

# Information Extraction From Free Text Comments in Questionnaires

Kartik Ramachandran

A thesis submitted to Auckland University of Technology for the fulfillment of the  
requirements for the degree of Master of Computer and Information Sciences (MCIS)



2017

School of Computing and Mathematical Sciences

Primary Supervisor: Shoba Tegginmath

# Table of Contents

List of Figures .....	6
List of Tables.....	7
List of Equation.....	8
Attestation of Authorship .....	9
Acknowledgment .....	10
Abstract.....	11
Chapter 1: Introduction.....	12
1.1 Introduction and background information .....	12
1.2 Introduction to NLP .....	14
1.3 Introduction to Data mining.....	14
1.4 Motivation .....	15
1.5 Research Problem and Question.....	16
1.6 Scope .....	16
1.7 Thesis structure .....	18
Chapter 2: Literature Review.....	20
2.1 Introduction.....	20
2.2 Sentiment Analysis Review.....	20
2.2.1 Areas of Application of Sentiment Analysis.....	21
2.2.2 Process of Sentiment Analysis.....	22
2.2.3 Classification of existing solutions .....	25
2.3 Sentiment Analysis Methods and Tools.....	26
2.3.1. Sentiment Classification on Online Customer Reviews .....	27
2.3.2. Concept-Level Sentiment Analysis.....	27
2.3.3. Interdependent Latent Dirichlet Allocation.....	28
2.3.4. A Joint Model of Feature Mining and Sentiment Analysis .....	29
2.3.5. Opinion Digger.....	30
2.3.6. Latent Aspect Rating Analysis .....	31
2.3.7. More approaches and tools .....	32
2.4 Sentiment analysis in the medical field .....	36
2.4.1 Sentiment analysis from the medical web .....	36
2.4.2 Sentiment analysis from biomedical literature .....	38

## Table of Contents

2.4.3 Sentiment analysis from other medical texts .....	39
2.4.4 Summary of medical opinion mining approaches .....	39
2.5 N-Grams.....	40
2.5.1 Calculating N-grams .....	40
2.5.2 Use and application of N-grams.....	41
2.5.3 Pseudo-code to generate N-grams .....	43
2.6 Gaps and key challenges.....	43
2.7 Summary.....	44
Chapter 3: Data mining and NLP .....	46
3.1 Introduction.....	46
3.2 Data Mining.....	46
3.3 Knowledge Discovery Process (KDP) .....	47
3.4 Stages of KDP.....	48
3.5 Text Mining .....	49
3.5.1 The Seven Practice Areas of Text Analytics.....	50
3.5.2 Interactions between Practice Area.....	51
3.6 NLP .....	52
3.6.1 Parsing.....	53
3.6.2 Discourse.....	53
3.6.3 Text Categorization .....	54
3.7 WordNet in NLP .....	56
3.7.1 How does WordNet work? .....	59
3.8 Word Sense Disambiguation (WSD) .....	61
3.8.1 Performing WSD.....	63
3.8.2 WSD using SentiWordNet.....	66
3.9 Summary.....	66
Chapter 4: Analyzing Negative Sentiments .....	67
4.1 Introduction .....	67
4.2 SentiWordNet.....	68
4.3 Negation Identification and Calculation in Sentiment Analysis.....	68
4.4 Summary.....	71
Chapter 5: Methodology and Framework .....	72
5.1 Introduction.....	72

## Table of Contents

5.2 Evaluation .....	72
5.2.1 The Test Set.....	72
5.2.2 Accuracy .....	73
5.2.3 Precision and Recall .....	74
5.3 Typical workflow of sentiment analysis module .....	75
5.4 Natural Language Toolkit (NLTK) .....	75
5.5 NLTK Processing Tasks.....	76
5.5.1 Tokenization and Stemming .....	76
5.5.2 Tagging .....	77
5.5.3 Chunking and Parsing .....	79
5.6 Classification Algorithms .....	80
5.6.1 Naïve Bayes Algorithm .....	81
5.6.2 Support Vector Machine (SVM) .....	81
5.6.3 Decision Tree Classification Algorithm .....	83
5.6.4 Decision Tree Algorithm – J48 .....	83
5.7 Summary.....	83
Chapter 6: Results .....	84
6.1 Introduction.....	84
6.2 Framework.....	84
6.2.1 Data Collection .....	84
6.2.2 Data Processing .....	85
6.2.3 Feature Extraction .....	85
6.2.4 POS Tagging.....	85
6.2.5 Calculate Sentiment Polarity .....	86
6.3 Project Setup Instructions .....	87
6.4 Results .....	90
6.5 Summary.....	92
Chapter 7: Review, limitations, and future research .....	92
7.1 Introduction.....	93
7.2 Research Review and Summary.....	93
7.3 Limitations.....	95
7.4 Future Work or Research.....	96
References .....	98

## Table of Contents

Appendix.....	103
---------------	-----

## List of Figures

Figure 1 Different Stages of Sentiment Analysis (Source: (Zhang & Desouza, 2014).....	17
Figure 2 Sentiment Analysis Process (Source: (Chandni et al., 2015).....	22
Figure 3 Sentiment Analysis Components (Source: Asghar et al., 2017).....	24
Figure 4 Sentiment Analysis Classification (Source: (Collomb et al., 2014).....	26
Figure 5 KDP Diagram (Source: (Fayyad et al., 1996) .....	48
Figure 6 Text Mining (Source: (G. Miner, January 2012).....	50
Figure 7 Interactions between Practice Areas Source: (G. Miner, January 2012).....	51
Figure 8 Application of NLP (Source: Chowdhury, 2003) .....	52
Figure 9 WordNet Synsets (Source: <a href="http://wordnetweb.princeton.edu/perl/webwn">http://wordnetweb.princeton.edu/perl/webwn</a> ).....	61
Figure 10 WSD Workflow (Source: Hung & Chen, 2016) .....	65
Figure 11 Negation Identification and Calculation (Source: Asmi & Ishaya, 2012).....	71
Figure 12 Precision and Recall (Source: NLTK, 2017) .....	74
Figure 13 Sentiment Analysis using SentiWordNet (Source: (Amiri & Chua, 2012) .....	87

# List of Tables

Table 1 Approaches and Techniques from Existing Solutions .....	25
---	----

## List of Equation

Equation 1 Calculating N-Gram (Source: Banerjee & Pedersen, 2003).....	41
Equation 2 F-Score or F-Measure Source: (NLTK, 2017).....	75
Equation 3 Bayes Formula Bayes Formula (Source: (Medagoda et al., 2015) .....	81
Equation 4 SVM Error Function Source: (Medagoda et al., 2015) .....	82
Equation 5 J48 Information Gain Source: (Medagoda et al., 2015) .....	83



## Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

- Kartik Ramachandran

# Acknowledgment

My Acknowledgement extends to all the people directly or indirectly for their help in completing my thesis.

Most importantly, I thank Dr. Shoba Tegginmath, my primary supervisor, for her guidance and patience throughout the entire course of my thesis. I also express my gratitude towards Dr. Parma Nand, who has provided me with valuable reviews which has helped me improve my written work.

I would also like to thank Program Administrators for the School of Computing and Mathematical Sciences at AUT, for their help throughout the course of my thesis.

I would like to express deepest gratitude to my mother, Kalyani Ramachandran and my brother, Kirti Ramachandran for their unwavering support, belief and encouragement throughout this journey.

Lastly, I would like to thank all my friends, who have been a gem and have supported and motivated me constantly during difficult times.

# Abstract

The last 15 years have seen a tremendous explosion in the amount of information available, encoded both in structured forms such as databases and XML files as well as free, naturally occurring forms such as HTML pages and word documents. This availability of free texts has created a need for automated text processing tools so that information can be extracted in a timely and effective manner.

This research investigated the extraction of information from free text responses to open-ended questions in questionnaires. The research undertook to develop a framework for analyzing open question responses to extract structured information which can then be conflated with the closed question responses in order to produce a more informative report from the survey, in particular to determine the sentiment expressed in the response.

Specifically, this research will help in understanding the positive or negative nature of the respondent's answers through the creation of software tools using Natural Language Toolkit (NLTK) and data mining and Natural Language Processing techniques and will help surveyors (Health centers, doctors, data analysts) obtain additional information from surveys. There is also a discussion of existing sentiment analysis solutions as well as the different components and ways of analyzing sentiment and creating a Natural Language Processing tool which would be interesting to future developers of such systems.

This research was successfully able to classify free text responses as positive or negative. While we appreciate that more time to fine tune the application and perform more training and testing would have been useful, the results obtained are promising. We have successfully developed a platform which can be used for generating a custom corpus and provide interested developers a starting framework to develop sentiment analysis tools.

# Chapter 1: Introduction

## 1.1 Introduction and background information

The world has seen a tremendous increase in the use of digital documents due to the increased availability of hardware tools used to digitize non-digital data and the increased availability of software tools which create digital data such as images or word documents (Sebastiani, 2005). At the same time, Natural language and data mining researchers have been striving to improve solutions for storing, organizing and most importantly retrieving huge amount of data in digital form generated every second from natural language text (Sebastiani, 2005).

Data or knowledge discovery, also known as data mining, is the process of evaluating data from various viewpoints and putting it together into novel and valuable information which can be used to decrease costs or increase revenue or both. Data mining allows us to view data from different angles and categorize, filter and summarize it and this helps users to find relationships within data. To summarize, data mining is the process of discovering patterns and relations between various fields in a relational database (Palace, 1996). Data mining consists of five major elements (Palace, 1996):

1. Extract, transform, and load transaction data onto the data warehouse system
2. Store and manage the data in a multidimensional database system
3. Provide data access to business analysts and information technology professionals
4. Analyze the data by application software
5. Present the data in a useful format, such as a graph or table

Natural language processing (NLP), an area at the intersection of artificial intelligence and linguistics, began in the 1950s and was considered to be different from information retrieval which uses statistics-based techniques to search and index huge volumes of data. Research in NLP has received increasing attention over the last 30 years and in the last decade, concrete commercial applications are being created for business, industry and services. With time however, the fields of data mining and NLP have converged (Nadkarni, Ohno-Machado, & Chapman, 2011).

The last 5 years have seen a significant shift in ways we communicate with others using short loosely structured or unstructured text (Nand & Perera, 2015). This has created the need to look for techniques which can help users conveniently access huge volume of unstructured repositories of text which can be done by:

- Creating powerful tools for finding relevant document(s) within a large repository which will accept a natural language query and give the user a list of documents according to the relevance of information user requires
- Creating tools powerful enough to convert unstructured repository of documents or data into a structured one which creates ease of storage, browsing and searching (Sebastiani, 2005)

Text classification is a sub discipline of data mining that is specifically concerned with building tools aimed at partitioning an unstructured collection of data or documents into a structured one. Text classification has two major variations. The first is *text clustering* which deals with finding undetected group structure in the repository and the second is *text categorization* which deals with categorizing or structuring the repository according to the scheme provided as input (Sebastiani, 2005).

A more recent discipline of computational linguistics, Opinion Mining is concerned with opinion, not the topic, expressed in a document. For instance, applications that determine opinions of users have helped in the review of products while others have helped in tracking general public attitude towards a political candidate.

Various sub-tasks of Opinion Mining have been identified (Esuli & Sebastiani, 2006):

1. Finding factual nature or opinions expressed in text on subject matter, which can be achieved by performing binary text categorization under subjective or objective categories.
2. Finding orientation of document *i.e.* determining whether a piece of subjective text expresses negative or positive opinion.
3. Evaluating strength of document orientation *i.e.* deciding for example whether opinion expressed is weakly positive, or mildly positive, or strongly positive.

## 1.2 Evaluation in NLP

While a good deal of time has been devoted to the study of computational models of languages and to their implementation into applications, very little attention has been given to evaluating their performance and accuracy. Two main reasons for this were that early results of NLP applications had a poor impact which did not push for the need for accurate performance evaluation and secondly, the unavailability of formal tools which can appropriately define NLP systems at different levels of abstraction such as linguistic models, external behavior and knowledge representation methods, knowledge bases and processing algorithms. This has impeded the development of methods for performance evaluation in NLP. Recently as a result of growing interest in NLP, performance evaluation has been seen as an important research problem and has begun to appeal to larger research interests as it helps to evaluate the results obtained from a system up to a certain point in development, and helps us to plan for the next stages of refinement and implementation. However it should be noted that several NLP applications that are available in the market do not consider performance evaluation (Nadkarni et al., 2011).

## 1.3 Introduction to Data mining

The aim of data mining is to understand large amounts of mostly unsupervised data, in various domains. This definition of data mining is intuitive and easy to understand. The users of data mining are often domain experts who not only own the data but also collect the data. It is generally assumed that data owners have some understanding of the data and the processes that generated the data. Businesses are the largest group of data mining users since they routinely collect massive amounts of data and have a vested interest in making sense of the data; their goal is to make their companies more competitive and profitable. Data owners desire not only to better understand their data but also to gain new knowledge about the domain that is present in their data for solving problems in novel, possibly better ways.

Data mining is not just an “umbrella” term coined for making sense of data. The major distinguishing characteristic of data mining is that it is data driven, as opposed to other

methods that are often model driven. In statistics, researchers often deal with the problem of finding the smallest data size that gives sufficiently confident estimates. In data mining it is the opposite, namely, data size is large and we are interested in building a data model that is small (not too complex) but still describes the data well.

Knowledge Discovery Process (KDP), also called knowledge discovery in databases, seeks new knowledge in some application domain. It is defined as the nontrivial process of finding valid, novel, potentially useful, and ultimately understandable patterns in data. The process generalizes to non-database sources of data although it emphasizes databases as a primary source of data. It consists of many steps (one of which is Data Mining), each attempting to complete a discovery task and each accomplished by the application of a discovery method. Knowledge discovery concerns the entire knowledge extraction process, including how data are stored and accessed, how to use efficient and scalable algorithms to analyse massive datasets, how to interpret and visualize the results, and how to model and support the interaction between human and machine. It is also concerned with support for learning and analysing the application domain. The KDP model consists of a set of processing steps to be followed by practitioners when executing a knowledge discovery project. The model describes procedures that are performed in each of its steps. It is primarily used to plan, work through, and reduce the cost of any given project.

## **1.4 Motivation**

Surveys are composed of closed and open ended questions. While closed questions restrict the frame of reference, open ended questions provide liberty to the user to express their opinion more freely. Some surveys require respondents to express their opinions using natural language text. Reading these surveys manually is time intensive and analyzing user opinion correctly becomes difficult, especially with the large amounts of text that need to be analyzed. Therefore this research has the aim of analyzing sentiment expressed in free text comments in questionnaires. Free text analysis will be done by automating the process which involves the creation of artifacts which can be

used to analyze the sentiment of free text, and determine the polarity (positive or negative) and strength of a free text comment.

There are tools in the market (commercial or open source) which help to find the sentiment expressed by a document. The available tools however lack documentation about the methodology used, algorithms applied to classify text and sentiment analysis algorithm used. This provided the main impetus for this research, which was to follow an accepted and viable research methodology to analyse the sentiment expressed, within a reusable framework, and to provide the documentation that is missing from tools available in the market. During this process, the goals were to: understand the process of sentiment analysis starting from text extraction to calculating positive and negative sentiment and most importantly create artifacts which can subsequently be used to analyze the sentiment of other types of free text. The framework created can be extended for future research in this field. The framework can also be used as a starting point to analyze document sentiment in various other languages.

## **1.5 Research Problem and Question**

The research problem addressed in this thesis can be stated as information extraction from free text comments in questionnaires using NLP and data mining techniques.

This thesis set out to answer the following research question:

How can an effective sentiment analysis tool be built to analyse free text comments in questionnaires?

A supplementary research question investigated is:

Will a corpus created from the data be useful in analyzing sentiment expressed in the data?

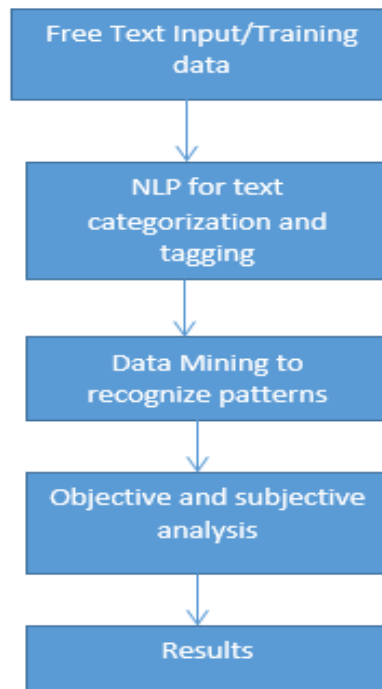
## **1.6 Scope**

The scope of the thesis is to study the different components of data mining and NLP and use this knowledge to create a tool which can extract free text responses from a



questionnaire; analyze the sentiments expressed in the responses and classify responses as positive or negative.

The different stages of sentiment analysis that we follow in this research are depicted in Figure 1:



*Figure 1 Different Stages of Sentiment Analysis (Source: (Zhang & Desouza, 2014))*

In the process of creating the software tool we also investigate the utility of a corpus in sentiment analysis—the corpus will be created from the data used in this research. The resulting corpus can be used as reference for future analysis of text. We will use different libraries and algorithms to classify text which is a well-rounded approach that is followed by software practitioners while creating language classification software. While the code will be tailored to the current study, *i.e.* to patient comments about doctors, the code can be extended in future to extract results related to demographics of patients such as race, sex, age.

In Section 1.2, we discussed the lack of attention towards the evaluation of performance and accuracy in NLP applications. Therefore, in this research we will be evaluating

results using accepted Information Science measures and this is discussed in Chapter 5.

## **1.7 Thesis structure**

The rest of the thesis is structured as described in the following paragraphs.

Chapter 2 reviews literature on sentiment analysis and provides information about the process and different components used to create an effective sentiment analysis tool and the areas of application of sentiment analysis, application in medical field. Then we take a detailed look at existing sentiment analysis. Chapter 2 also discusses research in a related area—that of n-grams—which is an important technique for solving the problem of language recognition, used in information retrieval. We look at how n-grams are calculated and then look at the use and areas of application and finish the chapter with pseudo code to generate n-grams and gaps and key challenges.

Chapter 3 begins with details about data mining and KDP, followed by text mining and the seven areas of practice in text mining, which is followed by a discussion on natural language processing, its components and various tools. Before diving deep into sentiment analysis, it is essential to discuss Word Sense Disambiguation (WSD) and performing WSD on a document. The chapter concludes with a discussion about Wordnet—a lexical resource and SentiWordnet, another lexical resource, which is used in this thesis.

We look at analyzing negative sentiment in Chapter 4, which is one of the most complex tasks in sentiment analysis and how it can be performed. We then look at pseudo code which can help us perform negation of sentiment polarity.

In Chapter 5, we discuss about the evaluation techniques for classification models. After looking at the sentiment analysis workflow, we look at the Natural Language Toolkit and processing tasks, which are widely used for sentiment analysis for educational and research purposes. We then study supervised classification algorithm which we will use in this thesis.

We present the software tool framework and set up instructions using the code in the appendix, in Chapter 6. We also present results obtained by running the tool on test data.

Finally, we review the thesis in Chapter 7, followed by limitations of the tool and areas for future research and work.

This thesis also has an extensive appendix, which provides further background information on the code written to create the custom corpus and perform various NLP tasks.

# Chapter 2: Literature Review

## 2.1 Introduction

This chapter discusses current literature and work in sentiment analysis along with the analysis process and classification methods used. Section 2.2 will discuss sentiment analysis followed by a section which discusses the areas of application of sentiment analysis. Classification of existing sentiment analysis method is discussed in the fourth section. The next section shows us the process of sentiment analysis and then we look at different sentiment analysis methods, after which we look at sentiment analysis in the medical field. WSD and N-grams are some of the most widely used techniques for information retrieval and it was essential to review the literature and understand n-grams before the framework for this thesis could be finalized. Finally, we look at the current gaps and challenges in the application of natural language processing to analyzing and classifying text and documents. Finally, a summary concludes the chapter.

## 2.2 Sentiment Analysis Review

Sentiment analysis is the computational study of human opinions, emotions, attitudes and thoughts towards an event or topic or individual. Opinion or sentiment mining, sentiment extraction and subjectivity analysis are other terms used for sentiment analysis (Chandni, Chandra, Pahade, & Gupta, 2015). Sentiment analysis uses natural language processing, computation techniques and text analysis to automate the entire process of extracting and classifying sentiment reviews. Sentiment analysis has spread across many fields such as marketing, consumer information, books, websites, application and social media. The main aim of analyzing sentiments in a variety of areas is to analyze and examine the reviews and score of sentiments (Hussein, 2016).

Opinions usually comprise of polarity which can positive or negative, the target or the aspects about which the sentiment was expressed, and the time at which the opinion was expressed.

Typically, we can perform sentiment analysis by using lexicon-based method or machine learning or a combination of the two methods also known as hybrid methods. Machine learning uses algorithm that needs to be trained with labelled data and then the model is used for classifying new documents. The labelled data is created by human annotator and is a labor-intensive process. Distant supervision is an alternative method which relies on usage of certain emoticons which signify sentiments. Although distant supervision has been proved to do well in classification, it is very difficult to integrate it with machine learning algorithm. Lexicon based methods make use of sentiment lexicons which associate terms with sentiment polarity (negative, positive or neutral) usually by using a numerical score that is an indicator of sentiment strength and dimension. But sentiment lexicons do not contain sentiment bearing and domain specific terms and this makes it difficult to classify high sentiment bearing words properly (Muhammad, Wiratunga, Lothian, & Glassey, 2013).

The goal of sentiment analysis is to analyze and examine the sentiments shown in the sentence and determine the polarity of the sentence. Sentiment analysis can be examined as a process of three systemization levels which are document level, aspect level and sentence level. Document level sentiment analysis sets to organize thoughts in a document as carrying a positive or negative or neutral sentiment. An aspect is a part of the product/movie that has been commented on in a review. For example, 'battery life' in the opinion phrase 'The battery life of this camera is too short'. Sentence level opinion mining first tries to recognize the sentence as subjective or objective and then if a sentence is subjective it tries to examine whether it displays positive or negative sentiment. There is no real distinction between document- and sentence-level analysis because sentences are considered small records from a document (Chandni et al., 2015).

### **2.2.1 Areas of Application of Sentiment Analysis**

Consumers make a choice or decision regarding a product from the information about the reputation of the product derived from the opinions of other users. When users choose a product, they are interested or attracted to certain aspects of the product and may comment on specific aspects in their review. A review is an assessment of the

quality of a product, for example, that is posted online. An aspect is a part of the product that has been commented on in a review. For example, 'battery life' in the opinion phrase 'The battery life of this camera is too short'. A rating is an intended interpretation of user satisfaction in terms of numerical values. Most review websites use ratings (number of stars) in the range from 1 to 5. Sentiment analysis can help collect the opinions of the reviewers and help estimate ratings on a specific aspect of the product, as a single global rating can be deceiving. Thus, sentiment analysis or mining can be used to give an indication or recommendation in choosing products. Another usage of sentiment analysis is for the organizations or companies to know the opinion consumers have of their products, which then can be used to improve on the aspects consumers did not like or found unsatisfactory. Sentiment analysis can also help companies understand which aspects consumers liked, and automatically suggest advertisement for other products that suit a viewer's opinion and this provides numerous opportunities in the human-machine interface domain.

### 2.2.2 Process of Sentiment Analysis

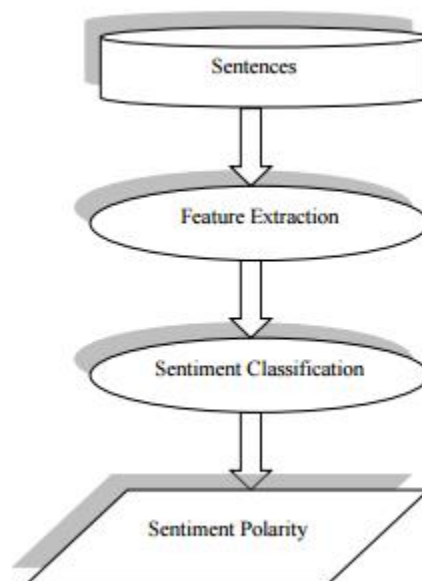


Figure 2 Sentiment Analysis Process (Source: (Chandni et al., 2015))

The process of sentiment analysis, as shown in Figure 2, may be simplistically viewed as a series of actions that begin with a sentence to be analysed and ends with the

determination of the polarity of the sentence. The feature extraction phase deals with features which may be of the following types in sentiment classification:

1. Part of speech (POS): It includes adjectives which are important for opinion or thought.
2. Presence of Terms and their Frequencies: These features are type of word i.e. individual word or N-gram words and their relative count of frequencies. Frequency count is used to show the relative value of features.
3. Words and Phrases for opinion: words and phrases that are commonly used to the opinions like love or hate, high or low.
4. Negation: as a negative word before any word may change the meaning of that word or opinion e.g. not love is similar to hate. (Chandni et al., 2015).

