**A Project Report**

**On**

**SENTIMENT ANALYSIS**

**CERTIFICATE**

**CANDIDATE’S DECLARATION**

**ACKNOWLEDGEMENT**

**ABSTRACT**

Sentiments, evaluations, attitudes, and emotions are the subjects of study of sentiment analysis and creating decisions based on various opinions. The inception and rapid growth of the field coincide with those of the social media on the Web, e.g., reviews, forum discussions, blogs, micro blogs, Twitter, and social networks, because it for the first time that we have kept a record of peoples’ view in the form of tweets and post on various topics. Since early 2000, sentiment analysis has grown to be one of the most active research areas in natural language processing. It is also widely observed in data mining, Web mining, and text mining. In fact, it has spread from computer science to management sciences and social sciences due to its importance to business and society as a whole. In recent years, industrial activities surrounding sentiment analysis have also thrived. Numerous startups have emerged. Many large corporations have built their own in-house capabilities. Sentiment analysis systems have found their applications in almost every business and social domain. The goal of this report is to give an introduction to this fascinating problem and to present a framework which will perform sentiment analysis on online mobile phone reviews by associating modified K means algorithm with Naïve bayes classification, KNN.

**CONTENT**

|  |  |  |
| --- | --- | --- |
|  |  | PAGE NO. |
| CERTIFICATE |  | **II** |
| DECLARATION |  | **III** |
| ACKNOWLEDGEMENT |  | **IV** |
| ABSTRACT |  | **V** |
| CONTENTS |  | **VI** |
| LIST OF FIGURES |  | **VIII** |
| LIST OF TABLES |  | **IX** |
|  |  |  |
| Chapter 1 | **INTRODUCTION** | **1** |
| 1.1  1.1.1  1.1.2  1.1.3  1.1.4  1.1.5  1.1.6  1.1.7  1.1.8  1.1.9  1.1.10 | **Introduction**  **Major tasks in NLP**  **Machine translation**  **Named entity recognition (NER)**  **Optical character recognition (OCR)**  **Part-of-speech tagging**  **Parsing**  **Sentiment analysis**  **Information retrieval (IR)**  **Objectives**  **Different types of evaluation** | 1 |
| 1.2  1.2.1  1.2.2 | **What is Sentiment?**  **Multi-level Analysis**  **Types of sentiment analysis** | 17  18  20 |
| 1.3 | **DATA SOURCE** | 22 |
| 1.4  1.4.1 | **SENTIMENT CLASSIFICATION**  **Machine Learning** | 23  23 |
| 1.5 | **Background Information** | **28** |
| 1.6 | **MODIFIED K-MEANS** | **30** |
| 1.7  1.7.1  1.7.2  1.7.3 | **ETL**  **Extract**  **Transform**  **Load** | **33**  **33**  **34**  **35** |
| 1.8 | **Organization of Thesis** | **37** |
| Chapter 2 | **LITERATURE REVIEW** | **38** |
| 2.1  2.2  2.2.1 | Overview  Literature Survey  Research Papers | 38  38  38 |
| Chapter 3 | **PROPOSED METHOD AND IMPLEMENTATION** | **46** |
| 3.1  3.1.1 | **Proposed System**  **Framework Model** | **46**  **46** |
| 3.2 | **TOOLS AND TECHNOLOGY** | **62** |
| Chapter 4  4.1 | **RESULT AND DISCUSSION**  Comparison Table | **68**  **68** |
| Chapter 5 | **CONCLUSION** | **69** |
| REFERENCES |  | **70** |

**LIST OF FIGURES**

|  |  |
| --- | --- |
|  |  |
| FIGURE 1.1: MULTI-LEVEL ANALYSIS | 19 |
| FIGURE 1.2: MULTI-LEVEL ANALYSIS CONTD… | 19 |
| FIGURE 3.1: FRAMEWORK MODEL | 46 |
| FIGURE 3.2: INFORMATION EXTRACTION PROCESS FROM WEBPAGES. | 47 |
|  |  |

**LIST OF TABLES**

|  |  |
| --- | --- |
|  |  |
| TABLE 4.1 Comparison Table | 68 |
|  |  |

**CHAPTER 1**

**INTRODUCTION**

**1.1 Introduction**

Natural Language Processing (NLP) engages with actual text data processing. The text data is categorized into machine understandable format by NLP algorithm. Artificial Intelligence (AI) uses information generated by the NLP algorithms and applies a lot of mathematics and implementation using data libraries to determine whether something is positive or negative. There are various algorithms exist to determine an author’s view on a topic from natural language textual information. Moving towards the application part, natural language processing algorithms deals with tracking the mood of the people regarding a particular product or topic. This software prototype provides automatic extraction of opinions, emotions and sentiments in text and also tracks attitudes and feelings using a predefined libraries. People publicize their views by writing blog posts, comments, reviews and tweets about all kinds of topics. NLP can be used to track products and campaign to determine whether they are viewed positively or negatively can be done using web.

Natural language processing (NLP) is a field of computer science, artificial intelligence, and computational linguistics concerned with the interactions between computers and human (natural) languages. As such, NLP is associated to the area of human–computer interaction and analysis. Many challenges in NLP involve: natural language understanding, enabling computers to derive meaning from human or natural language input; and others involve natural language generation.

The history of NLP generally goes back to 1950s, although work can be found from earlier periods. In 1950, Alan Turing published a paper titled "Computing Machinery and Intelligence" which stated what is now called the Turing test as a criterion of intelligence. The Georgetown experiment in 1954 implicated fully automatic translation of more than sixty Russian sentences into English. The authors asserted that within three or five years, machine translation would be a solved problem. However, real progress was much gradual, and after the ALPAC report in 1966, which stated that ten-year-long research had failed to fulfill the complete expectations, funding for machine translation was vividly reduced. Further research in machine translation was conducted until the late 1980s, when the first statistical machine translation system was developed. Some notably successful NLP systems developed in the 1960s were SHRDLU, a natural language system working in restricted "blocks worlds" with restricted vocabularies, and ELIZA, a simulation of a Rogerian psychotherapist, written by Joseph Weizenbaum between 1964 and 1966. Using almost no information about human mood or emotion, ELIZA sometimes provided an amazing human-like interaction. When the "patient" exceeded the very small knowledge base, ELIZA might provide a generic response, for example, responding to "My head hurts" with "Why do you say your head hurts?". During the 1970s many programmers began to analyze and writee 'conceptual ontologies', which structured real-world information into computer-understandable data. Certain examples are MARGIE (Schank, 1975), SAM (Cullingford, 1978), PAM (Wilensky, 1978), TaleSpin (Meehan, 1976), QUALM (Lehnert, 1977), Politics (Carbonell, 1979), and Plot Units (Lehnert 1981). During this time, many chatterbots were made including PARRY, Racter, and Jabberwacky. Up to the 1980s, most NLP systems were based on complex sets of hardcoded information. Starting in the late 1980s, however, there was a rocketing revolution in NLP with the introduction and understanding of machine learning algorithms for language processing. This was due to both the gradual increase in computational power (see Moore's Law) and the steady lessening of the dominance of Chomskyan theories of linguistics (e.g. transformational grammar), whose theoretical underpinnings dismayed the sort of corpus linguistics which underlies the machine-learning approach to language processing. Few of the earliest-used machine learning algorithms, such as decision based trees, produced systems of hard if-then rules similar to existing hardcoded rules. However, Part-of-speech tagging popularized the use of Hidden Markov Models to NLP, and increasingly, research has focused on statistical models, which make soft, probabilistic decisions analyzed on attaching real-valued weights to the features making up the input data. The cache language models on which many speech recognition systems rely are examples of similar statistical models. Such models are generally more hefty when given unfamiliar input, especially input that contains grammatical errors (as is very common for real-world data), and produce more reliable results when integrated into a larger system comprising multiple subunits. Many of the notable early successes occurred in the field of machine learning and translation, due especially to work at IBM Research Centers, where successively more and more complicated statistical models were developed. These systems were able to take usage of existing multilingual textual corpora that had been produced by the Parliament of Canada and the European Union as a result of laws asking for the translation of all governmental proceedings into all official languages of the corresponding systems of government in that particular region. However, most other systems depended on corpora specifically developed for the units implemented by these systems, which was (and often continues to be) a big limitation in the success of these systems. As a result, a great deal of research has been made into methods of more effectively learning from limited amounts of data. Recent research has increasingly focused on unsupervised and semi-supervised learning algorithms. Such algorithms are able to learn from data that has not been hard coded with the desired answers, or using a combination of annotated and non-coded data. Generally, this task is much more difficult than supervised learning, and typically produces quite less accurate results for a given amount of input data. However, there is an enormous amount of non-annotated data available (including, among other things, the entire content of the World Wide Web), which can often make up for the inferior results that it creates.

Modern NLP algorithms are completely based on machine learning, especially statistical machine learning. The paradigm of machine learning is quite different from that of most prior attempts of language processing. Prior implementations of language-processing tasks generally involved the direct hand coding of large sets of pre-defined rules. The machine-learning paradigm analyze and save instead for using general learning algorithms — often, although not always, grounded in statistical inference — to automatically learn such rules through the analysis of huge corpora of typical real-life examples. A corpus (plural, "corpora") is a set of documents (or sometimes, individual sentences) that have been hand-written with the correct values to be learned. Many different classes of machine learning algorithms have been applied to NLP tasks. These algorithms take as input a large set of "features" that are generated from the given input data. Some of the earliest-used algorithms, such as decision trees, produced systems of hard coded if-then rules similar to the systems of hand-written rules that were then common. Increasingly, however, research has focused on statistical models, which make soft, probabilistic decisions based on attaching real-valued weights to each given input feature. Such models have the benefit that they can express the relative certainty of many different possible answers rather than only one at a time, producing more reliable and analyzed results when such a model is included as a component of a larger system.

Systems based on machine-learning algorithms have many merits over hand-written rules: The learning procedures used in machine learning automatically focus on the most common cases, whereas when writing rules by hand it is often not at all obvious where the effort should be redirected. Automatic learning procedures can make the use of statistical inference algorithms to produce models that are strong to unfamiliar input (e.g. containing words or structures that have not been seen before) and to provided erroneous input (e.g. with misspelled words or words accidentally omitted). Generally, handling such input gracefully with hard coded rules — or more generally, creating systems of hand-written rules that can make soft decisions — is extremely difficult, error-prone and time-consuming. Systems based on automatically learning the rules can be made fiercely accurate simply by supplying more input data. However, systems based on hard-coded rules can only be made more accurate by increasing the complexity of the rules, which is a much tougher task. In particular, there is a limit to the complexity of systems based on hand-written rules, beyond which the systems become more and more chaotic. However, creating more data to input to machine-learning systems generally requires a corresponding increase in the number of man-hours worked, simply without significant increases in the complexity of the annotation process. The subunits of NLP devoted to learning approaches is known as Natural Language Learning (NLL) and its conference CoNLL and peak body SIGNLL are sponsored by ACL, recognizing also their links with Computational Linguistics and Language Acquisition system. When the aim of computational language learning research is to understand more about human language acquisition, or psycholinguistics, NLL overlaps into the related field of Computational Psycholinguistics systems.

**1.1.1 Major tasks in NLP**

The following is a list of few of the most commonly researched tasks in NLP. Note that some of these tasks have direct real-world applications, while others more commonly serve as subtasks that are used to aid in solving larger tasks. What differentiates these tasks from other potential and actual NLP tasks is not only the size of research devoted to them but the fact that for each one there is typically a very well-defined problem setting, a standard metric for evaluating the task, standard corpora on which the task can be evaluated, and competitions devoted to the defined task.

**Automatic summarization**

To produce a readable summary of a defined chunk of text. Often used to provide summaries of text of a known type cast, such as articles in the financial section of a newspaper.

**Coreference resolution**

Given a sentence or larger chunk of text, determine which words ("mentions") refer to the common objects ("entities"). Anaphora resolution is a given example of this task, and is specifically concerned with matching up of pronouns with the nouns or names that they refer to. The general task of reference resolution also includes identifying so-called "bridging relationships" involving referring expressions. For example, in a sentence such as "He entered into John's house through the front door", "the front door" is a referring expression and the bridging relationship to be identified is the fact that the door which is being referred to is the front door of John's house (rather than of some other structure that might also be referred to).

