
Emotion Classification on Social Networking Sites

Project Report submitted in partial fulfillment of the

requirement for the degree of

Bachelor of Technology.

in

Computer Science & Engineering

under the Supervision of

Dr. Pardeep Kumar

By

Geetanjali Vadehra (111341)



Jaypee University of Information and Technology

Waknaghat, Solan – 173234, Himachal Pradesh

Certificate

This is to certify that project report entitled “Emotion Classification on Social Networking Sites ”, submitted by Geetanjali Vadehra(111341) in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science & Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date:

Dr. Pardeep Kumar

**Assistant Professor (Senior Grade)
Computer Science & Engineering Department
Jaypee University of Information Technology, Solan (H.P) India**

Acknowledgement

“It is not possible to prepare a project without the assistance & encouragement of other people. This one is certainly no exception.”

On the very outset of this report, I would like to extend my sincere & heartfelt obligation towards all the personages who have helped me in this endeavor. Without their active guidance, help, cooperation & encouragement, I would not have made headway in the project.

I would like to show my greatest appreciation to Dr. Pardeep Kumar. I feel motivated every time I get his encouragement. For his coherent guidance throughout the tenure of the project, I feel fortunate to be taught by Dr. Pardeep Kumar, who gave me his unwavering support. Besides being my mentor, he taught me that there is no substitute for hard work.

I owe my heartiest thanks to Brig. (Retd.) S.P. Ghrera (HOD, CES/IT Department), who has always inspired me to take initiatives and showed me the path for achieving my goal.

In the light of new developments and recent findings, I devote the task that was asked from me at Jaypee University of Information Technology to **“Emotion Classification in Social Networking Sites”**.

Date:

Geetanjali Vadehra

Table of Content

S. No.	Topic	Page No.
1.	Introduction	1
1	Motivation	3
2	Aims and Objectives	3
2.	Project Requirements	4
1	Software Requirements	5
2	Hardware Requirements	5
3.	Feature Selection in Sentiment Classification	6
1	Feature Selection Methods	7
2	Search Based Feature Selection	10
4.	Sentiment Analysis Techniques	12
1	Machine Learning Approach	14
2	Lexicon Based Approach	18
5.	Lexicon Based Approach	21
6.	Naïve Bayes Classifier	38
7.	Technical Challenges	50
8.	Screenshots	53
9.	Conclusion	58
10.	Future Work	60
11.	References	62

List of Figures

S.No.	Title	Page No.
1.	Target of Sentiment Analysis	2
2.	Sentiment Analysis Techniques	13
3.	Support Vector Machines	16
4.	Lexicon Based Approach I	22
5.	Lexicon Based Approach II	25

List of Tables

S.No.	Title	Page No.
1.	Accuracy of different sentiment classifiers	20

Abstract

Sentiment Analysis (SA) is an ongoing field of research in text mining field. SA is the computational treatment of opinions, sentiments and subjectivity of text.

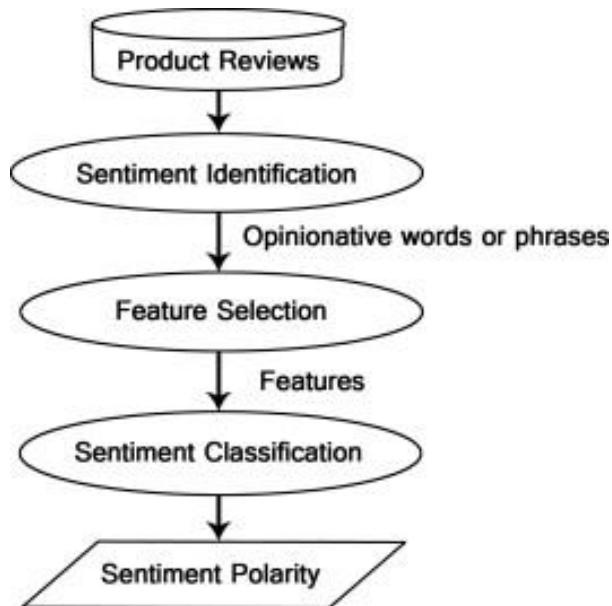
The decision-making process of people is affected by the opinions formed by thought leaders and ordinary people. When a person wants to buy a product online he or she will typically start by searching for reviews and opinions written by other people on the various offerings. Sentiment analysis is one of the hottest research areas in computer science. Over 7,000 articles have been written on the topic. Hundreds of startups are developing sentiment analysis solutions and major statistical packages such as SAS and SPSS include dedicated sentiment analysis modules. There is a huge explosion today of 'sentiments' available from social media including Twitter, Facebook, message boards, blogs, and user forums. These snippets of text are a gold mine for companies and individuals that want to monitor their reputation and get timely feedback about their products and actions. Sentiment analysis offers these organizations the ability to monitor the different social media sites in real time and act accordingly. Marketing managers, PR firms, campaign managers, politicians, and even equity investors and online shoppers are the direct beneficiaries of sentiment analysis technology.

This report tackles a comprehensive overview of the last update in this field and a sophisticated categorization of the techniques used in Sentiment Analysis.

The report also presents two methods to extract information about the users' sentiment polarity (positive, neutral or negative), as transmitted in the messages they write. It also mentions the potential improvements that can be made to these methods.

CHAPTER 1: INTRODUCTION

Sentiment Analysis (SA) or Opinion Mining (OM) is the computational study of people's opinions, attitudes and emotions toward an entity. The entity can represent individuals, events or topics. These topics are most likely to be covered by reviews. The two expressions SA or OM are interchangeable. They express a mutual meaning. However, some researchers stated that OM and SA have slightly different notions. Opinion Mining extracts and analyzes people's opinion about an entity while Sentiment Analysis identifies the sentiment expressed in a text then analyzes it. Therefore, the target of SA is to find opinions, identify the sentiments they express, and then classify their polarity as shown in figure.



Sentiment Analysis can be considered a classification process as illustrated in figure. There are three main classification levels in SA: document-level, sentence-level, and aspect-level SA. Document-level SA aims to classify an opinion document as expressing a positive or negative opinion or sentiment. It considers the whole document a basic information unit (talking about one topic). Sentence-level SA aims to classify sentiment expressed in each sentence. The first step is to identify whether the sentence is subjective or objective. If the sentence is subjective, Sentence-level SA will determine whether the sentence expresses positive or negative opinions. Sentiment expressions are not necessarily subjective in nature. However, there is no fundamental difference between document and sentence level classifications because sentences are just short documents. Classifying text at the document level or at the sentence level does not provide the necessary detail needed opinions on all aspects of the entity which is needed in many applications, to obtain these details; we need to go

to the aspect level. Aspect-level SA aims to classify the sentiment with respect to the specific aspects of entities. The first step is to identify the entities and their aspects. The opinion holders can give different opinions for different aspects of the same entity like this sentence “*The voice quality of this phone is not good, but the battery life is long*”.

1. Motivation

The social network sites and micro-blogging sites are considered a very good source of information because people share and discuss their opinions about certain topics freely. This project will be used to recognize people’s emotions about those topics. In political debates for example, we could figure out people’s opinions on a certain election candidates or political parties. The election results can also be predicted from political posts.

Sentiment analysis can also be used in adaptive E-learning systems. In particular, affective and emotional factors, among other aspects, seem to affect the student motivation and, in general, the outcome of the learning process. Therefore, in learning contexts, being able to detect and manage information about the students’ emotions at a certain time can contribute to know their potential needs at that time. On one hand, adaptive e-learning environments can make use of this information to fulfill those needs at runtime: they can provide the user with recommendations about activities to tackle or contents to interact with, adapted to his/her emotional state at that time. On the other hand, information about the student emotions towards a course can act as feedback for the teacher. This is especially useful for online courses, in which there is little (or none) face-to-face contact between students and teachers and, therefore, there are fewer opportunities for teachers to get feedback from the students.

Knowing the users’ emotions is useful not only in the educational context but also in many others (e.g., marketing, politics, online shopping, and so on).

2. Aim and Objective

The aim of this project is to extract information about the users’ sentiment as transmitted in the text they write on social networking sites, by assigning a score to each subsequent word in the text and computing the cumulative score

CHAPTER 2: PROJECT REQUIREMENTS

1. Software requirements

- Operating System: Windows 7/8/XP/Vista
- Software: Dev C++, Eclipse

1. Hardware requirements

- Processor: x86 compatible processor
- RAM: 512 MB or greater
- Hard Disk: 20 GB or greater
- Monitor: VGA/SVGA
- Keyboard: 104 keys standard
- Mouse: 2/3 button. Optical/ Mechanical.

CHAPTER 3:
FEATURE SELECTION IN SENTIMENT
CLASSIFICATION

Sentiment Analysis task is considered a sentiment classification problem. Feature selection is often integrated as the first step in machine learning algorithms like SVM, Neural Networks, k-Nearest Neighbors, etc. The main goal of the feature selection is to decrease the dimensionality of the feature space and thus computational cost. As a second objective, feature selection will reduce the overfitting of the learning scheme to the training data. During this process, it is also important to find a good tradeoff between the richness of features and the computational constraints involved when solving the categorization task. Some of the current features are:

Terms presence and frequency: These features are individual words or word n-grams and their frequency counts. It either gives the words binary weighting (zero if the word appears or one if otherwise) or uses term frequency weights to indicate the relative importance of features.

Parts of speech (POS): finding adjectives, as they are important indicators of opinions.

Opinion words and phrases: these are words commonly used to express opinions including *good or bad, like or hate*. On the other hand, some phrases express opinions without using opinion words. For example: *cost me an arm and a leg*.

Negations: the appearance of negative words may change the opinion orientation like *not good* is equivalent to *bad*.