(Asghar, Khan, Ahmad, Qasim, & Khan, 2017) detail many of the functions performed during lexicon-based sentiment analysis, presented here in Figure 3. Their method is based on the three steps: 1. Acquisition of data from different online resources; 2. Noise reduction, performed by applying different preprocessing techniques to refine the text that can be used for later processing, and 3. Classification techniques applied to classify the reviews into positive, negative or neutral.

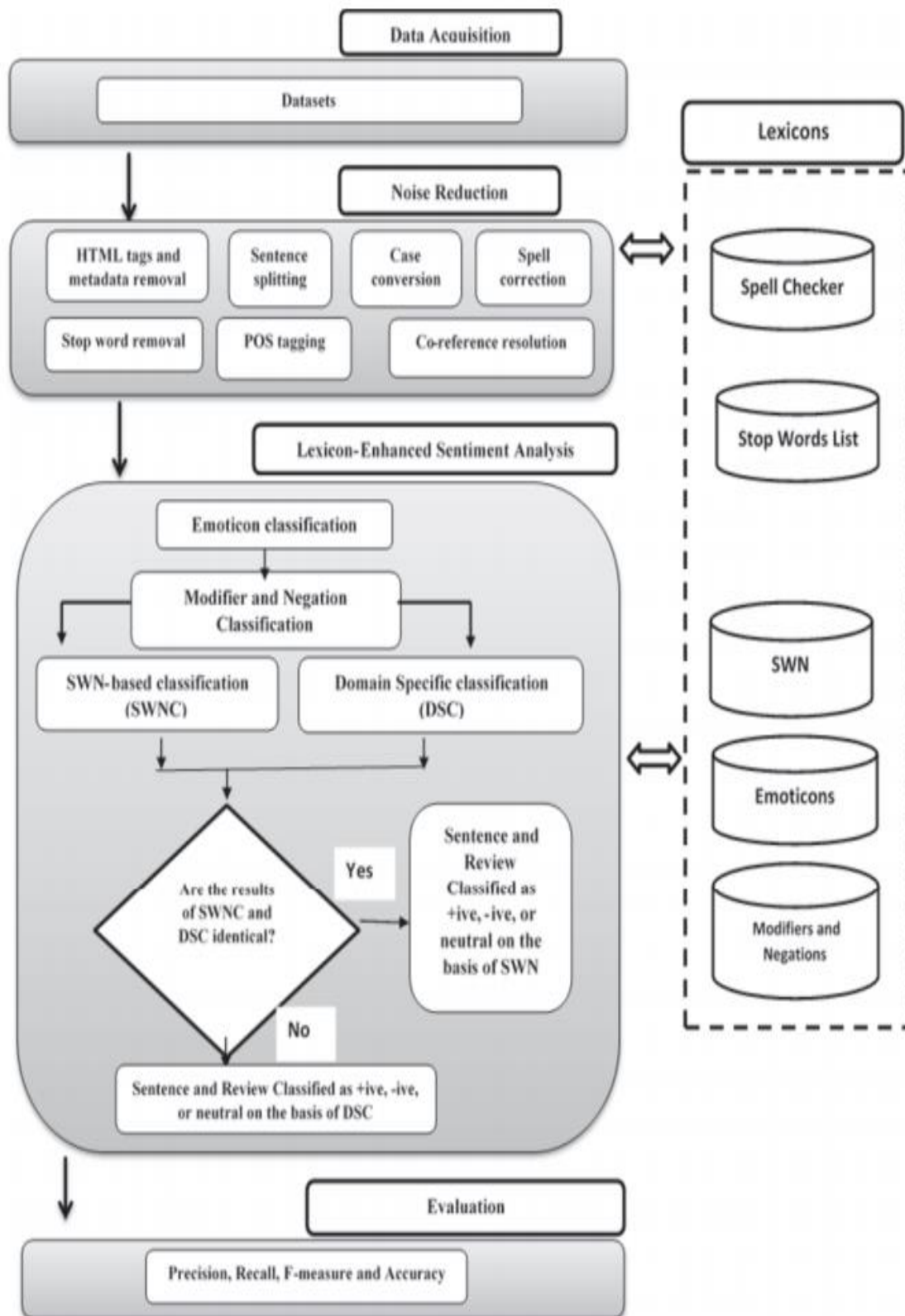


Figure 3 Sentiment Analysis Components (Source: Asghar et al., 2017)



In the following section, we look at the different ways existing solutions have been classified.

### 2.2.3 Classification of existing solutions

Existing solutions on sentiment analysis can be classified on different points such as technique used, view of text, level of detail of text analysis, rating level, etc. From a technical standpoint, we can classify based on the approaches.

**Table 1** Approaches and Techniques from Existing Solutions

Method/Approach	Measure/Technique Used
Machine Learning	Learning algorithms
Lexicon-based	Semantic orientation
Rule-based	Classification
Statistical	Multinomial distribution, Clustering

As shown in Table 1, the machine learning method uses several learning algorithms to determine the sentiment by training on a known dataset. The lexicon-based approach involves calculating sentiment polarity for a review using the semantic orientation of words or sentences in the review; “semantic orientation” is a measure of subjectivity and opinion in text. The rule-based approach looks for opinion words in a text and then classifies it based on the number of positive and negative words. It considers different rules for classification such as dictionary polarity, negation words, booster words, idioms, emoticons, mixed opinions, to mention a few. Statistical models represent each review as a mixture of latent aspects and ratings. It is assumed that aspects and their ratings can be represented by multinomial distributions and try to cluster head terms into aspects and sentiments into ratings.

Most of the solutions for review classification rely on polarity of the review and machine learning techniques. Solutions which aim for more detailed classification of user reviews use a great deal of linguistic features including negation, modality, intensification and

discourse structure. Figure 4 gives further detail on the classification of existing methods.

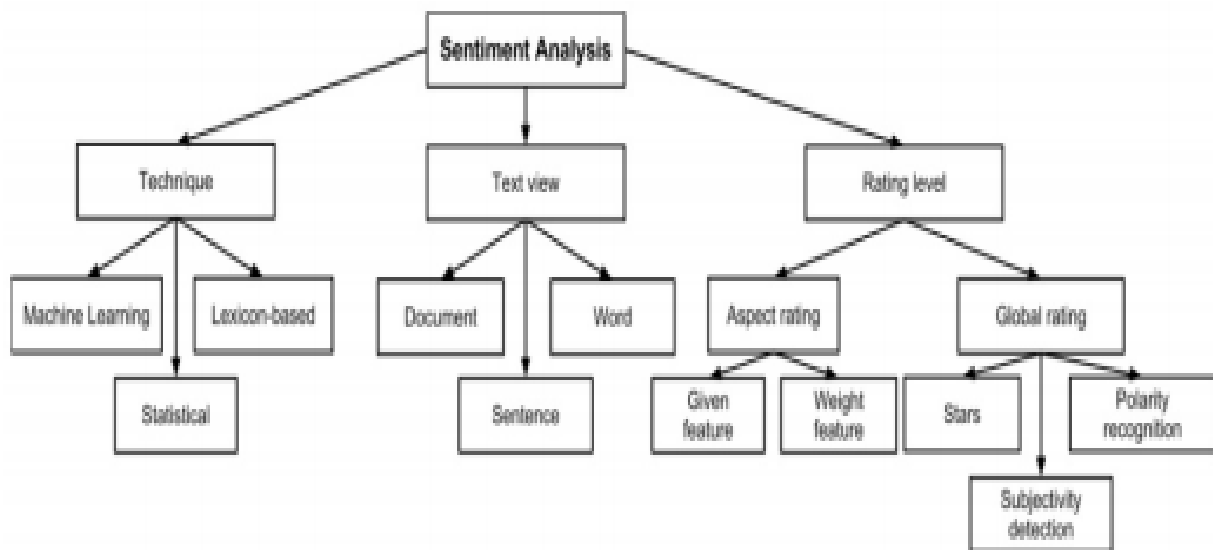


Figure 4 Sentiment Analysis Classification (Source: (Collomb et al., 2014))

Another classification of sentiment analysis is oriented more on the structure of the text: document level, sentence level or word/feature level classification. Document-level classification aims to find a sentiment polarity for the whole review, whereas sentence level or word-level classification can express a sentiment polarity for each sentence of a review and even for each word. (Collomb et al., 2014) state that most of the methods tend to focus on document-level classification. They also distinguish methods which measure sentiment strength for different aspects of a product and methods which attempt to rate a review on a global level.

In the next section, we look at some of the sentiment analysis methods from literature.

## 2.3 Sentiment Analysis Methods and Tools

In this section, we discuss various sentiment analysis methods and tools created by researchers in performing sentiment analysis.

### **2.3.1. Sentiment Classification of Online Customer Reviews**

(Khan, Baharudin, & Khan, 2011) presented a domain-independent rule-based method for classifying sentiments from customer reviews that works in three parts. First, the reviews are split into sentences, corrected and POS tagged, and the base word of each word in the sentence is stored. Next, opinion word extraction is used to find out the polarity of the sentence based on the contextual information and structure of the sentence. The noun phrases are the aspects of the product. The third part consists of classifying the sentence as subjective or objective—using rule based methods. Each word conveys opinion and has a semantic score which is calculated from SentiWordNet dictionary, and a weight is assigned to a sentence, by rating each term, to decide if it conveys positive or negative sentiment.

For evaluation, (Khan et al., 2011) collected three types of customer reviews (movie, hotel and airline reviews) which had an average of 1,000 movie and airlines reviews and 2,600 hotel reviews. The performance was assessed with an accuracy of 91% at the review level and 86% at the sentence level; moreover, the sentence level sentiment classification performed better than the word level. The accuracy of 70-75% seems better than the average results of other methods but there is no comparison provided with other lexicon-based methods, nor with learning-based methods.

### **2.3.2. Concept-Level Sentiment Analysis**

pSenti is a concept level sentiment analysis system which combines lexicon based and learning-based approaches. It measures and reports the overall sentiment of a review through a score that can be positive, negative or neutral or 1–5 stars classification. The main advantages and main interests of this article are the lexicon/learning symbiosis, the detection and measurement of sentiments at the concept level and the lesser sensitivity to changes in topic domain.

It works in four parts:

1. Pre-processing of the review where the noise (idioms and emoticons) is removed and each word is tagged and stored by the method Part of Speech (POS).

2. Aspects and views are extracted to generate a list of top 100 aspect groups and top 100 views. The aspects are identified as nouns and noun phrases, and the views as sentiment words, adjectives and known sentiment words which occur near an aspect.
3. Then the lexicon-based approach is used to give a “sentiment value” to any sentiment word and generates features for the supervised machine learning algorithm.
4. Algorithm generates a “feature vector” for each aspect which is either the sum of the sentiment value for a sentiment word or the number of occurrences of this word in relation with other adjectives.

To evaluate this method, experiments were conducted on software reviews (more than 10,000) and movie reviews (7,000) datasets. Software reviews were separated into software editor reviews and customer software reviews categories. In their experiments, pSenti’s accuracy was proved close to the pure learning-based system and higher than the pure lexicon-based method. It was also shown that the performance was not as good on customer software reviews as on software editor reviews because customer software reviews are usually much “noisier” (with comments that are irrelevant to the subject) than professional software editor reviews. Its accuracy was also affected by many reviews for which it did not detect any sentiment or assigned neutral score. However, the sentiment separability in movie reviews was much lower than in software reviews. One of the reasons is that many movie reviews have plot descriptions and quotes from the movie where words are identified as sentiments by the system (Collomb et al., 2014; Mudinas, Zhang, & Levene, 2012).

### **2.3.3. Interdependent Latent Dirichlet Allocation**

(Moghaddam & Ester, 2011), introduced Interdependent Latent Dirichlet Allocation (ILDA) in 2011. They introduced the probabilistic assumption that there is interdependency between an aspect and its corresponding rating. ILDA is a probabilistic graphical model which shows each review as a mixture of latent aspects and ratings. It assumes that aspects and their ratings can be represented by multinomial distributions and tries to cluster head terms into aspects and sentiments into ratings. ILDA relies on a

concept introduced in 2003 by (Blei, Ng, & Jordan, 2003). Latent Dirichlet Allocation (LDA). It is a generative probabilistic model for collections of discrete data such as text corpora. The basic idea is that each item of a collection is modeled as a finite mixture over an underlying set of latent variables.

Their experiments showed notable improved results for ILDA compared to the other two graphical models described in the paper (PLSI and LDA), gaining in average almost 20% for the accuracy of rating prediction. They obtain in average 83% accuracy in aspect identification and 73% accuracy in aspect rating.

#### **2.3.4. A Joint Model of Feature Mining and Sentiment Analysis**

This solution was introduced in 2011 by (de Albornoz, Plaza, Gervás, & Díaz, 2011). The authors propose a method that globally rates a product review into three categories by measuring the polarity and strength of the expressed opinion. This solution was chosen as a representative of the global rating solutions, as it goes further than other solutions; It tries to find the strength of the opinion as well as the relevance of the feature the opinion is about.

The mechanism of this method is straightforward:

1. Important features are found
2. Sentences having opinions on those features are found in the body of the review and polarity and strength are computed
3. Next, a global score is computed. The method does not rely on any earlier knowledge about the importance of the features to the customer, contrary to Hu and Liu, but learn it from a set of reviews using an unsupervised model. Another contribution is that each feature is automatically weighted. Feature importance and opinion extraction, as well as opinion rating rely on the WordNet lexical database for English. This can be an important disadvantage of the method, as it cannot be applied on reviews written in other languages.
4. The fourth step – rating reviews are predicted and these reviews are structured using Vector Feature Intensity (VFI) graph. It is constructed using the strength of the opinion and the relevance of the feature. This graph is fed as input to any machine learning algorithm that will classify the review into different rating categories.

This solution offers flexibility when it comes to choosing the best machine learning method for classifying reviews.

### **2.3.5. Opinion Digger**

The solution Opinion Digger was introduced in 2010 by (Moghaddam & Ester, 2010) and is a good and exact example of a completely unsupervised machine learning method. The particularity of this solution is to use as input a set of known aspects of a product and a ratings guideline (5 means “excellent” 4 means “good”). With these elements, Opinion Digger finds and outputs a set of other aspects and ratings in each aspect according to the guideline. The impetus for this research was based on the fact that many reviewing websites like amazon.com provide these input elements but there was no method that used them.

Opinion digger works in two steps:

1. In the first phase, Opinion Digger decides the set of aspects. After the pre-processing, each sentence is tagged with POS. It assumes that aspects are nouns so it first isolates the frequent nouns as potential aspects. With the sentences matching the known aspects, they determine opinion patterns as sequence of POS-tags that expressed opinion on an aspect. The frequent patterns used with known aspects are considered opinion patterns. If reviews with a “potential aspect” noun match at least two different opinion aspects, Opinion digger considers the noun as an aspect.
2. The second phase is rating the aspects. For each sentence having an aspect, Opinion Digger associates the closest adjective to the opinion. It searches two synonyms from the guideline in the WordNet synonymy graph. The estimated rating of the aspect is the weighted average of the corresponding rating in the guideline. Weight is calculated by the inverse of the smallest path distance between the opinion adjective and the guideline’s adjective in the WordNet hierarchy.

The experiments show good performance in aspect determination and an excellent accuracy in ratings. The evaluation of aspect ratings was made using only the known set of aspects and compared to 3 other unsupervised methods. Opinion Digger performs with an average ranking loss of only 0.49 which is the difference between

estimated and actual ratings. By incorporating new current information in the machine learning process, Opinion Digger increases the accuracy of the unsupervised machine-learning method.

### **2.3.6. Latent Aspect Rating Analysis**

This solution treats a special problem called Latent Aspect Rating Analysis with a model-based method. The model is called the Latent Rating Regression (LRR) model and was created by (Wang, Lu, & Zhai, 2010). It estimates ratings on different aspects in a review but also decides the emphasis of the author on each aspect. It uses a given set of aspects and the overall ratings of the review. It starts with an aspect-segmentation step. By recursively associating words with aspects, it builds an aspect dictionary and links each phrase of a review to the corresponding aspect, then it applies the model. The assumption of reviewer's rating behavior is as follows: to generate an opinionated review the reviewer first decides the aspects for reputation evaluation that she/he wants to comment on; and then for each aspect, the reviewer carefully chooses the words to express her/his opinions. The reviewer then forms a rating on each aspect based on the sentiments of words she/he used to discuss that aspect. Finally, the reviewer assigns an overall rating depending on a weighted sum of all the aspect ratings, where the weights reflect the relative emphasis she/he placed on each aspect. So, the overall rating is not directly decided by the words used in the review but rather by latent ratings on different aspects which are decided by the words.

With a probabilistic regression approach, Latent Aspect Rating Analysis converts the model into a Bayesian regression problem, and then decides the aspect ratings and weight with consideration to the author's intent. The overall rating  $r$  is assumed to be a sample drawn from a Gaussian distribution with variance  $\delta^2$  and mean the weighted sum of the aspect ratings  $S$ .  $S$  is the result of the weighted sum of the words  $W$  in the reviews. A multivariate Gaussian distribution is employed as the prior for aspect weight's  $\alpha$ .

The experimentation shows an average performance compared to other unsupervised methods in aspect ratings. However, it achieves what it set out to achieve—to estimate an aspect's weight (Collomb et al., 2014; Wang et al., 2010).

### 2.3.7. More approaches and tools

In this section, we go on to introduce some more research and tools in the area of sentiment analysis.

(Asghar et al., 2017) looked at enhancing the performance of sentiment analysis and resolving the issues of data sparseness and incorrect classification caused by the presence of noisy text, emoticons, modifiers and domain specific words (See Figure 3). The basic theme was to reduce noise from the review text by applying different pre-processing steps and processes through a variety of classifiers. The proposed method was used to test the text from different online forums; the reviews compiled from these sources were used as input items.

One simple way proposed to detect the polarity of a message is based on the emoticons it contains. Emoticons have become popular in recent years, to the extent that some (e.g. <3) are now included in English Oxford Dictionary. Emoticons are primarily face-based and represent happy or sad feelings, although a wide range of non-facial variations exist: for instance, <3 represents a heart and expresses love or affection. To extract polarity from emoticons, a set of common emoticons, which also includes popular variations that expresses positive, negative and neutral sentiments, are utilized (Gonçalves, Araújo, Benevenuto, & Cha, 2013).

Linguistic Inquiry and Word Count (LIWC) is a text analysis tool that evaluates emotional, cognitive, and structural components of a given text based on the use of a dictionary containing words and their classified categories. In addition to detecting positive and negative effects in each text, LIWC provides other sets of sentiment categories. For example, the word “agree” belongs to the following word categories: assent, affective, positive emotion, positive feeling, and cognitive process (Gonçalves et al., 2013).

Machine-learning-based methods are suitable for applications that need content-driven or adaptive polarity identification models. Several key classifiers for identifying polarity in online social network data have been proposed in the literature. A very comprehensive work developed SentiStrength which compared a wide range of



supervised and unsupervised classification methods, including simple logistic regression, SVM, J48 classification tree, JRip rule-based classifier, SVM regression, AdaBoost, Decision Table, Multilayer Perception, and Naive Bayes. It was shown that, SentiStrength implements the state-of-the-art machine learning method in the context of online social networks. SentiStrength version 2.0, is available at <http://sentistrength.wlv.ac.uk/Download> (Gonçalves et al., 2013).

SentiWordNet is a tool that is widely used in opinion mining, and is based on an English lexical dictionary called WordNet. Wordnet groups adjectives, nouns, verbs and other grammatical classes of a word into synonym sets called synsets. SentiWordNet associates three scores—positive, negative, and objective (neutral)—with synsets from the WordNet dictionary to indicate the sentiment of the text. The scores, which are in the values of [0, 1] and add up to 1, are obtained using a semi-supervised machine learning method SentiWordNet was evaluated with a labeled lexicon dictionary. To assign polarity based on this method, the average scores of all associated synsets of a given text are considered, and the text is considered to be positive if the average score of the positive affect is greater than that of the negative affect. Scores from objective sentiment were not used in determining polarity. SentiWordNet version 3.0, which is available at <http://sentiwordnet.isti.cnr.it/>. WordNet is discussed in Chapter 3 and SentiWordNet in Chapter 4 (Gonçalves et al., 2013).

SenticNet is a method of opinion mining and sentiment analysis that explores artificial intelligence and semantic Web techniques. SenticNet infers polarity of common sense concepts from natural language text at a semantic level (Gonçalves et al., 2013). The method uses NLP techniques to create a polarity for nearly 14,000 concepts. For example, to interpret a message “Boring, it’s Monday morning”, SenticNet first tries to identify concepts, which are “boring” and “Monday morning”. Then it assigns a polarity score to each concept, in this case, -0.383 for “boring”, and +0.228 for “Monday morning” (Gonçalves et al., 2013). The resulting sentiment score of SenticNet is an average of the polarity scores which is -0.077. The National Health Service in England used SenticNet to test and evaluate the polarity in opinions of patients about the health

service. We use SenticNet version 2.0, which is available at <http://sentic.net/> (Gonçalves et al., 2013).

SASA is a method based on machine learning techniques such as SentiStrength and was evaluated with 17,000 labeled tweets on the 2012 U.S. Elections. The open source tool was evaluated by the Amazon Mechanical Turk (AMT), where “turkers” were invited to label tweets as positive, negative, neutral, or undefined. The SASA python package version 0.1.3 is available at <https://pypi.python.org/pypi/sasa/0.1.3> (Gonçalves et al., 2013).

Happiness Index is a sentiment scale that uses the popular Affective Norms for English Words (ANEW). ANEW is a collection of 1,034 words commonly used associated with their affective dimensions of valence, arousal, and dominance. Happiness Index was constructed based on the ANEW terms and has scores for a given text between 1 and 9, indicating the amount of happiness existing in the text. The authors calculated the frequency that each word from the ANEW appears in the text and then computed a weighted average of the valence of the ANEW study words. The validation of the Happiness Index score is based on examples. ANEW was applied to a dataset of song lyrics, song titles, and blog sentences. It was found that the happiness score for song lyrics had declined from 1961 to 2007, while the same score for blog posts in the same period had increased (Gonçalves et al., 2013). To adapt Happiness Index for detecting polarity, any text that is classified with this method in the range of [1..5] is considered to be negative and in the range of [5..9] to be positive.

PANAS-t is a psychometric scale proposed for detecting mood fluctuations of users on Twitter. The method consists of an adapted version of the Positive Affect Negative Affect Scale (PANAS), which is a method in psychology. PANAS-t is based on a large set of words associated with eleven moods: joviality, assurance, serenity, surprise, fear, sadness, guilt, hostility, shyness, fatigue, and attentiveness (Gonçalves et al., 2013). This method is used to track any increase or decrease in sentiments over time and to associate text to a sentiment, PANAS-t first utilizes a baseline or the normative values of each sentiment based on the entire data. Then the method computes the P(s) score for each sentiment s for a given time as values between [-1.0, 1.0] to indicate the

change. For example, if a given set of tweets contain  $P(\text{"surprise"})$  as 0.250, then sentiments related to "surprise" increased by 25% compared to a typical day. Similarly,  $P(s) = -0.015$  means that the sentiment  $s$  decreased by 1.5% compared to a typical day (Gonçalves et al., 2013).

There are various other solutions in the market which offer a variety of opinion mining tools; most of them are custom made to analyze the sentiments from customer reviews about products and services by interpreting natural language. An example of a freely available application that simply analyzes terms can be found at <http://twitrratr.com/> (Cieliebak, Dürr, & Uzdilli, 2013).

