non_violent -1843.1340678781219 -1959.5333975779922
non_violent -1680.240875092838 -1772.1079197691738
non_violent -4784.171213964089 -5120.816637584143
non_violent -6795.557400107296 -7316.982193993823
non_violent -2787.1221323794407 -2953.345739631857
non_violent -3134.5862989756974 -3431.2912195908193
non_violent -617.734176737785 -647.4133216225543
non_violent -7126.894580905826 -7551.56179111771
non_violent -5723.804369220064 -6193.135080640284
non_violent -3102.6055201221325 -3374.707091958047
non_violent -7531.222118783328 -8029.636596678007
non_violent -2281.3718704235844 -2385.700585198711
non_violent -3062.962736310546 -3319.325949192802
non_violent -2590.099879766315 -2767.120347786638
non_violent -3039.811661973637 -3223.6503919975103
```

Figure A-2.1: Snapshot for the results of running bigram model on violent web documents

```
non_violent -1619.1730318577693 -1714.9563093334773
non_violent -833.2569845019598 -888.2533628460782
non_violent -1821.7290053319118 -1889.9277489498072
non_violent -1478.9894231733417 -1561.7495369233018
non_violent -1714.7134551744966 -1772.3606561286138
non_violent -596.4827047542428 -609.9769713740919
non_violent -836.0389688745748 -896.8374935102383
non_violent -3023.8333274193474 -3115.5231290476177
non_violent -1664.8433774649227 -1751.2222286463054
non_violent -569.95665627393 -586.5209245739903
non_violent -686.4160032906099 -734.4807150005522
non_violent -1163.1447754802675 -1231.4399600490704
non_violent -1774.5014949891365 -1835.1565242985644
non_violent -1336.9917071920013 -1388.7726302180574
non_violent -4111.978535884183 -4332.182417649397
non_violent -2242.772433270421 -2350.956169196693
non_violent -1892.537106907519 -1986.6538728618318
non_violent -5702.0201734411385 -5861.476502351099
non_violent -2487.9394199102326 -2657.4445766337267
non_violent -1761.0010002482525 -1878.4638590760062
non_violent -1110.2095695734727 -1174.0575318405456
non_violent -2130.127333439439 -2263.171194759003
non_violent -2248.5699528423543 -2374.3463837389318
non_violent -1064.9236827289417 -1148.91602021203
non_violent -911.9318776079966 -973.3128962304452
non_violent -6937.669556417636 -7268.215444290891
non_violent -1426.0711340258938 -1497.1638100119437
non_violent -1138.8137405601638 -1198.29422792127
non_violent -2223.4680717684596 -2328.9406752783793
non_violent -628.6527528377004 -638.3340731862695
non_violent -1554.9417247884376 -1655.3783284870995
non_violent -489.58259496505036 -506.17338561456774
non_violent -1576.1896364566458 -1643.6386534578412
non_violent -2343.4632530357776 -2425.451074800245
non_violent -904.1507486524305 -951.0485258690721
non_violent -393.61693420241534 -415.0435665796796
non_violent -4240.040090694275 -4390.3988436450645
non_violent -2120.029103439525 -2241.2981899327783
non_violent -5310.595607790276 -5554.409642590597
non_violent -1992.5481153651892 -2080.6805433578656
non_violent -1766.4151853789842 -1859.2691451030216
non_violent -1536.8479953270912 -1592.7673291953574
non_violent -1127.3593498606149 -1169.1613095410924
non_violent -600.2335103571809 -600.478655211773
non_violent -1691.9844136282466 -1800.3125267317694
non_violent -1658.1237078617594 -1714.3070228121933
non_violent -4006.162815057612 -4179.219845154189
non_violent -1685.5253748296725 -1743.6524082622855
non_violent -2371.6214611633154 -2492.6316143600548
non_violent -661.6893375882387 -686.0879723454792
non_violent -2435.0547202361977 -2588.1912958498237
non_violent -1262.2536530777456 -1341.4606220306764
non_violent -455.70103488238175 -476.72315460104534
non_violent -915.5393591430056 -962.3071767848865
non_violent -1219.3068565460471 -1276.662116278397
non_violent -1119.2807895141834 -1203.9457227049945
non_violent -2607.4145996697234 -2742.050113260572
non_violent -979.1863077756574 -994.5760775580902
```

Figure A-2.2: Snapshot for the results of running bigram model on non-violent web documents

Bibliography

[1] G. Chowdhury, 'Natural language processing', *Annual Review of Information Science and Technology*, vol. 37, no. 1, pp. 51-89, 2003.