1. Feature selection methods

Feature Selection methods can be divided into lexicon-based methods that need human annotation, and statistical methods which are automatic methods that are more frequently used. Lexicon-based approaches usually begin with a small set of ‘seed’ words. Then they bootstrap this set through synonym detection or on-line resources to obtain a larger lexicon. This proved to have many difficulties as reported by Whitelaw et al. Statistical approaches, on the other hand, are fully automatic.

The feature selection techniques treat the documents either as group of words (Bag of Words (BOWs)), or as a string which retains the sequence of words in the document. BOW is used more often because of its simplicity for the classification process. The most common feature selection step is the removal of stop-words and stemming (returning the word to its stem or root i.e. flies → fly).

In the next subsections, we present three of the most frequently used statistical methods in FS and their related articles. There are other methods used in FS like information gain and Gini index.

1.1. Point-wise Mutual Information (PMI)

The mutual information measure provides a formal way to model the mutual information between the features and the classes. This measure was derived from the information theory. The point-wise mutual information (PMI) $M_i(w)$ between the word w and the class i is defined on the basis of the level of co-occurrence between the class i and word w . The expected co-occurrence of class i and word w , on the basis of mutual independence, is given by $P_i \cdot F(w)$, and the true co-occurrence is given by $F(w) \cdot p_i(w)$.

The mutual information is defined in terms of the ratio between these two values and is given by the following equation:

$$M_i(w) = \log \left(\frac{F(w) \cdot p_i(w)}{F(w) \cdot P_i} \right) = \log \left(\frac{p_i(w)}{P_i} \right)$$

The word w is positively correlated to the class i , when $M_i(w)$ is greater than 0. The word w is negatively correlated to the class i when $M_i(w)$ is less than 0.

1.2. Chi-square (χ^2)

Let n be the total number of documents in the collection, $p_i(w)$ be the conditional probability of class i for documents which contain w , P_i be the global fraction of documents containing the class i , and $F(w)$ be the global fraction of documents which contain the word w . Therefore, the χ^2 -statistic of the word between word w and class i is defined as

$$\chi_i^2 = \frac{n \cdot F(w)^2 \cdot (p_i(w) - P_i)^2}{F(w) \cdot (1 - F(w)) \cdot P_i \cdot (1 - P_i)}$$

1.3. Latent Semantic Indexing (LSI)

Feature selection methods attempt to reduce the dimensionality of the data by picking from the original set of attributes. Feature transformation methods create a smaller set of features as a function of the original set of features. LSI is one of the famous feature transformation methods. LSI method transforms the text space to a new axis

system which is a linear combination of the original word features. Principal Component Analysis techniques (PCA) are used to achieve this goal. It determines the axis-system which retains the greatest level of information about the variations in the underlying attribute values. The main disadvantage of LSI is that it is an unsupervised technique which is blind to the underlying class-distribution. Therefore, the features found by LSI are not necessarily the directions along which the class-distribution of the underlying documents can be best separated

It is generally acknowledged that the ability to work with text on a semantic basis is essential to modern information retrieval systems. As a result, the use of LSI has significantly expanded in recent years as earlier challenges in scalability and performance have been overcome.

1.4 Categorical Proportional Difference (PD)

Categorical Proportional Difference (PD) is a metric which tells us how close to being equal two numbers are. We can use this to find unigrams that occur mostly in one class of documents or the other, by using the positive document frequency and negative document frequency of a unigram as the two numbers.

In other words if a unigram occurs predominantly in positive documents or predominantly in negative documents then the PD of the unigram will be close to one, whereas if it occurs in about as many positive documents as negative documents then its PD will be close to zero. A high score from this equation indicates that the unigram is telling us a lot, and a low score indicates that the unigram is telling us very little. For example if the word “actor” appears in exactly as many positive documents as negative documents then finding the word “actor” in a new document will tell us nothing about it and as such its PD score will be zero. Conversely, if the word “excellent” appears in only positive documents then finding the word “excellent” in a new document would give us a good clue that the document is positive, and as such it would have a PD score of one. So to use PD as a feature selector we simply need to remove any features where the result of the equation is less than or equal to some threshold value.

1.5 SWN Subjectivity Scores (SWNSS)

The SWN feature selector is actually able to distinguish objective and subjective terms, which is useful since only subjective terms should carry sentiment. To do this we use the SWN subjectivity score, which is found by adding the positive and negative SWN scores of a unigram together. To use it as a feature selector we simply remove any unigrams whose subjective score is less than a certain threshold. When this feature selector is used, unigrams that are not found in SWN, such as names and misspellings, are removed from the corpus as well (although arguably the names of certain actors could give strong clues about the quality of a movie).

1.6 SWN Proportional Difference (SWNPD)

While the SWN subjectivity feature selector can find words that have some a priori sentiment attached, it cannot tell us whether that sentiment is consistent or meaningful. It is entirely possible that a word may have a SWN subjectivity score of one, indicating that it is very subjective, but its positive and negative scores may be 0.5 each. This may make the word uninformative as a feature so there could be value in removing it. Similarly to PD, SWNPD will be high for words that are mostly positive or negative, and low for words that are a mix of both. By using this score we hope to remove subjective words that have an ambiguous polarity from the corpus.

2. Search based feature selection

An advantage of search based feature selection methods over rankings are usually more accurate results. These methods are based on both stochastic and heuristic searching strategies, what implies higher computational complexity, which for very large datasets that have a few thousands of variables may limit some algorithms usability.

Typical solutions of search based feature selection are forward/backward selection methods.

Forward selection

Forward selection starts from an empty feature set and, in each iteration, adds one new attribute, form the set of remaining. One that is added is this feature which maximizes certain criterion usually classification accuracy. To ensure the proper

outcome of adding a new feature to the feature subset, quality is measured in the cross validation process.

Backward elimination

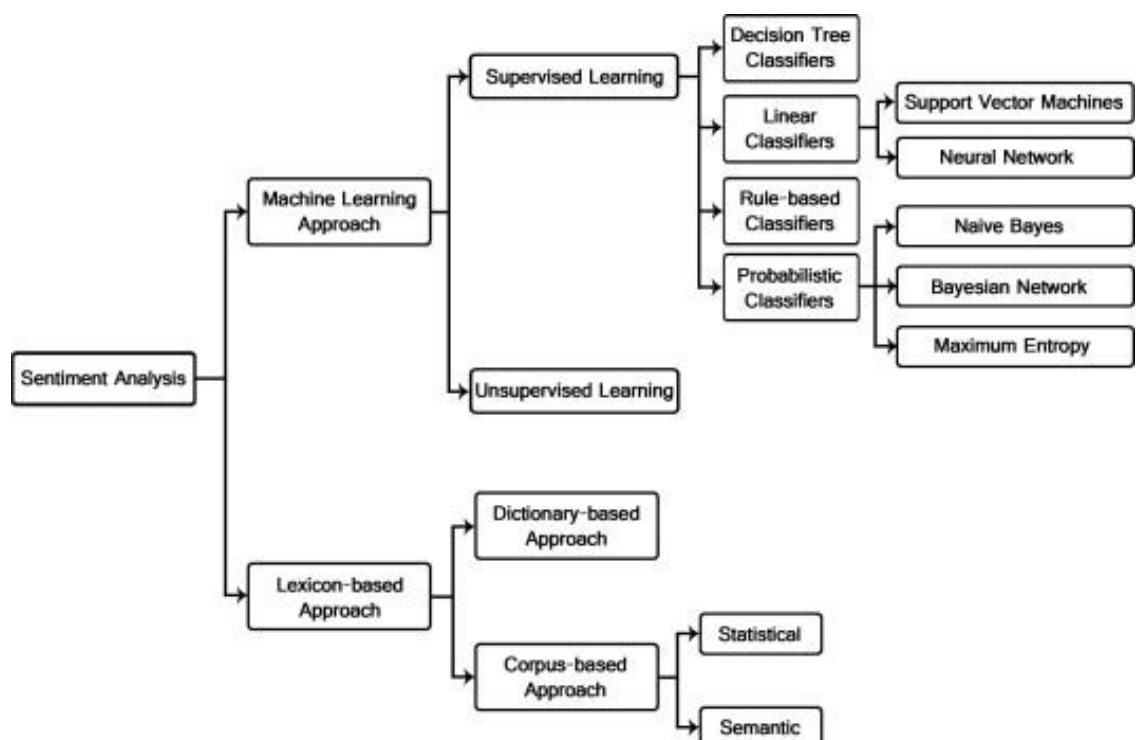
Backward elimination algorithm differs from forward selection by starting from the full feature set, and iteratively removes one by one feature. In each iteration only one feature is removed, which mostly affects overall model accuracy, as long as the accuracy stops increasing.

CHAPTER 4:
SENTIMENT ANALYSIS TECHNIQUES

Sentiment Classification techniques can be roughly divided into machine learning approach, lexicon based approach and hybrid approach. The *Machine Learning Approach (ML)* applies the famous ML algorithms and uses linguistic features. The *Lexicon-based Approach* relies on a sentiment lexicon, a collection of known and precompiled sentiment terms. It is divided into dictionary-based approach and corpus-based approach which use statistical or semantic methods to find sentiment polarity. The *hybrid Approach* combines both approaches and is very common with sentiment lexicons playing a key role in the majority of methods.

The text classification methods using ML approach can be roughly divided into supervised and unsupervised learning methods. The supervised methods make use of a large number of labeled training documents. The unsupervised methods are used when it is difficult to find these labeled training documents.

The lexicon-based approach depends on finding the opinion lexicon which is used to analyze the text. There are two methods in this approach. The dictionary-based approach which depends on finding opinion seed words, and then searches the dictionary of their synonyms and antonyms. The corpus-based approach begins with a seed list of opinion words, and then finds other opinion words in a large corpus to help in finding opinion words with context specific orientations. This could be done by using statistical or semantic methods. There is a brief explanation of both approaches' algorithms and related articles in the next subsections.