**Discourse analysis**

This rubric includes a number of related tasks. One of the task is identifying the discourse structure of connected text, i.e. the nature of the discourse relationships between the sentences (e.g. elaboration, explanation, contrast). Another possible task lies in recognizing and classifying the speech acts in a chunk of given text (e.g. yes-no question, content question, statement, assertion, etc.).

**1.1.2 Machine translation**

Automatically translate text from one human language to another. This is one of the most difficult problems, and is a member of a class of problems colloquially termed "AI-complete", i.e. requiring all of the different types of knowledge that humans possess (grammar, semantics, facts about the real world, etc.) in order to solve properly.

**Morphological segmentation**

Separate words into individual units and identify the class of them. The difficulty of this task depends greatly on the complexity of the morphology (i.e. the structure of words) of the language that’s being considered. English has fairly basic morphology, especially inflectional morphology, and thus it is generally possible to ignore this task entirely and simply model all possible forms of a given word (e.g. "open, opens, opened, opening") as separate words. In languages such as Turkish or Manipuri, a highly distinguished Indian language, however, such an approach is not possible, as each dictionary entry has thousands of possible word forms and there meanings.

**1.1.3 Named entity recognition (NER)**

Given a stream of text, to determine which items in the text map to proper names, such as people or places, and what is the type of each such name is (e.g. person, location, organization). Note that, although capitalization can aid in recognizing named entities in languages such as English, this information cannot aid in calculating the type of named entity, and in any case is often inaccurate. For example, the first word of a sentence is also capitalized, and named entities often span several words, only few of them are capitalized. Furthermore, many other languages in non-Western scripts (e.g. Chinese or Arabic) do not have any capitalization at all, and even languages with capitalization may not consistently use it to distinguish the given names. For example, German capitalizes all nouns, regardless of whether they refer to names, and French and Spanish do not capitalize names that serve as adjectives.

**Natural language generation**

Convert information from computer databases into readable human language. It could be said an NLG system is like a translator that converts a computer based representation into a natural language representation. However, the methods to produce the final language are different from those of a compiler due to the inherent expressivity of natural languages. NLG has existed for a long time but it is only recently that commercial NLG technology had become widely available and self-disciplined.

NLG may be viewed as the opposite of natural language understanding: whereas in natural language understanding the system needs to disambiguate the input sentence to produce the machine representation language, in NLG the system needs to make decisions about how to put a concept into words. Simple examples are systems that generate form letters. These do not typically involve grammar rules, but may generate a letter to a consumer, e.g. stating that a credit card spending limit was reached. More complex NLG systems dynamically create texts to meet their communicative goal. As in other areas of natural language processing, this can be done using either explicit models of language (e.g., grammars) and the domain, or using statistical models derived by analysing human-written codes.

**Natural language understanding**

Convert chunks of text into more formal representations such as first-order logic structures that are easier for computer programs to manipulate and extract results out of them. Natural language understanding involves the identification of the intended semantic from the multiple possible semantics which can be derived from a natural language expression which usually takes the form of organized notations of natural languages concepts. Introduction and creation of language metamodel and ontology are efficient but empirical solutions. An explicit formalization of natural languages semantics without confusions with implicit assumptions such as closed-world assumption (CWA) vs. open-world assumption, or subjective Yes/No vs. objective True/False is expected for the construction of a basis of semantics formalization.[5]

**1.1.4 Optical character recognition (OCR)**

Given an image representing printed text, determine the corresponding text. Optical character recognition (optical character reader) (OCR) is the mechanical or electronic conversion of images of typed, handwritten or printed text into machine-encoded text. It is widely used as a form of data entry from printed paper data records, whether passport documents, invoices, bank statements, computerised receipts, business cards, mail, printouts of static-data, or any formal documentation. It is a common method of digitising printed texts so that it can be electronically edited, searched, stored more compactly, displayed on-line, and used in machine processes such as machine translation, text-to-speech, key data and text mining. OCR is a field of coding and research in pattern recognition, artificial intelligence and computer vision.

Early versions needed to be trained with images of each character, and worked on one font at a time. Advanced systems capable of producing a higher degree of recognition accuracy for most fonts are now common. Some systems are capable of reproducing formatted output that closely approximates the original page including images, columns, and other non-textual components.

**1.1.5 Part-of-speech tagging**

Given a sentence, determine the part of speech for each word. Many words, especially the common ones, can serve as multiple parts of speech. For example, "book" can be a noun ("the book on the table") or verb ("to book a flight"); "set" can be a noun, verb or adjective; and "out" can be any of at least six different parts of speech. Some languages have more such ambiguity than others for the same word. Languages with little inflectional morphology, such as English are particularly prone to such ambiguity. Chinese is prone to such ambiguity because it is a tonal language during verbalization. Such inflection is not readily conveyed via the entities employed within the orthography to convey intended meaning.

**1.1.6 Parsing**

Determine the parse tree (grammatical analysis) of a given sentence. The grammar for natural languages is ambiguous and typical sentences have multiple possible analyses. In fact, perhaps surprisingly, for a given sentence there may be thousands of potential parses (most of which will seem completely nonsensical to a human).

**Question answering**

Given a human-language question, determine its answer. Typical questions have a specific right answer (such as "What is the capital of Canada?"), but sometimes open-ended questions are also considered (such as "What is the meaning of life?"). Recent works have looked at even more complex questions.

**Relationship extraction**

Given a chunk of text, identify the relationships among named entities (e.g. who is married to whom). A relationship extraction task requires the detection and classification of semantic relationship mentions within a set of artifacts, generally from text or XML documents. The task is very similar to that of information extraction (IE), but IE additionally requires the removal of repeated relations (disambiguation) and generally refers to the extraction of many different relationships.

**Sentence breaking (also known as sentence boundary disambiguation)**

Given a chunk of text, find the sentence boundaries. Sentence boundaries are often marked by periods or other punctuation marks, but these same characters can often serve other purposes (e.g. marking abbreviations).

**1.1.7 Sentiment analysis**

Extract subjective information usually from a set of documents, often using online reviews to determine "polarity" about specific objects. It is especially useful for identifying trends of public opinion in the social media, for the purpose of marketing.

## Benefits

### Evaluate sentiment and monitor changes over time.

The software automatically churns out sentiments in real time or over a period of time with a unique combination of statistical modeling and rule-based natural language processing techniques. Built-in reports show patterns and detailed reactions. So you can hone in on the sentiments that are expressed.

### Identify feedback sources to define new targets.

By actively monitoring internal collections and combining that with information from social networking sites, you can see where you're being discussed. Feedback is automatically extracted as the content is monitored. Important concepts are filtered so you can pursue the most promising opportunities.

### Continuously improve customer experience and competitive position.

The software searches for and evaluates internal and external content about your organization and competitors, identifying positive, negative, neutral and "no sentiment" texts – quantifying perceptions in the market.

### Promote discovery with a closed-loop, integrated analysis environment.

With ongoing evaluations, you can refine models and adjust classifications to reflect emerging topics and new terms relevant to your customers, organization or industry.

**Speech recognition**

Given a sound clip of a person or people speaking, determine the textual representation of that speech. This is the opposite of text to speech and is one of the extremely difficult problems colloquially termed "AI-complete" (see above). In natural speech there are hardly any pauses between successive words, and thus speech segmentation is a necessary subtask of speech recognition. Note also that in most spoken languages, the sounds representing successive letters blend into each other in a process termed coarticulation, so the conversion of the analog signal to discrete characters can be a very difficult process.

**Speech segmentation**

Given a sound clip of a person or people speaking, separate it into words. A subtask of speech recognition and typically grouped with it.

**Topic segmentation and recognition**

Given a chunk of text, separate it into units each of which is devoted to a topic, and identify the topic of the unit.

**Word segmentation**

Separate a chunk of continuous text into separate words. For a language like English, this is fairly trivial, since words are usually separated by spaces. However, some written languages like Chinese, Japanese and Thai do not mark word boundaries in such a fashion, and in those languages text segmentation is a significant task requiring knowledge of the vocabulary and modularity of words in the language.

**Word sense disambiguation**

Many words have more than one meaning; we have to select the meaning which makes the most sense in context. For this problem, we are typically given a list of words and associated word senses, e.g. from a dictionary or from an online resource library such as WordNet. In some cases, sets of related tasks are grouped into subfields of NLP that are often considered separately from NLP as a whole. Examples include:

**1.1.8 Information retrieval (IR)**

This is related with storing, searching and retrieving information. It is a separate field within computer science (closer to databases), but IR relies on some general NLP methods (for example, stemming). Some current research and applications seek to bridge the gap between IR and NLP.

**Information extraction (IE)**

This is concerned in general with the extraction of semantic information from text that pops out. This covers tasks such as named entity recognition, Coreference resolution, relationship extraction, etc.

**Speech processing**

This covers speech recognition, text-to-speech and related tasks.

Statistical natural-language processing uses stochastic, probabilistic, and statistical methods to resolve some of the difficulties discussed above, especially those which arise because longer sentences are highly ambiguous when processed with realistic grammars, yielding thousands or millions of possible analyzes. Methods for disambiguation often involve the use of corpora and Markov models. The ESPRIT Project P26 (1984 - 1988), led by CSELT, explored the problem of speech recognition comparing knowledge-based approach and statistical ones: the chosen result was a completely statistical model. One among the first models of statistical natural language understanding was introduced in 1991 by Roberto Pieraccini, Esther Levin, and Chin-Hui Lee from Bell Laboratories. NLP comprises all quantitative approaches to automated language processing, including probabilistic modeling, information theory, and linear algebra.[9] The technology for statistical NLP comes mainly from machine learning and data mining, both of which are fields of artificial intelligence that involve learning from raw information.

Evaluation of natural language processing

**1.1.9 Objectives**

The goal of NLP evaluation is to measure one or more qualities of an algorithm or a system, in order to determine: whether the given algorithm answers the goals of its designers, or if the system meets the needs of its users. Research in NLP evaluation has received considerable attention, because the definition of proper evaluation criteria is one way to specify precisely a NLP problem. The metric of NLP evaluation on an algorithmic system allows for the integration of language understanding and language generation on its own. A precise set of evaluation criteria, which include mainly evaluation of data and evaluation metrics can enable several teams to compare their solutions for a given NLP problem.

**Timeline of evaluation in NLP**

In 1987, the first evaluation campaign on written texts seems to be a campaign dedicated to message understanding (Pallet 1998).

The Parseval/GEIG project compared phrase-structure grammars (Black 1991).

There were series of campaigns within Tipster project on tasks like summarization, translation, and searching (Hirschman 1998).

In 1994, in Germany, the Morpholympics compared German morphological taggers.

The Senseval & Romanseval campaigns were conducted with the objectives of semantic disambiguation.

In 1996, the Sparkle campaign compared syntactic parsers in four different languages (English, French, German and Italian).

In France, the Grace project compared a set of 21 taggers for French in 1997 (Adda 1999).

In 2004, during the Technolangue/Easy project, 13 parsers for French were compared.

Large-scale evaluation of dependency parsers were performed in the context of the CoNLL shared tasks in 2006 and 2007.

In France, within the ANR-Passage project (end of 2007), 10 parsers for French were compared - passage web site.

In Italy, the EVALITA campaign was conducted in 2007,[10] 2009, 2011, and 2014[11] to compare various NLP and speech tools for Italian - EVALITA web site.

**1.1.10 Different types of evaluation**

Depending on the evaluation procedures, a number of distinctions are traditionally made in NLP evaluation.

**Intrinsic v. extrinsic evaluation**

Intrinsic evaluation considers an isolated NLP system and characterizes its performance with respect to a gold standard result as defined by the evaluators. Extrinsic evaluation, also called evaluation in use, considers the NLP system in a more complex setting as either an embedded system or a precise function for a human user. The extrinsic performance of the system is then characterized in terms of utility with respect to the overall task of the extraneous system or the human user. For example, consider a syntactic parser which is based on the output of some part of speech (POS) tagger. An intrinsic evaluation would run the POS tagger on structured data, and compare the system output of the POS tagger to the gold standard output. An extrinsic evaluation would run the parser with some other POS tagger, and then with the novel POS tagger, and compare the parsing accuracy.