[2] D. Jurafsky and J. Martin, *Speech and Language Processing*. Upper Saddle River, NJ: Pearson Prentice Hall, 2009.

[3] E. Kumar, *Natural Language Processing*. New Delhi: I.K. International Publishing House, 2011.

[4] S. Feldman, 'NLP meets the jabberwocky', *Online*, vol. 23, pp. 62-72, 1999.

[5] E. Liddy, 'Enhanced text retrieval using natural language processing', *Bulletin of the American Society for Information Science and Technology*, vol. 24, no. 4, pp. 14-16, 1998.

[6] D. Li, 'Between English and Esperanto: What does it take to be a world language?', *International Journal of the Sociology of Language*, vol. 2003, no. 164, 2003.

[7] C. Meierkord, 'Lingua Francas as Second Languages', *Encyclopedia of Language & Linguistics.* Elsevier, Amsterdam, pp. 163-171, 2006.

[8] D. Beeferman, A. Berger and J. Lafferty, 'Statistical Models for Text Segmentation', *Machine Learning*, vol. 34, no. 13, pp. 177-210, 1999.

[9] L. Rabiner, 'A tutorial on hidden Markov models and selected applications in speech recognition', *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257-286, 1989.

[10] P. Koehn, 'Europarl: A Parallel Corpus for Statistical Machine Translation', in *Conference Proceedings: The Tenth Machine Translation Summit*, Phuket, Thailand, 2005, pp. 7-86.

[11] I. Mani, *Automatic summarization*. Amsterdam: J. Benjamins Pub. Co., 2001.

[12] J. Quinteiro-Gonzalez, P. Hernandez-Morera and A. Lopez-Rodriguez, *Text Classification for Sentiment Analysis*, 1st ed.

[13] J. Piskorski and R. Yangarber, 'Information Extraction: Past, Present and Future', in *Multi-source, Multilingual Information Extraction and Summarization*, 1st ed., T. Poibeau, H. Saggion, J. Piskorski and R. Yangarber, Ed. Springer-Varlag Berlin Heidelberg, 2013, pp. 23-49.

[14] A. Aho and J. Ullman, *The theory of parsing, translation and compiling*. Englewood Cliffs (N.J.): Prentice-Hall, 1972.

[15] C. Akkaya, J. Wiebe and R. Mihalcea, 'Subjectivity Word Sense Disambiguation', in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, Singapore, 2009, pp. 190-199.

[16] S. Mittal and A. Mittal, 'Versatile question answering systems: seeing in synthesis', *International Journal of Intelligent Information and Database Systems*, vol. 5, no. 2, p. 119, 2011.

[17] F. Goldman-eisler, 'Speech production and the predictability of words in context', *Quarterly Journal of Experimental Psychology*, vol. 10, no. 2, pp. 96-106, 1958.

[18] H. Schmid, 'Probabilistic Part-of-Speech Tagging Using Decision Trees', in *International Conference on New Methods in Language Processing*, Manchester, UK, 1994.

[19] B. Liu, 'Sentiment analysis and opinion mining', *Synthesis Lectures on Human Language Technologies*, vol. 5, no. 1, pp. 1-167, 2012.

[20] S. Mukherjee and P. Bhattacharyya, 'Sentiment analysis: A literature survey', *arXiv preprint arXiv:1304.4520*, 2013.

[21] B. Pang, L. Lee and S. Vaithyanathan, 'Thumbs up?: Sentiment classification using machine learning techniques', in *Proceedings of the Acl-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, Association for Computational Linguistics, 2002, pp. 79-86.

[22] P. Cohen, 'The role of natural language in a multimodal interface', in *Proceedings of The 5th Annual ACM Symposium on User Interface Software and Technology*, ACM, 1992, pp. 143-149.

[23] A. Kawtrakul, 'An overview of a role of natural language processing in an intelligent information retrieval system'. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.12.5773&rep=rep1&type=pdf. [Accessed: 14- Oct- 2015].

[24] C. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press, 1999.

[25] P. Brown, P. Desouza, R. Mercer, V. Pietra and L. J. C., 'Class-based n-gram models of natural language', *Computational Linguistics*, vol. 18, no. 4, pp. 467-479, 1992.

[26] P. Clarkson and R. Rosenfeld, 'Statistical language modeling using the CMU-Cambridge toolkit', *Eurospeech*, vol. 97, pp. 2707-2710, 1997.

[27] C. Zhai and J. Lafferty, 'A study of smoothing methods for language models applied to ad hoc information retrieval', in *Proceedings of the 24th Annual International ACM Sigir Conference on Research and Development in Information Retrieval*, ACM, 2001, pp. 334-342.