1. Machine learning approach

Machine learning approach relies on the famous ML algorithms to solve the SA as a regular text classification problem that makes use of syntactic and/or linguistic features.

Text Classification Problem Definition: We have a set of training records $D = \{X_1, X_2, \dots, X_n\}$ where each record is labeled to a class. The classification model is related to the features in the underlying record to one of the class labels. Then for a given instance of unknown class, the model is used to predict a class label for it. The hard classification problem is when only one label is assigned to an instance. The soft classification problem is when a probabilistic value of labels is assigned to an instance.

1.1. Supervised learning

The supervised learning methods depend on the existence of labeled training documents. There are many kinds of supervised classifiers in literature. In the next subsections, we present in brief details some of the most frequently used classifiers in SA.

Probabilistic classifiers

Probabilistic classifiers use mixture models for classification. The mixture model assumes that each class is a component of the mixture. Each mixture component is a generative model that provides the probability of sampling a particular term for that component. These kinds of classifiers are also called *generative classifiers*. Three of the most famous probabilistic classifiers are discussed in the next subsections.

- **Naïve Bayes Classifier (NB):** The Naïve Bayes classifier is the simplest and most commonly used classifier. Naïve Bayes classification model computes the posterior probability of a class, based on the distribution of the words in the document. The model works with the BOWs feature extraction which ignores the position of the word in the document. It uses Bayes Theorem to predict the probability that a given feature set belongs to a particular label.

$$P(\text{label}|\text{features}) = \frac{P(\text{label}) * P(\text{features}|\text{label})}{P(\text{features})}$$

$P(\text{label})$ is the prior probability of a label or the likelihood that a random feature set the label. $P(\text{features}|\text{label})$ is the prior probability that a given feature set is being classified as a label. $P(\text{features})$ is the prior probability that a given feature set is occurred. Given the Naïve assumption which states that all features are independent, the equation could be rewritten as follows:

$$P(\text{label}|\text{features}) = \frac{P(\text{label}) * P(f_1|\text{label}) * \dots * P(f_n|\text{label})}{P(\text{features})}$$

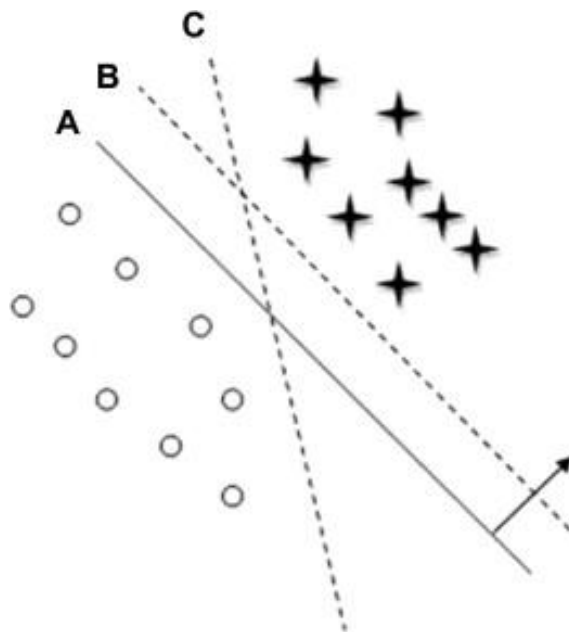
- **Bayesian Network (BN):** The main assumption of the NB classifier is the independence of the features. The other extreme assumption is to assume that all the features are fully dependent. This leads to the Bayesian Network model which is a directed acyclic graph whose nodes represent random variables, and edges represent conditional dependencies. BN is considered a complete model for the variables and their relationships. Therefore, a complete joint probability distribution (JPD) over all the variables is specified for a model. In Text mining, the computation complexity of BN is very expensive; that is why, it is not frequently used
- **Maximum Entropy Classifier (ME):** The Maxent Classifier (known as a conditional exponential classifier) converts labeled feature sets to vectors using encoding. This encoded vector is then used to calculate weights for each feature that can then be combined to determine the most likely label for a feature set. This classifier is parameterized by a set of $X\{weights\}$, which is used to combine the joint features that are generated from a feature-set by an $X\{encoding\}$. In particular, the encoding maps each $C\{featureset, label\}$ pair to a vector.

Linear Classifiers

Given $\bar{X} = \{x_1 \dots x_n\}$ is the normalized document word frequency, vector $\bar{A} = \{a_1 \dots a_n\}$ is a vector of linear coefficients with the same dimensionality as the feature space, and b is a scalar; the output of the linear predictor is defined as $p = \bar{A} \cdot \bar{X} + b$, which is the output of the *linear classifier*. The predictor p is a separating hyperplane between different classes. There are many kinds of linear classifiers; among them is *Support Vector Machines (SVM)* which is a form

of classifiers that attempt to determine *good* linear separators between different classes. Two of the most famous linear classifiers are discussed in the following subsections.

- **Support Vector Machines:** The main principle of SVMs is to determine linear separators in the search space which can best separate the different classes. In the following figure, there are 2 classes x, o and there are 3 hyperplanes A, B and C. Hyperplane A provides the best separation between the classes, because the normal distance of any of the data points is the largest, so it represents the maximum margin of separation. Text data are ideally suited for SVM classification because of the sparse nature of text, in which few features are irrelevant, but they tend to be correlated with one another and generally organized into linearly separable categories. SVM can construct a *nonlinear decision surface* in the original feature space by mapping the data instances non-linearly to an inner product space where the classes can be separated linearly with a hyperplane.



SVMs are used in many applications, among these applications are classifying reviews according to their quality.

- **Neural Network (NN):** Neural Network consists of many neurons where the neuron is its basic unit. The inputs to the neurons are denoted by the vector \overline{X}_i which is the word frequencies in the i th document. There are a set of weights A which are associated with each neuron used in order to compute a function of its inputs $f(\cdot)$. The linear function of the neural network

is: $p_i = A \cdot \bar{X}_i$. In a binary classification problem, it is assumed that the class label of \bar{X}_i is denoted by y_i and the sign of the predicted function p_i yields the class label. Multilayer neural networks are used for non-linear boundaries. These multiple layers are used to induce multiple piece-wise linear boundaries, which are used to approximate enclosed regions belonging to a particular class. The outputs of the neurons in the earlier layers feed into the neurons in the later layers. The training process is more complex because the errors need to be back-propagated over different layers.

Decision Tree Classifiers

Decision tree classifier provides a hierarchical decomposition of the training data space in which a condition on the attribute value is used to divide the data. The condition or predicate is the presence or absence of one or more words. The division of the data space is done recursively until the leaf nodes contain certain minimum numbers of records which are used for the purpose of classification.

There are other kinds of predicates which depend on the similarity of documents to correlate sets of terms which may be used to further partitioning of documents. The different kinds of splits are *Single Attribute split* which use the presence or absence of particular words or phrases at a particular node in the tree in order to perform the split. *Similarity-based multi-attribute split* uses documents or frequent words clusters and the similarity of the documents to these words clusters in order to perform the split. *Discriminant-based multi-attribute split* uses discriminants such as the Fisher discriminate for performing the split.

Rule Based Classifiers

In rule based classifiers, the data space is modeled with a set of rules. The left hand side represents a condition on the feature set expressed in disjunctive normal form while the right hand side is the class label. The conditions are on the term presence. Term absence is rarely used because it is not informative in sparse data.

There are numbers of criteria in order to generate rules, the training phase construct all the rules depending on these criteria. The most two common criteria are support and confidence. The *support* is the absolute number of instances in the training data set which are relevant to the rule. The *Confidence* refers to the conditional probability that the right hand side of the rule is satisfied if the left-hand side is satisfied.

1.2 Weakly, semi and unsupervised learning

The main purpose of text classification is to classify documents into a certain number of predefined categories. In order to accomplish that, large number of labeled training documents are used for supervised learning, as illustrated before. In text classification, it is sometimes difficult to create these labeled training documents, but it is easy to collect the unlabeled documents. The unsupervised learning methods overcome these difficulties.

2. Lexicon-based approach

Opinion words are employed in many sentiment classification tasks. Positive opinion words are used to express some desired states, while negative opinion words are used to express some undesired states. There are also opinion phrases and idioms which together are called *opinion lexicon*. There are three main approaches in order to compile or collect the opinion word list. *Manual approach* is very time consuming and it is not used alone. It is usually combined with the other two automated approaches as a final check to avoid the mistakes that resulted from automated methods. The two automated approaches are presented in the following subsections.

2.1. Dictionary-based approach

A small set of opinion words is collected manually with known orientations. Then, this set is grown by searching in the well known corpora WordNet or thesaurus for their synonyms and antonyms. The newly found words are added to the seed list then the next iteration starts. The iterative process stops when no new words are found. After the process is completed, manual inspection can be carried out to remove or correct errors.

The dictionary based approach has a major disadvantage which is the inability to find opinion words with domain and context specific orientations.

2.2. Corpus-based approach

The Corpus-based approach helps to solve the problem of finding opinion words with context specific orientations. Its methods depend on syntactic patterns or patterns that occur together along with a seed list of opinion words to find other opinion words in a large corpus. We start with a list of seed opinion adjectives, and use them along with a

set of linguistic constraints to identify additional adjective opinion words and their orientations. The constraints are for connectives like *AND*, *OR*, *BUT*, *EITHER-OR*.....; the conjunction *AND* for example says that conjoined adjectives usually have the same orientation. This idea is called *sentiment consistency*, which is not always consistent practically. There are also adversative expressions such as *asbut*, *however* which are indicated as opinion changes. In order to determine if two conjoined adjectives are of the same or different orientations, learning is applied to a large corpus. Then, the links between adjectives form a graph and clustering is performed on the graph to produce two sets of words: positive and negative.