**Black-box v. glass-box evaluation**

Black-box evaluation requires someone to run an NLP system on a sample data set and to measure a number of parameters related to: the quality of the process, such as speed, reliability, resource consumption; and most importantly, the quality of the result, such as the accuracy of data annotation or the fidelity of a translation. Glass-box evaluation looks at the: design of the system; the algorithms that are implemented; the linguistic resources it uses, like vocabulary size or expression set cardinality. Given the complexity of NLP problems, it is often difficult to predict performance only on the basis of glass-box evaluation; but this type of evaluation is more informative with respect to error analysis or future developments of a system.

**Automatic v. manual evaluation**

In many cases, automatic procedures can be defined to evaluate an NLP system by comparing its output with the gold standard one. Although the cost of reproducing the gold standard can be quite high, bootstrapping automatic evaluation on the same input data can be repeated as often as needed without inordinate additional costs. However for many NLP problems the precise definition of a gold standard is a complex task and it can prove impossible when inter-annotator agreement is insufficient. Manual evaluation is best performed by human judges instructed to estimate the quality of a system, or most often of a sample of its output, based on a number of criteria. Although, thanks to their linguistic competence, human judges can be considered as the reference for a number of language processing tasks, there is also considerable variation across their ratings. That is why automatic evaluation is sometimes referred to as objective evaluation while the human evaluation is perspective.

However, they all come under the umbrella of sentiment analysis or opinion mining. Sentiment classification, feature based sentiment classification and opinion summarization are few main fields of research predominate in sentiment analysis.

In recent years, we have witnessed that opinionated postings in social media have helped reshape businesses, and sway public sentiments and emotions, which have profoundly impacted on our social and political systems. Such postings have also mobilized masses for political changes such as those happened in some Arab countries in 2011. It has thus become a necessity to collect and study opinions on the Web. Of course, opinionated documents not only exist on the Web (called external data), many organizations also have their internal data, e.g., customer feedback collected from emails and call centers or results from surveys conducted by the organizations.

Opinion mining can be useful in several ways. For example, in marketing, it tracks and judges the success rate of an ad campaign or launch of new product, determine popularity of products and services with its versions also tell us about demographics which like or dislike particular features. For example, a review might be about a digital camera might be broadly positive, but be specifically negative about how heavy it is. The vendor gets a much clearer picture of public opinion than surveys or focus groups, if this kind of information is indentified in a systematic way.

The technique to detect and extract subjective information in text documents is opinion mining and sentiment analysis. In general, the overall contextual polarity or sentiment of a writer about some aspect can be determined using sentiment analysis. The main challenge in this area is the sentiment classification in which the sentiment may be a judgment, mood or evaluation of an object namely film, book, product, etc which can be in the form of document or sentence or feature that can be labeled as positive or negative.

Classifying entire documents according to the opinions towards certain objects is called as

Sentiment classification. One form of opinion mining in product reviews is also to produce feature-based summary. To produce a summary on the features, product features are first identified, and positive and negative opinions on them are aggregated. Features are product

attributes, components and other aspects of the product. The effective opinion summary, grouping feature expressions which are domain synonyms is critical. It is very time consuming and tedious for human users to group typically hundreds of feature expressions that can be discovered from text for an opinion mining application into feature categories. Some automated assistance is needed. Opinion summarization does not summarize the reviews by selecting a subset or rewrite some of the original sentences from the reviews to capture the main points as the classic text summarization.

**1.2 What is Sentiment?**

One of the challenges of Sentiment Analysis is defining the objects of the study – opinions and subjectivity. Originally, subjectivity was defined by linguists, most prominently, Quirk defines private state as something that is not open to objective observation or verification. These private states include emotions, opinions, and speculations, among others. Wiebe, a prominent Natural Language Processing (NLP) researcher, used Quirk’s definition of the private state when tracking point of view in narrative. She defines private state as a tuple (p, experience, attitude, object) relating experience’s state p to his/her attitude possibly toward an object. In practice, a simplified version of this model,

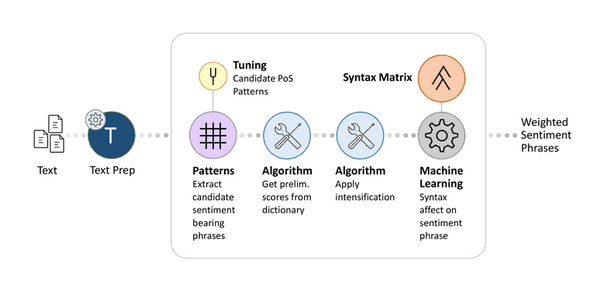
where we look only at polarity and the target of the sentiment, is usually used. In fact, many researchers define sentiment loosely, as a negative or positive opinion. Some researchers use products that provide pre-compiled lists of words in various groupings, some of which are related to emotional states. These include Linguistic Inquiry and Word Count (LIWC)3 and Profile of Mood States (POMS)4

A common use case for this technology is to discover how people feel about a particular topic. Say you want to know if people on Twitter think that Chinese food in San Francisco is good or bad. Analyzing tweets for sentiment will answer this question. You can even learn why people think the food is good or bad, by extracting the exact words that indicate why people did or didn't like the food. If "too salty" shows as a common theme, for example, you immediately have a better idea of why consumers aren’t happy. This is the kind of insight one aims to find through market research, but why devote enormous budgets and countless man-hours to conducting surveys and cold calling? Through Lexalytics text mining tools, you’ll get answers in seconds from the comfort of your office chair. Lexalytics provides sentiment analysis solutions directly to businesses, as well as offering APIs for integration into our client’s own products. Hundreds of companies around the world rely on Lexalytics sentiment analysis solutions to track and monitor public opinion of their products, services, or organization in general. If someone is attacking your brand on social media, our sentiment analysis systems score the relevant posts as extremely negative, and a social media monitoring solution flags them for immediate response. Salience and Semantria process billions of documents every day in a wide range of industries, from Hospitality to Financial Services to Customer Experience Management and beyond. Our results drive better business decisions for scores of companies of all sizes around the world.

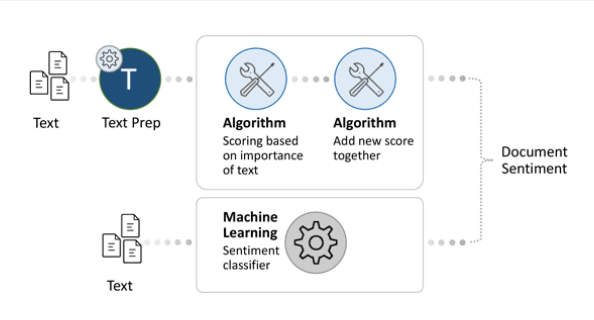
**1.2.1 Multi-level Analysis**

Lexalytics’ sentiment analysis techniques can be configured to determine sentiment on a range of levels. We’ll score sentiment on a document level (does this express a general positive or negative tone), but we’ll also score the sentiment of individual words or phrases in the document. Sentiment analysis (also known as opinion mining) refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials. Sentiment analysis is widely applied to reviews and social media for a variety of applications, ranging from marketing to customer service. Generally speaking, sentiment analysis aims to determine the attitude of a speaker or a writer with respect to some topic or the overall contextual polarity of a document. The attitude may be his or her judgment or evaluation (see appraisal theory), affective state (that is to say, the emotional state of the author when writing), or the intended emotional communication (that is to say, the emotional effect the author wishes to have on the reader).

A weak sentiment analysis system will score "love the summer" as positive and "hate the winter" as negative, but will report the entire comment’s sentiment as neutral (the positive "love" and the negative "hate" cancelling each other out). Lexalytics recognize the importance of that middle word, "but", and so we report separate sentiment for the first and second parts of the sentence.

**Figure 1.1: Multi-level Analysis**

In order to determine the sentiment of the overall document, we can use our own scoring algorithms – using the weighted phrases from the previous side, and then using our proprietary way of adding them up. We can also take a set of sentiment tagged content and build a document-level sentiment classifier.



**Figure 1.2: Multi-level Analysis contd…**

First identify the sentiment phrases (and not, say, a proper noun like “Good Morning America”), apply things like “intensification” and “negation” – for stuff like “good” “very good” “not very good”, etc. Then, in English, use the syntax matrix to determine the syntactic effect of the ordering of the words.

**1.2.2 Types of sentiment analysis**

A basic task in sentiment analysis is classifying the polarity of a given text at the document, sentence, or feature/aspect level — whether the expressed opinion in a document, a sentence or an entity feature/aspect is positive, negative, or neutral. Advanced, "beyond polarity" sentiment classification looks, for instance, at emotional states such as "angry," "sad," and "happy."

Early work in that area includes Turney and Pang who applied different methods for detecting the polarity of product reviews and movie reviews respectively. This work is at the document level. One can also classify a document's polarity on a multi-way scale, which was attempted by Pang and Snyder among others: Pang and Lee expanded the basic task of classifying a movie review as either positive or negative to predicting star ratings on either a 3 or a 4 star scale, while Snyder performed an in-depth analysis of restaurant reviews, predicting ratings for various aspects of the given restaurant, such as the food and atmosphere (on a five-star scale). Even though in most statistical classification methods, the neutral class is ignored under the assumption that neutral texts lie near the boundary of the binary classifier, several researchers suggest that, as in every polarity problem, three categories must be identified.

A different method for determining sentiment is the use of a scaling system whereby words commonly associated with having a negative, neutral or positive sentiment with them are given an associated number on a -10 to +10 scale (most negative up to most positive) and when a piece of unstructured text is analyzed using natural language processing, the subsequent concepts are analyzed for an understanding of these words and how they relate to the concept.[citation needed] Each concept is then given a score based on the way sentiment words relate to the concept, and their associated score. This allows movement to a more sophisticated understanding of sentiment based on an 11 point scale. Alternatively, texts can be given a positive and negative sentiment strength score if the goal is to determine the sentiment in a text rather than the overall polarity and strength of the text.

**Subjectivity/objectivity identification**

This task is commonly defined as classifying a given text (usually a sentence) into one of two classes: objective or subjective. This problem can sometimes be more difficult than polarity classification. The subjectivity of words and phrases may depend on their context and an objective document may contain subjective sentences (e.g., a news article quoting people's opinions). Moreover, as mentioned by Su, results are largely dependent on the definition of subjectivity used when annotating texts. However, Pang showed that removing objective sentences from a document before classifying its polarity helped improve performance.

**Feature/aspect-based sentiment analysis**

It refers to determining the opinions or sentiments expressed on different features or aspects of entities, e.g., of a cell phone, a digital camera, or a bank. A feature or aspect is an attribute or component of an entity, e.g., the screen of a cell phone, the service for a restaurant, or the picture quality of a camera. The advantage of feature-based sentiment analysis is the possibility to capture nuances about objects of interest. Different features can generate different sentiment responses, for example a hotel can have a convenient location, but mediocre food. This problem involves several sub-problems, e.g., identifying relevant entities, extracting their features/aspects, and determining whether an opinion expressed on each feature/aspect is positive, negative or neutral. The automatic identification of features can be performed with syntactic methods or with topic modeling. More detailed discussions about this level of sentiment analysis can be found in Liu's work.

**1.3 DATA SOURCE**

People and companies across disciplines exploit the rich and unique source of data for varied purposes. The major criterion for the improvement of the quality services rendered and enhancement of deliverables are the user opinions. Blogs, review sites and micro blogs provide a good understanding of the reception level of products and services.

**Blogs**

The name associated to universe of all the blog sites is called blogosphere. People write about the topics they want to share with others on a blog. Blogging is a happening thing because of its ease and simplicity of creating blog posts, its free form and unedited nature. We find a large number of posts on virtually every topic of interest on blogosphere. Sources of opinion in many of the studies related to sentiment analysis, blogs are used.