Wordclouds are also becoming more and more used in making sense of large quantities of information in a snapshot and is a popular solution for word visualization. Such tools are also extremely simplified and only offer a visualization of the most commonly used terms, which gives an idea of what the document is about. Tools such as those available at [www.wordle.com](http://www.wordle.com) offer an appealing design solution that can serve as an entry level in the opinion mining market (Cieliebak et al., 2013).

Another way of classifying or making sense of large amount of information is to rely on human effort using collective intelligence and or crowdsourcing, where people will not only filter but also signal the most important ones. The website [www.uservice.com](http://www.uservice.com) provides such a tool which allows users to send feedback and rate other users' ideas, and this helps in creating new ideas (Cieliebak et al., 2013).

There is a flourishing market of enterprise-level software for opinion mining with much more advanced features. These tools are largely in use by companies to monitor their reputation and the feedback about products on social media. In the government context, opinion mining has long been in used as an intelligence tool to detect hostile or negative communications. These tools rely on machine learning for finding and classifying relevant comments, using a combination of latent semantic analysis, support vector machines, "bag of words" and semantic orientation. These processes need significant human effort aided by machines; tools in the market rely on a combination of machine and human analysis, typically using machines to augment human capacity to classify,

code and label comments. Automated analysis is based on a combination of semantic and statistical analysis. Recently, because of the sheer increase in the quantity of datasets available, statistical analysis is becoming more important (Cieliebak et al., 2013).

Table 2 lists some of the commercially available sentiment analysis tools which can analyze arbitrary texts, with free API access and are available free of charge (Cieliebak et al., 2013).

**Table 2** Commercial Tools. (Source: Cieliebak et al., 2013)

Tool	Short Name	URL
AlchemyAPI	alc	<a href="http://www.alchemyapi.com">www.alchemyapi.com</a>
Lymbix	lym	<a href="http://www.lymbix.com">www.lymbix.com</a>
ML Analyzer	mla	<a href="http://www.mashape.com/mlanalyzer/ml-analyzer">www.mashape.com/mlanalyzer/ml-analyzer</a>
Repustate	rep	<a href="http://www.repustate.com">www.repustate.com</a>
Semantria	sma	<a href="http://www.semantria.com">www.semantria.com</a>
Sentigem	sen	<a href="http://www.sentigem.com">www.sentigem.com</a>
Skyttle	sky	<a href="http://www.skyttle.com">www.skyttle.com</a>
Textalytics	tex	<a href="http://core.textalytics.com">core.textalytics.com</a>
Text-processing	txp	<a href="http://www.text-processing.com">www.text-processing.com</a>

## 2.4 Sentiment analysis in the medical field

As the goal of this thesis is to analyze sentiments expressed by patients of a medical centre, a review of existing research on sentiment analysis in the medical field was undertaken. Such literature can be grouped based on textual source (e.g. medical web content, biomedical literature and clinical notes), task (e.g. polarity analysis, outcome classification), method (e.g. rule-based, machine-learning based) and level (e.g. word level, sentence level).

### 2.4.1 Sentiment analysis from the medical web

Most research on sentiment analysis in the domain of medicine considers web data such as medical blogs or forums for mining or studying patient opinions or measuring quality. For example, a method was introduced that separates factual texts from

experiential texts to measure content quality and credibility in patient-generated content. As factual content is better than affective content since more information is given (in contrast to moods and feelings), a system has been developed using subjectivity words and a medical ontology to evaluate the factual content of medical social media.

As in general sentiment analysis, existing approaches to sentiment analysis from medical web data are either machine-learning based or rule-based. Most of the work focuses on polarity classification.

(Xia, Gentile, Munro, & Iria, 2009) introduced a multi-step approach to patient opinion classification. Their approach decides the topic and the polarity expressed towards it. An F-measure of around 0.67 was reported.

(Sokolova, Matwin, Jafer, & Schramm, 2013) tested several classifiers including naive Bayes, decision trees and support vector machines for the sentiment classification of tweets. Texts were represented as bags of words. Two classification tasks were considered: three-class (positive, negative and neutral) and two-class (positive, negative). The best F-measure of 0.69 was achieved with an SVM classifier.

The work by (Biyani et al., 2013) focused on determining the polarity of sentiments expressed by users in online health communities. More specifically, they performed sentiment classification of user posts in an online cancer support community (cancer survivors network) by exploiting domain-dependent and domain-independent sentiment features as the two complementary views of a post and exploiting them for post-classification in a semi-supervised setting employing a co-training algorithm. This work was later extended with features derived from a dynamic sentiment lexicon, while the previous work used a general sentiment lexicon to extract features.

(Smith & Lee, 2012) studied another aspect of sentiment in patient feedback, namely discourse functions such as expressiveness and persuasiveness. A classifier was evaluated based on a patient feedback corpus from NHS Choices. The results illustrate that the multinomial naive Bayes classifier with frequency-based features can achieve the best accuracy (83.53%). Further, the results showed that a classification model trained solely on an expressive corpus can be directly applied to the persuasive corpus

and achieve a performance comparable to the training based on the corpus with the same discourse function.

(Sharif, Zaffar, Abbasi, & Zimbra, 2014) presented an interesting application of sentiment analysis with their extracts of semantic, sentiment and affect cues for detecting adverse drug events reported by patients in medical blogs. This approach can reflect the experiences of people when they discuss adverse drug reactions as well as the severity and emotional impact of their experiences.

(Na et al., 2012) presented a rule-based linguistic approach for the sentiment classification of drug reviews. They used existing resources for sentiment analysis, namely SentiWordNet and the Subjectivity Lexicon, and came up with linguistic rules for classification. Their approach achieved an F-measure of 0.79. Additional work has tackled the detection and analysis of emotion in medical web documents.

(Sokolova & Bobicev, 2013) considered the categories encouragement (e.g. hope, happiness), gratitude (e.g. thankfulness), confusion (e.g. worry, concern, doubt), facts, and facts + encouragement. They used the affective lexicon WordNetAffect for emotion analysis of forum entries. However, the f-score achieved, with a naive Bayes classifier, was 0.518.

Also, it is interesting to note the work of (Melzi et al., 2014) who applied an SVM classifier on a feature set comprising unigrams, bigrams and specific attributes to classify sentences into one of six emotion categories.

#### **2.4.2 Sentiment analysis from biomedical literature**

In addition to medical social media data, biomedical literature has been analyzed with respect to the outcome of a medical treatment. In this context, sentiment refers to the outcome of a treatment or intervention. Four classes were considered in existing work: positive, negative, neutral outcome and no outcome. (Niu, Zhu, Li, & Hirst, 2005) used a supervised method to classify the (outcome) polarity at sentence level. Unigrams, bigrams, change phrases, negations and categories were employed as features. As per the results, the algorithm's accuracy was improved by the usage of category information

and context information derived from a medical terminology ontology—the unified medical language system.

(Sarker, Mollá-Aliod, & Paris, 2011) developed a new feature called the relative average negation count (RANC) to calculate polarity with respect to the number and position of the negations. This count suggests that a larger total number of negations reflects a negative outcome. The experimental corpus was collected from medical research papers, which are related to the practice of evidence-based medicine. An NGram feature set with RANC exploited by an SVM classifier achieved an accuracy of 74.9%.

#### **2.4.3 Sentiment analysis from other medical texts**

Researchers have focused medical texts to apply sentiment analysis methods to suicide notes which was a shared task in an i2b2 challenge. (Cambria, Benson, Eckl, & Hussain, 2012) introduced Sentic PROMs, where emotion analysis methods were integrated into a framework to measure healthcare quality. In a questionnaire, patients answered questions regarding their health status. From the free text entered, emotion terms such as “happy” and “sad” were detected using the semantic resources WordNet-Affect and ConceptNet. The extractions were assigned to one of 24 affective clusters following the concept of hourglass of emotions. Performance was promising with an F-score of 0.61 being achieved with an SVM classifier. This concept presents the affective common-sense knowledge in terms of a vector, which shows the location in the affective space.

#### **2.4.4 Summary of medical opinion mining approaches**

In summary, existing methods for sentiment analysis in the medical domain so far have focused on processing web content or biomedical literature. The clinical narratives which are used to record the activities and observations of physicians as well as patient records have not yet been analyzed in this context. In terms of methods, rule-based approaches are presented, but most papers report on machine-learning methods (SVM, naive Bayes, and regression tree) using features such as parts of speech and uni-band trigrams. Although general sentiment lexicons are exploited, experiments showed that they are not well suited for capturing the meanings in medical texts. In contrast to

“normal” sentiment analysis, additional domain-specific features have been explored in some approaches, mainly UMLS concepts reflecting medical conditions and treatments. The main tasks considered have been polarity classification, but new tasks are emerging including outcome classification, information content classification or emotion analysis. However, the existing work on medical sentiment analysis does not cover all facets of sentiment analysis described in Section 2. In summary, there is still a huge potential for future research.

## **2.5 N-Grams**

N-grams is one of the most used techniques for solving the problem of language recognition that is used in information retrieval (Jacob and Gokhale, 2007). N-gram based techniques are used in NLP and its applications where they are used as features to create vector space and then classification algorithms are applied to this model. The values of these features are n-grams frequencies. Traditional N-grams can be a sequence of words in a text, POS tags, or any other elements as they appear one after the other. N-grams correspond to the number of elements in a sequence (Sidorov, Velasquez, Stamatatos, Gelbukh, & Chanona-Hernández, 2014). N-grams are substrings of a large string of length n which is split in to strings of fixed length. For example, the string “MALWARE”, can be segmented into several 4-grams: “MALW”, “ALWA”, “LWAR”, “WARE” and so on (Santos, Peña, Devesa, & Bringas, 2009).

N-gram technique has been used for analysis for quite a long time in the field of NLP for tasks such as language modelling and speech recognition. In 1994, character n-gram was used mainly for text categorization, but currently, common n-gram (CNG) analysis has been successfully applied to authorship attribution, detection of dementia and text clustering.

### **2.5.1 How N-grams Work**

Text n-grams are used widely in NLP for text mining tasks.

N-grams are co-occurring words in each window selected from a sentence and while computing n-gram we move forward on words.



Consider the following sentence "*The cow jumps over the moon*". To calculate bigram where  $N=2$ , N-gram for the sentence would be:

- the cow
- cow jumps
- jumps over
- over the
- the moon

For  $N=3$ , the n-grams would be:

- the cow jumps
- cow jumps over
- jumps over the
- over the moon

So, bigram generated 5 n-grams whereas trigram generated 4 n-grams. For unigram,  $N=1$  and this is essentially the individual words in a sentence. When  $N>3$  this is usually referred to as four grams or five grams and so on.

If  $X$  = Number of words in each sentence  $K$ , the number of n-grams for sentence  $K$  would be:

$$N_{gramsK} = X - (N - 1)$$

*Equation 1 Calculating N-Gram (Source: Banerjee & Pedersen, 2003)*

It is essential to identify tokens in a sentence as N-grams are formed by connecting tokens (Banerjee & Pedersen, 2003).

### **2.5.2 Use and application of N-grams**

N-grams are used for a variety of tasks such as developing a language model which can be unigram or bigram or trigram model. Microsoft and Google have developed web scale n-gram models which are used for a variety of tasks such as spelling correction, text summarization and word breaking.

N-grams are also used in developing features for supervised machine learning classification algorithms such as SVMs, MaxEnt models, Naive Bayes, etc. The idea is to use tokens such as bigrams in the feature space instead of just unigrams. But the

use of bigrams and trigrams in feature space may not necessarily yield any significant improvement.

N-gram methodologies are used a great deal in statistical modeling which is used for predicting the next word in any sentence. The language model predicts that the probability of the next word depends on last  $N-1$  words.

Shannon game is an application which tries to guess the next letter (Shannon, 1951). (Damashek, 1995) measured topical similarity in unrestricted text using n-grams while (Huang, Peng, Schuurmans, Cercone, & Robertson, 2003) identified boundaries of sessions using n-grams in a large collection of Livelink log data. (Cavnar & Trenkle, 1994) researched electronic documents, and they calculated the frequency of n-grams in terms of textual errors, such as spelling and grammatical errors. (Roark, Saraclar, & Collins, 2007) used a discriminative n-gram approach for speech recognition. N-gram language modeling can be used for optical or speech character recognition, handwriting recognition, spelling correction and statistical machine translation. Spelling errors can be detected using character n-grams and is used in predicting topic continuations in search engine queries, more than word n-grams.

(Mcnamee & Mayfield, 2004) used the character n-gram method for multilingual text retrieval. They aimed to show that the character n-gram tokenization can provide retrieval accuracy better than the other language specific approaches. (Liu & Kešelj, 2007) studied automatic classification of web user navigation patterns, and they implemented the character n-gram method for capturing textual content of web pages. (Kanaris & Stamatatos, 2007) studied about webpage genre identification for improving the quality of search engines, and they applied the character n-gram method to identify of webpage genres. (Chau, Lu, Fang, & Yang, 2009) researched the character usage of Chinese search logs from Chinese search engines; since the character n-gram method is independent from language, they implemented this method to their study without any difficulty. (Vilares, Vilares, & Otero, 2011) used the classic stemming based methods and the character n-gram method for identifying spelling mistakes and make corrections in Spanish. They compared these methods and showed performance results in their study. In addition to these studies, (El-Nasan A. & M., 2002; Senda & Yamada, 2001)

used the character n-gram method in handwriting recognition (Gencosman, Ozmutlu, & Ozmutlu, 2014).

Next, we look at code which can be used to generate N-grams.

### 2.5.3 Pseudo-code to generate N-grams

The following code may be used to generate N-gram(s); given length of n-gram to be generated and a sentence, a list is returned which will hold the list of n-grams generated.

```
void GenerateNGrams(int N, String sent) {
    String [] tokens = sent.split("\\s+"); //split sentence into tokens
    List<string> ngramList = new List<string>();
    //GENERATE THE N-GRAMS
    for (int k=0; k < (tokens.length - N+1); k++) {
        String s="";
        int start=k;
        int end=k+N;
        for (int j=start; j<end; j++) {
            s=s+""+tokens[j];
        }
        //Add n-gram to a list
        ngramList.add(s);
    }
}
```

## 2.6 Gaps and Key Challenges

Solutions for sentiment analysis are being developed, typically by reducing the amount of human effort needed to classify text. But there are challenges that have been identified and are applicable to this thesis.

1. Detecting fake reviews and spam, which is done by identifying duplicates and outliers and the reputation of reviewers.

Fake reviews refer to fake or bogus reviews which misguide the users or customers by providing them 'false' positive or negative opinion about any object. Spam makes opinion or sentiment analysis useless in many areas and is a challenge faced by sentiment analysis and researchers (Chandni et al., 2015).

2. The integration of opinion with behavior and implicit data, to validate and provide further analysis into the data beyond opinion expressed.
3. Availability of opinion mining software, currently can only be afforded by organizations and governments, but not by citizens. In other words, governments have the means today to monitor public opinion in ways that are not available to the average citizen. Citizens produce and publish content but are unable to analyze it.
4. The usability and user-friendliness of tools need to be improved so as to be usable by citizens and not just by data analysts (Osimo & Mureddu, 2012).
5. Language Problem: Researchers always face a challenge for building lexicons, corpora and dictionaries for any language although there are a number of resources available for English language.
6. NLP processing needs more enhancement with respect to domain-dependent sentiment analysis and or context-based mining, which will give good results compared to domain independent corpus. But domain-dependent corpora are more difficult to build (Chandni et al., 2015).

## 2.7 Summary

This chapter presented a literature review of areas closely aligned with the topic of this thesis. It was found that current research focusses on: Reduction of human effort needed to analyze content; Semantic analysis through lexicon/corpus of words with known sentiment for sentiment classification; Identification of opinionated material to be analyzed; and Computer-generated reference corpuses in the healthcare field. We then looked at N-Grams, which is a technique widely used for text mining, the algorithm used

to calculate n-grams and the pseudo code which can be written in any programming language to generate n-grams.

Current gaps in research and key challenges in the field were also presented.

# Chapter 3: Data mining and NLP

## 3.1 Introduction

Literature on sentiment analysis was reviewed in the previous chapter where we also looked at the different tools and methods, and areas of application of sentiment analysis. This chapter focuses on data mining and the process and stages of knowledge discovery. Section 3.2 introduces data mining while in the next sections we discuss KDP, followed by stages of KDP. In Section 3.5, we discuss text mining where we look at the seven practice areas where text mining is applied and the interaction between these areas. After this, we discuss NLP and its components which is one of the key practice areas in the discussion of text mining.

## 3.2 Data Mining

Data mining is mainly used to make sense of huge volumes of unsupervised data in different domains. Data mining users are domain experts who own and collect data which means that they understand data and the processes that are used to generate it. Businesses routinely collect huge volumes of data and invest huge amount of time to make sense of that data. Business houses often use data mining to increase profit, gain competitive advantages and gain better insight of the domain to create novel approaches to problem solving.

Data mining is mainly classified into three major activities:

1. Making sense of data is the first key activity which varies depending on the user's experience.
2. Knowledge derived should be useful, meaningful, valid and novel for the data owners to understand and create models that can be described in easy to understand terms, making this the second key activity which requires generated models to be valid.
3. Finally discovered knowledge must be novel. We should understand that data mining is about analyzing large amounts of data and requires the use of data mining

techniques to analyze and reduce the data in terms of both dimensionality and quantity.

Data mining mostly deals with unsupervised data as it is much easier and less costly to collect unsupervised data as with supervised data we need to have known inputs that correspond to known outputs which are determined by domain experts. This makes it important to understand the process that leads to new knowledge discovery, which is a sequence of steps to be followed to discover patterns in data. These steps can be discovered with the help of an open source or commercial software tool. Process models are used to formalize KDP and this helps institutions to understand, plan and execute KDP which in turn helps save time and cost. Steps in process model are non-trivial and involve multiple iterations of interaction with data owners (Baitharu & Pani, 2016).

### **3.3 Knowledge Discovery Process (KDP)**

As previously mentioned in Chapter 1, Knowledge Discovery is the process of evaluating data from various sources and viewpoints and putting it together into useful information. KDP needs to be structured as a standardized process model for the following reasons:

1. Product of data mining should be useful to the owners. Unstructured and blind application of data mining frequently results in meaningless knowledge which does not contribute to problem solving, ultimately leading to failure of a project. A well-defined process would result in an understandable, novel and valid product.
2. A logical, well-thought out structured approach to KDP can help any decision maker understand the importance and value as well as the mechanics behind KDP. There is huge untapped potential knowledge available in possibly valuable data which humans may fail to understand. Decision makers often do not want to put in the time and money on formal methods of knowledge extraction from data but often rely on domain experts for information. However, being the ultimate decision maker, they frequently end up trying to understand the technology applied to create

solutions. Logical and structured process models help resolve any doubts and questions they may have.

3. A solid grounded framework is required for knowledge discovery projects and require significant project management effort, team work and careful scheduling and planning.
4. Knowledge discovery projects should use well-defined models like waterfall or agile methodologies like software engineering does, which is a relatively new and dynamic field.
5. Modern data miners need to learn accepted industry standards and standardization would help in creating news methods and procedures which will enable end users to deploy their projects easily and would directly lead to projects which are cheaper, faster, manageable and more reliable. Altogether, this would promote the creation of business terminologies which will result in greater acceptability and exposure for the field of knowledge discovery (Fayyad, Piatetsky-Shapiro, & Smyth, 1996).

### 3.4 Stages of KDP

KDP depends heavily on techniques from statistics and machine learning and makes use of ideas from database query, machine learning, visualization and artificial intelligence areas, the focus being the creation of techniques for extracting knowledge. Figure 4.1 shows five important steps in KDP and these steps are explained subsequently.

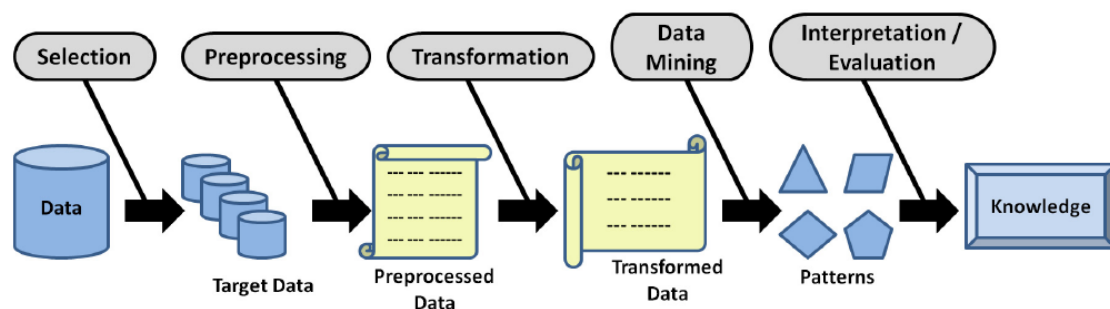


Figure 5 KDP Diagram (Source: (Fayyad et al., 1996)

1. Data selection: In this step, the task related to analysis task is selected from datasets or databases.



2. Data preprocessing: Missing observations are replaced, extreme values, data noise and inconsistencies are removed.
3. Data transformation and reduction: In this step, data is converted into convenient structures. Here we try to find useful structures to implement data mining.
4. Data mining: We select suitable KDP or data mining algorithm to extract data patterns.
5. Interpretation or evaluation: Is used by user to understand and extract knowledge from the patterns mined, this interpretation is typically carried out by visualizing the models, patterns or the data for the models (Kurgan & Musilek, 2006; Reinartz, 2002).

After looking into data mining and KDP, next we look at text mining and text analytics to understand how data mining and NLP are used to extract, analyze and process structured and unstructured data.

### 3.5 Text Mining

Text mining and text analytics are terms used to describe a range of technologies for processing and analyzing semi or unstructured text data. We need to know both types of techniques so we can apply powerful algorithms to large document databases helping to convert them to structured and numerical formats, which can be used to classify documents. Text mining is emerging out of a group of related and distinct but related fields and due to the disparity and breadth of these distinct fields it becomes difficult to characterize what text mining is. This situation is further complicated by the fact that different areas of text mining are in different stages of maturity. There is a total of seven different text mining practices that we need to consider when talking about text mining (G. Miner, January 2012). Figure 6 shows the seven practice areas and these are explained in the section that follows.

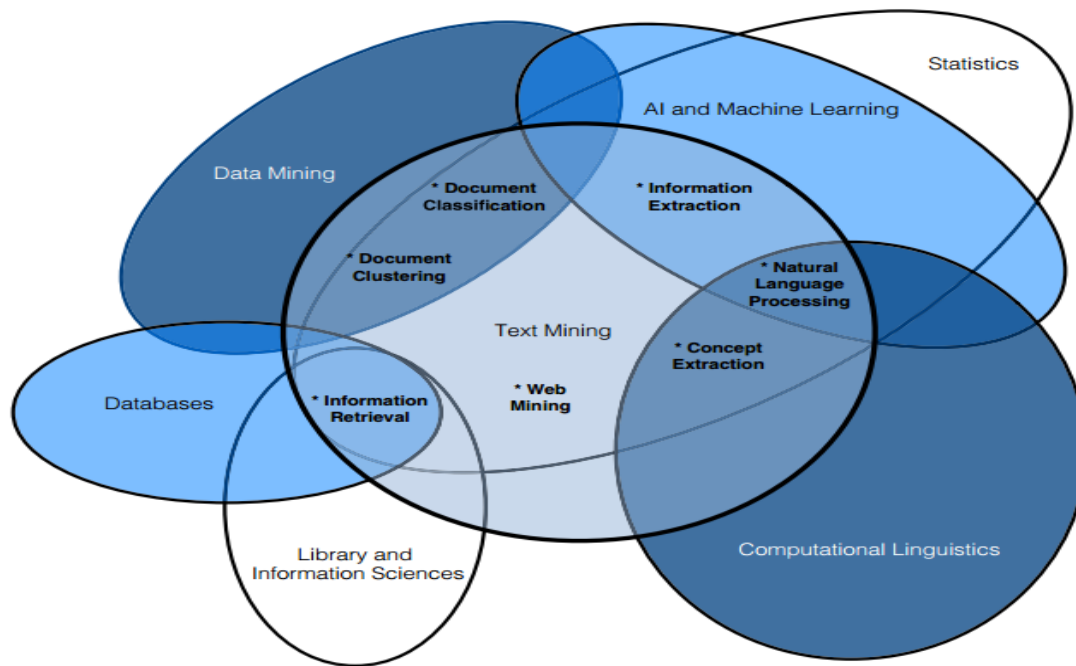


Figure 6 Text Mining (Source: (G. Miner, January 2012))

### 3.5.1 The Seven Practice Areas of Text Analytics

Text mining has been divided into seven practice areas, based on the uniqueness of the characteristics of each of these areas. Though these practice areas are distinct they are interrelated as well, as explained here:

1. Search and Information Retrieval (IR): Storage and retrieval of text documents, including search engines and keyword search.
2. Document clustering: Grouping and categorizing terms, snippets, paragraphs, or documents, using data mining clustering methods.
3. Document classification: Grouping and categorizing snippets, paragraphs, or documents, using data mining classification methods based on models trained on labeled examples.
4. Web mining: Data and text mining on the Internet, with a specific focus on the scale and interconnectedness of the web.