Using the corpus-based approach alone is not as effective as the dictionary-based approach because it is hard to prepare a huge corpus to cover all English words, but this approach has a major advantage that can help to find domain and context specific opinion words and their orientations using a domain corpus. The corpus-based approach is performed using statistical approach or semantic approach as illustrated in the following subsections:

Statistical approach

Finding co-occurrence patterns or seed opinion words can be done using statistical techniques. This could be done by deriving posterior polarities using the co-occurrence of adjectives in a corpus, as proposed by Fahrni and Klenner. It is possible to use the entire set of indexed documents on the web as the corpus for the dictionary construction. This overcomes the problem of the unavailability of some words if the used corpus is not large enough.

The polarity of a word can be identified by studying the occurrence frequency of the word in a large annotated corpus of texts. If the word occurs more frequently among positive texts, then its polarity is positive. If it occurs more frequently among negative texts, then its polarity is negative. If it has equal frequencies, then it is a neutral word. The similar opinion words frequently appear together in a corpus. This is the main observation that the state of the art methods are based on. Therefore, if two words appear together frequently within the same context, they are likely to have the same polarity. Therefore, the polarity of an unknown word can be determined by calculating the relative frequency of co-occurrence with another word. This could be done using PMI.

Latent Semantic Analysis (LSA) is a statistical approach which is used to analyze the relationships between a set of documents and the terms mentioned in these documents in order to produce a set of meaningful patterns related to the documents and terms.

Semantic approach

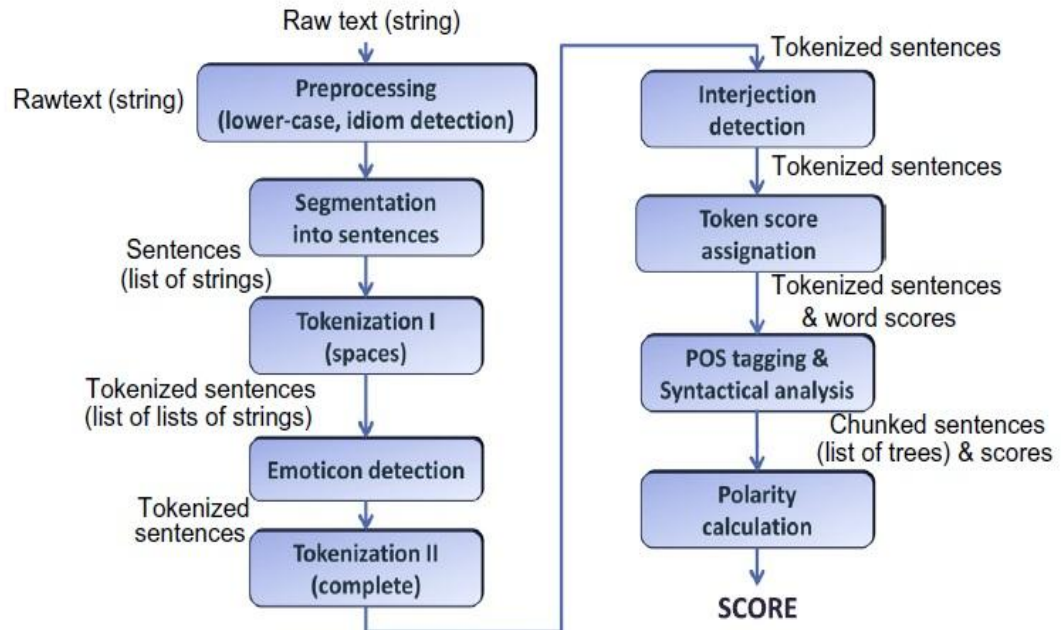
The Semantic approach gives sentiment values directly and relies on different principles for computing the similarity between words. This principle gives similar sentiment values to semantically close words. WordNet for example provides different kinds of semantic relationships between words used to calculate sentiment polarities. WordNet could be used too for obtaining a list of sentiment words by iteratively expanding the initial set with synonyms and antonyms and then determining the sentiment polarity for an unknown word by the relative count of positive and negative synonyms of this word.

Accuracy of proposed classifiers.

Method	Accuracy (%)
Lexicon-based approach	80.02
Tree decision (J48-C4.5) + Lexicon-based tagging	83.17
Naïve-Bayes + Lexicon-based tagging	83.13
SVM + Lexicon-based tagging	83.27

CHAPTER 5:
LEXICON BASED APPROACH

1. Method I



Extracting sentiments from texts: message classification

1.1. Preprocessing (lower-case, idiom detection)

The first step consists of preprocessing the message to convert all the words into lower-case. Afterwards, it detects idioms and joins the words involved in each of them.

1.2. Segmentation into sentences

Then, the message is divided into sentences. Dots are the only punctuation marks considered as separators at this step, since others such as commas or semicolons can be part of emoticons.

1.3. Tokenization I (partial)

In the next step, tokens are extracted from each sentence. At this time, only whitespaces are taken into consideration to separate tokens, since other separators, such as hyphens, can be part of emoticons.

1.4. Emoticon detection

Next, emoticons are detected. In order to detect them, the classifier in the text all the emoticons stored in two text files, containing positive and negative emoticons, respectively.

1.5. Tokenization II (complete)

During this second tokenization phase, all the punctuation marks (including commas, semicolons and so on) are considered as separators leading to obtain the final sets of tokens for each sentence.

1.6. Interjection detection

The next step consists of detecting and labeling interjections. Those that express laughs, such as “hehehe” or “hahaha”, are marked as positive whereas interjections such as “aaah” or “aaargh” are marked as negative. This is implemented through regular expressions, because in most of the cases, the interjections are intensified by repeating letters or sets of letters contained in the own word.

1.7. Token score assignation

The next phase consists of assigning a score to each token: 1 to 5 if it transmits a positive sentiment, 0 if it is neutral, and -1 to -5 if it is negative. To assign a score, the classifier checks if the token is a positive/ negative emoticon, a positive/negative interjection, or whether it matches one of the words stored in the sentiment lexicon (L).

1.8. Removing repetitive letters

The next step is, for each token, to check whether it appears in any of the two dictionary categories and, if it is the case, to tag it as either positive or negative, accordingly. The messages written by users in Facebook usually contain very casual language. It is frequent to find words with repeated letters or with non-alphabetic characters. Consequently, for each token, if it was not found in the dictionary in the form in which it appears in the message, letters occurring more than twice in a row are replaced by only one occurrence and the new token is looked for in

the dictionary. Was this version of the token not found in the dictionary yet, then it is reduced to its lexeme.

1.9. Spelling checking

If the token does not match any word yet, then it is checked with a spelling checker. Since Facebook messages usually contain misspellings, a spelling checker is incorporated into the classifier. However, the spelling checker must be applied carefully, since some of the corrections it suggests produce bad results in the classifier. With the purpose of avoiding these situations, a list of words that should not be corrected, including names and surnames, is created and incorporated within the dictionary, so that, since they are found in the dictionary, they are not checked by the spelling checker. Finally, if a token is not classified into positive/ negative in any of the previous steps (even after all those considerations) the token is labeled as neutral.

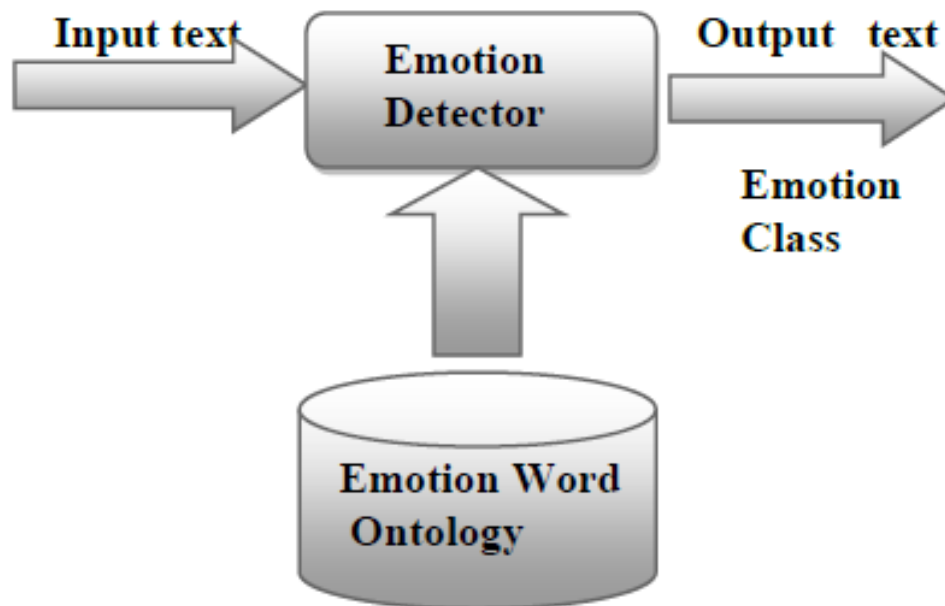
1.10. Syntactical analysis

Once each token has received a positive/neutral/negative score, each sentence is syntactically analyzed in order to check whether any score (positive/negative) should be reversed (e.g., because of negations). Firstly, we apply part of speech (POS) tagging to discriminate words that do not reflect any sentiment (e.g. articles) and to disambiguate words with multiple semantic meaning (e.g., words that can be both a noun and a verb). Afterwards, suffixes are analyzed. And then, negations are detected.

1.11. Polarity calculation

In order to calculate the polarity of a sentence, the number of tokens susceptible of conveying sentiments according to their grammatical category (i.e. noun, adjective, interjection or verb) is calculated. Other types of words, which appear frequently in texts (i.e. determinant, prepositions, etc.), are not considered, because they are “stopwords” for a sentiment analysis. Then, once each token has been scored, the final polarity score of a sentence is calculated.