**Review Sites**

Opinions are the decision makes for any user in making a purchase. The user generated reviews for products and services are largely available on internet. The sentiment classification uses reviewer’s data collected from the websites like www.gsmarena.com (mobile reviews), www.amazon.com (product reviews), www. CNETdownload.com (product reviews), which hosts millions of product reviews by consumers.

**Micro-blogging**

A very popular communication tool among Internet users is micro-blogging. Millions of messages appear daily in popular web-sites for micro-blogging such as Twitter, Tumbler, Face book. Twitter messages sometimes express opinions which are used as data source for classifying sentiment.

**1.4 SENTIMENT CLASSIFICATION**

Sentiment classification or Polarity classification is the binary classification task of labeling an opinionated document as expressing either an overall positive or an overall negative opinion. A technique for analyzing subjective information in a large number of texts, and many studies is sentiment classification. A typical approach for sentiment classification is to use machine learning algorithms.

**1.4.1 Machine Learning**

A system capable of acquiring and integrating the knowledge automatically is referred as machine learning. The systems that learn from analytical observation, training, experience, and other means, results in a system that can exhibit self-improvement, effectiveness and efficiency. Machine learning is a subfield of computer science[1] and statistics that evolved from the study of pattern recognition and computational learning theory in artificial intelligence.[1] In 1959, Arthur Samuel defined machine learning as a "Field of study that gives computers the ability to learn without being explicitly programmed".[2] Machine learning explores the study and construction of algorithms that can learn from and make predictions on data.[3] Such algorithms operate by building a model from example inputs in order to make data-driven predictions or decisions expressed as outputs,[4]:2 rather than following strictly static program instructions.

Machine learning is closely related to and often overlaps with computational statistics; a discipline which also focuses in prediction-making through the use of computers. It has strong ties to mathematical optimization, which delivers methods, theory and application domains to the field. Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms is infeasible. Example applications include spam filtering, optical character recognition (OCR),[5] search engines and computer vision. Machine learning is sometimes conflated with data mining,[6] where the latter sub-field focuses more on exploratory data analysis and is known as unsupervised learning.[4]:vii[7]

Within the field of data analytics, machine learning is a method used to devise complex models and algorithms that lend themselves to prediction. These analytical models allow researchers, data scientists, engineers, and analysts to "produce reliable, repeatable decisions and results" and uncover "hidden insights" through learning from historical relationships and trends in the data.

Knowledge and a corresponding knowledge organization are usually used by a machine learning system to test the knowledge acquired, interpret and analyze. One of the machine learning algorithms is taxonomy based depending on outcome of the algorithm or type of input available.

* Supervised learning generates a function which maps inputs to desired outputs also called as labels because they are training examples labeled by human experts. Since it is a text classification problem, any supervised learning method can be applied, e.g., Naïve Bayes classification, and KNN.
* Unsupervised learning models a set of inputs, like clustering, labels are not known during training. Classification is performed using some fixed syntactic patterns which are used to express opinions. The part-of-speech (POS) tags are used to compose syntactic patterns.
* Semi-supervised learning generate an appropriate function or classifier in which both labeled and unlabelled examples are combined.

**Types of problems and tasks**

Machine learning tasks are typically classified into three broad categories, depending on the nature of the learning "signal" or "feedback" available to a learning system. These are

Supervised learning: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.

Unsupervised learning: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).

Reinforcement learning: A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle), without a teacher explicitly telling it whether it has come close to its goal. Another example is learning to play a game by playing against an opponent.

Between supervised and unsupervised learning is semi-supervised learning, where the teacher gives an incomplete training signal: a training set with some (often many) of the target outputs missing. Transduction is a special case of this principle where the entire set of problem instances is known at learning time, except that part of the targets is missing.

Among other categories of machine learning problems, learning to learn learns its own inductive bias based on previous experience. Developmental learning, elaborated for robot learning, generates its own sequences (also called curriculum) of learning situations to cumulatively acquire repertoires of novel skills through autonomous self-exploration and social interaction with human teachers, and using guidance mechanisms such as active learning, maturation, motor synergies, and imitation.

Another categorization of machine learning tasks arises when one considers the desired output of a machine-learned system:

In classification, inputs are divided into two or more classes, and the learner must produce a model that assigns unseen inputs to one or more (multi-label classification) of these classes. This is typically tackled in a supervised way. Spam filtering is an example of classification, where the inputs are email (or other) messages and the classes are "spam" and "not spam".

In regression, also a supervised problem, the outputs are continuous rather than discrete.

In clustering, a set of inputs is to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.

Density estimation finds the distribution of inputs in some space.

Dimensionality reduction simplifies inputs by mapping them into a lower-dimensional space. Topic modeling is a related problem, where a program is given a list of human language documents and is tasked to find out which documents cover similar topics. A core objective of a learner is to generalize from its experience.[17][18] Generalization in this context is the ability of a learning machine to perform accurately on new, unseen examples/tasks after having experienced a learning data set. The training examples come from some generally unknown probability distribution (considered representative of the space of occurrences) and the learner has to build a general model about this space that enables it to produce sufficiently accurate predictions in new cases.

The computational analysis of machine learning algorithms and their performance is a branch of theoretical computer science known as computational learning theory. Because training sets are finite and the future is uncertain, learning theory usually does not yield guarantees of the performance of algorithms. Instead, probabilistic bounds on the performance are quite common. The bias–variance decomposition is one way to quantify generalization error.

How well a model trained with existing examples predicts the output for unknown instances is called generalization. For best generalization, complexity of the hypothesis should match the complexity of the function underlying the data. If the hypothesis is less complex than the function, we've underfitted. Then, we increase the complexity, the training error decreases. But if our hypothesis is too complex, we've overfitted. After then, we should find the hypothesis that has the minimum training error.[19]

In addition to performance bounds, computational learning theorists study the time complexity and feasibility of learning. In computational learning theory, a computation is considered feasible if it can be done in polynomial time. There are two kinds of time complexity results. Positive results show that a certain class of functions can be learned in polynomial time. Negative results show that certain classes cannot be learned in polynomial time.

**Sentiment Analysis Tasks**

Sentiment analysis tasks mainly consists of classifying the polarity of a given text at the Document, sentence or feature/aspect level expressing the opinion as positive, negative or neutral.

The sentiment analysis can be performed at one of the three levels: the document level, sentence level, feature level.

Document Level Sentiment Classification: In document level sentiment analysis main challenge is to extract informative text for inferring sentiment of the whole document. The learning methods can be confused because of objective statements are rendered by subjective statements and complicate further for document categorization task with conflicting sentiment. [6]

Sentence Level Sentiment Classification: The sentiment classification is a fine-grained level than document level sentiment classification in which polarity of the sentence can be given by three categories as positive, negative and neutral. The challenge faced by sentence level sentiment classification is the identification features indicating whether sentences are on-topic which is kind of co-reference problem

Feature Level Sentiment Classification: Product features are defined as product attributes or components. Analysis of such features for identifying sentiment of the document is called as feature based sentiment analysis. In this approach positive or negative opinion is identified from the already extracted features. It is a fine grained analysis model among all other models

**1.5 Background Information**

For this thesis, thorough and extensive knowledge of Big Data and Data mining is a must. We are deluged by data be it scientific or financial all around us. As time progresses and human effort becomes more and more precious, human attention to such data has become of paramount importance. As a result, automatic mining tools were developed which help reduce this human effort.

Our capabilities of both geenerating and collecting data have been increasing raapidly. Computerization and increased connectivity, releasing global barriers has led to development of various Data Mining Systems. The urgent need to analyze and interpret this data can transform vast amounts of data in a concise manner for easier interpretation.

Data Mining is a multidisciplinary field, drawing work from areas including databases techology, machine learning, statistics, probability, pattern recognition etc. Data Mining refers to extracting knowledge from large amounts of data. A misnomer of the commonly used word 'Mining', its meaning is more appropriately reflected by the phrase mining knowledge from data. However, once coined, this term has been used for decades and analysts have become accustomed to this word. Also known as Knowledge Discovery from Data, or KDD, Data Mining is an essential step in knowldege discovery.

Regarding the actual Sentiment Analyzer, Python will be used. Python is part of a winning formula for productivity, software quality, and maintainability at many companies and institutions across the globe. It's extensively used in GIS, DIP, Computer Graphics, Big Data Analytics etc. From Desktop GUIs to Server Back ends, Python has slowly dominated the world of Computers.

Microsoft .net is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data streuctures, combined with dynamic scripting and runtime binding, make it a lucarative option for Application Development in short time. It can also be used as a language used to connect existing highly cohesive and loosely coupled components together. It is an easy, simple to lern language which poses no difficulty for learning and memorizing syntax. It stresses on interpretation and therefore reduces program maintenance cost. Enforcing the OOP concepts like those of Modularity and Polymorphism, the .net and its readily availbale libraries present no challenge to the developer. It also offers cross platform independence.

With its increased productivity, programmers prefer .net as it’s a compilation free language. Also, the edit-test-debug cycle is incredibly fast. Testing and Debuhging .net programs is easy: a bug or bad input will never cause a segmentation fault. Raising exceptions instead of erros, it is a user friendly language. In case the exception isnt thrown, the interpreter prints a stack trace. The debugger is written in .net itself, testifying to .net introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

**1.6 MODIFIED K-MEANS:**

This paper presents a data clustering approach using modified K-Means algorithm based on the improvement of the sensitivity of initial center of clusters. This algorithm partitions the whole space into different segments and calculates the frequency of data point in each segment. The segment which shows maximum frequency of data point will have the maximum probability to contain the centroid of cluster. The number of cluster's centroid (k) will be provided by the user in the same manner like the traditional K-mean algorithm and the number of division will be k\*k (`k' vertically as well as `k' horizontally). If the highest frequency of data point is same in different segments and the upper bound of segment crosses the threshold `k' then merging of different segments become mandatory and then take the highest k segment for calculating the initial centroid of clusters. In this paper we also define a threshold distance for each cluster's centroid to compare the distance between data point and cluster's centroid with this threshold distance through which we can minimize the computational effort during calculation of distance between data point and cluster's centroid. It is shown that how the modified k-mean algorithm will decrease the complexity & the effort of numerical calculation, maintaining the easiness of implementing the k-mean algorithm. It assigns the data point to their appropriate class or cluster more effectively.

We have presented a modified k-means algorithm which eliminates the problem of generation of empty clusters (with some exceptions). Here, the basic structure of the original k-means is preserved along with all its necessary characteristics. A new center vector computation strategy enables us to redefine the clustering process and to reach our goal. The modified algorithm is found to work very satisfactorily, with some conditional exceptions which are very rare in practice.

**Modified approach K-mean algorithm:**

The K-mean algorithm is a popular clustering algorithm and has its application in data mining, image segmentation, bioinformatics and many other fields. This algorithm works well with small datasets. In this paper we proposed an algorithm that works well with large datasets. Modified k-mean algorithm avoids getting into locally optimal solution in some degree, and reduces the adoption of cluster -error criterion.

Algorithm: Modified approach (S, k), S={x1,x2,…,xn }

Input: The number of clusters k1 (k1> k) and a dataset containing n objects(Xij+).

Output: A set of k clusters (Cij) that minimize the Cluster - error criterion.

**Algorithm**

1. Compute the distance between each data point and all other data- points in the set D

2. Find the closest pair of data points from the set D and form a data-point set Am (1<= p <= k+1) which contains these two data- points, Delete these two data points from the set D

3. Find the data point in D that is closest to the data point set Ap, Add it to Ap and delete it from D

4. Repeat step 4 until the number of data points in Am reaches (n/k)

5. If p<k+1, then p = p+1, find another pair of data points from D between which the distance is the shortest, form another data-point set Ap and delete them from D, Go to step4.