5. Information extraction (IE): Identification and extraction of relevant facts and relationships from unstructured text; the process of making structured data from unstructured and semi structured text.
6. NLP: Low-level language processing and understanding tasks (e.g., tagging part of speech); often used synonymously with computational linguistics.
7. Concept extraction: Grouping of words and phrases into semantically similar groups (G. Miner, January 2012).

### 3.5.2 Interactions between Practice Area

The seven practice areas of text analytics overlap considerably, since many practical text mining tasks sit at the intersection of multiple practice areas. A visualization of this overlap between practice areas is shown as a Venn diagram in Figure 7. For example, entity extraction draws from the practice areas of information extraction and text classification, and document similarity measurement draws from the practice areas of document clustering and information retrieval. (G. Miner, January 2012)

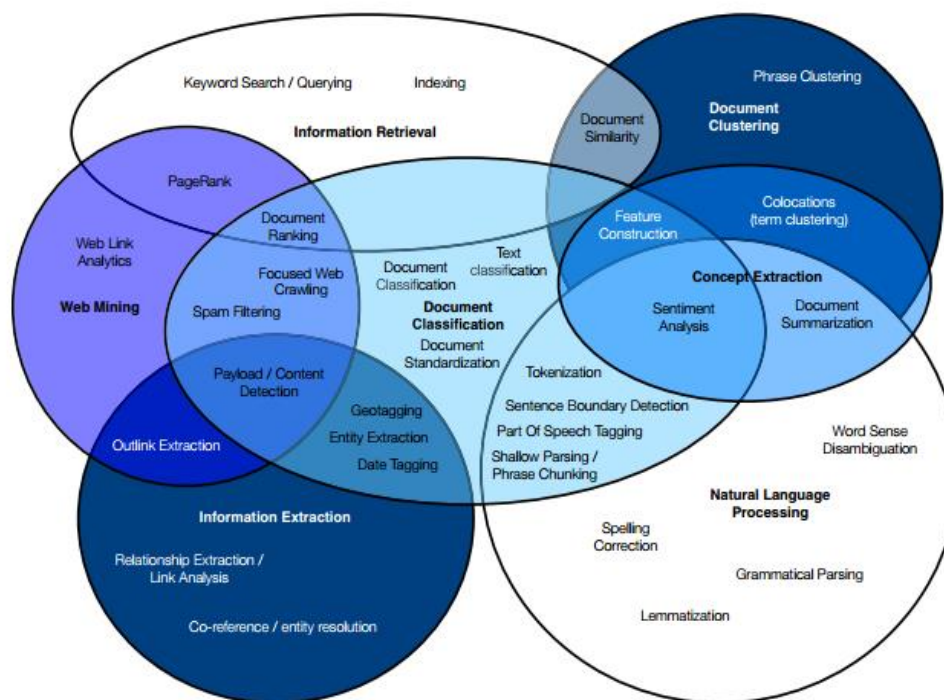


Figure 7 Interactions between Practice Areas Source: (G. Miner, January 2012)

In the next section, we discuss NLP and its components which are important for document classification.

### 3.6 NLP

NLP is an area of research and application that explores how computers can be used to understand and manipulate natural language text or speech to do useful things. NLP researchers aim to gather knowledge on how human beings understand and use language so that appropriate tools and techniques can be developed to make computer systems understand and manipulate natural languages to perform the desired tasks. The foundations of NLP lie in several disciplines—computer and information sciences, linguistics, mathematics, electrical and electronic engineering, artificial intelligence and robotics, and psychology, to name a few (Chowdhury, 2003).

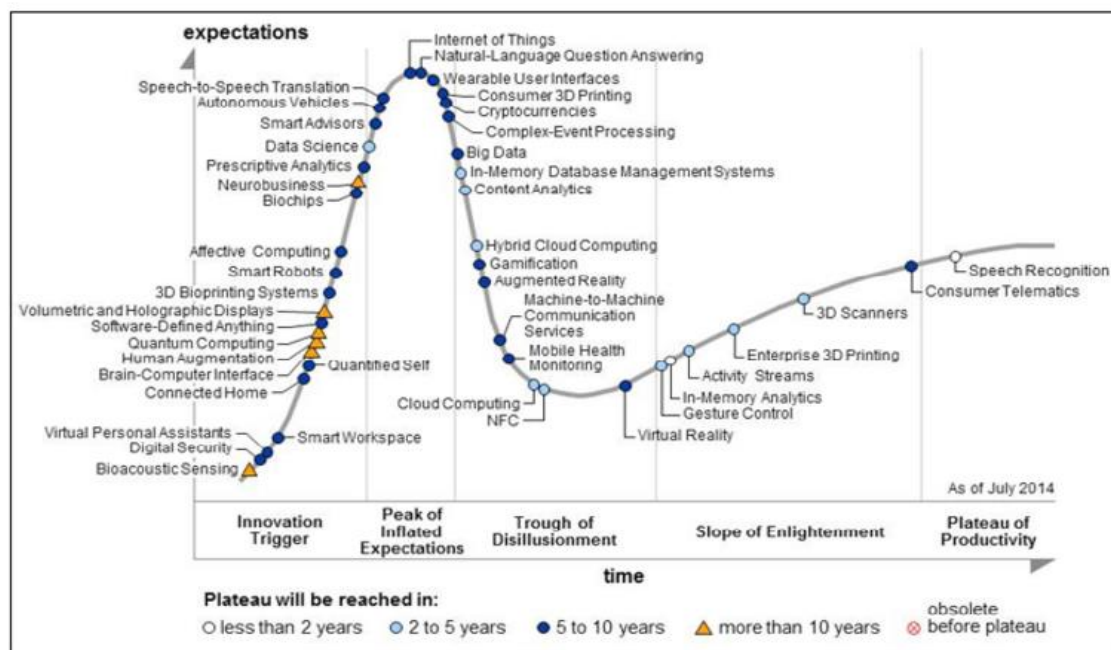


Image is taken from <http://www.gartner.com/newsroom/id/2819918>

Figure 8 Application of NLP (Source: Chowdhury, 2003)

Some of the applications of NLP are: Spelling Correction; Search engines; Speech engines; Spam classifiers; News feeds; Machine translation; IBM Watson. The development of these areas is shown in Figure 8. NLP is one of the most niche areas

and requires a very specific skill set with understanding of the language and the tools to process the language efficiently (Hardeniya, Perkins, Chopra, Joshi, & Mathur, 2016).

### **3.6.1 Parsing**

The process of parsing helps with automatic analysis of a sentence, or an entire discourse, that is viewed as a sequence of words which allows us to decide the syntactic structure of a sentence. The grammar for natural languages is ambiguous and typical sentences have multiple possible analyses. In fact, for a typical sentence there may be thousands of potential parses most of which will seem completely nonsensical to a human (Clark, Fox, & Lappin, 2010). Parsing needs a mathematical model of syntax of language of interest that is expected to be formal grammar, which consists of a collection of rules that specify elements of the language (words) and how they are used to create a sentence. Parsing is the core central component of an NLP tool and helps represent the structure of a sentence as a list or tree. Bottom-up, top-down, left-corner, head-corner and statistical parsing are some of the strategies used to parse a sentence. Parsed information is passed to modules that implement pragmatic, semantic and discourse processing. Therefore, parsing is an important aspect in showing meaning and determining the structure of a sentence in language processing.

### **3.6.2 Discourse**

Informal language does not consist of isolated pieces of sentences or text but of words which together form a unified whole which we call a discourse. As discussed, parsing passes useful information to the module that implements discourse processing, which tells us how these two components i.e. parsing and discourse, interact with each other. In this section we discuss discourse, discourse structure and related terms.

Discourse is a piece of writing or serious speech on a subject or conversation or plain discussion between different people in a language. The text so produced is a set of coherent and cohesive sentences which helps achieve a communication goal.

Coherence specifically has more to do with a meaningful relation between two words and how they combine to produce a meaningful discourse structure. The sequence of a

discourse topic is typical of certain genre or document and accounts for the way texts are segmented and organized by topics (Eugenio, 2005). Interpretation, production and utterances of phrases whose meaning depends on the context of the discourse and the fact that a sequence of utterances conveys a meaning which is more than the sum of the individual utterances are two phenomena which are intrinsic to discourse processing.

Discourse generation basically deals with generation of coherent text and discourse processing is the last stage in interpretation of a language after parsing and semantic analysis. Language generation begins with non-linguistic illustration of information, after which we can perform discourse processing which imposes order and structure over the set of messages. This is followed by linearization and planning, which includes sentence aggregation and relating individual terms to their entities of interest. The final step in discourse generation is linguistic realization proper, namely, applying the rules of grammar to produce a text that is syntactically and morphologically correct.

Anaphora or co-reference, helps in setting up referential dependency between two or more expressions and refers to referentially dependent expressions in natural language which contribute their meaning to a sentence by helping identify another expression which provides them their semantic value. For example, consider the following statement: Mark felt that there was someone watching him. Knowing that 'him' refers to 'Mark', the pronoun is an anaphor and 'Mark' is an antecedent. Both refer to the same person 'Mark'. Co-Reference is often used to describe this relation between an antecedent and anaphor (Liddy, 1990).

### **3.6.3 Text Categorization**

Text categorization has become increasingly important and deals with automated assignment of natural language texts to predefined categories. Text categorization has found primary application in assigning subject categories to documents to help information retrieval. NLP uses text categorization heavily for data extraction. Categorization is used to filter out unnecessary parts of the document or the whole document, which is unlikely to have the text or can also be used to route words to

category specific processing modules or create fillers for some fields. In general, text categorization tries to recreate human categorization judgement.

One approach to building a text categorization system is to manually assign documents to certain categories and then use inductive learning to assign documents to categories automatically in future, based on the texts they have. Rules, or boolean expressions, can be created which capture certain combination of keywords that indicate a class. Class is more often a subject area such as coffee or person. Apart from manual classification or hand-crafted rules by domain experts which may exceed or rival the accuracy of automatic classification, there exists machine learning based text classification where the rules or decision criteria are more often learned from the training data. This approach is also known as statistical text classification if the learning method or algorithm used is statistical. Statistical approach require a large number of documents for each class and completely eliminates the need for manual classification (Lewis & Ringuette, 1994; Manning, Raghavan, & Schütze, 2008). Information about individual words is stored in a lexicon and computational lexical representation techniques have been created to model lexical knowledge. Computational lexicon helps perform mapping from phonology or orthography (the system of contrastive relationships among the speech sounds that constitute the fundamental components of a language) to some combination of semantic, syntactic and pragmatic information. It is used to deliver information to modules which is used to analyze or generate speech or text. Information contained in the computational lexicon depends on the system. For example, a lexicon for a Part of Speech (POS) tagger would be much simpler as compared to a lexicon for a natural language interface. Generally we think of commonly used approaches to lexical representation as forms of attribute-value representation, and in such formalism information is represented as pairing of attributes and associated values (Pustejovsky, 2005).

As human beings cannot read, understand or synthesize megabytes or terabytes of data or text daily, researchers have explored various information management strategies and the most common of these are information filtering and Information Retrieval (IR). IR systems are harvesters that bring back useful information from vast

fields of raw information. Information is often available in journal articles or newspapers and IR systems can retrieve articles that are relevant to the search criteria.

IE systems can transform huge amount of useful information into required text. IE begins with collecting texts and then transforming them into data that is readable and can be analyzed. It helps retrieve text fragments and extract information from these fragments and then puts them together into a coherent framework. Information retrieved is stored in traditional databases and then data can be retrieved using standard queries. Currently, IE systems are partly accurate and deal with specific types of texts. IE, from the perspective of NLP, is a set of tasks which are well defined, use real-world texts, poses difficult and interesting NLP problems and performance can be compared to human performance on the same task (Cowie & Lehnert, 1996).

After discussing the different components of an NLP system, we move on to discuss Word Sense Disambiguation, beginning with a discussion about WordNet.

### **3.7 WordNet in NLP**

WordNet lexicon database is used widely for information retrieval and translation which require WSD and is a tool that is widely used by the natural language community. WordNet, which started as a project in 1985, is a semantic dictionary designed to represent words and related concepts as an interrelated system which is consistent with the way any human would organise his own mental lexicons. WordNet is not a traditional dictionary or thesaurus, and is also different from most of the other lexicon dictionaries compiled before, but it combines the features of both dictionary and thesaurus (Kreutzer & Witte, 2013).

There were three assumptions made at the initial stages of development, one was separability hypothesis, which says that every lexical component of any language can be isolated and studied separately from other components. This idea seemed promising as a person's vocabulary grows with time, and phonology and grammar do not change. The second was patterning, which suggested that people could not master all the lexical knowledge of any language by remembering every single word's meaning



separately, but rather by understanding the patterns and relations between different words. The third assumption was comprehensive hypothesis which suggested a need for lexical storage like a human's, and must understand the process of natural language. At the same time, computational linguists were looking for alternative theories to express word semantics which would not depend on the decomposition of the word. Thus, they came up with networks and diagrams to represent semantic relations between words. In the early days, WordNet was used to decide if relational semantics could be applied to large lexicons (Kreutzer & Witte, 2013).

WordNet usage for WSD, to begin with, was limited by sparsity of its arcs. For example, the verb interest is connected to the adjective interesting and noun interest. But since a noun has more than one meaning or sense, not all of the synonyms are related to the adjective sense or verb sense and these must be entered manually. Adding more information about their meaning is useful for machine and human users. A user who wishes to better understand the definition or gloss can refer to the synsets (set of synonyms) for more information on verbs, adjectives, nouns and adverbs in that gloss. This work resulted in the creation of a semantically better annotated corpus, WordNet, that can be used to test and train natural language systems. (Boyd-Graber, Fellbaum, Osherson, & Schapire, 2006) tried to improve the density of WordNet and make it a more efficient tool for natural language by collecting more than 120,000 ratings from human annotators. The strength links in WordNet are weighted and directed which helps in expressing more discernable semantic relations. The WordNet lexicon has nouns, verbs, adjectives and adverbs. Lexical information is organized in terms of word meanings rather than word forms. (Navigli & Lapata, 2010)

The corpus used for the initial development of WordNet was the Brown corpus. Adjective pairs were also incorporated along with various synonyms and antonym dictionaries. A year later, Fred Chang's list of words were used and added as input. COMLEX lexicon which had 39,143 words were added as well. As the list of words grew it became necessary to divide the database. The division into database took the syntactic categories of the words into account and different files for verbs, nouns, adjectives and later adverbs. Later verbs and nouns were divided into different classes.

The WordNet team created a lemmatization program which returned the base form of any word, as WordNet was not able to recognise plurals. In 1991, ConText was developed which pre-processed text by performing various NLP tasks and the output was stored in WordNet. In this way, semantic tagging was used to greatly improve the coverage for words and meanings appearing in WordNet (Kreutzer & Witte, 2013).

The number of glosses in WordNet has grown steadily since 1989 when it had 37,409 synsets and no glosses, until 1995 when it had 91,050 synsets and 75,389 glosses. The current number is 117597 synsets. The first version of WordNet that was publicly available was version 1.0 in 1991. However, WordNet is still being worked on. The current version, 2.1, is available for free download on the WordNet website <http://wordnet.princeton.edu/>.

The structure of WordNet is based on the word as a basic unit and thus WordNet does not decompose words into smaller meaningful units. WordNet also does not contain units larger than words, such as scripts or frames, which have been proposed as building blocks for other lexicons. For example, a frame would be a lexicalized concept that is relevant to a certain type of situation. Frames include both verbs and nouns and their relations that hold true in the situation in question. Even if WordNet does not have a frame lexicon its relational semantics network still reflects some of the structure of frame semantics. For example, verbs like sell and buy are related in the WordNet lexicon to “commercial transaction”.

The division of words into four separate nets, one for each open word class, also entails that WordNet has no information about the syntagmatic properties of words. Another characteristic of WordNet is that unlike other dictionaries it has short phrases, such as “bad person,” that are thought to be not para-phrasal by single words. These phrases are needed to fill lexical gaps by serving as connections between two words when there is no single word with the desired meaning to connect them. These gaps are not structural artefacts, but quite often, they are lexicalized in other languages, just not in English.

Also, a distinction that has been made between meanings of words is the separation of meanings into the camps of lexical and encyclopedic knowledge. But this distinction was not incorporated into WordNet since WordNet does not try to include any of the latter, although the synonym set information provided by WordNet often goes beyond lexical meaning. Although initially the only information synsets were supposed to hold was pointers to other synsets, it was felt that a description of the current sense would help in distinguishing highly similar synsets for words that appear in many synsets. Later, this was found to be helpful in dealing with many technical concepts whose lexical definitions intermixed with encyclopedic knowledge. Thus, overall WordNet found the need to store more than only lexical meaning, even if it was meant only for reasoning and inference purposes (Kreutzer & Witte, 2013).

### 3.7.1 How does WordNet work?

Lexical data is arranged in terms of the word meanings rather than the word forms. Senses in WordNet database (<http://wordnetweb.princeton.edu/perl/webwn>) are represented by synonym sets or synsets with words in each synset sharing a common sense of the word. For example, consider the senses of verb drink: “consume liquids”, “consume alcohol” and “toast”; See Figure 9 for a representation of this information.

Each word in a synset is associated with a part of speech which is denoted by a subscript where n stands for noun, v for verb, a for adjective, and r for adverb. Each synset is associated with a textual definition which explains the meaning of the word. The synsets are ranked as per the frequency of occurrence in the SemCor corpus which is a subset of Brown corpus with word senses. A superscript denotes the ranking of the sense of the word. For example, the 5<sup>th</sup> sense of the word drink in the figure above would be denoted as drink<sub>v</sub><sup>5</sup>. The latest WordNet version 3.0 contains 155,000 words arranged in more than 117,000 synsets. Semantic and lexical relations are also encoded in WordNet. Lexical relations connect pairs of word senses, whereas, semantic relations relate synsets. Lexical relations in WordNet are nominalization (e.g., the noun drinking<sub>n</sub><sup>1</sup> is a nominalization of the verb drink<sub>v</sub><sup>1</sup>), antonymy (e.g., cold<sub>a</sub><sup>1</sup> is an antonym of hot<sub>a</sub><sup>1</sup>), pertainymy (e.g., dental<sub>a</sub><sup>1</sup> pertains to tooth<sub>n</sub><sup>1</sup>), and so on (Navigli & Lapata, 2010).

## Noun

- **S: (n) drink** (a single serving of a beverage) *"I asked for a hot drink"; "likes a drink before dinner"*
- **S: (n) drink, drinking, boozing, drunkenness, crapulence** (the act of drinking alcoholic beverages to excess) *"drink was his downfall"*
- **S: (n) beverage, drink, drinkable, potable** (any liquid suitable for drinking) *"may I take your beverage order?"*
- **S: (n) drink** (any large deep body of water) *"he jumped into the drink and had to be rescued"*
- **S: (n) swallow, drink, deglutition** (the act of swallowing) *"one swallow of the liquid was enough"; "he took a drink of his beer and smacked his lips"*

## Verb

- **S: (v) drink, imbibe** (take in liquids) *"The patient must drink several liters each day"; "The children like to drink soda"*
- **S: (v) hit the bottle, drink, booze, fuddle** (consume alcohol) *"We were up drinking all night"*
- **S: (v) toast, drink, pledge, salute, wassail** (propose a toast to) *"Let us toast the birthday girl!"; "Let's drink to the New Year"*
- **S: (v) drink in, drink** (be fascinated or spell-bound by; pay close attention to) *"The mother drinks in every word of her son on the stage"*
- **S: (v) drink, tope** (drink excessive amounts of alcohol; be an alcoholic) *"The husband drinks and beats his wife"*

## Verb

- **S: (v) absorb, suck, imbibe, soak up, sop up, suck up, draw, take in, take up** (take in, also metaphorically) *"The sponge absorbs water well"; "She drew strength from the minister's words"*
- **S: (v) assimilate, imbibe** (take (gas, light or heat) into a solution)
- **S: (v) drink, imbibe** (take in liquids) *"The patient must drink several liters each day"; "The children like to drink soda"*
- **S: (v) imbibe** (receive into the mind and retain) *"Imbibe ethical principles"*

## Noun

- **S: (n) toast** (slices of bread that have been toasted)
- **S: (n) toast** (a celebrity who receives much acclaim and attention) *"he was the toast of the town"*
- **S: (n) goner, toast** (a person in desperate straits; someone doomed) *"I'm a goner if this plan doesn't work"; "one mistake and you're toast"*
- **S: (n) pledge, toast** (a drink in honor of or to the health of a person or event)

## Verb

- **S: (v) crispen, toast, crisp** (make brown and crisp by heating) *"toast bread"; "crisp potatoes"*
- **S: (v) toast, drink, pledge, salute, wassail** (propose a toast to) *"Let us toast the birthday girl!"; "Let's drink to the New Year"*

Figure 9 WordNet Synsets (Source: <http://wordnetweb.princeton.edu/perl/webwn>)

In this research, we use SentiWordNet, which was created from WordNet and contains all semantically similar synsets of WordNet, to help calculate the sentiment score of a sentence. This is discussed in Chapter 4.

### 3.8 Word Sense Disambiguation (WSD)

Humans use words that can be interpreted in many ways; such ambiguity is common and based on the context of the conversation or sentence.

1. I like poaching
2. I like poaching eggs

The above sentences clearly have different meanings. A machine needs to break down sentences into textual information and create data structures which are analyzed to understand the meaning of the sentences. Demand for automatic methods of processing large amount of unstructured data such as web pages, data warehouses and corpora has increased in the last few decades. Traditional methods of information retrieval always fail when they are applied to such data as they do not extract relevant information and are not able to discard documents which fail to meet users' queries. WSD is an intermediate task for text disambiguation that can be configured as a stand-alone module or integrated into a bigger application. (Navigli, 2009)

WSD helps with computational interpretation of meaning of words and it relies heavily on the approach to word sense representation, sense inventory granularity, unrestricted nature versus domain-orientated nature of words and the set of words to disambiguate the meaning. WSD can be summarized as a technique applied to a set of words to associate suitable sense using one or more knowledge sources such as a corpus, which can be annotated or unlabeled or more structured resources such as machine readable dictionaries or semantic networks (Navigli, 2009).

However, WSD is a difficult task when you consider the computational limitations. Knowledge resources, information about words, senses and context in the target word were some of the areas which are considered important for NLP. Generalization was difficult back in the 70's considering the limitation of computing power. Large scale lexical resources were released in the 80's which helped in extracting knowledge using automatic methods. The 90's saw the creation of periodic evaluation campaigns and employment of massive statistical methods which has continued to present days (Navigli, 2009).

Improved WSD systems were developed with the availability of annotated corpora. Supervised algorithms greatly outperform unsupervised algorithms but they often need huge amounts of annotated training data, and creating this data is labor intensive and an expensive task and must be repeated for new languages or domains. Given the data requirements for supervised classification and the current scarcity of suitable data for many text genres and languages, unsupervised approaches offer hope for large-scale sense disambiguation. Unsupervised approaches do not use labeled training data to perform sense disambiguation. Unsupervised approaches exploit the structure and relations per pre-existing sense of the repository or inventory to perform disambiguation task accurately. A restrictive view of "unsupervised" applies to methods for sense disambiguation, which tries to automatically find all senses of a word without labeled training data (Navigli & Lapata, 2010). Graph-based methods which represent a knowledge base in graphs have become a popular choice for domain-independent knowledge-based WSD systems and offer the advantage of scanning through the entire knowledge base during the disambiguation process.