2. Method II



The Framework is divided into two main components: Emotion Ontology, Emotion Detector.

2.1. Emotion Ontology

Ontology is an explicit specification of conceptualization. Ontologies have definitional aspects like high level schemas and aspects like entities and attributes; interrelationship is between entities, domain vocabulary. Ontologies provide an understanding of particular domain. Ontologies allow the domain to be communicated between persons, institutions, and application systems. Emotion word hierarchy is converted into ontology. An ontology development tool is used to develop emotion ontology. The ontology has class and subclass relationship format. Emotion classes at the primary level in emotion hierarchy are at the top of emotion ontology and emotion classes at the tertiary level are at the bottom of ontology. High weight age is assigned to the upper level emotion classes and low to the lower level emotion classes.

2.2. Emotion Detector Algorithm

Emotion of the textual data can be recognized with the help of this emotion detection algorithm. The algorithm calculates weight for particular emotion by adding weights

assigned at each level of hierarchy and also calculates same for its counter emotion, then compares the both scores and greater one is taken as the detected emotion.

2.3. Parameters Used

Algorithm is to calculate weight age to be assigned to different emotion words so that they can be sorted according to it. Certain parameters are required for this purpose. The first step is calculation of parameters. This task is achieved with the help of Jena library which allows traversal and parsing of ontology.

Different parameters are calculated as follows:

Parent-Child relationship

If a text document belongs to a child; it also indirectly refers to the parent of it. Hence if a certain value is added to the child's score, parent score also need to be modified. This is achieved by traversing the ontology model in a breadth first manner using Jena API. When any node is encountered all of its children are retrieved. Then same method is applied to every child.

Depth in Ontology

This is required as it gives an idea about how specific is the term in relation to its corresponding ontology structure. The more specific it is the more weight age should be given to it. This value is calculated simultaneously while traversing the ontology tree.

Frequency in Text document

This is also an important parameter as more is the frequency more will be the importance of that term. This value is calculated by parsing the text document and searching for occurrences of the words.

2.4 Algorithm

For each word in input text **i**

```
{
    For each word in emotion lexicon e
    {
        Calculate length of i;
        compare each letter of i with each letter of e;
        if((no. of letters successfully compared=length of i)&(next
letter of e is an alphabet))
            go to next input word;
```

```

        if((no. of letters successfully compared=length of i)|(next
letter of e is '*'))
            { assign score of e to score of i;
              go to next input word;
            }
        }
    }
}

```

3. Approaches to Building Sentiment Dictionaries

The computational speed and efficiency of dictionary-based approaches to sentiment analysis, together with their intuitive appeal, make such approaches an attractive alternative for extracting emotional context from text. At the same time, both types of dictionary based approaches offer potential limitations as well. Pre-constructed dictionaries for use with modern standard U.S. English have the advantage of being exceptionally easy to use and extensively validated, making them strong contenders for applications where the emotional content of the language under study is expressed in conventional ways. At the same time, the validity of such dictionaries rests critically on such conventional usage of emotional words and phrases. Conversely, custom dictionaries developed for specific contexts are sensitive to variations in word usage, but come with a high cost of creation and limited future applicability.

What we term specialized vocabularies arise in situations when the standard emotional valences associated with particular words are no longer correct, either because words that typically convey emotional content do not do so in the context in question or vice versa. For example, in colloquial English the word “love” almost always carries a positive valence (and its inclusion in pre-constructed sentiment dictionaries reflects this fact) while the word “bagel” does not. For professional and amateur tennis players, however, the two words might mean something very different; “love” means no points scored (a situation which has, if anything, a negative valence) and the word “bagel” refers specifically to the (negative) event of losing a set 6-0 (e.g., “putting up a bagel in the first set”). It is easy to see how the application of a standard sentiment dictionary to a body of text generated from a discussion of tennis could easily lead to inaccurate inferences about its content.

In such circumstances, an ideal approach is to develop a sentiment dictionary that reflects the emotional valence of the words as they are used in that context. Such dictionaries reflect the emotional valence of the language as it is used in context, and so are more likely to yield accurate estimates of sentiment in specialized vocabularies. Such dictionaries, however, are also difficult and time-consuming to construct, since they typically involve specifying every emotionally-valenced word or phrase that could be encountered in that context. The challenge, then, is to develop an approach for building sentiment dictionaries in the context of specialized vocabularies that is substantially more efficient and less costly than simple human coding.

4. Code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#define SIZE 2557
#define SMILE 112
struct{
char word[40];
char score[4];

} rec[SIZE];
struct{
char smi[25];
char score[4];

} emo[SMILE];
int main()
{

int i,j,k,l,count,pos,neg,tot;
signed int s;
FILE *in_file;
```

```

char sent[25][15];
char nega[16][15];
int negs=0;
char in[200];
int len;
int temp;
int words;
int sc[25];
int scor[25];

in_file = fopen("neg.txt", "r");
if (in_file == NULL) {
    fprintf(stderr, "Could not open file\n");
    exit(8);
}
for (i=0;i<17;i++)
{
    fscanf(in_file, "%15[^\n]",nega[i]);
    // printf("Name of Word:%s\n",nega[i]);

}
fclose(in_file);

in_file = fopen("emo.txt", "r");
if (in_file == NULL) {
    fprintf(stderr, "Could not open file\n");
    exit(8);
}
for (i=0;i<=SIZE;i++)
{
    fscanf(in_file, "%39[^\n],%s",rec[i].word, rec[i].score);
    //printf("Name of Word:%s - Score:%s\n",rec[i].word, rec[i].score);

}

```

```

fclose(in_file);
in_file = fopen("smile.txt", "r");
if (in_file == NULL) {
    fprintf(stderr, "Could not open file\n");
    exit(8);
}
for (i=0;i<=SMILE;i++)
{
    fscanf(in_file, "%24[^k]k%s",emo[i].smi, emo[i].score);
    // printf("Name of Word:%s - Score:%s\n",emo[i].smi, emo[i].score);

}
fclose(in_file);
for(i=0;i<25;i++)
{ for(j=0;j<15;j++)
    sent[i][j]='0';
}
for(i=0;i<25;i++)
    sc[i]=0;

for(i=0;i<25;i++)
    scor[i]=0;

printf("Enter the sentence to be evaluated\n\n");
scanf ("%s", in);

len=strlen(in);
i=0; j=0;
for(k=0;k<len;k++)
{ if(in[k]!=' ')
    { sent[i][j]=in[k];
    // printf("%c", sent[i][j]);
    j++;
}
}

```

```

else
{ i++;
  j=0;
}
}
//for(i=0;i<15;i++)
//printf("%c %d\n",nega[1][i],i);

words=i;

//negations
i=0;
while(i<=words)
{
  for(k=0;k<17;k++)
  { j=0;
    for(l=0;l<15;l++)
    { if(sent[i][l]=='0'||sent[i][l]=='!'||sent[i][l]=='.'||sent[i][l]==',')
      break;
    }

    count=1;

    while(sent[i][j]==nega[k][count])
    {
      j++;
      count++;
    }

    if((j==1)&&((int)nega[k][count]>=97&&(int)nega[k][count]<=122))
    continue;
    if(j==1)
    {

```

```

    negs++;

    break;
}
}
i++;
}
//if(negs!=1&&negs!=0)
//negs++;

//emotion words
i=0;
while(i<=words)
{
    if((sent[i][0]=='n'&&
sent[i][1]=='e'&&
sent[i][2]=='v'&&
sent[i][3]=='e'&&
sent[i][4]=='r'&&
sent[i][5]=='0')||
(sent[i][0]=='n'&&
sent[i][1]=='o'&&
sent[i][2]=='t'&&
sent[i][3]=='0'))
    { i++;
      continue;
    }
for(k=0;k<SIZE;k++)
{ j=0;
  for(l=0;l<15;l++)
  { if(sent[i][l]=='0'||sent[i][l]=='!'||sent[i][l]=='.'||sent[i][l]==',')
    break;
  }
}
}

```

```

count=0;
while((int)rec[k].word[count]<97||((int)rec[k].word[count]>122)
count++;

while(sent[i][j]==rec[k].word[count])
{
j++;
count++;
}

if((j==1)&&((int)rec[k].word[count]>=97&&(int)rec[k].word[count]<=122))
continue;
if(j==1||rec[k].word[count]=='*')
{
if(rec[k].score[0]!='-')
s=rec[k].score[0]-'0';
else
s=0-(rec[k].score[1]-'0');
sc[i]=s;

break;
}
}
i++;
}

//emoticons

i=0;
while(i<=words)
{

```



```

if((int)sent[i][0]>=97&&(int)sent[i][0]<=122)
{ //printf("%d\n",i);
  i++;
  continue;
}
for(k=0;k<SMILE;k++)
{ j=0;

    for(l=0;l<15;l++)
    { if(sent[i][l]=='0')
      break;
    }
// printf("%d",l);
  count=1;
  /*
  while(emo[k].smi[count]!='% '||
emo[k].smi[count]!='('||
emo[k].smi[count]!=')'||
emo[k].smi[count]!='* '||
emo[k].smi[count]!='-'||
emo[k].smi[count]!='3 '||
emo[k].smi[count]!='8 '||
emo[k].smi[count]!=':'||
emo[k].smi[count]!=';'||
emo[k].smi[count]!='<'||
emo[k].smi[count]!='='||
emo[k].smi[count]!='>'||
emo[k].smi[count]!='@ '||
emo[k].smi[count]!='B '||
emo[k].smi[count]!='D '||
emo[k].smi[count]!='X '||
emo[k].smi[count]!='x '||
emo[k].smi[count]!=' '||

```