**Algorithm A**

1. For each data-point set Am (1<=p<=k) find the arithmetic mean of the vectors of data points Cp(1<=p<=k) in Ap.

2. Select nearest object of each Cp(1<=p<=k) as initial centroid.

3. Compute the distance of each data-point di (1<=i<=n) to all the centroids cj (1<=j<=k+1) as d(di, cj)

4. For each data-point di, find the closest centroid cj and assign di to cluster j

Set ClusterId[i]=j; // j:Id of the closest cluster

5. Set Nearest\_Dist[i++]= d(di, cj)

6. For each cluster j (1<=j<=k), recalculate the centroids

7. Repeat

**Algorithm B**

1. For each data-point di

**2.** Compute its distance from the centroid of the present nearest cluster

**3.** If this distance is less than or equal to the present nearest distance, the data-point stays in the cluster

Else ;

4. For every centroid cj (1<=j<=k) Compute the distance (di, cj); Endfor

5. Assign the data-point di to the cluster with the nearest centroid Cj

6. Set ClusterId[i] =j

7. Set Nearest\_Dist[i] = d (di, cj); Endfor

**1.7 ETL**

In computing, Extract, Transform, Load (ETL) refers to a process in database usage and especially in data warehousing. The ETL process became a popular concept in the 1970s. Data extraction is where data is extracted from homogeneous or heterogeneous data sources; data transformation where the data is transformed for storing in the proper format or structure for the purposes of querying and analysis; data loading where the data is loaded into the final target database, more specifically, an operational data store, data mart, or data warehouse.

Since the data extraction takes time, it is common to execute the three phases in parallel. While the data is being extracted, another transformation process executes while processing the data already received and prepares it for loading while the data loading begins without waiting for the completion of the previous phases.

ETL systems commonly integrate data from multiple applications (systems), typically developed and supported by different vendors or hosted on separate computer hardware. The disparate systems containing the original data are frequently managed and operated by different employees. For example, a cost accounting system may combine data from payroll, sales, and purchasing.

**1.7.1 Extract**

The first part of an ETL process involves extracting the data from the source system(s). In many cases, this represents the most important aspect of ETL, since extracting data correctly sets the stage for the success of subsequent processes. Most data-warehousing projects combine data from different source systems. Each separate system may also use a different data organization and/or format. Common data-source formats include relational databases, XML and flat files, but may also include non-relational database structures such as Information Management System (IMS) or other data structures such as Virtual Storage Access Method (VSAM) or Indexed Sequential Access Method (ISAM), or even formats fetched from outside sources by means such as web spidering or screen-scraping. The streaming of the extracted data source and loading on-the-fly to the destination database is another way of performing ETL when no intermediate data storage is required. In general, the extraction phase aims to convert the data into a single format appropriate for transformation processing.

An intrinsic part of the extraction involves data validation to confirm whether the data pulled from the sources has the correct/expected values in a given domain (such as a pattern/default or list of values). If the data fails the validation rules it is rejected entirely or in part. The rejected data is ideally reported back to the source system for further analysis to identify and to rectify the incorrect records. In some cases, the extraction process itself may have to do a data-validation rule in order to accept the data and flow to the next phase.

**1.7.2 Transform**

In the data transformation stage, a series of rules or functions are applied to the extracted data in order to prepare it for loading into the end target. Some data does not require any transformation at all; such data is known as "direct move" or "pass through" data.

An important function of transformation is the cleaning of data, which aims to pass only "proper" data to the target. The challenge when different systems interact is in the relevant systems' interfacing and communicating. Character sets that may be available in one system may not be so in others.

In other cases, one or more of the following transformation types may be required to meet the business and technical needs of the server or data warehouse:

Selecting only certain columns to load: (or selecting null columns not to load). For example, if the source data has three columns (aka "attributes"), roll\_no, age, and salary, then the selection may take only roll\_no and salary. Or, the selection mechanism may ignore all those records where salary is not present (salary = null).

Translating coded values: (e.g., if the source system codes male as "1" and female as "2", but the warehouse codes male as "M" and female as "F")

Encoding free-form values: (e.g., mapping "Male" to "M")

Deriving a new calculated value: (e.g., sale\_amount = qty \* unit\_price)

Sorting or ordering the data based on a list of columns to improve search performance

Joining data from multiple sources (e.g., lookup, merge) and deduplicating the data

Aggregating (for example, rollup — summarizing multiple rows of data — total sales for each store, and for each region, etc.)

Generating surrogate-key values

Transposing or pivoting (turning multiple columns into multiple rows or vice versa)

Splitting a column into multiple columns (e.g., converting a comma-separated list, specified as a string in one column, into individual values in different columns)

Disaggregating repeating columns

Looking up and validating the relevant data from tables or referential files

Applying any form of data validation; failed validation may result in a full rejection of the data, partial rejection, or no rejection at all, and thus none, some, or all of the data is handed over to the next step depending on the rule design and exception handling; many of the above transformations may result in exceptions, e.g., when a code translation parses an unknown code in the extracted data

**1.7.3 Load**

The load phase loads the data into the end target, which may be a simple delimited flat file or a data warehouse. Depending on the requirements of the organization, this process varies widely. Some data warehouses may overwrite existing information with cumulative information; updating extracted data is frequently done on a daily, weekly, or monthly basis. Other data warehouses (or even other parts of the same data warehouse) may add new data in a historical form at regular intervals—for example, hourly. To understand this, consider a data warehouse that is required to maintain sales records of the last year. This data warehouse overwrites any data older than a year with newer data. However, the entry of data for any one year window is made in a historical manner. The timing and scope to replace or append are strategic design choices dependent on the time available and the business needs. More complex systems can maintain a history and audit trail of all changes to the data loaded in the data warehouse.

As the load phase interacts with a database, the constraints defined in the database schema — as well as in triggers activated upon data load — apply (for example, uniqueness, referential integrity, mandatory fields), which also contribute to the overall data quality performance of the ETL process.

For example, a financial institution might have information on a customer in several departments and each department might have that customer's information listed in a different way. The membership department might list the customer by name, whereas the accounting department might list the customer by number. ETL can bundle all of these data elements and consolidate them into a uniform presentation, such as for storing in a database or data warehouse.

Another way that companies use ETL is to move information to another application permanently. For instance, the new application might use another database vendor and most likely a very different database schema. ETL can be used to transform the data into a format suitable for the new application to use.

An example would be an Expense and Cost Recovery System (ECRS) such as used by accountancies, consultancies, and legal firms. The data usually ends up in the time and billing system, although some businesses may also utilize the raw data for employee productivity reports to Human Resources (personnel dept.) or equipment usage reports to Facilities Management.

**1.8 Organization of Thesis**

Report is organized as follows:

Chapter 2 begins with the briefing about the literature survey and discusses the various approaches studied and their comparative analysis

Chapter 3 outlines the proposed approach and explains the same with flow graphs and algorithms

Chapter 4 gives the implementations details of the project

Chapter 5 gives the results and detailed analysis

Chapter 6 concludes the work

**CHAPTER 2**

**LITERATURE REVIEW**

**2.1 OVERVIEW**

Chapter 2 presents a brief introduction to the literature survey which was being performed to extract the available information related to the dissertation from various sources like technical books, previous IEEE papers and ITU-T standards. Section 2.2.1 is devoted to selected Research papers. This chapter also includes the objective of the thesis which is discussed in section 2.3.

**2.2 LITERATURE SURVEY**

The following sub-sections give the information extracted from the technical books and IEEE papers.

**2.2.1 RESEARCH PAPERS**

There is a huge collection of papers related to NLP, Modified K-Means, Naïve Bayes and KNN. After reviewing them, following papers are found relevant for the present work of my dissertation:-

Jalaj S. Modha, Prof & Head Gayatri S. Pandi Sandip J. Modha, **Automatic Sentiment Analysis for Unstructured Data**, International Journal of Advanced Research in Computer Science and Software Engineering , Volume 3, Issue 12, December 2013

In this thesis they discussed about exiting methods, approaches to do sentimental analysis for unstructured data which reside on web. Currently, Sentiment Analysis concentrates for subjective statements or on subjectivity and overlook objective statements which carry sentiment(s). So, they proposed new approach classify and handle subjective as well as objective statements for sentimental analysis.

Proposed Approach:

In Sentiment Analysis, numbers of sentences or sentences of documents. All these documents or sentences may convey opinion or maybe not. Formally, there is document set D= {d1, d2, ..., dN}, sentence set S= {S1, S2, ..., Sn} and all these documents and sentences belong to some specific entity e where e is a product, service, topic, issue, person, organization, or event

They followed four steps of classification.

1.) First step: First classify sentences or sentences of documents into two categories Opinionated and No- Opinionated, regardless whether it is subjective or objective.

2.) Second Step: In this step we have opinionated sentences so now they are classified as subjective sentences and Objective sentences.

3.) Third Step: The third step is classifying subjective sentences into positive, negative or neutral category. For complex type of sentences we may need to attach context or semantic orientation

4.) Fourth Step: The fourth step is classifying objective sentences into positive, negative or neutral category. Here also we have to provide context or sentiment orientation as and when needed.

[2] R M. Chandrasekaran , G.Vinodhini, **Sentiment Analysis and Opinion Mining: A Survey** International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 6, June 2012 [14]

Sentiment Analysis for objective sentences is very trending research topic now-a-days because there are so many data sources which have objective sentences that carry sentiment but because of lake of proper algorithms and contexts we can’t get the fruitful result from the objective sentences. According to recent article published by Ronen Feldman express that objective sentences that carry sentiment should be analyzed for getting efficient sentiment analysis and this is one of the challenging task in sentiment analysis.

Source of objective sentences are including news articles, blogs, social media etc. where we get good amount of objective sentences.

We consider following examples which are objective sentences but still carry sentiment.

* “Firefox keeps crashing.” defined sentences carry negative sentiment about Firefox web browser.
* “The earphone broke in two days.” defined sentence carry negative sentiment about the earphones.
* “I get relaxed time after today’s session.” define positive sentiment about person’s routine.

In this particular area just challenges are proposed but still researchers are trying to find out efficient solution to get analyzed these kinds of implicit opinions in the objective sentences. Available sentiment dictionaries don‟t have enough vocabulary to get analyzed objective sentences and categorized them efficiently into positive, negative or neutral. Provide proper context or semantic orientation is also very important part of sentiment analysis of objective Sentences

Bing Liu. **Sentiment Analysis and Opinion Mining,** Morgan & Claypool Publishers, May 2012.

Opinions and its related concepts such as sentiments, evaluations, attitudes, and emotions are the subjects of study of sentiment analysis and opinion mining. The inception and rapid growth of the field coincide with those of the social media on the Web, e.g., reviews, forum discussions, blogs, microblogs,

Twitter, and social networks, because for the first time in human history, we have a huge volume of opinionated data recorded in digital forms. Since early 2000, sentiment analysis has grown to be one of the most active research areas in natural language processing. It is also widely studied

in data mining, Web mining, and text mining. In fact, it has spread from computer science to management sciences and social sciences due to its importance to business and society as a whole. In recent years, industrial activities surrounding sentiment analysis have also thrived. Numerous startups have emerged. Many large corporations have built their own in-house capabilities. Sentiment analysis systems have found their applications in almost every business and social domain.

The goal of this book is to give an in-depth introduction to this fascinating problem and to present a comprehensive survey of all important research topics and the latest developments in the field. As evidence of that, this book covers more than 400 references from all major conferences and journals. Although the field deals with the natural language text, which is often

Considered the unstructured data, this book takes a structured approach in introducing the problem with the aim of bridging the unstructured and structured worlds and facilitating qualitative and quantitative analysis of opinions. This is crucial for practical applications. In this book, defined the problem in order to provide an abstraction or structure to the problem.

[4] Arti Buche, Dr. M. B. Chandak, Akshay Zadgaonkar,**OPINION MINING AND ANALYSIS: A SURVEY,** International Journal on Natural Language Computing (IJNLC) Vol. 2, No.3, June 2013 [16]

The current research is focusing on the area of Opinion Mining also called as sentiment analysis due to sheer volume of opinion rich web resources such as discussion forums, review sites and blogs are available in digital form. One important problem in sentiment analysis of product reviews is to produce summary of opinions based on product features. We have surveyed and analyzed in this thesis, various techniques that have been developed for the key tasks of opinion mining. They have provided an overall picture of what is involved in developing a software system for opinion mining on the basis of our survey and analysis.Classifying entire documents according to the opinions towards certain objects is called as sentiment classification. One form of opinion mining in product reviews is also to produce feature-based summary. To produce a summary on the features, product features are first identified, and positive and negative opinions on them are aggregated. Features are product attributes, components and other aspects of the product. The effective opinion summary, grouping feature expressions which are domain synonyms is critical. It is very time consuming and tedious for human users to group typically hundreds of feature expressions that can be discovered from text for an opinion mining application into feature categories. Some automated assistance is needed. Opinion summarization does not summarize the reviews by selecting a subset or rewrite some of the original sentences from the reviews to capture the main points as the classic text summarization.