There are three mainstream approaches to word sense disambiguation:

1. Supervised WSD: this approach uses machine learning to understand a classifier for a target word from the labeled training data. Among supervised methods, memory based and SVM approaches have been proven to be best systems.
2. Knowledge-based WSD: these methods exploit knowledge resources such as dictionaries or thesauri to decide the senses of words in context. They have the advantage of a wider coverage, thanks to the use of large amounts of structured knowledge. The best knowledge-based systems in the literature, such as Degree or Personalized PageRank, exploit WordNet or other resources to build a semantic graph and exploit the structural properties of the graph to choose the proper senses of words in context.
3. Unsupervised WSD: these are Word Sense Induction techniques aimed at discovering senses automatically based on unlabeled corpora to offer a sense choice for a word in context. They do not exploit any manually sense-tagged corpus.

The question of which approach is best in general, and for which application, is still very much open. In fact, until recently, the general belief was that supervised WSD performed better than knowledge-based WSD. However, recent results show that, in the presence of enough knowledge or within a domain, knowledge-rich systems can beat supervised approaches while offering, at the same time, much wider coverage (Navigli, 2009).

### **3.8.1 Performing WSD**

The main goal of sentiment analysis is to use automated methods to extract emotions from text or documents, while analyzing the overall sentiment of the document which needs us to extract information from the text. Sentiment lexicons are often employed to find the words used in the document which tells us about the sentiment of the word. WordNet is the foundation for SentiWordNet lexicon which uses a semi-supervised approach to constructing a vocabulary database which helps us find out the polarity of

the document. SentiWordNet can be considered as a general sentimental vocabulary database which helps us find sentimental words in a word of mouth document. Word sense identification affects sentiment analysis which is due to the fact that one word can have more than one meaning and the meanings expressed by words change, based on background and environment. Sentimental attitude changes when meaning of the word changes, which affects the overall sentiment analysis results. SentiWordNet provides us with meanings of a word which are considered as senses and each sense is assigned sentiment polarity score; sense 1 characterizes the sense used most often in general situations.

Two approaches are used to select sentiment scores for words that have multiple meanings. The first is to pick sense 1 of the word to serve as the meaning of the word in the text, this method does not consider the domain knowledge and usually results in biased sentiment analysis results. For example, in the context of movie reviews, the word “suck” is most likely to mean “inappropriate or lousy”, and is used to express the opinion that the movie is very poor. The sentiment, in this case, is negative. However, in SentiWordNet, the meaning represented by sense 1 of the word is “a sucking action” and is classified as having neutral sentiment. If the sense 1 meaning is automatically selected, the sentiment would clearly be incorrect. The second method is to take the average of the sense scores for all meanings for words with multiple meanings, and use the average sentiment score to conduct analysis. However, this method does not take the effects of domain knowledge into consideration, and could also result in issues with the accuracy of sentiment analysis.

(Hung & Chen, 2016) recognized that words used in different domains may have different senses, different sentiment values and even different sentiment orientations. Figure 9 shows the structure of their approach.



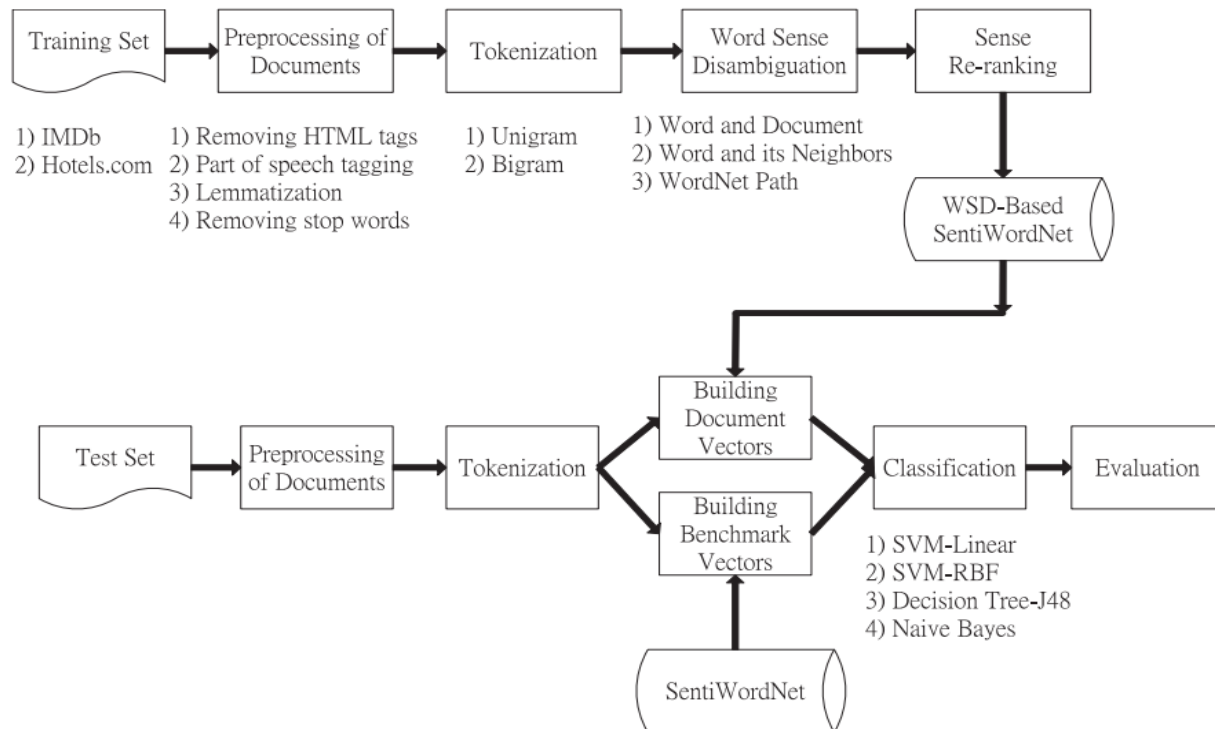


Figure 10 WSD Workflow (Source: Hung & Chen, 2016)

## Preprocessing of documents

First, we remove all the non-necessary tags from document and then as words may have different meanings we use different parts of speech (POS) for a word. Third we can lemmatize the word to its base form using SentiWordNet or WordNet and then perform word cleansing by removing stop words.

## Tokenization

Tokens are features of text and the basic unit for processing. Two approaches which stand out are word-based and phrase-based approaches. Word based tokenization, which is also known as unigram-based approach, treats each single word as a feature. After preprocessing a word of mouth text, we use term frequency to select other important features from the remaining word. Term frequency of a word basically implies that it is significant in relation to the text. If a word occurs multiple number of times in a document, it is an important concept to the document. Word-based tokenization does not understand the relationship between words. Phrase-based tokenization, also known

as bigram language model, helps decide the relationship between two words through calculation of the number of times any two words appear together in a text document (Hung & Chen, 2016).

### **3.8.2 WSD using SentiWordNet**

SentiWordNet is based on WordNet, which shows possible senses for a word and lists each sense in order of usage frequency. A word may have different senses with different sentiment values or even different sentiment orientations. SentiWordNet can provide a word with a proper sentiment orientation and sentiment value, only if the sense of this word is clear. Thus, a WSD-based lexicon is domain oriented. A word may have several senses depending on the correct part of speech. The proper sense of a word can be found by comparing the target document with the glosses (definitions) of each sense of this word, defined in WordNet (Hung & Chen, 2016).

## **3.9 Summary**

To summarize, we looked at data mining, KDP and the stages of KDP. Next, we considered text mining and the seven practice areas of text analytics and following this, we looked at NLP which is one of the seven practice areas. We then looked at the different components of NLP, followed by WordNet and finally WSD. Word sense disambiguation is one of the most important phases of sentiment analysis and can be performed in a supervised or unsupervised manner or by using knowledge-based ways of performing WSD.

# Chapter 4: Analyzing Negative Sentiments

## 4.1 Introduction

Sentiment analysis is one of the major sub-problems in NLP and when applied to analyzing review comments, it helps in understanding the aspect or topic being reviewed, and whether the review are positive or negative. Data can be structured or unstructured; while structured data can be easily analyzed, analyzing unstructured data needs complex algorithms. Different levels of sentiment analysis can be performed. Document-level analysis looks at analyzing a document while sentence-level analysis helps analyze one sentence at a time and provides us with a summary of the sentence analyzed. Aspect-level analysis, analyzes the overall opinion on an entity and is the most basic form of sentiment analysis. Sentiment words have an important role in identifying the sentiments in a document which are combined to form a sentiment lexicon. Sentiment lexicons are used by complex algorithms to analyze the sentiments of a document which can be subjective or objective. Subjective sentences provide opinion about a person or subject and are easy to analyze whereas objective sentences express irony or negation and are more of a challenge to analyze (Medhat, Hassan, & Korashy, 2014).

Supervised and unsupervised learning algorithms or techniques are two of the main techniques used for sentiment classification. Text classification based on classifiers are used for supervised learning which uses frequency and terms, phrases and words, sentiment shifters and part of speech, negation and so on, to analyze or classify a document. Unsupervised techniques use fixed syntactic patterns and POS tagging to identify the entity, aspects and opinions. Maintaining a sentiment word dictionary like SentiWordNet or WordNet, based on the weight of the opinions is another approach for unsupervised learning which helps in understanding the effect of negation or sentiment shifting words. (Medhat et al., 2014)

## 4.2 SentiWordNet

SentiWordNet was created by automatic annotation of all semantically similar synsets of WordNet, based on the notion of negativity, positivity and neutrality. Each synset from WordNet is related to three numerical scores denoting sentiment, which indicates how negative (Neg), positive (Pos) or neutral ((Obj) for objective) the terms contained in the synset are. Different opinion-related properties are found for different senses of the same term. Each of the scores are in the range 0.0. to 1.0 and the sum of the three equals 1.0 for each synset. The scores show that the synset has each of the opinion-related properties to some degree which gives us an idea about the sense of the corresponding term. For example, the synset for adjective estimable<sup>1</sup> in SentiWordNet 1.0 for the sense “deserving of respect or high regard” has a Pos score of 0.75. Neg score of 0.0 and Obj score of 0.25, while the synset for adjective estimable<sup>3</sup> belonging to the sense “may be computed or estimated” has a Pos and Neg score of 0 but an Obj score of 1.0. SentiWordNet 3.0 can be freely downloaded from <http://sentiwordnet.isti.cnr.it/> for non-profit research purposes (Baccianella, Esuli, & Sebastiani, 2010).

## 4.3 Negation Identification and Calculation in Sentiment Analysis

Sentiment analysis is used to find out positive and negative sentiments (feelings, opinions and emotions) in text which are based on the meaning of words used in different situations and scenarios. Different grammatical rules can be used to express similar feelings in written text which may have negations which can change the meaning of words. Therefore, it becomes necessary to identify negation and its scope within a sentence to correctly identify sentiments expressed (Asmi & Ishaya, 2012).

Finding negation is a complex task and its complexity increases with use of different negation words. Along with negation words such as nor, not, and so on, we must look out for prefixes, suffixes, diminishers and word intensifiers which can introduce negation in a sentence and there is a need for considerable effort to enlist such words. Negation sentences have been considered. The pseudo code that follows calculates negative polarity for a word that the word has a higher negative score (Asmi & Ishaya, 2012).

```

Function CalculatePolarity Returns Polarity {
    Double polarity = 0
    For Each nounPhraseOfSentence {
        get SentiWordNet value of all Adjectives and Nouns of noun-phrase
        If (Sentence is Marked NEGATION by Syntax Parser) {
            Reverse the SentiWordNet values of related Nouns/Adjectives
        }
        For Each Noun and Adjective {
            polarity += [( 1 – Noun/Adjective) * Noun/Adjective]
        }
    }
    For Each verbPhraseOfSentence {
        get SentiWordNet value of all Adverbs and Verbs of verb-phrase;
        If (Sentence is Marked NEGATION by Syntax Parser) {
            Reverse the SentiWordNet values of related Verbs/Adverbs
        }
        For Each Verb and Adverb {
            polarity += [( 1 – Verb/Adverb) * Verb/Adverb]
        }
    }
    Return polarity
}

```

**Table 3:** Negation Identification (Source: Asmi & Ishaya, 2012)

First Word /Phrase /Clause	Second Word /Phrase /Clause	Negation	Result
Positive	Positive	True	Negative
Positive	Positive	False	Positive
Positive	Negative	True	Positive
Positive	Negative	False	Negative
Negative	Positive	True	Positive
Negative	Positive	False	Negative
Negative	Negative	True	Negative
Negative	Negative	False	Positive

Table 3 can be used for identifying negation in 'part of sentence'. Part of sentence is used to calculate the polarity of a sentence. A sentence can have either simple or complex part of sentence like Noun phrase which is a pronoun and a noun, or Verb

phrase which is a verb and a noun. The following details possible parts of sentence in a sentence; the subsequent example farther down the page works with an actual complete sentence.

```
(Sentence
  (Noun Phrase (Pronoun, Noun))
  (Adverbial Phrase (Adverb))
  (Verb Phrase (Verb)
    (Sentence
      (Verb Phrase (Verb)
        (Noun Phrase (Noun)))
    )
  )
)
```

Sentiment polarity calculation has always been a nested process and this process helps us to calculate the sentiment of the inner most level first and then moves on to higher levels which is called as sentiment propagation. The process helps us to calculate the intensity and polarity of the phrases and words and negative polarity is also considered while calculating the sentiment. The following examples illustrate the process of polarity calculation (Asmi & Ishaya, 2012).

'They have not succeeded, and will never succeed, in breaking the will of this valiant people.'

```
(Sentence
  (Pronoun They)
  (Verb Phrase
    (Verb Phrase (have not)
      (Verb Phrase (Verb succeeded)))
    (and)
    (Verb Phrase (will)
      (Adverbial Phrase (Adverb never))
      (Verb Phrase (succeed)))
    (Prepositional Phrase (in)
      (Sentence
        (Verb Phrase (breaking)
          (Noun Phrase
            (Noun Phrase (the will)))
          (Prepositional Phrase (of)
            (Noun Phrase (this valiant people))))))
    )
  )
)
```

The negation word 'not' is affecting 'succeeded' (+) (which is a positive word) while never is affecting succeed (+). Both successes are in breaking (-) the will of people who are valiant (+) people. As they have not succeeded in doing something negative and the polarity of sentence is positive as shown in Figure 11.

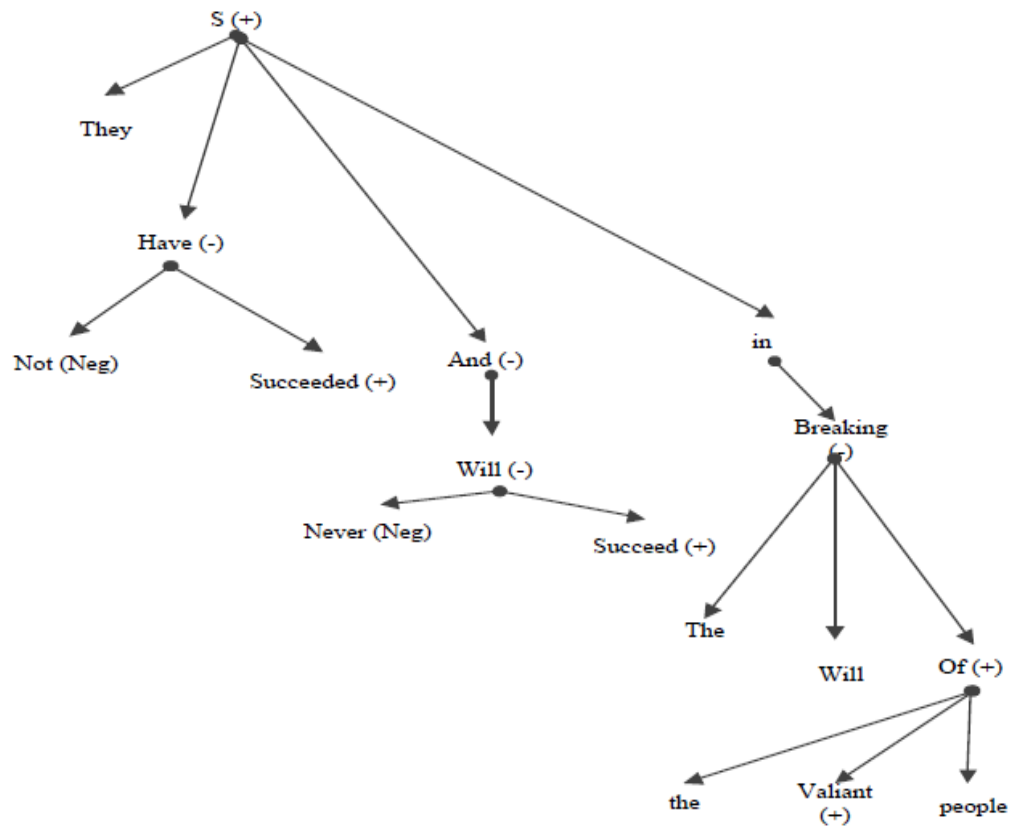


Figure 11 Negation Identification and Calculation (Source: Asmi & Ishaya, 2012)

## 4.4 Summary

In this chapter, we looked at SentiWordNet for use in sentiment analysis. We also considered the effect of negative words and calculating negation in sentiment analysis.

# Chapter 5: Methodology and Framework

## 5.1 Introduction

In this chapter, we start by first looking at evaluation measures used in NLP and the typical workflow of a sentiment analysis module. We then discuss Natural Language Toolkit (NLTK) which is a python module and then we go on to look at the different components of NLTK which can be used to process and analyze the sentiments expressed by a document. Finally, we discuss various classification algorithms that can be used in sentiment analysis classification.

## 5.2 Evaluation

Several different evaluation techniques are required to be used to verify whether the classification models produce the correct output. And this result is necessary to decide the accuracy of the model and the purposes it can be used for. Evaluating a tool is also considered to be an effective mechanism to make future improvements to a model (NLTK, 2017).

### 5.2.1 The Test Set

Evaluation techniques generally compute a score for a model by comparing the labels generated for input in a test set with the correct labels for that input. Test and training sets used need to have the same format, but they should be unique enough so that the classification model learns to generalize to new samples and will not give incorrect high scores. The least frequently occurring labels should occur at least 50 times in a test set for a classification task that has many labels or may include uncommon labels. If the test set has a number of labels which are closely related, then the size of the test set needs to be increased to make up for the lack of diversity which will produce skewed evaluation results. For example, consider the following code sample where we create test set and training set by randomly assigning sentences from the data source (brown corpus – category ‘news’).

```
import random
```



```
from nltk.corpus import brown
tagged_sents = list(brown.tagged_sents(categories='news'))
random.shuffle(tagged_sents)
size = int(len(tagged_sents) * 0.1)
train_set, test_set = tagged_sents[size:], tagged_sents[:size]
```

The above code sample lets us create test and training set which would be very similar and would generate results with high score. So, a better approach would be to create test and training set from different documents as follows

```
file_ids = brown.fileids(categories='news')
size = int(len(file_ids) * 0.1)
train_set = brown.tagged_sents(file_ids[size:])
test_set = brown.tagged_sents(file_ids[:size])
```

A more stringent evaluation can be achieved by using test set from a document that is related to training set

```
train_set = brown.tagged_sents(categories='news')
test_set = brown.tagged_sents(categories='fiction')
```

A classifier that performs well on this test set, can be confidently used to generalize well beyond the data that it was trained on.

### 5.2.2 Accuracy

Accuracy measures the percentage of inputs in the test set which have been labelled correctly by the classifier. For example, a gender name classifier which predicts the correct name for 60 times against a test set containing 80 names would have an accuracy of 75% (i.e. 60/80).

`nltk.classify.accuracy()` is the function used to calculate the accuracy of a classifier on a test set:

```
classifier = nltk.NaiveBayesClassifier.train(train_set)
print 'Accuracy: %4.2f' % nltk.classify.accuracy(classifier, test_set)
Accuracy: 0.75
```

It is also important to take into consideration the frequencies of each class label in a test set. For example, consider a classifier that decides the correct sense of the word 'bank',

where different word senses may be: financial institution; earth mass or hill side; or a row of objects. If we evaluate this classifier on a financial news-wire text, we would find that bank is matched to financial-institution sense 19 out of 20 times with an accuracy of 95%. Instead, if we used a more balanced corpus where the most frequent word sense has a frequency of 40%, then a high accuracy score would be a much more significant result.

### 5.2.3 Precision and Recall

Search related or IR task can provide us with misleading accuracy. A model which labels every document as irrelevant will give us an accuracy score of 100% as can be seen in Figure 9, as the number of irrelevant documents outweighs the ones that are relevant for the task.

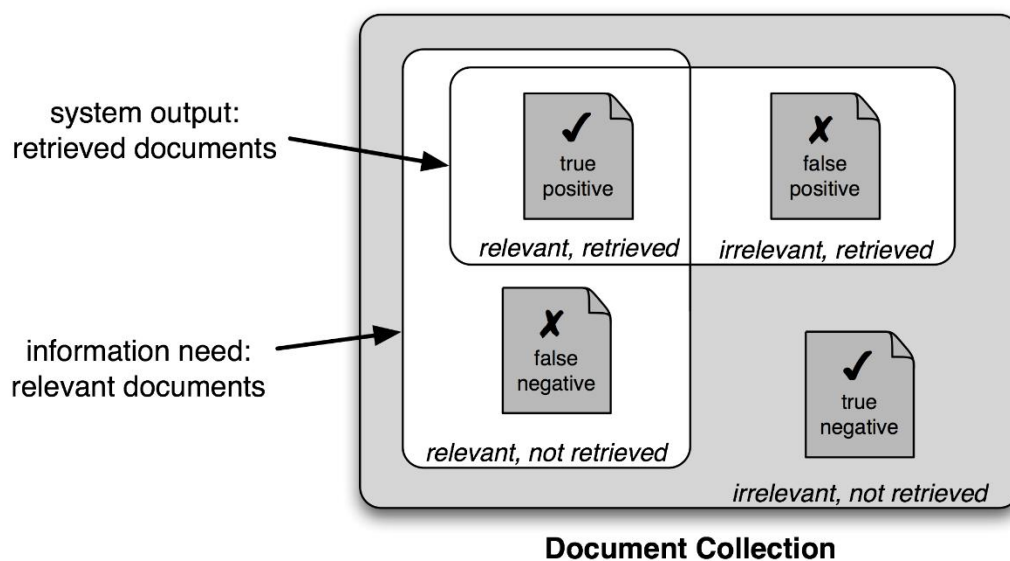


Figure 12 Precision and Recall (Source: NLTK, 2017)

Hence it is necessary to use a different set of measures for search tasks, based on the number of items in each of the four categories as shown in Figure 9:

- **True positives:** are relevant items that were correctly found as relevant.
- **True negatives:** are irrelevant items that were correctly found as irrelevant.
- **False positives:** are irrelevant items that were incorrectly found as relevant.
- **False negatives:** are relevant items that were incorrectly found as irrelevant.

We can define the following metrics based on the above four numbers:

- **Precision**, which shows how many of the items that were found were relevant, which is  $TP/(TP + FP)$ .
- **Recall**, which shows how many of the relevant items that were found, is  $TP/(TP + FN)$ .
- The **F-Score or F-Measure** combines the precision and recall giving a single score and is defined as the harmonic mean of the precision and recall:

$$(2 * Precision * Recall)/(Precision + Recall)$$

*Equation 2 F-Score or F-Measure Source: (NLTK, 2017)*

### 5.3 Typical workflow of sentiment analysis module

Having looked at evaluation measures that we will use in this thesis, we go on to consider the typical workflow of a sentiment analysis module (Zhang & Desouza, 2014).