```

    emo[k].smi[count]!='}'||
    emo[k].smi[count]!='^'
    )
    count++;
*/

    while(sent[i][j]==emo[k].smi[count])
    {
        j++;
        count++;
    }

// if((j==1)&&((int)rec[k].word[count]>=97&&(int)rec[k].word[count]<=122))
// continue;
    //if(j==1||rec[k].word[count]=='*')
    if(j==1)
    {
        if(emo[k].score[0]!='-')
            s=emo[k].score[0]-'0';
        else
            s=0-(emo[k].score[1]-'0');
        scor[i]=s;

        break;
    }
    }
    i++;
}

for(i=0;i<=words;i++)
{ j=0;

```

```

    while(sent[i][j]!='0')
    {
    printf("%c",sent[i][j]);
    j++;
    }
    printf("(%d) ",sc[i]+scor[i]);
    if(i==words)
    break;
}
printf("\n\nNo. of negations: %d\n\n",negs);

```

```

pos=0;
neg=0;
tot=0;
for(i=0;i<=words;i++)
{ if(sc[i]>=0)
pos=pos+sc[i];
else
neg=neg+sc[i];
}
if(negs%2!=0)
{ temp=pos;
pos=-1*neg;
neg=-1*temp;
}
for(i=0;i<=words;i++)
{ if(scor[i]>=0)
pos=pos+scor[i];
else
neg=neg+scor[i];
}

```

```
printf("\n\nThe ratings are on a scale of -5(strongly -ve) to 0(neutral) to 5(strongly
+ve)");
printf("\n\nPositive emotion rating is: %d\nNegative emotion rating is: %d",pos,neg);
    printf("\n\nTotal score: %d",pos+neg);
    if(pos>0-neg)
        printf("\n\nThe sentence is positive");
    else if(0-neg>pos)
        printf("\n\nThe sentence is negative");
    else
        printf("\n\nThe sentence is neutral");
    getch();
    return (0);
}
```

CHAPTER 6:
NAIVE BAYES CLASSIFIER

Simple Bayesian classifiers have been gaining popularity lately, and have been found to perform surprisingly well. These probabilistic approaches make strong assumptions about how the data is generated, and posit a probabilistic model that embodies these assumptions; then they use a collection of labeled training examples to estimate the parameters of the generative model. Classification on new examples is performed with Bayes' rule by selecting the class that is most likely to have generated the example. The naive Bayes classifier is the simplest of these models, in that it assumes that all attributes of the examples are independent of each other given the context of the class. This is the so-called "Naive Bayes assumption." While this assumption is clearly false in most real-world tasks, naive Bayes often performs classification very well. This paradox is explained by the fact that classification estimation is only a function of the sign (in binary cases) of the function estimation; the function approximation can still be poor while classification accuracy remains high. Because of the independence assumption, the parameters for each attribute can be learned separately, and this greatly simplifies learning, especially when the number of attributes is large. Document classification is just such a domain with a large number of attributes. The attributes of the examples to be classified are words, and the number of different words can be quite large indeed. While some simple document classification tasks can be accurately performed with vocabulary sizes less than one hundred, many complex tasks on real-world data from the Web, UseNet and newswire articles do best with vocabulary sizes in the thousands. Naive Bayes has been successfully applied to document classification in many research efforts. Despite its popularity, there has been some confusion in the document classification community about the "Naive Bayes" classifier because there are two different generative models in common use, both of which make the "Naive Bayes assumption." Both are called "Naive Bayes" by their practitioners. One model specifies that a document is represented by a vector of binary attributes indicating which words occur and do not occur in the document. The number of times a word occurs in a document is not captured. When calculating the probability of a document, one multiplies the probability of all the attribute values, including the probability of non-occurrence for words that do not occur in the document. Here we can understand the document to be the "event," and the absence or presence of words to be attributes of the event. This describes a distribution based on a multi-variate Bernoulli event model. This approach is more traditional in the field of Bayesian networks, and is appropriate for tasks that

have a fixed number of attributes. The approach has been used for text classification by numerous people. The second model specifies that a document is represented by the set of word occurrences from the document. As above, the order of the words is lost; however, the number of occurrences of each word in the document is captured. When calculating the probability of a document, one multiplies the probability of the words that occur. Here we can understand the individual word occurrences to be the “events” and the document to be the collection of word events. We call this the multinomial event model. This approach is more traditional in statistical language modeling for speech recognition, where it would be called a “unigram language model.” This approach has also been used for text classification by numerous people. Results indicate that the multi-variate Bernoulli model sometimes performs better than the multinomial at small vocabulary sizes. However, the multinomial usually outperforms the multi-variate Bernoulli at large vocabulary sizes, and almost always beats the multi-variate Bernoulli when vocabulary size is chosen optimally for both. While sometimes the difference in performance is not great, on average across data sets, the multinomial provides a 27% reduction in error over the multi-variate Bernoulli.

1. Probabilistic Framework of Naive Bayes

Consider the task of text classification in a Bayesian learning framework. This approach assumes that the text data was generated by a parametric model, and uses training data to calculate Bayes-optimal estimates of the model parameters. Then, equipped with these estimates, it classifies new test documents using Bayes’ rule to turn the generative model around and calculate the posterior probability that a class would have generated the test document in question. Classification then becomes a simple matter of selecting the most probable class.

Both scenarios assume that text documents are generated by a mixture model parameterized by θ . The mixture model consists of mixture components $c_j \in C = \{c_1, \dots, c_{|C|}\}$. Each component is parameterized by a disjoint subset of θ . Thus a document, d_i , is created by (1) selecting a component according to the priors, $P(c_j | \theta)$, then (2) having the mixture component generate a document according to its own parameters, with distribution $P(d_i | c_j; \theta)$. We can characterize the likelihood of a document with a sum of total probability over all mixture components:

$$P(d_i|\theta) = \sum_{j=1}^{|\mathcal{C}|} P(c_j|\theta)P(d_i|c_j; \theta). \quad (1)$$

Each document has a class label. We assume that there is a one-to-one correspondence between classes and mixture model components, and thus use c_j to indicate both the j^{th} mixture component and the j^{th} class. In this setting, (supervised learning from labeled training examples), the typically “hidden” indicator variables for a mixture model are provided as these class labels.

1.1 Multi-variate Bernoulli Model

In the multi-variate Bernoulli event model, a document is a binary vector over the space of words. Given a vocabulary V , each dimension of the space t , $t \in \{1, \dots, |V|\}$, corresponds to word w_t from the vocabulary. Dimension t of the vector for document d_i is written B_{it} , and is either 0 or 1, indicating whether word w_t occurs at least once in the document. With such a document representation, we make the naive Bayes assumption: that the probability of each word occurring in a document is independent of the occurrence of other words in a document. Then, the probability of a document given its class from Equation 1 is simply the product of the probability of the attribute values over all word attributes:

$$P(d_i|c_j; \theta) = \prod_{t=1}^{|V|} (B_{it}P(w_t|c_j; \theta) + (1 - B_{it})(1 - P(w_t|c_j; \theta))). \quad (2)$$

Thus given a generating component, a document can be seen as a collection of multiple independent Bernoulli experiments, one for each word in the vocabulary, with the probabilities for each of these word events defined by each component, $P(w_t|c_j; \theta)$. This is equivalent to viewing the distribution over documents as being described by a Bayesian network, where the absence or presence of each word is dependent only on the class of the document.

Given a set of labeled training documents, $D = \{d_1, \dots, d_{|D|}\}$, learning the parameters of a probabilistic classification model corresponds to estimating each of these class-conditional word probabilities. The parameters of a mixture

component are written $\theta_{w_t|c_j} = P(w_t|c_j; \theta)$, where $0 \leq \theta_{w_t|c_j} \leq 1$. We can calculate Bayes-optimal estimates for these probabilities by straightforward counting of events, supplemented by a prior. We use the Laplacean prior, priming each word's count with a count of one to avoid probabilities of zero or one. Define $P(c_j|d_i) \in \{0, 1\}$ as given by the document's class label. Then, we estimate the probability of word w_t in class c_j with:

$$\hat{\theta}_{w_t|c_j} = P(w_t|c_j; \theta) = \frac{1 + \sum_{i=1}^{|\mathcal{D}|} B_{it} P(c_j|d_i)}{2 + \sum_{i=1}^{|\mathcal{D}|} P(c_j|d_i)}. \quad (3)$$

The class prior parameters, θ_{c_j} , are set by the maximum likelihood estimate:

$$\hat{\theta}_{c_j} = P(c_j|\hat{\theta}) = \frac{\sum_{i=1}^{|\mathcal{D}|} P(c_j|d_i)}{|\mathcal{D}|}. \quad (4)$$

Note that this model does not capture the number of times each word occurs, and that it explicitly includes the non-occurrence probability of words that do not appear in the document

1.2 Multinomial Model

In contrast to the multi-variate Bernoulli event model, the multinomial model captures word frequency information in documents. Consider, for example, the occurrence of numbers in the Reuters newswire articles; our tokenization maps all strings of digits to a common token. Since every news article is dated, and thus has a number, the number token in the multi-variate Bernoulli event model is uninformative. However, news articles about earnings tend to have a lot of numbers compared to general news articles. Thus, capturing frequency information of this token can help classification. In the multinomial model, a document is an ordered sequence of word events, drawn from the same vocabulary V . We assume that the lengths of documents are independent of class. We again make a similar Naive Bayes assumption: that the probability of each word event in a document is independent of the word's context and position in the document. Thus, each document d_i is drawn from a multinomial distribution of words with as many independent trials as the length of d_i . This yields the familiar "bag of words" representation for documents. Define N_{it} to be the count of the

number of times word w_t occurs in document d_i . Then, the probability of a document given its class from Equation 1 is simply the multinomial distribution:

$$P(d_i|c_j; \theta) = P(|d_i|) |d_i|! \prod_{t=1}^{|V|} \frac{P(w_t|c_j; \theta)^{N_{it}}}{N_{it}!}. \quad (5)$$

2. Classification

Given estimates of these parameters calculated from the training documents, classification can be performed on test documents by calculating the posterior probability of each class given the evidence of the test document, and selecting the class with the highest probability. We formulate this by applying Bayes' rule:

$$P(c_j|d_i; \hat{\theta}) = \frac{P(c_j|\hat{\theta})P(d_i|c_j; \hat{\theta}_j)}{P(d_i|\hat{\theta})}. \quad (7)$$

The right hand side may be expanded by first substituting using Equations 1 and 4. Then the expansion of individual terms for this equation is dependent on the event model used. Use Equations 2 and 3 for the multi-variate Bernoulli event model. Use Equations 5 and 6 for the multinomial.

3. Code for training dataset