[5]. Fred Popowich**, Using Text Mining and Natural Language Processing for Health Care Claims Processing**, SIGKDD Explorations. Volume 7, Issue 1 - Page 59 [17]

The application makes use of a natural language processing (NLP) engine, together with application-specific knowledge, written in a concept specification language. Using NLP techniques, the entities and relationships that act as indicators of recoverable claims are mined from management notes, call centre logs and patient records to identify medical claims that require further investigation. Text mining techniques can then be applied to find dependencies between different entities, and to combine indicators to provide scores to individual claims. Claims are scored to determine whether they involve potential fraud or abuse, or to determine whether claims should be paid by or in conjunction with other insurers or organizations. Dependencies between claims and other records can then be combined to create cases. Issues related to the design of the application are discussed, specifically the use of rule-based techniques which provide a capability for deeper analysis than traditionally found in statistical techniques.

**[6].Research By: K. Bun and M. Ishizuka.**

**Title: “‘Topic extraction from news archive using TF\*PDF algorithm’”**

Since the Web became widespread, the amount of electronically available information online, especially news archives, has proliferated and threatens to become overwhelming. We propose an information system that will extract main topics in a news archive on a weekly basis. By obtaining a weekly report, a user can know what the main news events were in the past week

In general, related research on subject identification is classified into two types.

First one is term weighting method to extract useful terms that is relevant to collected documents and modeled also. Second is TF-IDF mostly used for term weighting in Natural language processing and information extraction process [7].

**[7] Research by: Jacques Savoy, Olena Zubaryeva**

**Titled: “Classification Based on Specific Vocabulary”**

1. Text Organization: This is description based method. This description method known as Lemma. Lemma causes same set of words like eat, eats etc.
2. Support Vector Machine (SVM): Derived from vector space model. TF-IDF method use to find weight of each term.[8]

**[8] Research by: Dengya Zhu, Jitian .**

**Titled: “R-tfidf, a Variety of tf-idf Term Weighting Strategy in Document Categorization”**

‘R-TF-IDF’: this formula is an enhancement over ‘TF-IDF’. Here we multiply the TF-IDF formula with an adjusting factor. This factor increase the importance of term frequency in a document and punish the terms that appear less frequently in a document where as have relatively higher term frequency weighting [9] .

**[9]. Research by: Catherine Blake published in IEEE 2010**

**Titled: “A Comparison of Document, Sentence, and Term Event Spaces “**

Zipf’s Law: f = ( k / r )

where f 🡪 frequency of the word

r 🡪 rank of the word.

But this can be used only for short text e.g. News article [10].

**[10]. Research by: Ying Chen, Wenping Guo, Xiaoming Zhao, 2010, IEEE**

**Titled: “A semantic Based Information Retrieval Model for Blog [11]”.**

Metadata Management: This management includes retrieval of metadata, processing in semantically way, semantically encoding of Metadata [11].

Log retrieval module: Log retrieval includes extend query semantically, process query semantically based on particular domain semantic selective ontology and SPARQL [11].

**[11].Research By: Mukhrjee, A. and B. Liu, 2010. Titled: “Improving Gender Classification of Blog Authors”,**

F-measure: F-measure explores the notion of implicitness of text and is a unitary measure of text relative contextuality and formality. Contextuality and formality can be captured by certain part of speech [12].

EFS Algorithm: EFS takes the best of both worlds. It first uses a number of feature selection criteria to rank the features following the filter model. Upon ranking, the algorithm generates some candidate feature subsets which are used to find the final feature set based on classification accuracy using the wrapper model [12].

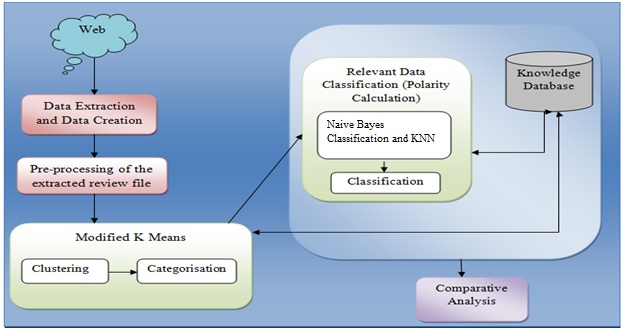
**CHAPTER 3**

**PROPOSED METHODOLOGY AND IMPLEMENTATION DETAIL**

**3.1. PROPOSED SYSTEM**

The proposed architecture of four modules: user interface, log preprocessing, Feature Clustering using Modified K-means, and Naïve Bayes for Classification, Training and testing using KNN for more accurate categorization of opinion. This system can solve irrelevant data and more accuracy by associating KNN with Naïve Bayes Classification algorithm.

**3.1.1 Framework Model**

****

**Figure 3.1: Framework Model**

A. TEXT PRE-PROCESSING MODULE: Text Pre-processing for extract relevant information from WebPages can be divided into three major stages:

1) Information collects from WebPages as unstructured corpus 2) Remove irrelevant data from WebPages; and 3) Information extraction from content blocks. Further this information will be use for classification of sentiment

**i. Text Preprocessing Structure**

In Text Preprocessing, extract relevant information from WebPages, we are using Crawler and parser for extract the information regarding blogging sites, crawled and parsed for information collection and for text processing on those data. Fig. 1 shows the process. First of all we need to extract relevant text from irrelevant data.

**Remove irrelevant data**

**Unstructured Corpus**

**Web Crawler**

**Static Data**

**Parser**

**Information extraction from relevant data**

**Words**

**Structured corpus**

**Dictionary**

**Fig 3.2:** Information Extraction Process from WebPages.

**ii. Remove irrelevant data**

For extraction of the relevant text from the irrelevant one, we are using Jsoup. To deal with real world HTML there is a java library named Jsoup. It gives a very comfort API for manipulating and extracting data, by using the best of DOM, CSS, and jquery like methods. jsoup implements the WHATWG HTML5 specification and parses HTML to the same DOM as modern browsers do.The functions of the Jsoup are:

* Scrape and parse HTML from a URL, file, or string.
* Find and extract data, using DOM traversal or CSS selectors.
* Manipulate the HTML elements, attributes, and text.
* Clean user-submitted content against a safe white-list, to prevent XSS attacks.
* Output  tidy HTML

**iii.Feature Definition**

Words and Stems

Though one may certainly represent a document by the raw words in it, a classic technique in information retrieval is to stem the words to their morphological roots. Stemmed feature vectors are smaller in size, they aggregate across occurrences of variants of a given word. Stemming has had mixed success in both information retrieval and text mining., for example, show that stemming produces mixed results on different datasets. They conclude that “corpus of reviews is highly sensitive to minor details of language, and these may be glossed over by the stemmer”. An example they observe is that negative reviews tend to occur more frequently in the past tense, since the product might have been returned.

Binary versus Term Frequency Weights

A standard approach in information retrieval is to use term frequency (TF) weights to indicate the relative importance of features in document representations. However, some research has shown that binary weighting (0 if the word appears in the document, 1 otherwise) is more beneficial for polarity classification . In a study of the standard information retrieval weighting schemes in SA, [60] found that using binary features is better than raw term frequency, though a scaled TF version28 performs as well as binary. Thus, we include runs in our experiments which compare the two weighting schemes.

**Negations**

Negations such as not and never are often included in stopword lists, and hence are removed from the text analysis. Combined with other words, though, negations reverse the polarity of words. Because polarity classification may be affected by negations, SA researchers have tried incorporating them into the feature vector. We take the approach of [13] who use a heuristic to identify negated words and create a new feature by appending NOT- to the words (for example, a phrase “don’t like” results in feature NOT-like).

**N-grams**

Negation phrases discussed above can be considered as a special case of ngrams, which are ordered sets of words. The benefit of using n-grams instead of single words as features comes in being able to capture some dependencies between the words and the importance of individual phrases. In a study of subjective text fragments, [12] found a significant improvement in polarity classification task using high n (up to 6). However, it is unclear if n-grams are as useful in a smaller dataset where there may not be enough data to capture information about their occurrence patterns. In our experiments, we generate features of up to 3-grams using CMU

**Phrases**

Since n-grams are often synthetic, in that they do not necessarily represent a semantically cohesive part of text, we explore the use of grammatical phrases as features. Using a CRF-based phrase chunker (http://jtextpro.sourceforge.net/), we break the text into phrases and use these as features. We further explore phrase features with modifications below.

**1V.Feature Selection**

**Frequency-Based Selection**

In text modeling, it is often the practice to remove words which appear rarely in the corpus. These are presumed to be perhaps misspellings, that do not help in generalization during classification. On the other hand, words that occur only once in a given corpus have been found to be high-precision indicators of subjectivity [89].

Rare terms, thus, may serve an important role in classification, and so we test various cutoffs using frequency counts.

**Mutual Information Based Selection**

The performance of the classifier may also be improved by removing some of the less useful features. One of the common feature selection measurements is expected Mutual Information [46]. Usually the features are scored by the expected MI and top several are taken as the most useful in classification. This is also the approach we take.

**Part of Speech-Based Selection**

In particular for SA, certain POS have been determined to be more useful in classification tasks. For example, [3] show that using adjectives and adverbs works better than using adjectives alone. [9] also use verbs for sentiment classification. If indeed adjectives are important factors in predicting sentiment polarity [63], limiting the feature space to only these may improve classifier performance by removing less useful words. We test this notion by retaining only words that are adjectives, verbs, and nouns individually and in combination.

**Lexicon-Based Selection**

Similarly, sentiment-annotated lexicons may be used for feature selection. By selecting terms which are indicative of strong sentiment, less useful features may be excluded from the feature set. Popular lexicons are the extensions of WordNet (http://wordnet.princeton.edu/), a large lexical database of English. SentiWordNet, for example, contains polarity and objectivity labels for the WordNet terms [21]. In WordNet-Affect [80] take advantage of synsets - word groupings in WordNet - to label31 each synset with affective labels. Both have been widely used in the community, and we use both lexicons in our analysis.

**V. Feature Generalization**

Phrase Generalization (POS-driven)

To avoid the problem of data sparsity we generalize the phrases described earlier by replacing some of the words in each phrase with their POS. The most drastic generalization is replacing all words with POS, though this may remove too much information from the phrase. Instead, we may want to retain words belonging to important POS and generalize others. As discussed above, adjectives, verbs, and nouns may be indicative of sentiment polarity. We explore just how much these POS individually and in combination help in classification by generalizing all words by their POS except for adjectives. Likewise we study verbs and nouns.

Phrase Generalization (Lexicon-driven)

We may also wish to generalize phrases by considering sentiment-annotated lexicon words as important. We experiment with the three above lexicons: Affect Control Theory (ACT), SentiWordNet (SWN) and WordNet-Affect (WNA).

**K- Means**

K-Means tries to find the natural clusters in the data, by calculating the distance from the centers of the clusters. The position of centers is iteratively changed until the distances between all the points are minimal. The centers are initially randomly assigned. K-Means can find only local maximum, and the final label assignment can be suboptimal. Common practice is to repeat the algorithm on the same data multiple times, and to report the best result. We have repeated the procedure 10 times in our experiments. We have used Euclidean distance as dissimilarity metric between feature vectors. We use B3 measure to evaluate the performance of the classifiers. K-means clustering is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. K-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells. The problem is computationally difficult (NP-hard); however, there are efficient heuristic algorithms that are commonly employed and converge quickly to a local optimum. These are usually similar to the expectation-maximization algorithm for mixtures of Gaussian distributions via an iterative refinement approach employed by both algorithms. Additionally, they both use cluster centers to model the data; however, k-means clustering tends to find clusters of comparable spatial extent, while the expectation-maximization mechanism allows clusters to have different shapes. The algorithm has a loose relationship to the k-nearest neighbor classifier, a popular machine learning technique for classification that is often confused with k-means because of the k in the name. One can apply the 1-nearest neighbor classifier on the cluster centers obtained by k-means to classify new data into the existing clusters. This is known as nearest centroid classifier or Rocchio algorithm.