- Step 1. Set up input parameters & extract raw text data (online reviews, blogs, Tweets, or other documents).
- Step 2. Process raw data:
  - Clean up text
  - Remove stop words
  - Stemming
  - Translate text into corpus matrices
- Step 3. Conduct a few classifiers to calculate the polarity of the formatted data.
- Step 4. Evaluate the accuracy and efficiency of each algorithm.
- Step 5. Produce outputs e.g. sentiment scores, spreadsheets, graphs.

### 5.4 Natural Language Toolkit (NLTK)

NLTK is a python module which offers many NLP processing tasks along with data types, corpora and readers, tutorials, animated algorithms and problem sets. Data types in NLTK include tags, chunks, trees, token and feature structures. Implementation for stemmers, taggers, parsers, chunkers, classifiers, clusterers and tokenizers are offered in interface definitions and reference. Corpus readers and samples include Chunking corpus, Brown Corpus, Treebank, SentiWordNet and CMU Pronunciation Dictionary. NLTK is ideally suited for learning or conducting research in NLP and has been used

successfully in prototyping platforms, building research systems and teaching or individual study tool. Python as a programming language has a minimal learning curve compared to other languages in the market and it allows users to explore via its interactive interpreter. Code in python can be encapsulated and reused easily, it has an extensive library and offers tools for graphical and numerical processing. NLTK has grown significantly as each new processing task added requirements on input and output data. As the numbers of tasks multiply, data management becomes more and more difficult. For more information, including documentation, download pointers, and links to courses that have adopted NLTK, please see: <http://nltk.sourceforge.net/> (Bird, 2006).

Next, we look at Natural Language Toolkit, a module which can be used to build a sentiment classifier and analyzer, and which was used in this research.

## 5.5 NLTK Processing Tasks

In this section, we look at some of the NLTK processing tasks that are used in every sentiment analysis process (Perkins, 2014).

### 5.5.1 Tokenization and Stemming

The following three lines of a program show how to import a tokenize package, define a text string and tokenize the string on whitespace to create a list of tokens. There are several other tokenizers which can be used.

```
text = 'This is a test.'  
list(tokenize.whitespace(text))  
['This', 'is', 'a', 'test.']
```

Then we can stem the output from tokenizer as follows:

```
text = 'stemming is exciting'  
tokens = tokenize.whitespace(text)  
porter = stem.Porter()  
for token in tokens:  
    print porter.stem(token)
```

**Output: stem word is excit**

The corpora included with NLTK come with corpus readers that understand the file structure of the corpus, and load the data into Python data structures. For example, the following code reads part of the Brown Corpus. It prints a list of tuples, where each tuple consists of a word and its tag.

```
for sent in brown.Tagged('a'):
```

```
    print sent
```

```
[(('The', 'at'), ('Fulton', 'np-tl'), ('County', 'nn-tl'), ('Grand', 'jj-tl'), ('Jury', 'nn-tl'), ('said', 'vbd'), ...]
```

NLTK also offers support for conditional frequency distributions, making it easy to count items of interest in specified contexts. Such information may be useful for studies in stylistics or in text categorization.

**5.5.2 Tagging**

The simplest possible tagger assigns the same tag to each token (Perkins, 2014):

```
my_tagger = tag.Default('nn')
```

```
list(my_tagger.tag(tokens))
```

```
[('John', 'nn'), ('saw', 'nn'), ('3', 'nn'), ('polar', 'nn'), ('bears', 'nn'), ('.', 'nn')]
```

Simple tagger will tag only 10–20% of the tokens correctly. However, it is a reasonable tagger to use as a default if a more advanced tagger fails to determine a token's tag.

The regular expression tagger assigns a tag to a token per a series of string patterns. For instance, the following tagger assigns *cd* to cardinal numbers, *nns* (nouns) to words ending in the letter *s*, and *nn* (noun) to everything else:

```
patterns = [
```

```
(r'\d+(\.\d+)?$', 'cd'),
```

```
(r'\.*s$', 'nns'),
```

```
(r'.*', 'nn')]
```

```
simple_tagger = tag.Regexp(patterns)
```

```
list(simple_tagger.tag(tokens))
```

```
[('John', 'nn'), ('saw', 'nn'), ('3', 'cd'), ('polar', 'nn'), ('bears', 'nns'), ('.', 'nn')]
```

The `tag.Unigram` class implements a simple statistical tagging algorithm where it assigns the tag for every token. For example, it will assign the tag `jj` to any occurrence of the word `frequent`, since `frequent` is used as an adjective (e.g. a frequent word) more often than it is used as a verb (e.g. I frequent this cafe). Before a unigram tagger can be used, it must be trained on a corpus, as shown below for the first section of the Brown Corpus.

```
unigram_tagger = tag.Unigram()
unigram_tagger.train(brown('a'))
```

Once a unigram tagger has been trained, it can be used to tag new text. Note that it assigns the default tag `'None'` to any token that was not encountered during training.

```
text = "John saw the books on the table"
tokens = list(tokenize.whitespace(text))
list(unigram_tagger.tag(tokens))
[('John', 'np'), ('saw', 'vbd'), ('the', 'at'), ('books', None), ('on', 'in'), ('the', 'at'), ('table', None)]
```

We can instruct the unigram tagger to `'back off'` to our default `simple_tagger` when it cannot assign a tag itself. Now all the words are guaranteed to be tagged:

```
unigram_tagger = tag.Unigram(backoff=simple_tagger)
unigram_tagger.train(train_sents)
list(unigram_tagger.tag(tokens))
[('John', 'np'), ('saw', 'vbd'), ('the', 'at'), ('books', 'nns'), ('on', 'in'), ('the', 'at'), ('table', 'nn')]
```

We can go on to define and train a bigram tagger, as shown below:

```
bigram_tagger = tag.Bigram(backoff=unigram_tagger)
bigram_tagger.train(brown.tagged('a'))
```

We can easily evaluate this tagger against some gold-standard tagged text, using the `tag.accuracy()` function. NLTK also includes a Brill tagger and an HMMtagger.

### 5.5.3 Chunking and Parsing

Chunking is a technique for shallow syntactic analysis of text. Chunk data can be loaded from files that use the common bracket or IOB notations. We can define a regular-expression based chunk parser for use in chunking tagged text. NLTK also supports simple cascading of chunk parsers. Corpus readers for chunked data in Penn Treebank and CoNLL-2000 are provided in NLTK, along with comprehensive support for evaluation and error analysis.

NLTK offers several parsers for context-free phrase-structure grammars. Grammars can be defined using a series of productions as follows (Perkins, 2014):

```
grammar = cfg.parse_grammar("""
S -> NP VP
VP -> V NP | V NP PP
V -> "saw" | "ate"
NP -> "John" | Det N | Det N PP
Det -> "a" | "an" | "the" | "my"
N -> "dog" | "cat" | "ball"
PP -> P NP
P -> "on" | "by" | "with"
""")
```

Now we can tokenize and parse a sentence with a recursive descent parser. Note that we avoided left-recursive productions in the above grammar, so that this parser does not get into an infinite loop.

```
text = "John saw a cat with my ball"
sent = list(tokenize.whitespace(text))
rd = parse.RecursiveDescent(grammar)
```

Now we apply it to our sentence, and iterate over all the parses that it generates. Observe that two parses are possible, due to prepositional phrase attachment ambiguity.

```
for p in rd.get_parse_list(sent):
    print p
    (S:
    (NP: 'John')
```

```
(VP:  
(V: 'saw')  
(NP:  
(Det: 'a')  
(N: 'cat')  
(PP: (P: 'with')  
(NP: (Det: 'my') (N: 'ball'))))))  
(S:  
(NP: 'John')  
(VP:  
(V: 'saw')  
(NP: (Det: 'a') (N: 'cat'))  
(PP: (P: 'with')  
(NP: (Det: 'my') (N: 'ball'))))))
```

The same sentence can be parsed using a grammar with left-recursive productions, so long as we use a chart parser. We can invoke NLTK's chart parser with a bottom-up rule-invocation strategy with `chart.ChartParse(grammar, chart.BU STRATEGY)`. Tracing can be turned on to display each step of the process. NLTK also supports probabilistic context free grammars, and offers a Viterbi-style PCFG parser, together with a suite of bottom-up probabilistic chart parsers.

## 5.6 Classification Algorithms

Classification is the process to find the properties that help us find the group to which each word or case belongs. There are two methods of finding the semantic orientation which helps find the polarity of the sentence: supervised and unsupervised classification techniques. For this thesis, we will be experimenting or creating our own corpus from the training document which will help us classify the test document. Hence, a supervised classification algorithm seems fit for this thesis. Most common among the supervised algorithm for sentiment analysis is the Naïve Bayes and Support Vector machine. Naïve Bayes is the simplest yet effective supervised classification algorithm and is most widely used (Perkins, 2014).



### 5.6.1 Naïve Bayes Algorithm

Naive Bayes classification technique is based on Bayesian theorem and is particularly suited when the dimensionality of inputs is high. Let  $R = \{R_1, R_2, R_3, \dots, R_n\}$  denotes the set of training opinions, where each opinion is labeled with one of the coding in  $C = \{P, N, O\}$ . Given some new opinion, the aim is to estimate the probability of each code. Using Bayes formula,

$$p\left(\frac{c}{r}\right) = \frac{p\left(\frac{r}{c}\right)p(c)}{p(r)}$$

*Equation 3 Bayes Formula Bayes Formula (Source: (Medagoda et al., 2015))*

We are interested in the relative order of codes for a given opinion R, if  $p(r)$  is independent of codes, then we can consider

$$p\left(\frac{c}{r}\right) = p\left(\frac{r}{c}\right)p(c)$$

If F is the ordered sequence of the features that compose the opinion R then  $F = \{w_1, w_2, w_3, w_4\}$  Where,  $w_1$  and  $w_3$  are adjective positive score and  $w_2$  and  $w_4$  are adjective negative score

$$p\left(\frac{c}{r}\right) = p\left(\frac{r}{c}\right)p(c) = p(c) \prod_{k=1}^p p(w_k/c)$$

And classify r into the most possible code c using

$$\arg \max_c p\left(\frac{c}{r}\right)$$

(Medagoda et al., 2015)

### 5.6.2 Support Vector Machine (SVM)

SVM is the best binary classification method and a non-probabilistic classification technique that looks for a hyperplane with the maximum margin between the positive and negative examples of the training opinions. Support Vector Machines are based on the concept of decision planes that define decision boundaries. A decision plane is one that forms a separation between a set of objects, which have different class memberships (Medagoda et al., 2015).

Decision planes are the classifiers either a line or a curve. A simple classifier may use linear decision planes rather than more complex structures. Classification tasks based on drawing separating lines to distinguish between objects of different class memberships are known as hyperplane classifiers. SVM is primarily a classification method that performs classification tasks by constructing hyperplanes in a multidimensional space that separates cases of different class labels. SVM supports both regression and classification tasks and it can handle multiple continuous and categorical variables (Medagoda et al., 2015).

To construct an optimal hyperplane, SVM employs an iterative training algorithm; this is used to minimize an error function. According to the form of the error function, SVM models can be classified into distinct groups. In the simplest SVM, training involves the minimization of the error function (Medagoda et al., 2015).

$$\frac{1}{2} w^T w + C \sum_{i=0}^n \xi_i$$

*Equation 4 SVM Error Function Source: (Medagoda et al., 2015)*

Subject to the constraints

$$y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0, i = 1, \dots, N$$

Where C is the capacity constant w is the vector of coefficients, b, a constant and  $\xi$  are parameters for handling non-separable data (inputs). The index i labels the N training cases. Note that y ( $\in \pm 1$ ) is the class label and  $x_i$  is the independent variables. The kernel  $\phi$  is used to transform data from the input (independent) to the feature space. It should be noted that the larger the C, the more the error is penalized. Thus, C should be chosen with care to avoid over fitting. It is suggested in that SVM does not depend on the dimensionality of the problem when compared with other machine learning methods. The success of SVM in text categorization lies in its automatic capacity tuning by minimizing, i.e. the extraction of a small number of support vectors from the training data that are relevant for the classification (Medagoda et al., 2015).

### 5.6.3 Decision Tree Classification Algorithm

Decision tree is a graph with branches that represent every possible outcome of a decision. The rules produced by a decision tree model are human readable and are easily interpretable. The classification task using decision tree technique can be performed without complicated computations and the technique can be used for both continuous and categorical variables. In this work, a decision tree model was tested to classify comments broken down to Positive, Negative or Neutral and then the rules generated by the decision trees were investigated (Medagoda et al., 2015).

### 5.6.4 Decision Tree Algorithm – J48

J48 is a univariate decision tree classification method which creates trees based on the information gain. It tests whether all cases belong to the same class; if true then the tree is a leaf and is labeled as a class. Next for each attribute calculate the information gain. The information gain can be calculated as (Medagoda et al., 2015)

$$Gain(p, j) = Entropy(p) - entropy\left(\frac{i}{p}\right)$$

*Equation 5 J48 Information Gain Source: (Medagoda et al., 2015)*

Where,

$$Entropy\left(\frac{j}{p}\right) = \frac{pj}{p} \log pj/p$$

Finally, find the best fitting attribute based on the current selection criteria. Once the initial tree is constructed using the entropy then pruning is carried out to remove the

## 5.7 Summary

In this chapter, we discussed the evaluation methods that can be applied to a classification model and a typical sentiment analysis workflow. Then we looked at Natural Language Toolkit, a python module, which is widely used for studies and research in sentiment analysis and its classification algorithms, after which we looked at some of the supervised algorithms available in NLTK which we use in this research.

# Chapter 6: Results

## 6.1 Introduction

In this chapter, we discuss the custom tool framework and compare with some of the commercially available tools from Microsoft, IBM and Google and then finally how to start up the tool to create custom corpus and obtain the results. The tool we create will be used to study the different components involved in the process of sentiment analysis and this will help us understand the rigorous process of creating a commercially viable tool, while addressing outliers and overfitting.

## 6.2 Framework

To create a workable solution, we use a lexicon-based method which is based on SentiWordNet and we use aspect level sentiment analysis to obtain positive and negative features of any given text or we can train and test using a classification algorithm. The main aim is to find and extract the features to be analyzed and calculate its polarity. The steps we perform are discussed in the following subsections. The code used to perform the experiments is presented in the Appendix.

### 6.2.1 Data Collection

Designing a dataset is the first step in opinion mining where opinions are collected from various sources like reviews, blogs etc. of a domain. For the thesis, we use secondary data—an excel spreadsheet provided by the Health Centre, which has review comments from patients about doctors, along with other demographic information like patient and doctor gender, and so on. Review comments from patients are in column #23 of the spreadsheet and the first row contains the header for the column.

Therefore, data collection for this thesis is effectively done by Health Centre by means of an online review or by asking patients to submit paper forms at the Health Centre reception.

### **6.2.2 Data Preprocessing**

Dataset is cleaned and preprocessed and some common steps include removing non-textual contents and HTML tags and removing information about the reviews that are not needed for sentiment analysis, such as review dates and reviewers' names.

At this stage, for the tool created, headers for the columns are ignored as they do not convey any sentiment which would add up to the sentiment polarity of the document. We make sure that stop words are removed and the correct column which contains patient reviews is chosen for calculating sentiment of the document.

### **6.2.3 Feature Extraction**

Identification and selection of features is perhaps the most important task of opinion mining. There can be more than one name for the same aspect, for example “story of the book is good” or someone else may use “the storyline of the book is fantastic”, where story and storyline have the same meaning.

This stage of the framework is taken care of by the tool, where the tool is trained by a movie review corpus and a custom corpus in turn, which will train the tool in identification and selection of the features.

### **6.2.4 POS Tagging**

POS tagger parses a sentence or document and tags each term with its part of speech. For POS tagging we used the Stanford POS tagger. This tagger is used to split text data into sentences and to produce the part-of-speech tag for each word (whether the word is a noun, verb, adjective, etc.). The following shows a sentence that is parsed and POS tags applied.

“The feel of the phone is the best of the series.”

When we apply the POS-tagger, it generates the following parts of speech for the sentence.

“The\_DT feel\_NN of\_IN the\_DT phone\_NN is\_VBZ the\_DT best\_JJS of\_IN the\_DT series\_NN.”

This would be one of the most important stages of the tool, where the test set is tokenized and POS tagged for sentiment analysis. NLTK module for tokenizing and POS tagging is used to perform this part of the framework. See Appendix.

### **6.2.5 Calculate Sentiment Polarity**

At this stage, we calculate the sentiment polarity using SentiWordNet which is a lexical resource used for opinion mining. We can also train and test using classification algorithms. SentiWordNet synset has three scores: Positive, negative and objective, which tells us how positive or negative or objective the term in the synset is. Each of the three is assigned a score from 0.0 to 1.0 and their sum is 1.0 for each synset and the entries contain the parts of speech category of the displayed entry, its positivity, its negativity, and the list of synonyms. The word presented in the form of lemma #sense-number, where the first sense corresponds to the most frequent use of the word and the different word senses can have different polarities.

In the second approach, we use SentiWordNet to check the polarity of the test sentences and create a custom corpus which we use for sentiment analysis prediction. This, we estimate, will help achieve better accuracy in analyzing the sentiments.

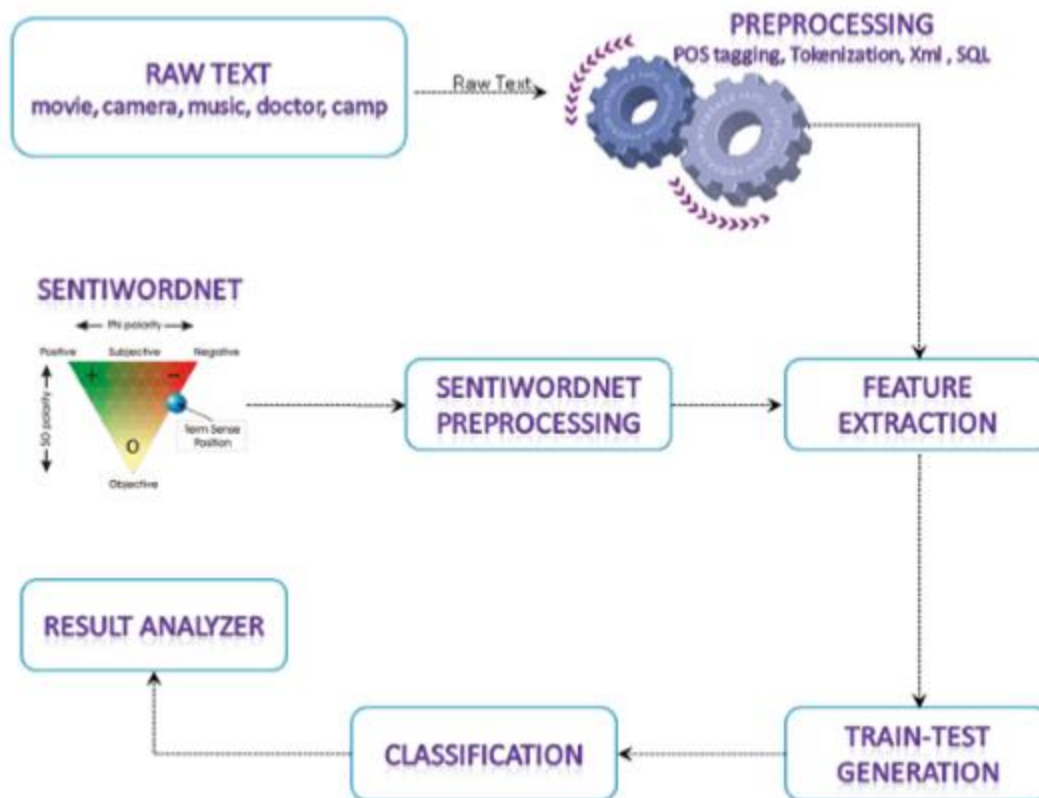
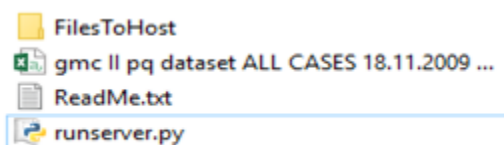


Figure 13 Sentiment Analysis using SentiWordNet (Source: (Amiri & Chua, 2012))

### 6.3 Project Setup Instructions

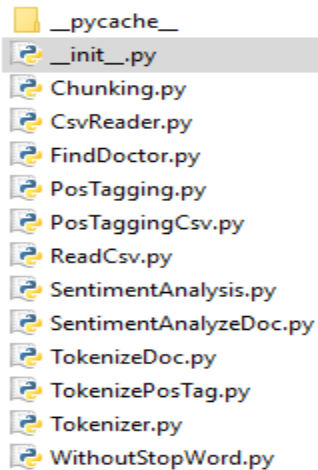
1. Install latest Python from <https://www.python.org/> and NPM (install Nodejs which installs NPM on your machine)
2. Install NLTK and dependencies from NLTK website (<http://www.nltk.org/install.html>)
3. Install FLASK and FLASK CORS latest version (<http://flask.pocoo.org/docs/0.10/installation/>)
4. Create a folder structure as shown below.

Note that the data set is contained in gmc II pq dataset.csv file.



Runserver.py is used to run flask server and access methods as services such as reading csv files, tokens, tagging, classification and ascertaining document sentiment. Currently all services are programmed to be accessed as 'GET' methods.

\_init\_.py file inside the 'FilesToHost' folder is first executed when Flask server is started.



Start the 'Flask' server by navigating to the folder where we have RunServer.py file and open command prompt and type in python.exe runserver.py

File paths may change based on the location of files on your machine.

Once the server starts we will see the following message on the command prompt.

```
D:\College Studies\MasterThesis\SentimentAnalyzer\FlaskSample>C:\Python34\python.exe "D:\College Studies\MasterThesis\SentimentAnalyzer\FlaskSample\runserver.py"
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger pin code: 296-742-626
```


To access any 'GET' service type in [http://127.0.0.1:5000/readcsv/gmc II pq dataset ALL CASES 18.11.2009 CW.csv](http://127.0.0.1:5000/readcsv/gmc%20pq%20dataset%20ALL%20CASES%2018.11.2009%20CW.csv) in your favorite browser and if the service works you would see the comments in the file as shown: `
















["All my treatment is first class.", "Very courteous and helpful.", "This doctor is very down to Earth", "Very understanding and patient doctor.", "This doctor is always my first choice. Keep up the good work every time I've attended the clinic.", "Very happy with the care and treatment for my child's condition.", "As always", "The doctor was very good. We didn't feel rushed", "Excellent manner", "One of the first visit.", "Took time to listen and explain - not rushed.", "Always received excellent treatment from this doctor and treated with great respect on all visits.", "This doctor is excellent. He consistently puts the world.", "Nearly ran out of medication", "A very caring and understanding consultant that put me completely at ease.", "Excellent", "Had to ask to be put in this doctor's clinic - was no problem but I had already been to.", "The doctor appeared to give my eyes a thorough examination and concluded that no treatment was required at this time.", "I was extremely impressed with how the doctor put me at ease after appointment without my own GP. I had to wait a long time and that is very disappointing.", "2", "I have seen the doctor for many years and have never had a complaint.", "I was very happy with the time taken is so soon in the new year.", "He had a very calm and pleasant manner which was both instantly reassuring and empathetic throughout.", "I was still drowsy when this doctor came to see me after my operation thought that this doctor was a lovely man and explained everything very well to me. He was so polite and made me feel comfortable - I thought he was wonderful.", "This consultant has the most delightful cc very polite and courteous and very willing to listen and answer questions. A most pleasant man.", "The doctor saw me for a specific reason following referral from my own doctor.", "I saw this doctor for the first appointment for a post-operation assessment but not with this doctor.", "The doctor made me feel at ease when I was upset by another doctor. I was really happy with their decision and really excited about feeling 2", "I was very pleasantly surprised after previous poor experiences with the other doctors. I cannot recommend this doctor enough. the doctor is an excellent doctor with a wonderful bedside manner.", "I saw at this practice."]