[28] B. MacCartney, *NLP lunch tutorial: Smoothing*, 1st ed. 2005.

[29] L. Hai Son, A. Allauzen and F. Yvon, 'Measuring the influence of long range dependencies with neural network language models', in *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, Association for Computational Linguistics, 2015, pp. 1-10.

[30] F. Song and W. Croft, 'A general language model for information retrieval', in *Proceedings of the Eighth International Conference on Information and Knowledge Management*, ACM, 1999, pp. 316-321.

[31] E. Charniak, *Statistical Language Learning*. Cambridge, MA: MIT Press, 1993.

[32] D. Lewis, 'Representation and learning in information retrieval,' Ph.D. dissertation, University of Massachusetts, Massachusetts, MA, 1992.

[33] D. Lewis, 'Feature selection and feature extraction for text categorization', in *Proceedings of the Workshop on Speech and Natural Language*, Association for Computational Linguistics, 1992, pp. 212-217.

[34] D. Lewis, 'An evaluation of phrasal and clustered representations on a text categorization task', in *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, 1992, pp. 37-50.

[35] C. Tan, Y. Wang and C. Lee, 'The use of bigrams to enhance text categorization', *Information Processing & Management*, vol. 38, no. 4, pp. 529-546, 2002.

[36] D. Mladenic and M. Grobelnik, 'Word sequences as features in-text learning', in *Proceedings of the 17th Electrotechnical and Computer Science Conference (ERK-98)*, Ljubljana, Slovenia, IEEE Section, 1998, pp. 145-148.

[37] J. Fürnkranz, 'A study using n-gram features for text categorization', *Austrian Research Institute for Artificial Intelligence*, vol. 3, pp. 1-10, 1998.

[38] H. Schütze, D. Hull and J. Pederson, 'A comparison of classifiers and document representations for the routing problem', *in Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, 1995, pp. 229-237.

[39] K. Dave, S. Lawrence and D. Pennock, 'Mining the peanut gallery: Opinion extraction and semantic classification of product reviews', in *Proceedings of the 12th International Conference on World Wide Web*, ACM, 2003, pp. 519-528.

[40] D. D and B. S, 'Identifying Emotional Expressions, Intensities and Sentence level Emotion Tags using a Supervised Framework', *PACLIC 24 Proceedings*, vol. 24, pp. 95-104, 2010.

[41] M. Ghiurcau, C. Rusu and J. Astola, 'Speaker recognition in an emotional environment', in *Proceedings of Signal Processing and Applied Mathematics for Electronics and Communications (SPAMEC 2011)*, 2011, pp. 81-84.

[42] I. Shahin, 'Speaker identification in emotional environments', *Iranian Journal of Electrical and Computer Engineering*, vol. 8, no. 1, pp. 41-46, 2009.

[43] I. Shahin, 'Improving Speaker Identification Performance Under the Shouted Talking Condition Using the Second-Order Hidden Markov Models', *EURASIP Journal on Advances in Signal Processing*, vol. 2005, no. 4, pp. 482-486, 2005.

[44] A. Hussainalsaid, B. Zahir Azami and A. Abhari, 'Automatic Classification of the Emotional Content of URL Documents Using NLP Algorithms,' in *SpringSim2015*, Alexandria, VA, USA, 2015.

[45] M. Goodrich, R. Tamassia and M. Goldwasser, *Data Structures and Algorithms in Python.* Hoboken, NJ: Wiley, 2013.

[46] G. van Rossum and F. Drake, 'Python Tutorial', 2015. [Online]. Available: http://docs.python.org/tut/tut.html. [Accessed: 14- Oct - 2015].

[47] S. Loria, 'TextBlob 0.11.0 Documentation', Textblob.readthedocs.org, 2015. [Online]. Available: https://textblob.readthedocs.org/en/dev/api_reference.html?highlight=main%20function. [Accessed: 14- Oct - 2015].

[48] S. Loria, *Textblob Documentation*, Release 0.10.0., 1st ed. 2015.

[49] Docs.python.org, 'CSV File Reading and Writing'. [Online]. Available: https://docs.python.org/2/library/csv.html. [Accessed: 14- Oct- 2015].

[50] Docs.python.org, 'Operator', 2015. [Online]. Available: https://docs.python.org/2/library/operator.html. [Accessed: 14- Oct- 2015].

[51] Docs.python.org, 'Math'. [Online]. Available: https://docs.python.org/2/library/math.html. [Accessed: 14- Oct- 2015].