**Standard algorithm**

The most common algorithm uses an iterative refinement technique. Due to its ubiquity it is often called the k-means algorithm; it is also referred to as Lloyd's algorithm, particularly in the computer science community. Given an initial set of k means m1(1),…,mk(1) (see below), the algorithm proceeds by alternating between two steps:

Assignment step: Assign each observation to the cluster whose mean yields the least within-cluster sum of squares (WCSS). Since the sum of squares is the squared Euclidean distance, this is intuitively the "nearest" mean.[8] (Mathematically, this means partitioning the observations according to the Voronoi diagram generated by the means).

S\_i^{(t)} = \big \{ x\_p : \big \| x\_p - m^{(t)}\_i \big \|^2 \le \big \| x\_p - m^{(t)}\_j \big \|^2 \ \forall j, 1 \le j \le k \big\},

where each x\_p is assigned to exactly one S^{(t)}, even if it could be assigned to two or more of them.

Update step: Calculate the new means to be the centroids of the observations in the new clusters.

m^{(t+1)}\_i = \frac{1}{|S^{(t)}\_i|} \sum\_{x\_j \in S^{(t)}\_i} x\_j

Since the arithmetic mean is a least-squares estimator, this also minimizes the within-cluster sum of squares (WCSS) objective. The algorithm has converged when the assignments no longer change. Since both steps optimize the WCSS objective, and there only exists a finite number of such partitionings, the algorithm must converge to a (local) optimum. There is no guarantee that the global optimum is found using this algorithm.

The algorithm is often presented as assigning objects to the nearest cluster by distance. The standard algorithm aims at minimizing the WCSS objective, and thus assigns by "least sum of squares", which is exactly equivalent to assigning by the smallest Euclidean distance. Using a different distance function other than (squared) Euclidean distance may stop the algorithm from converging.[citation needed] Various modifications of k-means such as spherical k-means and k-medoids have been proposed to allow using other distance measures.

**Naive Bayes Classifier**

A Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem (from Bayesian statistics) with strong (naive) independence assumptions. A more descriptive term for the underlying probability model would be "independent feature model".

In simple terms, a naive Bayes classifier assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 4" in diameter. Even if these features depend on each other or upon the existence of the other features, a naive Bayes classifier considers all of these properties to independently contribute to the probability that this fruit is an apple.

Depending on the precise nature of the probability model, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model without believing in Bayesian probability or using any Bayesian methods.

In spite of their naive design and apparently over-simplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. In 2004, analysis of the Bayesian classification problem has shown that there are some theoretical reasons for the apparently unreasonable efficacy of naive Bayes classifiers. Still, a comprehensive comparison with other classification methods in 2006 showed that Bayes classification is outperformed by more current approaches, such as boosted trees or random forests.

An advantage of the naive Bayes classifier is that it requires a small amount of training data to estimate the parameters (means and variances of the variables) necessary for classification. Because independent variables are assumed, only the variances of the variables for each class need to be determined and not the entire covariance matrix.

Naive Bayes is a simple model which works well on text

categorization [5]. We use a multinomial Naive Bayes model.

Class c∗ is assigned to tweet d, where

c∗ = argmaccPNB(c|d)

PNB(c|d) := (P(c) ∑m i=1 P(f|c)ni(d)  ) / P(d)

In this formula, f represents a feature and ni(d) represents the count of feature fi found in tweet d. There are a total of m features. Parameters P(c) and P(f|c) are obtained through maximum likelihood estimates, and add-1 smoothing is utilized for unseen features

**Probabilistic model**

Abstractly, naive Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector \mathbf{x} = (x\_1, \dots, x\_n) representing some n features (independent variables), it assigns to this instance probabilities

p(C\_k \vert x\_1, \dots, x\_n)\,

for each of K possible outcomes or classes.[7]

The problem with the above formulation is that if the number of features n is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable. Using Bayes' theorem, the conditional probability can be decomposed as

p(C\_k \vert \mathbf{x}) = \frac{p(C\_k) \ p(\mathbf{x} \vert C\_k)}{p(\mathbf{x})} \,

In plain English, using Bayesian probability terminology, the above equation can be written as

\mbox{posterior} = \frac{\mbox{prior} \times \mbox{likelihood}}{\mbox{evidence}} \,

In practice, there is interest only in the numerator of that fraction, because the denominator does not depend on C and the values of the features F\_i are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model

p(C\_k, x\_1, \dots, x\_n)\,

which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

\begin{align} p(C\_k, x\_1, \dots, x\_n) & = p(x\_1, \dots, x\_n, C\_k) \\ & = p(x\_1 \vert x\_2, \dots, x\_n, C\_k) p(x\_2, \dots, x\_n, C\_k) \\ & = p(x\_1 \vert x\_2, \dots, x\_n, C\_k) p(x\_2 \vert x\_3, \dots, x\_n, C\_k) p(x\_3, \dots, x\_n, C\_k) \\ & = \dots \\ & = p(x\_1 \vert x\_2, \dots, x\_n, C\_k) p(x\_2 \vert x\_3, \dots, x\_n, C\_k) \dots p(x\_{n-1} \vert x\_n, C\_k) p(x\_n \vert C\_k) p(C\_k) \\ \end{align}

Now the "naive" conditional independence assumptions come into play: assume that each feature F\_i is conditionally independent of every other feature F\_j for j\neq i, given the category C. This means that

p(x\_i \vert x\_{i+1}, \dots ,x\_{n}, C\_k ) = p(x\_i \vert C\_k)\,,

for i =1, \dots ,n-1 . Thus, the joint model can be expressed as

\begin{align} p(C\_k \vert x\_1, \dots, x\_n) & \varpropto p(C\_k, x\_1, \dots, x\_n) \\ & \varpropto p(C\_k) \ p(x\_1 \vert C\_k) \ p(x\_2\vert C\_k) \ p(x\_3\vert C\_k) \ \cdots \\ & \varpropto p(C\_k) \prod\_{i=1}^n p(x\_i \vert C\_k)\,. \end{align}

This means that under the above independence assumptions, the conditional distribution over the class variable C is:

p(C\_k \vert x\_1, \dots, x\_n) = \frac{1}{Z} p(C\_k) \prod\_{i=1}^n p(x\_i \vert C\_k)

where the evidence Z = p(\mathbf{x}) is a scaling factor dependent only on x\_1, \dots, x\_n, that is, a constant if the values of the feature variables are known.

**Constructing a classifier from the probability model**

The discussion so far has derived the independent feature model, that is, the naive Bayes probability model. The naive Bayes classifier combines this model with a decision rule. One common rule is to pick the hypothesis that is most probable; this is known as the maximum a posteriori or MAP decision rule. The corresponding classifier, a Bayes classifier, is the function that assigns a class label \hat{y} = C\_k for some k as follows:

\hat{y} = \underset{k \in \{1, \dots, K\}}{\operatorname{argmax}} \ p(C\_k) \displaystyle\prod\_{i=1}^n p(x\_i \vert C\_k).

**Parameter estimation and event models**

A class' prior may be calculated by assuming equiprobable classes (i.e., priors = 1 / (number of classes)), or by calculating an estimate for the class probability from the training set (i.e., (prior for a given class) = (number of samples in the class) / (total number of samples)). To estimate the parameters for a feature's distribution, one must assume a distribution or generate nonparametric models for the features from the training set.

The assumptions on distributions of features are called the event model of the Naive Bayes classifier. For discrete features like the ones encountered in document classification (include spam filtering), multinomial and Bernoulli distributions are popular. These assumptions lead to two distinct models, which are often confused.[9][10]

**Gaussian naive Bayes**

When dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a Gaussian distribution. For example, suppose the training data contain a continuous attribute, x. We first segment the data by the class, and then compute the mean and variance of x in each class. Let \mu\_c be the mean of the values in x associated with class c, and let \sigma^2\_c be the variance of the values in x associated with class c. Then, the probability distribution of some value given a class, p(x=v|c), can be computed by plugging v into the equation for a Normal distribution parameterized by \mu\_c and \sigma^2\_c. That is,

p(x=v|c)=\frac{1}{\sqrt{2\pi\sigma^2\_c}}\,e^{ -\frac{(v-\mu\_c)^2}{2\sigma^2\_c} }

Another common technique for handling continuous values is to use binning to discretize the feature values, to obtain a new set of Bernoulli-distributed features; some literature in fact suggests that this is necessary to apply naive Bayes, but it is not, and the discretization may throw away discriminative information.[4]

**Multinomial naive Bayes**

With a multinomial event model, samples (feature vectors) represent the frequencies with which certain events have been generated by a multinomial (p\_1, \dots, p\_n) where p\_i is the probability that event i occurs (or K such multinomials in the multiclass case). A feature vector \mathbf{x} = (x\_1, \dots, x\_n) is then a histogram, with x\_i counting the number of times event i was observed in a particular instance. This is the event model typically used for document classification, with events representing the occurrence of a word in a single document (see bag of words assumption). The likelihood of observing a histogram x is given by

p(\mathbf{x} \vert C\_k) = \frac{(\sum\_i x\_i)!}{\prod\_i x\_i !} \prod\_i {p\_{ki}}^{x\_i}

The multinomial naive Bayes classifier becomes a linear classifier when expressed in log-space:

\begin{align} \log p(C\_k|\mathbf{x}) & \varpropto \log \left( p(C\_k) \prod\_{i=1}^n {p\_{ki}}^{x\_i} \right) \\ & = \log p(C\_k) + \sum\_{i=1}^n x\_i \cdot \log p\_{ki} \\ & = b + \mathbf{w}\_k^\top \mathbf{x} \end{align}

where b = \log p(C\_k) and w\_{ki} = \log p\_{ki}.

If a given class and feature value never occur together in the training data, then the frequency-based probability estimate will be zero. This is problematic because it will wipe out all information in the other probabilities when they are multiplied. Therefore, it is often desirable to incorporate a small-sample correction, called pseudocount, in all probability estimates such that no probability is ever set to be exactly zero. This way of regularizing naive Bayes is called Laplace smoothing when the pseudocount is one, and Lidstone smoothing in the general case.

**KNN: -** h

The k-Nearest Neighbors algorithm (or k-NN for short) is a non-parametric method used for classification and regression.[1] In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression:

In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors.

k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all machine learning algorithms.

Both for classification and regression, it can be useful to assign weight to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of 1/d, where d is the distance to the neighbor.[2]

The neighbors are taken from a set of objects for which the class (for k-NN classification) or the object property value (for k-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

A shortcoming of the k-NN algorithm is that it is sensitive to the local structure of the data. The algorithm has nothing to do with and is not to be confused with k-means, another popular machine learning technique.

**Algorithm:-**

Example of k-NN classification. The test sample (green circle) should be classified either to the first class of blue squares or to the second class of red triangles. If k = 3 (solid line circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If k = 5 (dashed line circle) it is assigned to the first class (3 squares vs. 2 triangles inside the outer circle).

The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples.

In the classification phase, k is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the k training samples nearest to that query point.

A commonly used distance metric for continuous variables is Euclidean distance. For discrete variables, such as for text classification, another metric can be used, such as the overlap metric (or Hamming distance). In the context of gene expression microarray data, for example, k-NN has also been employed with correlation coefficients such as Pearson and Spearman.[3] Often, the classification accuracy of k-NN can be improved significantly if the distance metric is learned with specialized algorithms such as Large Margin Nearest Neighbor or Neighbourhood components analysis.