### Creating the Custom Corpus

We create two folders first programmatically

 pos	5/8/2016 2:18 AM	File folder
 neg	5/8/2016 2:11 AM	File folder

We then feed the program with sample csv files containing our test data, created from the dataset, which can be used to train the classifier and create a corpus which can be used for future analysis. The Appendix contains the code used to create the custom corpus.

 pos2016-05-08-01-45-56.txt	5/8/2016 1:45 AM	Text Document	1 KB
 pos2016-05-08-01-45-57.txt	5/8/2016 1:45 AM	Text Document	1 KB
 pos2016-05-08-01-45-58.txt	5/8/2016 1:45 AM	Text Document	1 KB
 pos2016-05-08-01-45-59.txt	5/8/2016 1:45 AM	Text Document	1 KB
 pos2016-05-08-01-46-00.txt	5/8/2016 1:46 AM	Text Document	1 KB
 pos2016-05-08-01-46-01.txt	5/8/2016 1:46 AM	Text Document	1 KB
 pos2016-05-08-01-46-02.txt	5/8/2016 1:46 AM	Text Document	1 KB
 pos2016-05-08-01-46-03.txt	5/8/2016 1:46 AM	Text Document	1 KB
 pos2016-05-08-01-46-04.txt	5/8/2016 1:46 AM	Text Document	1 KB
 pos2016-05-08-01-46-05.txt	5/8/2016 1:46 AM	Text Document	1 KB
 pos2016-05-08-01-46-06.txt	5/8/2016 1:46 AM	Text Document	1 KB
 pos2016-05-08-01-46-07.txt	5/8/2016 1:46 AM	Text Document	1 KB
 pos2016-05-08-01-46-08.txt	5/8/2016 1:46 AM	Text Document	1 KB

 neg2016-05-08-01-45-56.txt	5/8/2016 1:45 AM	Text Document	1 KB
 neg2016-05-08-01-45-59.txt	5/8/2016 1:45 AM	Text Document	1 KB
 neg2016-05-08-01-46-03.txt	5/8/2016 1:46 AM	Text Document	1 KB
 neg2016-05-08-01-46-04.txt	5/8/2016 1:46 AM	Text Document	1 KB
 neg2016-05-08-01-46-09.txt	5/8/2016 1:46 AM	Text Document	1 KB
 neg2016-05-08-01-46-12.txt	5/8/2016 1:46 AM	Text Document	1 KB
 neg2016-05-08-01-46-15.txt	5/8/2016 1:46 AM	Text Document	1 KB
 neg2016-05-08-01-46-17.txt	5/8/2016 1:46 AM	Text Document	1 KB
 neg2016-05-08-01-46-19.txt	5/8/2016 1:46 AM	Text Document	1 KB
 neg2016-05-08-01-46-20.txt	5/8/2016 1:46 AM	Text Document	1 KB
 neg2016-05-08-01-46-21.txt	5/8/2016 1:46 AM	Text Document	1 KB
 neg2016-05-08-01-46-22.txt	5/8/2016 1:46 AM	Text Document	1 KB
 neg2016-05-08-01-46-28.txt	5/8/2016 1:46 AM	Text Document	1 KB

## 6.4 Results

After multiple runs on the test data set, we achieved the following result sets.

- 1. Using Naive Bayes Classifier:** The following result was obtained after creating a custom corpus and running the tool to analyze sentiments on the questionnaire using Naive Bayes Classifier.

### 1<sup>st</sup> run

Document sentiment	Positive
Accuracy	0.935
positive precision	0.959
positive recall	0.958
negative precision	0.847
negative recall	0.850

### 2<sup>nd</sup> run

Document sentiment	Positive
Accuracy	0.939
positive precision	0.942
positive recall	0.982

negative precision	0.924
negative recall	0.785

As we can see, we get a very high precision and high recall, which tells us that the classifier used is very good. Training set used trained the classifier which helped us classify the document with good accuracy. We can say this is almost an ideal test result one should expect in almost all situations to ensure document is classified correctly and proper results are obtained.

**2. Using SentiwordNet:** The tool was programmed to run and analyze the questionnaire directly using SentiwordNet and after multiple runs the document was analyzed as positive. SentiwordNet has positive and negative index values for a word which we use to categorize every word used in the document and calculate the overall sentiment of the document.

**3. Using movie review corpus:** In this test run we use movie review corpus, which is widely used to classify natural languages and is a huge collection of natural language sentences.

**1<sup>st</sup> run**

Document sentiment	Positive
Accuracy	0.967
positive precision	0.938
positive recall	1.0
negative precision	1.0
negative recall	0.935

**2<sup>nd</sup> run**

Document sentiment	Positive
Accuracy	0.967
positive precision	0.938

positive recall	1.0
negative precision	1.0
negative recall	0.935

Comparing this run to the previous runs with custom corpus, we obtain good results. This tells us that, we were able to create a good quality custom corpus which gives us precision and recall as close to a research quality corpus.

**4. Using Support Vector Machine:** The document was classified as negative using Support Vector Machine with Linear Support Classification variation after both creating a custom corpus, and using movie review and running the tool to analyze sentiments.

SVC and NuSVC can also be used as variations with SVM.

We were not able to evaluate the results on using SVM; we could not obtain the accuracy, precision and recall as it requires usage of sci-kit, which would push the timelines for this research. This can however be undertaken as a separate research topic.

In conclusion, with the direct use of SentiWordNet, the document is classified as a positive document. However, when using corpora, we get a contrasting result. When executing the application using Bayes and custom corpus the document is classified as positive. When using SVM with both the custom corpus and movie corpus, the document is classified as negative.

## 6.5 Summary

In this chapter, we looked at the framework which we used in the creation of sentiment analysis tools. We then looked at the steps to set up the project and finally presented the results obtained.

The results obtained show us that while we could classify the document as a positive or a negative document, we would have obtained better results if we had the time to fine tune the application and use expert help in creating the training and test sets. However, the results obtained are promising within the time constraints of this research.

# Chapter 7: Review, limitations, and future research

## 7.1 Introduction

In this Chapter, we discuss the results obtained and the implications, followed by the limitations of the tool and the changes that it needs to be a commercial viable tool. Finally, we consider about the future work or research that can be carried out to take this research further, or generally further the research in this area.

## 7.2 Research Review and Summary

This thesis set out to investigate the question:

1. How can an effective sentiment analysis tool be built to analyze free text comments in questionnaires?

And a supplementary question:

2. Will a corpus created from the data be useful in analyzing sentiment expressed in the data?

We built a basic sentiment analysis tool which can use SentiWordNet or custom corpus or movie reviews corpus to analyze sentiments in free text responses.

The software tool at this stage needs fine tuning in the following areas:

1. Improve and increase the size of custom corpus by using more test to obtain good precision and recall.
2. This would also provide us better precision and recall scores.
3. Consider the demographic information available and create graphs or visual representation for different parameters like gender, age of the patient and doctors.

Throughout the thesis, we have tried to study, research and understand different components that we need to create a simple but effective sentiment analysis and classification tool.

Literature review was conducted to understand sentiment analysis and its process and sentiment analysis methods and tools. In this Chapter, we looked at sentiment analysis from the medical web, biomedical literature and other medical texts, which shows us that SVM classifier is widely used for classification of medical field data (refer Section 2.4). For the tool, we created a classifier using Naive Bayes and SVM. Though at this stage the tool requires further improvement, we were able to successfully create a tool which would tell us the overall sentiment conveyed by the patient review document (response). Further this tool can also be used as a data mining tool that would help us filter data by patient or doctor's age or sex.

Next, we briefly looked at N-Gram which is another effective approach to study sentiments. Although we did not directly use N-grams in our research, this can be an alternative we can add to the tool giving the user a choice of different approaches.

After discussing the fundamental concepts needed to understand the different processes in sentiment analysis in chapter 3, we look at SentiWordNet, which is mainly used to calculate sentiment polarity (positive, negative or neutral). We use SentiWordNet to create our custom corpus to help us obtain better precision with classification tasks. We also used SentiWordNet to create an alternative approach to analyze document sentiments and we achieved a positive result. After SentiWordNet, it was necessary to study negation identification in sentiment analysis, which is one of the most complex tasks. Pseudocode provided was used in the tool to consider negative words so that their polarity of the negative word can be reversed. Code used in this research is presented in the Appendix.

In Chapter 5, we discussed about NLTK which is one of the widely used NLP toolkit. We look at the different processing task and classification algorithms which we applied in our tool. We presented a framework and project setup instructions, which can be used to effectively create a custom NLP tool.

To summarize, following are the major steps involved in sentiment analysis:

1. Data Collection
2. Data Preprocessing

3. Feature Extraction
4. POS Tagging
5. Calculate Sentiment Polarity

While the tool developed needs more training and testing before it can be a practical solution for future work, we have successfully developed a platform which can be used for generating custom corpus and provide anyone with an understanding to help them start to develop commercial sentiment analysis tools.

### **7.3 Limitations**

The tool developed as a part of this thesis, provides us with ample opportunities to learn about the process of developing a commercially feasible sentiment analysis tool but when compared to Microsoft, IBM or Google sentiment analysis API's or tools in the market, it has limitations which can be tackled or can be overcome over a period of time.

1. NLTK, python module used for creating the tool, as mentioned in (Bird, 2006), is ideally suited for learning or conducting research in NLP and has been used successfully in prototyping platforms, building research systems and teaching or individual study tool.

However, to build a commercial tool we cannot wholly rely on NLTK module to provide us with solutions to perform different processes. A mix of system and programming languages need to be used to create a platform which can hopefully provide better results.

2. We have not included any mechanism to detect fake or duplicate review or to check the reputation of the reviewer.
3. As previously stated, we could not obtain the accuracy, precision and recall scores when using SVM, due to the time constraints, as it requires usage of sci-kit.

4. As mentioned in Section 2.8, NLP needs more enhancement with respect to domain-dependent sentiment analysis, which would need considerable changes to the underlying NLTK module used.
5. SentiWordNet, although a good general-purpose lexicon database for use in sentiment analysis in English, researchers face a challenge in building lexicons, corpora and dictionaries for any other language.

## 7.4 Future Work or Research

The tool developed reads patient review comments about doctors which can be extended to comments collected from social media and other sources about the doctor or hospital.

1. We have created a corpus which is used to further classify the document; this can be extended to create an elaborate domain dependent lexicon database which will help classify negative sentiments in future.
2. Visualization is one of the main aspects of any data mining program. Popular libraries like D3.js can be used to present data extracted in form of graphs.
3. It would be interesting to use sci-kit with NLTK in order to experiment with SVM.
4. Different algorithms along with their variations can be studied to improve the performance and accuracy of the results obtained. In this thesis, we have mostly looked at using supervised algorithms like Naïve Bayes or SVM. Future work can be done on usage of unsupervised classification algorithms.
5. Different stages of data mining or NLP can be studied and improved by using different algorithms than the ones used by default in NLTK.
6. As mentioned above different components can be studied and improved, and this can be done by using more efficient programming languages which can be used to replace NLTK module or create tools which would give results faster or help create better solution.



7. API solutions offered by Microsoft, IBM and Google, which can be good substitute for NLTK to understand the sentiments expressed by reviewers.

# References

- Amiri, H., & Chua, T.-S. (2012). Sentiment Classification Using the Meaning of Words Symposium conducted at the meeting of the Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence
- Asghar, M. Z., Khan, A., Ahmad, S., Qasim, M., & Khan, I. A. (2017). Lexicon-enhanced sentiment analysis framework using rule-based classification scheme. *PLOS ONE*, 12(2), e0171649. doi:10.1371/journal.pone.0171649
- Asmi, A., & Ishaya, T. (2012). Negation identification and calculation in sentiment analysis Symposium conducted at the meeting of the The Second International Conference on Advances in Information Mining and Management
- Baccianella, S., Esuli, A., & Sebastiani, F. (2010). SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining Symposium conducted at the meeting of the LREC
- Baitharu, T. R., & Pani, S. K. (2016). Analysis of Data Mining Techniques for Healthcare Decision Support System Using Liver Disorder Dataset. *Procedia Computer Science*, 85, 862-870.
- Banerjee, S., & Pedersen, T. (2003). The design, implementation, and use of the ngram statistics packageSpringer. Symposium conducted at the meeting of the CICLing
- Bird, S. (2006). NLTK: the natural language toolkitAssociation for Computational Linguistics. Symposium conducted at the meeting of the Proceedings of the COLING/ACL on Interactive presentation sessions
- Biyani, P., Caragea, C., Mitra, P., Zhou, C., Yen, J., Greer, G. E., & Portier, K. (2013). Co-training over domain-independent and domain-dependent features for sentiment analysis of an online cancer support communityACM. Symposium conducted at the meeting of the Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining
- Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan), 993-1022.
- Boyd-Graber, J., Fellbaum, C., Osherson, D., & Schapire, R. (2006). Adding dense, weighted connections to WordNet Symposium conducted at the meeting of the Proceedings of the third international WordNet conference
- Cambria, E., Benson, T., Eckl, C., & Hussain, A. (2012). Sentic PROMs: Application of sentic computing to the development of a novel unified framework for measuring health-care quality. *Expert Systems with Applications*, 39(12), 10533-10543.
- Cavnar, W. B., & Trenkle, J. M. (1994). N-gram-based text categorization. *Ann Arbor MI*, 48113(2), 161-175.
- Chandni, Chandra, N., Pahade, R., & Gupta, S. (2015). Sentiment Analysis and its Challenges.
- Chau, M., Lu, Y., Fang, X., & Yang, C. C. (2009). Characteristics of character usage in Chinese Web searching. *Information Processing & Management*, 45(1), 115-130.
- Chowdhury, G. G. (2003). Natural language processing. *Annual review of information science and technology*, 37(1), 51-89.
- Cieliebak, M., Dürr, O., & Uzdilli, F. (2013). Potential and Limitations of Commercial Sentiment Detection Tools Symposium conducted at the meeting of the ESSEM@ AI\* IA

- Clark, A., Fox, C., & Lappin, S. (2010). The handbook of computational linguistics and natural language processing (Vol. 57): Wiley. com.
- Collomb, A., Costea, C., Joyeux, D., Hasan, O., & Brunie, L. (2014). A study and comparison of sentiment analysis methods for reputation evaluation. Rapport de recherche RR-LIRIS-2014-002.
- Cowie, J., & Lehnert, W. (1996). Information extraction. *Communications of the ACM*, 39(1), 80-91.
- Damashek, M. (1995). Gauging similarity with n-grams: Language-independent categorization of text. *Science*, 267(5199), 843.
- de Albornoz, J., Plaza, L., Gervás, P., & Díaz, A. (2011). A joint model of feature mining and sentiment analysis for product review rating. *Advances in information retrieval*, 55-66.
- Esuli, A., & Sebastiani, F. (2006). Determining Term Subjectivity and Term Orientation for Opinion Mining Symposium conducted at the meeting of the EACL
- Eugenio, B. D. (2005). Discourse processing. Retrieved 26/10/2013, 2013, from [http://www.credoreference.com.ezproxy.aut.ac.nz/entry/wileycs/discourse\\_processing](http://www.credoreference.com.ezproxy.aut.ac.nz/entry/wileycs/discourse_processing)
- Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI magazine*, 17(3), 37.
- G. Miner, D. D., J. Elder, A. Fast, T. Hill, and R. Nisbet, Elsevier. (January 2012). The Seven Practice Areas of Text Analytics.
- Gencosman, B. C., Ozmutlu, H. C., & Ozmutlu, S. (2014). Character n-gram application for automatic new topic identification. *Information Processing & Management*, 50(6), 821-856.
- Gonçalves, P., Araújo, M., Benevenuto, F., & Cha, M. (2013). Comparing and combining sentiment analysis methodsACM. Symposium conducted at the meeting of the Proceedings of the first ACM conference on Online social networks
- Hardeniya, N., Perkins, J., Chopra, D., Joshi, N., & Mathur, I. (2016). Natural Language Processing: Python and NLTK. Birmingham, UNKNOWN: Packt Publishing. Retrieved from <http://ebookcentral.proquest.com/lib/aut/detail.action?docID=4747560>
- Huang, X., Peng, F., Schuurmans, D., Cercone, N., & Robertson, S. E. (2003). Applying machine learning to text segmentation for information retrieval. *Information Retrieval*, 6(3), 333-362.
- Hung, C., & Chen, S.-J. (2016). Word sense disambiguation based sentiment lexicons for sentiment classification. *Knowledge-Based Systems*, 110, 224-232.
- Hussein, D. M. E.-D. M. (2016). A survey on sentiment analysis challenges. *Journal of King Saud University-Engineering Sciences*.
- Kanaris, I., & Stamatatos, E. (2007). Webpage genre identification using variable-length character n-gramsIEEE. Symposium conducted at the meeting of the Tools with Artificial Intelligence, 2007. ICTAI 2007. 19th IEEE International Conference on
- Khan, A., Baharudin, B., & Khan, K. (2011). Sentiment classification from online customer reviews using lexical contextual sentence structureSpringer. Symposium conducted at the meeting of the International Conference on Software Engineering and Computer Systems

## References

- Kreutzer, J., & Witte, N. (2013). Opinion Mining Using SentiWordNet. Uppsala University.
- Kurgan, L. A., & Musilek, P. (2006). A survey of Knowledge Discovery and Data Mining process models. *The Knowledge Engineering Review*, 21(1), 1-24.
- Lewis, D. D., & Ringuette, M. (1994). A comparison of two learning algorithms for text categorization. *CiteSeer*. Symposium conducted at the meeting of the Third annual symposium on document analysis and information retrieval
- Liddy, E. D. (1990). Anaphora in natural language processing and information retrieval. *Information processing & management*, 26(1), 39-52.
- Liu, H., & Kešelj, V. (2007). Combined mining of Web server logs and web contents for classifying user navigation patterns and predicting users' future requests. *Data & Knowledge Engineering*, 61(2), 304-330.
- Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to information retrieval (Vol. 1): Cambridge University Press Cambridge.
- McNamee, P., & Mayfield, J. (2004). Character n-gram tokenization for European language text retrieval. *Information Retrieval*, 7(1), 73-97.
- Medagoda, N., Shanmuganathan, S., & Whalley, J. (2015). Sentiment lexicon construction using SentiWordNet 3.0. *IEEE. Symposium conducted at the meeting of the Natural Computation (ICNC), 2015 11th International Conference on*
- Medhat, W., Hassan, A., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, 5(4), 1093-1113.
- Melzi, S., Abdaoui, A., Azé, J., Bringay, S., Poncelet, P., & Galtier, F. (2014). Patient's rationale: Patient Knowledge retrieval from health forums. *Symposium conducted at the meeting of the eTELEMED: eHealth, Telemedicine, and Social Medicine*
- Moghaddam, S., & Ester, M. (2010). Opinion digger: an unsupervised opinion miner from unstructured product reviews. *ACM. Symposium conducted at the meeting of the Proceedings of the 19th ACM international conference on Information and knowledge management*
- Moghaddam, S., & Ester, M. (2011). ILDA: interdependent LDA model for learning latent aspects and their ratings from online product reviews. *ACM. Symposium conducted at the meeting of the Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*
- Mudinas, A., Zhang, D., & Levene, M. (2012). Combining lexicon and learning based approaches for concept-level sentiment analysis. *ACM. Symposium conducted at the meeting of the Proceedings of the First International Workshop on Issues of Sentiment Discovery and Opinion Mining*
- Muhammad, A., Wiratunga, N., Lothian, R., & Glassey, R. (2013). Domain-Based Lexicon Enhancement for Sentiment Analysis. *Symposium conducted at the meeting of the SMA@ BCS-SGAI*
- Na, J.-C., Kyaing, W. Y. M., Khoo, C. S., Foo, S., Chang, Y.-K., & Theng, Y.-L. (2012). Sentiment classification of drug reviews using a rule-based linguistic approach. *Springer. Symposium conducted at the meeting of the International Conference on Asian Digital Libraries*
- Nadkarni, P. M., Ohno-Machado, L., & Chapman, W. W. (2011). Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, 18(5), 544-551.

## References

- Nand, P., & Perera, R. (2015). An evaluation of POS tagging for tweets using HMM modeling.
- Navigli, R. (2009). Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2), 10.
- Navigli, R., & Lapata, M. (2010). An experimental study of graph connectivity for unsupervised word sense disambiguation. *IEEE transactions on pattern analysis and machine intelligence*, 32(4), 678-692.
- Niu, Y., Zhu, X., Li, J., & Hirst, G. (2005). Analysis of polarity information in medical text American Medical Informatics Association. Symposium conducted at the meeting of the AMIA annual symposium proceedings
- NLTK. (2017). NLTK 3.2.4 documentation. Retrieved from <http://www.nltk.org/>
- Osimo, D., & Mureddu, F. (2012). Research challenge on opinion mining and sentiment analysis. Université de Paris-Sud, Laboratoire LIMSIS-CNRS, Bâtiment, 508.
- Palace, B. (1996). Data Mining. Retrieved from [http://www.anderson.ucla.edu/faculty/jason.frand/teacher/technologies/palace/data\\_mining.htm](http://www.anderson.ucla.edu/faculty/jason.frand/teacher/technologies/palace/data_mining.htm)
- Perkins, J. (2014). *Python 3 Text Processing with NLTK 3 Cookbook*: Packt Publishing Ltd.
- Pustejovsky, J. (2005). *Lexicon: Natural Language Processing*. Retrieved 26/10/2013, 2013, from <http://www.credoreference.com.ezproxy.aut.ac.nz/entry/wiley/cs/lexicon>
- Reinartz, T. (2002). *Stages of the discovery process* Oxford University Press, Inc. Symposium conducted at the meeting of the Handbook of data mining and knowledge discovery
- Roark, B., Saraclar, M., & Collins, M. (2007). Discriminative n-gram language modeling. *Computer Speech & Language*, 21(2), 373-392.
- Santos, I., Peña, Y. K., Devesa, J., & Bringas, P. G. (2009). N-grams-based File Signatures for Malware Detection. *ICEIS* (2), 9, 317-320.
- Sarker, A., Mollá-Alíod, D., & Paris, C. (2011). Outcome polarity identification of medical papers.
- Sebastiani, F. (2005). *Text Categorization*.
- Shannon, C. E. (1951). Prediction and entropy of printed English. *Bell Labs Technical Journal*, 30(1), 50-64.
- Sharif, H., Zaffar, F., Abbasi, A., & Zimbra, D. (2014). Detecting adverse drug reactions using a sentiment classification framework.
- Sidorov, G., Velasquez, F., Stamatatos, E., Gelbukh, A., & Chanona-Hernández, L. (2014). Syntactic n-grams as machine learning features for natural language processing. *Expert Systems with Applications*, 41(3), 853-860.
- Smith, P., & Lee, M. (2012). Cross-discourse development of supervised sentiment analysis in the clinical domain Association for Computational Linguistics. Symposium conducted at the meeting of the Proceedings of the 3rd Workshop in Computational Approaches to Subjectivity and Sentiment Analysis
- Sokolova, M., & Bobicev, V. (2013). What Sentiments Can Be Found in Medical Forums? Symposium conducted at the meeting of the RANLP

## References

- Sokolova, M., Matwin, S., Jafer, Y., & Schramm, D. (2013). How Joe and Jane Tweet about Their Health: Mining for Personal Health Information on Twitter Symposium conducted at the meeting of the RANLP
- Vilares, J., Vilares, M., & Otero, J. (2011). Managing misspelled queries in IR applications. *Information Processing & Management*, 47(2), 263-286.
- Wang, H., Lu, Y., & Zhai, C. (2010). Latent aspect rating analysis on review text data: a rating regression approachACM. Symposium conducted at the meeting of the Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining
- Xia, L., Gentile, A. L., Munro, J., & Iria, J. (2009). Improving Patient Opinion Mining through Multi-step ClassificationSpringer. Symposium conducted at the meeting of the TSD
- Zhang, Y., & Desouza, P. (2014). Enhance the Power of Sentiment Analysis. *International Journal of Computer, Information, Systems and Control Engineering*.