```
import weka.core.Instances;
import weka.filters.Filter;
import weka.filters.unsupervised.attribute.StringToWordVector;
import weka.classifiers.Evaluation;
import java.util.Random;
import weka.classifiers.bayes.NaiveBayes;
import weka.classifiers.meta.FilteredClassifier;
import weka.core.converters.ArffLoader.ArffReader;
import java.io.*;

public class text {
```

```

Instances trainData;

StringToWordVector filter;

FilteredClassifier classifier;

public void loadDataset(String fileName) {
    try {
        BufferedReader reader = new BufferedReader(new
FileReader(fileName));
        ArffReader arff = new ArffReader(reader);
        trainData = arff.getData();
        System.out.println("==== Loaded dataset: " + fileName + "
====");
        reader.close();
    }
    catch (IOException e) {
        System.out.println("Problem found when reading: " +
fileName);
    }
}

public void evaluate() {
    try {
        trainData.setClassIndex(0);
        filter = new StringToWordVector();
        filter.setAttributeIndices("last");
        classifier = new FilteredClassifier();
        classifier.setFilter(filter);
        classifier.setClassifier(new NaiveBayes());
        Evaluation eval = new Evaluation(trainData);
    }
}

```

```

        eval.crossValidateModel(classifier, trainData, 4, new
Random(1));
        System.out.println(eval.toSummaryString());
        System.out.println(eval.toClassDetailsString());
        System.out.println("=====  

dataset done =====");
    }
    catch (Exception e) {
        System.out.println("Problem found when evaluating");
    }
}

```

```

public void learn() {
    try {
        trainData.setClassIndex(0);
        filter = new StringToWordVector();
        filter.setAttributeIndices("last");
        classifier = new FilteredClassifier();
        classifier.setFilter(filter);
        classifier.setClassifier(new NaiveBayes());
        classifier.buildClassifier(trainData);

        System.out.println(classifier);
        System.out.println("=====  

dataset done =====");
    }
    catch (Exception e) {
        System.out.println("Problem found when training");
    }
}

```

```

public void saveModel(String fileName) {

```

```

        try {
            ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream(fileName));
            out.writeObject(classifier);
            out.close();

            System.out.println("==== Saved model: " + fileName + "
====");
        }
        catch (IOException e) {
            System.out.println("Problem found when writing: " +
fileName);
        }
    }

    public static void main (String[] args) {

        text learner;

        learner = new text();
        learner.loadDataset("C:\\Users\\Vadehra\\Desktop\\sms.txt");

        learner.evaluate();
        learner.learn();
        learner.saveModel("classy.dat");

    }
}

```

4. Code for testing data

```

import weka.core.*;
import weka.core.FastVector;
import weka.classifiers.meta.FilteredClassifier;

```

```

import java.util.List;
import java.util.ArrayList;
import java.io.*;

public class text2 {

    String text;

    Instances instances;

    FilteredClassifier classifier;

    public void load(String fileName) {
        try {
            BufferedReader reader = new BufferedReader(new
FileReader(fileName));
            String line;
            text = "";
            while ((line = reader.readLine()) != null) {
                text = text + " " + line;
            }

            System.out.println("==== Loaded text data: " + fileName + "
====");

            reader.close();
            System.out.println(text);
        }
        catch (IOException e) {
            System.out.println("Problem found when reading: " +
fileName);
        }
    }
}

```

```

public void loadModel(String fileName) {
    try {
        ObjectInputStream in = new ObjectInputStream(new
FileInputStream(fileName));
        Object tmp = in.readObject();
        classifier = (FilteredClassifier) tmp;
        in.close();
        System.out.println("==== Loaded model: " + fileName + "
====");
    }
    catch (Exception e) {
        System.out.println("Problem found when reading: " +
fileName);
    }
}

```

```

public void makeInstance() {
    FastVector fvNominalVal = new FastVector(2);
    fvNominalVal.addElement("spam");
    fvNominalVal.addElement("ham");
    Attribute attribute1 = new Attribute("class", fvNominalVal);
    Attribute attribute2 = new Attribute("text", (FastVector) null);
    FastVector fvWekaAttributes = new FastVector(2);
    fvWekaAttributes.addElement(attribute1);
    fvWekaAttributes.addElement(attribute2);
    instances = new Instances("Test relation", fvWekaAttributes, 1);
    instances.setClassIndex(0);
    Instance instance = new Instance(2);
    instance.setValue(attribute2, text);
    instances.add(instance);
    System.out.println("==== Instance created with reference dataset
====");
}

```

```

        System.out.println(instances);
    }

    public void classify() {
        try {
            double pred = classifier.classifyInstance(instances.instance(0));
            System.out.println("===== Classified instance =====");
            System.out.println("Class      predicted:      "      +
instances.classAttribute().value((int) pred));
        }
        catch (Exception e) {
            System.out.println("Problem found when classifying the text");
        }
    }

    public static void main (String[] args) {

        text2 classifier;
    {
        classifier = new text2();
        classifier.load("C:\\Users\\Vadehra\\Desktop\\smstest.txt");
        classifier.loadModel("classy.dat");
        classifier.makeInstance();
        classifier.classify();
    }
    }

```

CHAPTER 7:
TECHNICAL CHALLENGES

-
- When inspecting the sentences misclassified by my project, I can see that most of the classification errors are related to the underlying ambiguity of the natural language. In the following examples, I have first presented sentences that were incorrectly labeled as negative. In the second case, there are sentences incorrectly classified as having a positive sentiment by the model. All these examples demonstrate the complexity of all natural language and the need for developing language specific heuristics to better capture phraseological expressions, contrasting statements, sarcasm, and allusions made by the writer

Positive sentences classified as negative.

1. “Longley has constructed a remarkably coherent, horrifically vivid snapshot of those turbulent days.”
2. “Romanek keeps the film constantly taut... reflecting the character's instability with a metaphorical visual style and an unnerving, heartbeat-like score.”
3. “Compelling revenge thriller, though somewhat weakened by a miscast leading lady.”

Negative sentences classified as positive.

1. “In the book-on-tape market, the film of "the kid stays in the picture" would be an abridged edition.”
2. “A mechanical action-comedy whose seeming purpose is to market the charismatic Jackie Chan to even younger audiences.”
3. “It's not so much a movie as a joint promotion for the national basketball association and teenaged rap and adolescent poster-boy lil' bow wow.”
4. “Director Tom Dey demonstrated a knack for mixing action and idiosyncratic humor in his charming 2000 debut shanghai noon, but showtime's uninspired send-up of tv cop show cliches mostly leaves him shooting blanks.”

- Another technical challenge is the inability to detect and rectify textese, or SMS language, such as that used on Twitter.
- Using emotion keywords is a straightforward way to detect associated emotions, the meanings of keywords could be multiple and vague, as most words could change their meanings according to different usages and contexts. Moreover, even the minimum set of emotion labels (without all their

synonyms) could have different emotions in some extreme cases such as ironic or cynical sentences.

- Sentences without any keyword would imply that they do not contain any emotion at all, which is obviously wrong. For example, “I passed my qualify exam today” and “Hooray! I passed my qualify exam today” should imply the same emotion (joy), but the former without “hooray” could remain undetected if “hooray” is the only keyword to detect this emotion.
- Syntax structures and semantics also have influences on expressed emotions. For example, “I laughed at him” and “He laughed at me” would suggest different emotions from the first person’s perspective. As a result, ignoring linguistic information also poses a problem to keyword-based methods.

CHAPTER 8: SCREENSHOTS

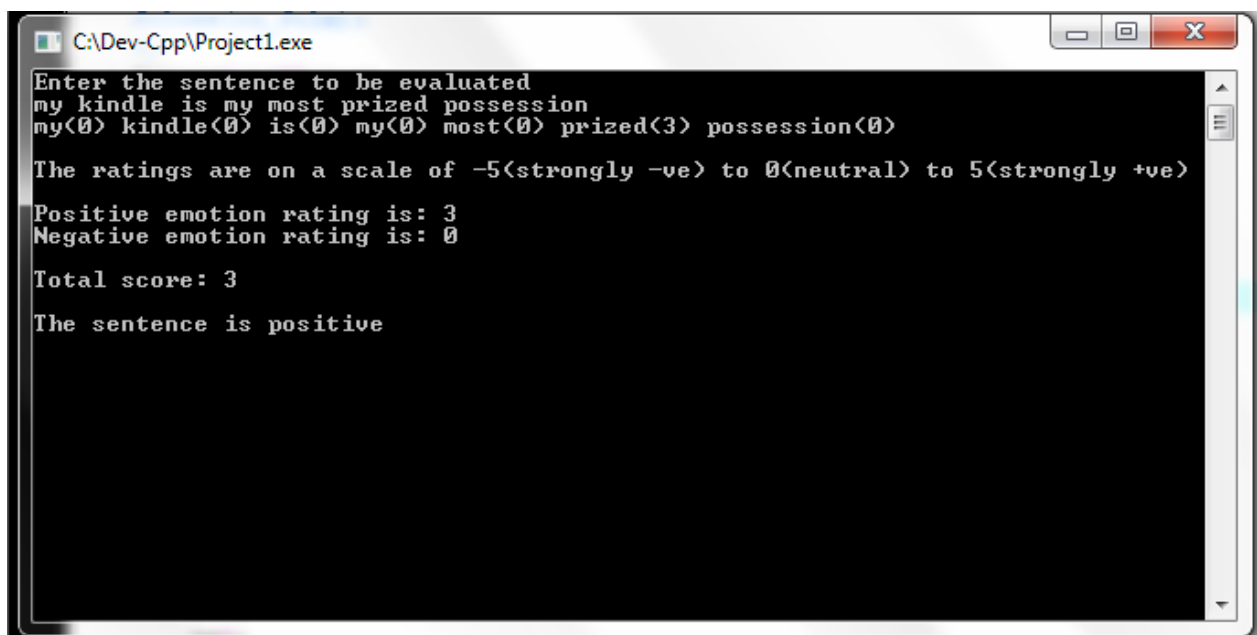
1. Lexicon Approach