# Appendix

1. First we read the entire csv and then read a particular column which contains user comments and excludes first row and cells which have no comments. In a csv, columns with no comments have '999'

```
from FilesToHost import app
import nltk
import json
import numpy
import csv
@app.route("/readcsv/<csvfilename>")
def readcsv(csvfilename):
    returnlist=[]
    with open(csvfilename, newline='') as csvfile:
        dataReader = csv.reader(csvfile, delimiter=',', quotechar='|')
        for row in dataReader:
            commentobj = row[23]
            if (commentobj == "999" or commentobj == "comments"):
                commentobj = 'na'
            else:
                returnlist.append(commentobj)
    return json.dumps(returnlist)
```

## 2. Tokenization

The following code helps us to tokenize the data.

```
from FilesToHost import app
import nltk
import json
import numpy
@app.route("/tokenize/<word>")
def tokenize(word):
    return json.dumps(Tokenizer(commentobj))
def Tokenizer(tokens):
    return nltk.word_tokenize(tokens)
```

### 3. Fetch Stopwords

In the following code we remove all the stopwords by importing all the stopwords from nltk corpus. We can add our own list of stopwords in the following code.

```
from FilesToHost import app
import nltk
import json
import numpy
from nltk.corpus import stopwords
stopwordlist = stopwords.words('english')
@app.route("/stopwords")
def stopwords():
    return json.dumps(stopwordlist)
```

### 4. POS Tagging

In the following code we POS tag the sentence by using nltk's pos\_tag method.

```
from FilesToHost import app
import nltk
import json
import numpy
@app.route("/postag/<sentence>")
def PosTaggingCsv (sentence):
    return json.dumps(nltk.pos_tag(sentence.split()))
```

### 5. Positive sentiment

In the following code we obtain the positive score of the word using SentiWordNet. We iterate over the list of word we obtain after tokenizing them and then we pass it to SentiWordNet and positive score is fetched by calling pos\_score() method.

```
from FilesToHost import app
import json
import nltk
from nltk.corpus import sentiwordnet as swn
@app.route("/sentiwordnetpos/<word>/<pos>")
def sentiwordnetpos(word, pos):
    returnlist=[]
    posscore=0
    test = list(swn.senti_synsets(word))
    for iterating_var in test:
```



```

        posscore += iterating_var.pos_score()
    return json.dumps(posscore)

```

## 6. Negative sentiment

In the following code we obtain the negative score of the word using SentiWordNet. We iterate over the list of word we obtain after tokenizing them and then we pass it to SentiWordNet and negative score is fetched by calling `neg_score()` method.

```

from FilesToHost import app
import json
import nltk
from nltk.corpus import sentiwordnet as swn
@app.route("/sentiwordnetneg/<word>/<pos>")
def sentiwordnetneg(word, pos):
    returnlist=[]
    negscore=0
    test = list(swn.senti_synsets(word))
    for iterating_var in test:
        negscore += iterating_var.neg_score()
    return json.dumps(negscore)

```

## 7. Creating custom corpus

As a part of creating a customized solution to study the process of Sentiment analysis, a python program was written which could create custom corpus for any dataset. This program would go through the entire dataset and would then classify the sentences as positive or negative and this can be used in future to obtain better results.

```

from FilesToHost import app
import json
import nltk
from nltk.corpus import sentiwordnet as swn
from nltk.sentiment.util import *
from nltk.classify import NaiveBayesClassifier
import datetime
import time
@app.route("/createpersonalcorpus/<sentence>")

```

```

def createpersonalcorpus(sentence):
    posscore=0
    negscore=0
    totalscore=0
    personalCorpusDir = "D:\\College
Studies\\MasterThesis\\SentimentAnalyzer\\FlaskSample\\Corpus\\"
    #get negated words
    negatedwords = negation(sentence)
    #print(negatedwords)
    #tokenize
    tokens = Tokenizer(sentence)
    #cleantokens
    cleanedTokens = CleanTokens(tokens)

    #postag
    postaggedsentence = PosTag (cleanedTokens)

    #get sentiword net score for each word in the pos tagged array
    for items in postaggedsentence:
        mappedValue = MapValue(items[1])
        if (mappedValue != ""):
            sentiword = list(swn.senti_synsets(items[0], mappedValue))
            for word in sentiword:
                posscore += word.pos_score()
                negscore += word.neg_score() * -1
                totalscore = totalscore + posscore + negscore

    ts = time.time()
    dt = datetime.datetime.fromtimestamp(Fayyad et al.).strftime('%Y-%m-%d-%H-
%M-%S')

    #after calculating scores write sentences
    if totalscore >= 0:
        fo = open(personalCorpusDir + "pos\\" + "pos"+ dt + ".txt", "a+")
        fo.write(sentence + "\n");
        fo.close()
    else:
        fo = open(personalCorpusDir + "neg\\" + "neg"+ dt + ".txt", "a+")
        fo.write(sentence + "\n");
        fo.close()

```

```
        return json.dumps(totalscore)

def Tokenizer(sentence):
    return nltk.word_tokenize(sentence)

def CleanTokens(tokens):
    returnlist=[]
    donothin = ""
    for token in tokens:
        if (token == '.' or token == ''):
            donothin = 'need to check this again'
        else:
            returnlist.append(token)
    return returnlist

def negation(sentence):
    sent = sentence.split()
    return mark_negation(sent)

def PosTag (tokens):
    return nltk.pos_tag(tokens, 'universal')

def MapValue(val):
    if val == 'NOUN':
        return 'n'
    elif val == 'ADJ':
        return 'a'
    elif val == 'VERB':
        return 'v'
    elif val == 'ADV':
        return 'r'
    else:
        return ""
```

## 8. Sentiment analysis using Movie Review

```

from FilesToHost import app

import nltk.classify.util
import json
import csv
from nltk import precision
from nltk import recall
import collections
import nltk.classify.util, nltk.metrics
from nltk.classify import NaiveBayesClassifier
from nltk.corpus import movie_reviews as mr

from nltk.corpus import stopwords
stopwordlist = stopwords.words('english')

@app.route("/sentimentanalyzedoc")
def sentimentanalyzedoc():
    return sentiment_analyze(word_feats)

def sentiment_analyze(feats):
    negids = mr.fileids('neg')
    posids = mr.fileids('pos')

    posfeats = [(featx(mr.words(fileids=[f])), 'pos') for f in posids]
    negfeats = [(featx(mr.words(fileids=[f])), 'neg') for f in negids]

    trainfeats = negfeats + posfeats

    classifier = NaiveBayesClassifier.train(trainfeats)

    tokenizeddoc = word_feats(TokenizeDoc())
    observed = classifier.classify(tokenizeddoc)
    print ('doc sentiment: ' + observed)

    testsets = collections.defaultdict(set)
    refsets = collections.defaultdict(set)
    for i, (feats, label) in enumerate(trainfeats):
        refsets[label].add(i)
        observed = classifier.classify(feats)
        testsets[observed].add(i)

```

```

    print ('accuracy:', nltk.classify.util.accuracy(classifier, trainfeats))
    print ('pos precision:', precision(refsets['pos'], testsets['pos']))
    print ('pos recall:', recall(refsets['pos'], testsets['pos']))
    print ('neg precision:', precision(refsets['neg'], testsets['neg']))
    print ('neg recall:', recall(refsets['neg'], testsets['neg']))
    print ('most important information:', classifier.show_most_informative_features())

    return json.dumps('nothing')

def word_feats(words):
    return dict([(word, True) for word in words])

def TokenizeDoc():
    returnlist=[]
    donothing = "
    with open('D:\\College Studies\\MasterThesis\\SentimentAnalyzer\\gmc II pq
dataset ALL CASES 18.11.2009.csv', newline=") as csvfile:
        dataReader = csv.reader(csvfile, delimiter=',', quotechar='|')
        for row in dataReader:
            commentobj = row[23]
            if (commentobj == "999" or commentobj == "comments"):
                commentobj = 'na'
            else:
                tokenized = Tokenizer(commentobj)
                for token in tokenized:
                    tempObj = token.split(',')
                    if (tempObj[0] == '.' or tempObj[0] == '' or
tempObj[0].lower() in stopwordlist):
                        donothin = 'need to check this again'

                    else:
                        returnlist = returnlist + tempObj

    return returnlist

def Tokenizer(tokens):
    return nltk.word_tokenize(tokens)

```

## 10. Sentiment analysis using custom corpus

```

from FilesToHost import app
import json

```

```

import csv
import collections
import nltk.classify.util, nltk.metrics
from nltk.classify import NaiveBayesClassifier
from nltk import precision
from nltk import recall
import os
from nltk.corpus.reader.plaintext import PlaintextCorpusReader
from nltk.corpus import CategorizedPlaintextCorpusReader

@app.route("/finalclassification")
def finalclassification():
    mydir = 'D:\\College
Studies\\MasterThesis\\SentimentAnalyzer\\FlaskSample\\Corpus\\'
    mr = CategorizedPlaintextCorpusReader(mydir, r'(!G. Miner).*\\.txt',
cat_pattern=r'(neg|pos)/.*', encoding='ascii')
    posids = mr.fileids('pos')
    negids = mr.fileids('neg')
    posfeats = [(ConvertToDictionary(mr.words(fileids=[f])), 'pos') for f in posids]
    negfeats = [(ConvertToDictionary(mr.words(fileids=[f])), 'neg') for f in negids]
    trainsets = posfeats + negfeats
    classifier = GetClassifier(trainsets)
    filepath = 'D:\\College
Studies\\MasterThesis\\SentimentAnalyzer\\FlaskSample\\gmc II pq dataset ALL CASES
18.11.2009.csv'
    tokenizeddoc = ConvertToDictionary(TokenizeDoc(filepath))
    observed = classifier.classify(tokenizeddoc)
    print ('doc sentiment: ' + observed)
    testsets = collections.defaultdict(set)
    refsets = collections.defaultdict(set)
    for i, (feats, label) in enumerate(trainsets):
        refsets[label].add(i)
        observed = classifier.classify(feats)
        testsets[observed].add(i)
    print ('accuracy:', nltk.classify.util.accuracy(classifier, trainsets))
    print ('pos precision:', precision(refsets['pos'], testsets['pos']))
    print ('pos recall:', recall(refsets['pos'], testsets['pos']))
    print ('neg precision:', precision(refsets['neg'], testsets['neg']))
    print ('neg recall:', recall(refsets['neg'], testsets['neg']))

```

```

        print ('most important information:',
classifier.show_most_informative_features())

        return json.dumps(observed)

def GetClassifier (trainSets):
    return NaiveBayesClassifier.train(trainSets)

def ConvertToDictionary(words):
    return dict([(word, True) for word in words])

def TokenizeDoc(filepath):
    returnlist=[]
    donothing = ''
    with open(filepath, newline='') as csvfile:
        dataReader = csv.reader(csvfile, delimiter=',', quotechar='|')
        for row in dataReader:
            commentobj = row[23]
            if (commentobj == "999" or commentobj == "comments"):
                commentobj = 'na'
            else:
                tokenized = Tokenizer(commentobj)
                for token in tokenized:
                    tempObj = token.split(',')
                    if (tempObj[0] == '.' or tempObj[0] == ''):
                        donothin = 'need to check this again'

                else:
                    returnlist = returnlist + tempObj

    return returnlist

def Tokenizer(tokens):
    return nltk.word_tokenize(tokens)

```

We then use Javascript Ajax to call these services and use HTML to view the data

```

<!DOCTYPE html>
<html>
<head>

```

```

        <title>Doctor Review</title>
</head>
<body>
<p> View Results</p>

<input type="file" name="filename" id="filename">
<br/>
<br/>

<select id="doctorFilter">
    <option selected></option>
    <option selected>All</option>
</select>
<select id="patientFilter">
    <option selected></option>
    <option selected>All</option>
</select>
<br/>
<br/>

<input type="hidden" id="csvSentences" value=""/>

<input type="hidden" id="commentArr" value=""/>
<input type="hidden" id="negatedArr" value=""/>
<input type="hidden" id="posTaggedArr" value=""/>
<input type="hidden" id="negatedWordArr" value=""/>
<input type="hidden" id="totalPosScore" value=""/>
<input type="hidden" id="totalNegScore" value=""/>
<input type="hidden" id="totalDocScore" value=""/>
<input type="hidden" id="csv" value=""/>

<div id="score"></div>
<div id="div1"></div>

<input type="button" id="posTag" value="Get Pos Tagged"/>
<input type="button" id="getNegatedWords" value="Get Negated Words"/>
<input type="button" id="getPosScore" value="Get Positive Score"/>
<input type="button" id="getNegScore" value="Get Negative Score"/>
<input type="button" id="getDocScore" value="Get Document Score"/>

```



```

<br />
<br />
<input type="button" id="createCorpus" value="Create Corpus"/>
<input type="button" id="classifyDocument" value="Classify"/>

<script src="https://code.jquery.com/jquery-2.2.3.min.js"></script>
<script>
$(function() {
    var stopwordsArr = "";
    var commentArr = [];
    var posTagged = [];
    var negationDetected = [];
    var allDocIdArr = [];
    var allPatientIdArr = [];
    var csvData = [];

    $("#filename").change(function(e) {
        var ext = $("input#filename").val().split(".").pop().toLowerCase();

        if (e.target.files != undefined) {
            var reader = new FileReader();
            reader.onload = function(e) {
                var csvval=e.target.result.split("\n");

                var csvCounter = 1;
                var prevDocId = 0;
                var prevPatientId = 0;

                for (var i=1; i < csvval.length; i++)
                {
                    csvCounter++;
                    if (csvval[i].split(",")[24] != '999' &&
csvval[i].split(",")[24] != undefined)
                    {
                        if (csvval[i].split(",")[0] != undefined)
                        {
                            if (prevDocId !=
parseInt(csvval[i].split(",")[0]))
                                {

```

```

                                var tempDocId =
csvval[i].split(",")[0]

                                var options = "<option>" +

parseInt(tempDocId) + "</option>";

                                $('#doctorFilter').append(options);
                                }

                                prevDocId =

parseInt(csvval[i].split(",")[0]);
                                }

                                if (csvval[i].split(",")[1] != undefined)
                                {
                                    if (prevPatientId !=

parseInt(csvval[i].split(",")[1]))
                                    {

                                        var options = "<option>" +

parseInt(csvval[i].split(",")[1]) + "</option>";

                                        $('#patientFilter').append(options);
                                        }

                                        prevPatientId =

parseInt(csvval[i].split(",")[1]);
                                }

                                var tempData = csvval[i].split(",")[24];

                                if ($('#csvSentences').val() == "")
                                {
                                    $('#csvSentences').val(tempData);
                                }
                                else
                                {
                                    var stored = $('#csvSentences').val();

```

```

                                $('#csvSentences').val(stored + "," +
tempData);
                                }

                                csvData.push(csvval[i].split(",")[0] + ";" +
csvval[i].split(",")[1] + ";" + tempData);

                                //tokenize and add to array
                                $.ajax({
                                type: "GET",
                                url:
"http://localhost:5000/removestopword/" + tempData,
                                async: true,
                                cache: false,
                                processData: false,
                                success: function(data, textStatus, xhr) {

                                commentArr.push(JSON.parse(data));

                                }});

                                $.ajax({
                                type: "GET",
                                url: "http://localhost:5000/negation/" +
tempData,

                                async: true,
                                cache: false,
                                processData: false,
                                success: function(data, textStatus, xhr) {

                                negationDetected.push(JSON.parse(data));

                                }});

                                }
                                }

                                $('#csv').val(csvData);

```

```

        var interval = setInterval(function(){
            if(csvCounter == csvval.length){
                clearInterval(interval);
                var negText = [];
                for(var i=0; i < negationDetected.length; i++)
                {
                    for(var j=0; j <
negationDetected[i].length; j++){

                        negText.push(negationDetected[i][j].replace(".", ""))
                        }
                    }
                $('#negatedArr').val("");
                $('#negatedArr').val(negText);

                var allText = "";
                for(var j=0; j < commentArr.length; j++)
                {
                    if (allText == "") {
                        allText = commentArr[j];
                    }
                    else{
                        allText = allText + "," +
commentArr[j];

                    }

                }
                $('#commentArr').val("");
                $('#commentArr').val(allText);
            }
        }, 2000);
    };
    reader.readAsText(e.target.files.item(0));
}
return false;
});

$('#posTag').click(function(){
    $.ajax({
        type: "GET",

```

```

        url: "http://localhost:5000/postag/" + $('#commentArr').val(),

        async: true,
        cache: false,
        processData: false,
        success: function(data, textStatus, xhr) {
            $('#posTaggedArr').val("");
            posTagged.push(data);
            $('#posTaggedArr').val(posTagged);
        }
    });

});

$('#patientFilter').change(function(){
    var commentArr = [];
    var negationDetected = [];
    var counter = 0;
    var selectedText = $( "select#patientFilter option:selected" ).text();

    var csvData = $('#csv').val().split(",");

    for(var count = 0; count < csvData.length; count++)
    {
        var csvRow = csvData[count].split(";");
        counter++;
        if (csvRow[1] == selectedText)
        {
            var tempData = csvRow[2];

            //tokenize and add to array
            $.ajax({
                type: "GET",
                url: "http://localhost:5000/removestopword/" + tempData,

                async: true,
                cache: false,
                processData: false,
                success: function(data, textStatus, xhr) {
                    commentArr.push(JSON.parse(data));
                }
            });
        }
    }
});

```

```

        });

$.ajax({
    type: "GET",
    url: "http://localhost:5000/negation/"+ tempData,

    async: true,
    cache: false,
    processData: false,
    success: function(data, textStatus, xhr) {
        negationDetected.push(JSON.parse(data));
    }
});

}

}

var interval = setInterval(function(){
    if(counter == csvData.length){
        clearInterval(interval);
        var negText = [];
        for(var i=0; i < negationDetected.length; i++)
        {
            for(var j=0; j < negationDetected[i].length; j++){

negText.push(negationDetected[i][j].replace(".", ""))
            }
        }
        $('#negatedArr').val("");
        $('#negatedArr').val(negText);

        var allText = "";
        for(var j=0; j < commentArr.length; j++)
        {
            if (allText == "") {
                allText = commentArr[j];
            }
            else{
                allText = allText + "," + commentArr[j];
            }
        }
    }
});

```

```

        }
        $('#commentArr').val("");
        $('#commentArr').val(allText);
    }
    }, 2000);
});

$('#doctorFilter').change(function(){
    var commentArr = [];
    var negationDetected = [];
    var counter = 0;
    var selectedText = $( "select#doctorFilter option:selected" ).text();

    var csvData = $('#csv').val().split(",");

    for(var count = 0; count < csvData.length; count++)
    {
        var csvRow = csvData[count].split(";");
        counter++;
        if (csvRow[0] == selectedText)
        {

            var tempData = csvRow[2];

            //tokenize and add to array
            $.ajax({
                type: "GET",
                url: "http://localhost:5000/removestopword/"+ tempData,

                async: true,
                cache: false,
                processData: false,
                success: function(data, textStatus, xhr) {
                    commentArr.push(JSON.parse(data));
                }
            });

            $.ajax({
                type: "GET",

```

```

        url: "http://localhost:5000/negation/"+ tempData,

        async: true,
        cache: false,
        processData: false,
        success: function(data, textStatus, xhr) {
            negationDetected.push(JSON.parse(data));
        });
    }
}

var interval = setInterval(function(){
    if(counter == csvData.length){
        clearInterval(interval);
        var negText = [];
        for(var i=0; i < negationDetected.length; i++)
        {
            for(var j=0; j < negationDetected[i].length; j++){

negText.push(negationDetected[i][j].replace(".", ""))
            }
        }
        $('#negatedArr').val("");
        $('#negatedArr').val(negText);

        var allText = "";
        for(var j=0; j < commentArr.length; j++)
        {
            if (allText == "") {
                allText = commentArr[j];
            }
            else{
                allText = allText + "," + commentArr[j];
            }
        }
        $('#commentArr').val("");
        $('#commentArr').val(allText);
    }
}, 2000);

```



```

});

$('#getPosScore').click(function(){
    var parsed = JSON.parse($('#posTaggedArr').val());
    var posScoreArr = [];

    for(var count=0; count < parsed.length; count++){
        var mapped = Map(parsed[count][1]);
        parsed[count][1] = mapped;

        if (parsed[count][1] != undefined && (parsed[count][0].indexOf("")))
< 0))
        {
            $.ajax({
                type: "GET",
                url: "http://localhost:5000/sentiwordnetpos/" +
parsed[count][0] + "/" + parsed[count][1],
                async: false,
                cache: false,
                processData: false,
                success: function(data, textStatus, xhr) {
                    posScoreArr.push(parseFloat(data));
                });
        }
    }
    $('#totalPosScore').val("");
    $('#totalPosScore').val(posScoreArr);
});

$('#getNegScore').click(function(){
    var parsed = JSON.parse($('#posTaggedArr').val());
    var negScore = 0;
    var negScoreArr = [];

    for(var count=0; count < parsed.length; count++){
        var mapped = Map(parsed[count][1]);
        parsed[count][1] = mapped;

        if (parsed[count][1] != undefined && (parsed[count][0].indexOf("")))
< 0))

```

```

        {
            $.ajax({
                type: "GET",
                url: "http://localhost:5000/sentiwordnetneg/" +
                parsed[count][0] + "/" + parsed[count][1],
                async: false,
                cache: false,
                processData: false,
                success: function(data, textStatus, xhr) {
                    negScore = parseFloat(data) * -1;
                    negScoreArr.push(negScore);
                });
        }
    }
    $('#totalNegScore').val("");
    $('#totalNegScore').val(negScoreArr);
});

$('#getDocScore').click(function(){
    var parsed = JSON.parse($('#posTaggedArr').val());

    var docScore = 0;
    var counter = 0;
    var posScore = 0;
    var negScore = 0;

    var posScoreArr = $('#totalPosScore').val().split(',');
    var negScoreArr = $('#totalNegScore').val().split(',');
    var negatedWordArr = $('#negatedWordArr').val().split(',');

    for(var count=0; count < negatedWordArr.length; count++)
    {
        //reverse negated words
        if (negatedWordArr[count] != "")
        {
            if (posScoreArr[count] > 0)
            {
                negScore = (1 - parseFloat(posScoreArr[count])) *
                (parseFloat(posScoreArr[count]) * -1);
            }
        }
    }
});

```

```

        }
        else if (negScore < 0)
        {
            posScore = (1 - parseFloat(negScoreArr[count])) *
(parseFloat(negScoreArr[count]) * -1);
        }
    }
    else
    {
        posScore = parseFloat(posScoreArr[count]);
        negScore = parseFloat(negScoreArr[count]);
    }

    docScore = docScore + posScore + negScore;
}

if (docScore > 0){
    $('#div1').html("The document has positive reviews overall");
    $('#score').html("Overall doc score: " + docScore);
}
else{
    $('#div1').html("The document has negative reviews overall");
    $('#score').html("Overall doc score: " + docScore);
}
});

$('#createCorpus').click(function(){
    var csvsentences = $('#csvSentences').val().split(',');

    for(var count = 0; count < csvsentences.length; count++){
        $.ajax({
            type: "GET",
            url: "http://localhost:5000/createpersonalcorpus/"+
csvsentences[count],
            async: false,
            cache: false,
            processData: false,
            success: function(data, textStatus, xhr) {

```

```

    });
    }
});

$('#classifyDocument').click(function(){
    var csvsentences = $('#csvSentences').val().split(',');

    for(var count = 0; count < csvsentences.length; count++){
        $.ajax({
            type: "GET",
            url: "http://localhost:5000/finalclassification",
            async: false,
            cache: false,
            processData: false,
            success: function(data, textStatus, xhr) {

                });
            }
        });

    $('#getNegatedWords').click(function(){
        var parsed = JSON.parse($('#posTaggedArr').val());

        var counter = 0;
        var negatedWordArr = [];
        var negatedWord = "";
        var negText = $('#negatedArr').val().split(',');

        for(var count=0; count < parsed.length; count++){
            var mapped = Map(parsed[count][1]);
            parsed[count][1] = mapped;

            if((parsed[count][0].indexOf("") > 0 ||
            parsed[count][0].indexOf("") < 0)){

                if (parsed[count][1] != undefined && negText[counter] !=
            undefined){

                    if (negText[counter].indexOf("_NEG") >=0)
                    {

```

```

        negatedWordArr.push(negText[counter].substr(0,
negText[counter].indexOf("_NEG")));
            }
            else
            {
                negatedWordArr.push("");
            }
        }
        counter++;
    }
}
$('#negatedWordArr').val("");
$('#negatedWordArr').val(negatedWordArr);
});
});
function Map(param)
{
    switch(param)
    {
        case 'NOUN':
            return 'n';
        case 'ADJ':
            return 'a';
        case 'VERB':
            return 'v';
        case 'ADV':
            return 'r';
    }
}

}

</script>
</body>
</html>

```