```
C:\Dev-Cpp\Project1.exe
Enter the sentence to be evaluated
i'm anxious but hopeful
i'm<0> anxious<-3> but<0> hopeful<2>

The ratings are on a scale of -5<strongly -ve> to 0<neutral> to 5<strongly +ve>
Positive emotion rating is: 2
Negative emotion rating is: -3

Total score: -1
The sentence is negative_
```



```
C:\Dev-Cpp\Project1.exe
Enter the sentence to be evaluated
my kindle is my most prized possession
my<0> kindle<0> is<0> my<0> most<0> prized<3> possession<0>

The ratings are on a scale of -5<strongly -ve> to 0<neutral> to 5<strongly +ve>
Positive emotion rating is: 3
Negative emotion rating is: 0

Total score: 3
The sentence is positive
```

```
C:\Dev-Cpp\Project1.exe
Enter the sentence to be evaluated
alas! this was an absurd story
alas!<-2> this<0> was<0> an<0> absurd<-2> story<0>

The ratings are on a scale of -5<strongly -ve> to 0<neutral> to 5<strongly +ve>
Positive emotion rating is: 0
Negative emotion rating is: -4
Total score: -4
The sentence is negative
```

```
C:\Dev-Cpp\Project1.exe
Enter the sentence to be evaluated
he suffers from an incurable disease
he<0> suffers<-4> from<0> an<0> incurable<-2> disease<-3>

The ratings are on a scale of -5<strongly -ve> to 0<neutral> to 5<strongly +ve>
Positive emotion rating is: 0
Negative emotion rating is: -9
Total score: -9
The sentence is negative_
```

2. Machine Learning Approach

```
Problems @ Javadoc Declaration Console
<terminated> text [Java Application] C:\Program Files\Java\jdk1.8.0_05\bin\javaw.exe (May 13, 2015, 3:35:44 PM)
Usage: java MyLearner <fileData> <fileModel>
===== Loaded dataset: C:\Users\Vadehra\Desktop\try.txt =====

Correctly Classified Instances      145          85.7988 %
Incorrectly Classified Instances    24           14.2012 %
Kappa statistic                     0.6766
Mean absolute error                 0.1408
Root mean squared error             0.359
Relative absolute error             31.9983 %
Root relative squared error         76.599 %
Total Number of Instances          169

=== Detailed Accuracy By Class ===

      TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
      0.895   0.218   0.895     0.895   0.895     0.939   positive
      0.782   0.105   0.782     0.782   0.782     0.939   negative
Weighted Avg.   0.858   0.181   0.858     0.858   0.858     0.939

===== Evaluating on filtered (training) dataset done =====
===== Training on filtered (training) dataset done =====
===== Saved model: classy.dat =====
```

The model used for training the dataset has given an accuracy of 85% on 169 instances.

```
Problems @ Javadoc Declaration Console
<terminated> text2 [Java Application] C:\Program Files\Java\jdk1.8.0_05\bin\javaw.exe (May 13, 2015, 3:50:25 PM)
===== Loaded text data: C:\Users\Vadehra\Desktop\smstest.txt =====
I agree with Capital punishment today we find criminality is increasing very rapidly particularly increasing rate of rapes and murders
===== Loaded model: classy.dat =====
===== Instance created with reference dataset =====
@relation 'Test relation'

@attribute text string
@attribute class {positive,negative}

@data
'I agree with Capital punishment today we find criminality is increasing very rapidly particularly increasing rate of rapes and murders
===== Classified instance =====
Class predicted: positive
```

The model correctly classified this test instance as positive.

```
Problems @ Javadoc Declaration Console
<terminated> text2 [Java Application] C:\Program Files\Java\jdk1.8.0_05\bin\javaw.exe (May 13, 2015, 3:53:16 PM)
===== Loaded text data: C:\Users\Vadehra\Desktop\smstest.txt =====
I think capital or death punishment should be put to an end this is a topic going in global world and most of the countries are in favor o
===== Loaded model: classy.dat =====
===== Instance created with reference dataset =====
@relation 'Test relation'

@attribute text string
@attribute class {positive,negative}

@data
' I think capital or death punishment should be put to an end this is a topic going in global world and most of the countries are in favor o
===== Classified instance =====
Class predicted: negative
```

The model correctly classified this test instance as negative

```
Problems @ Javadoc Declaration Console
<terminated> text2 [Java Application] C:\Program Files\Java\jdk1.8.0_05\bin\javaw.exe (May 13, 2015, 3:55:24 PM)
===== Loaded text data: C:\Users\Vadehra\Desktop\smstest.txt =====
As rightly said An eye for eye makes the whole world blind Those who commit rape or heinous crimes like that need more suffering than death My point of
===== Loaded model: classy.dat =====
===== Instance created with reference dataset =====
@relation 'Test relation'

@attribute text string
@attribute class {positive,negative}

@data
' As rightly said An eye for eye makes the whole world blind Those who commit rape or heinous crimes like that need more suffering than death My point of
===== Classified instance =====
Class predicted: negative
```

The model correctly classified this test instance as negative

**CHAPTER 9:
CONCLUSION**

Emotion Detection can be seen as an important field of research in human-computer interaction. A sufficient amount of work has been done by researchers to detect emotion from facial and audio information whereas recognizing emotions from textual data is still a fresh and hot research area.

The report demonstrates that it is feasible to extract information about a person's sentiments from the text they write on social networking sites with high accuracy. The method applied supports to get information about the users' sentiment polarity(positive, neutral or negative) according to the messages they write.

The classification method implemented follows a lexical-based approach.

Adaptive and recommendation systems in general can take advantage of knowing the users' sentiments at a certain time, as well as significant emotional changes with respect to their usual state. However, asking the users about their sentiments is intrusive, can be bothering and, furthermore, in some contexts there is a high probability that they do not admit negative sentiments, because of, e.g., their cultural background or the need of social approval. In the educational context, in particular, it is not highly probable that a student directly transmits the teacher his/her feelings towards a subject or methodology when they are negative. The process to get this information is usually harder (e.g., anonymous/ distinctive surveys at the end of the semester, or teachers receiving comments through class delegates during the course, at most). The work presented in this report demonstrates that it is possible to extract it from the messages the students write on social networking sites.

This can be considered as input for adaptive e-learning systems to provide sentiment-based adaptation, making it possible, e.g., to recommend the most suitable activities to be performed by each student at a certain time according to his/her sentiment at that time; another application of this work in the learning context deals with extracting feedback for teachers about the sentiments of their students towards their courses or teaching methodologies.

**CHAPTER 10:
FUTURE WORK**

-
- One potential improvement to the project will be the ability to detect not just the user's emotions but also the significant emotional changes over a period of time, like a day or a week. This can be useful in adaptive E-learning systems to assess what effects the subjects taught are having on the student's state of mind.
 - Another improvement is the ability to rectify and detect textese.
 - One other change could be the ability to detect the emotion behind a word based on the context it has been used in.
 - The emotion of a word also changes based on its position in the sentence. A potential improvement would be to take this position under account.

CHAPTER 11: REFERENCES

-
- [1] Walaa Medhat, Ahmed Hassan and Hoda Korashy, “**Sentiment analysis algorithms and applications: A survey**”, Ain Shams Engineering Journal, April 2014
- [2] Alvaro Ortigosa, José M. Martín and Rosa M. Carro, “**Sentiment analysis in Facebook and its application to e-learning**”, Computers in Human Behavior, 2013
- [3] XIONG XiaoBing, ZHOU Gang, HUANG YongZhong, CHEN HaiYong and XU Ke, “**Dynamic evolution of collective emotions in social networks**”, July 2013, Vol. 56
- [4] Shiv Naresh Shivhare and Prof. Saritha Khethawat, “**Emotion Detection from Text**”, Department of CSE and IT, Maulana Azad National Institute of Technology, 2011, Bhopal, Madhya Pradesh, India
- [5] Cecilia Ovesdotter Alm, Dan Roth and Richard Spooth, “**Emotions from text: machine learning for text-based emotion prediction**”, Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), October 2005, pages 579–586
- [6] Bo Pang, Lillian Lee and Shivakumar Vaithyanathan, “**Thumbs up? Sentiment Classification using Machine Learning Techniques**”, Proceedings of EMNLP 2002, pp. 79–86
- [7] Douglas R. Rice and Christopher Zorn, “**Corpus-Based Dictionaries for Sentiment Analysis of Specialized Vocabularies**”, Prepared for presentation at the New Directions in Analyzing Text as Data Workshop, 2013, Version 0.1, September 2013
- [8] Andrew McCallum and Kamal Nigam, “**A Comparison of Event Models for Naive Bayes Text Classification**”