A drawback of the basic "majority voting" classification occurs when the class distribution is skewed. That is, examples of a more frequent class tend to dominate the prediction of the new example, because they tend to be common among the k nearest neighbors due to their large number.[4] One way to overcome this problem is to weigh the classification, taking into account the distance from the test point to each of its k nearest neighbors. The class (or value, in regression problems) of each of the k nearest points is multiplied by a weight proportional to the inverse of the distance from that point to the test point. Another way to overcome skew is by abstraction in data representation. For example, in a self-organizing map (SOM), each node is a representative (a center) of a cluster of similar points, regardless of their density in the original training data. K-NN can then be applied to the SOM

**3.2. TOOLS & TECHNOLOGY**

**HARDWARE REQUIREMENTS**

Processor PC with a Pentium II-class processor, 600 MHz (Recommended)

RAM 256 MB (Recommended)

Hard disk 10GB (4GB space required to install VS .NET and related software’s)

**SOFTWARE REQUIREMENTS**

**System Software**

The software can be executed in the Windows 2000 (Professional or Server) or Windows XP Professional environment.

A high-level programming language developed by Sun Microsystems. Java was originally called OAK, and was designed for handheld devices and set-top boxes. Oak was unsuccessful so in 1995 Sun changed the name to Java and modified the language to take advantage of the burgeoning World Wide Web.

Java is an object-oriented language similar to C++, but simplified to eliminate language features that cause common programming errors. Java source code files (files with a .java extension) are compiled into a format called bytecode (files with a .class extension), which can then be executed by a Java interpreter. Compiled Java code can run on most computers because Java interpreters and runtime environments, known as Java Virtual Machines (VMs), exist for most operating systems, including UNIX, the Macintosh OS, and Windows. Byte code can also be converted directly into machine language instructions by a just-in-time compiler (JIT).

Java is a general purpose programming language with a number of features that make the language well suited for use on the World Wide Web. Small Java applications are called Java applets and can be downloaded from a Web server and run on your computer by a Java-compatible Web browser, such as Netscape or Microsoft Internet Explorer.

Java is a programming language originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from Cand C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to bytecode (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is a general-purpose, concurrent, class-based, object-oriented language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere". Java is currently one of the most popular programming languages in use, and is widely used from application software to web applications.

The original and reference implementation Java compilers, virtual machines, and class libraries were developed by Sun from 1995. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of its Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java, GNU Classpath, and Dalvik.

**Principles of Java**

There were five primary goals in the creation of the Java language:

1. It should be "simple, object oriented and familiar".

2. It should be "robust and secure".

3. It should be "architecture neutral and portable".

4. It should execute with "high performance".

5. It should be "interpreted, threaded, and dynamic".

**Java Platform**

One characteristic of Java is portability, which means that computer programs written in the Java language must run similarly on any supported hardware/operating-system platform. This is achieved by compiling the Java language code to an intermediate representation called Java bytecode, instead of directly to platform-specific machine code. Java bytecode instructions are analogous to machine code, but are intended to be interpreted by a virtual machine (VM) written specifically for the host hardware. End-users commonly use a Java Runtime Environment (JRE) installed on their own machine for standalone Java applications, or in a Web browser for Java applets.

Standardized libraries provide a generic way to access host-specific features such as graphics, threading, and networking.

A major benefit of using bytecode is porting. However, the overhead of interpretation means that interpreted programs almost always run more slowly than programs compiled to native executables would. Just-in-Time compilers were introduced from an early stage that compile bytecodes to machine code during runtime. Over the years, this JVM built-in feature has been optimized to a point where the JVM's performance competes with natively compiled C code.

**Implementations**

Sun Microsystems officially licenses the Java Standard Edition platform for Linux, Mac OS X and Solaris. Although in the past Sun has licensed Java to Microsoft, the license has expired and has not been renewed. Through a network of third-party vendors and licensees,[24] alternative Java environments are available for these and other platforms.

Sun's trademark license for usage of the Java brand insists that all implementations be "compatible". This resulted in a legal dispute with Microsoft after Sun claimed that the Microsoft implementation did not support RMI or JNI and had added platform-specific features of their own. Sun sued in 1997, and in 2001 won a settlement of US$20 million, as well as a court order enforcing the terms of the license from Sun.As a result, Microsoft no longer ships Java with Windows, and in recent versions of Windows, Internet Explorer cannot support Java applets without a third-party plugin. Sun, and others, have made available free Java run-time systems for those and other versions of Windows.

Platform-independent Java is essential to the Java EE strategy, and an even more rigorous validation is required to certify an implementation. This environment enables portable server-side applications, such as Web services, Java Servlets, and Enterprise JavaBeans, as well as with embedded systems based on OSGi, using Embedded Java environments. Through the new GlassFish project, Sun is working to create a fully functional, unified open source implementation of the Java EE technologies.

Sun also distributes a superset of the JRE called the Java Development Kit (commonly known as the JDK), which includes development tools such as the Java compiler, Javadoc, Jar, and debugger.

**Performance**

Programs written in Java have a reputation for being slower and requiring more memory than those written in C. However, Java programs' execution speed improved significantly with the introduction of Just-in-time compilation in 1997/1998 for Java 1.1,the addition of language features supporting better code analysis (such as inner classes, StringBuffer class, optional assertions, etc.), and optimizations in the Java Virtual Machine itself, such as HotSpot becoming the default for Sun's JVM in 2000.

To boost even further the speed performances that can be achieved using the Java language, Systronix made JStik, a microcontroller based on the aJile Systems line of embeddedJava processors. In addition, the widely used ARM family of CPUs has hardware support for executing Java bytecode through its Jazelle option.

**Automatic memory management**

Java uses an automatic garbage collector to manage memory in the object lifecycle. The programmer determines when objects are created, and the Java runtime is responsible for recovering the memory once objects are no longer in use. Once no references to an object remain, the unreachable memory becomes eligible to be freed automatically by the garbage collector. Something similar to a memory leak may still occur if a programmer's code holds a reference to an object that is no longer needed, typically when objects that are no longer needed are stored in containers that are still in use. If methods for a nonexistent object are called, a "null pointer exception" is thrown.

One of the ideas behind Java's automatic memory management model is that programmers can be spared the burden of having to perform manual memory management. In some languages, memory for the creation of objects is implicitly allocated on the stack, or explicitly allocated and deallocated from the heap. In the latter case the responsibility of managing memory resides with the programmer. If the program does not deallocate an object, a memory leak occurs. If the program attempts to access or deallocate memory that has already been deallocated, the result is undefined and difficult to predict, and the program is likely to become unstable and/or crash. This can be partially remedied by the use of smart pointers, but these add overhead and complexity. Note that garbage collection does not prevent "logical" memory leaks, i.e. those where the memory is still referenced but never used.

Garbage collection may happen at any time. Ideally, it will occur when a program is idle. It is guaranteed to be triggered if there is insufficient free memory on the heap to allocate a new object; this can cause a program to stall momentarily. Explicit memory management is not possible in Java.

Java does not support C/C++ style pointer arithmetic, where object addresses and unsigned integers (usually long integers) can be used interchangeably. This allows the garbage collector to relocate referenced objects and ensures type safety and security.

As in C++ and some other object-oriented languages, variables of Java's primitive data types are not objects. Values of primitive types are either stored directly in fields (for objects) or on the stack (for methods) rather than on the heap, as commonly true for objects (but see Escape analysis). This was a conscious decision by Java's designers for performance reasons. Because of this, Java was not considered to be a pure object-oriented programming language. However, as of Java 5.0, auto boxing enables programmers to proceed as if primitive types were instances of their wrapper class.

**CHAPTER 4**

**RESULT AND DISCUSSION**

**Comparsion** Table 4.1

|  |  |  |
| --- | --- | --- |
| **Name of Algorithm** | **Dataset** | **Accuracy(%)** |
| Naive Bayes | 500 mobile dataset | 79.66 |
| KNN | 500 mobile dataset | 83.59 |
| Modified K-Means +NB | 500 mobile dataset | 89 |
| Modified K-Means + KNN +Naïve Bayes | 500 mobile dataset | 91 |

**Comparison table**

**CHAPTER 5**

**CONCLUSION**

Above methods has been applied on mobile review .We proposed a method using Naïve Bayes, KNN and modified k means clustering and found that it is more accurate than naïve bayes and KNN techniques individually. We obtained an overall classification accuracy of 91% on the test set of 500 mobile reviews. The running time of our algorithm is O(n + V log V) for training and O(n) for testing, where n is the number of words in the documents (linear) and V the size of the reduced vocabulary. It is much faster than other machine learning algorithms like Maxent classification or Support Vector Machines which take a long time to converge to the optimal set of weights. The accuracy is comparable to that of the current state-of-the-art algorithms used for sentiment classification on mobile reviews.

It achieved a better or similar accuracy when compared to more complicated models like SVMs, auto encoders, contextual valence shifters, matrix factorization, appraisal groups etc.

From our point of view MKM and Naïve Bayes is best suitable for text based classification and KNN for social interpretation. In future we will be finding out the best result of sentiment analysis by applying other method on social networking reviews.

**REFERENCE**

[1] G.Vinodhini and RM.Chandrasekaran, **“Sentiment Analysis and Opinion Mining: A Survey”,** Volume 2, Issue 6, June 2012 ISSN: 2277 128X International Journal of Advanced Research in Computer Science and Software Engineering

[2] Zhongwu Zhai, Bing Liu, Hua Xu and Hua Xu, **“Clustering Product Features for Opinion Mining”**, WSDM’11, February 9–12, 2011, Hong Kong, China. Copyright 2011 ACM 978-1-4503-0493- 1/11/02...$10.00

[3] Singh and Vivek Kumar, “**A clustering and opinion mining approach to socio-political analysis of the blogosphere”**, Computational Intelligence and Computing Research (ICCIC), 2010 IEEE International Conference.

[4] Alexander Pak and Patrick Paroubek, **“Twitter as a Corpus for Sentiment Analysis and Opinion Mining,”**

[5] Bing Liu. **“Sentiment Analysis and Opinion Mining, Morgan & Claypool Publishers, May 2012.”**

[6] V. S. Jagtap and Karishma Pawar, **“Analysis of different approaches to Sentence-Level Sentiment Classification”,** International Journal of Scientific Engineering and Technology (ISSN : 2277-1581) Volume 2 Issue 3, PP : 164-170 1 April 2013

[7]. K. Bun and M. Ishizuka, **“Topic extraction from news archive using TF\*PDF algorithm”,** In Proceedings of Third International Conference on Web Information System Engineering.

[8].Jacques Savoy, Olena Zubaryeva, **“Classification Based on Specific Vocabulary”** published in 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology 978-0-7695-4513-4/11 2011 IEEE

[9]. Dengya Zhu, Jitian XIAO, **“R-tfidf, a Variety of tf-idf Term Weighting Strategy in Document Categorization”,** published in 2011 Seventh International Conference on Semantics, Knowledge and Grids.

[10]. Catherine Blake **“A Comparison of Document, Sentence, and Term Event Spaces”** published in IEEE 2010

[11]. Ying Chen, Wenping Guo, Xiaoming Zhao, **“A semantic Based Information Retrieval Model for Blog”,** Third International Symposium on Electronic Commerce and Security, 2010, IEEE

[ 12]. Mukhrjee, A. and B. Liu, **“Improving gender classification of weblog authors. Proceedings of Conference on Empirical Methods in Natural Language Processing”,** (EMNLP’ 10), 10RDF Primer. W3C Recommendation. http://www.w3.org/TR/rdf-primer, 2004.

[13] Jalaj S. Modha, Prof & Head Gayatri S. Pandi Sandip J. Modha, **“Automatic Sentiment Analysis for Unstructured Data”,** International Journal of Advanced Research in Computer Science and Software Engineering , Volume 3, Issue 12, December 2013

[14] R M. Chandrasekaran , G.Vinodhini, **“Sentiment Analysis and Opinion Mining: A Survey”,** International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 6, June 2012

[15] Bing Liu., **“Sentiment Analysis and Opinion Mining”,** Morgan & Claypool Publishers, May 2012.

[16] Arti Buche, Dr. M. B. Chandak, Akshay Zadgaonkar, **“OPINION MINING AND ANALYSIS: A SURVEY”**, International Journal on Natural Language Computing (IJNLC) Vol. 2, No.3, June 2013

[17]. Fred Popowich, **“Using Text Mining and Natural Language Processing for Health Care Claims Processing”,** SIGKDD Explorations. Volume 7, Issue 1 - Page 59

[18] Ms. Gaurangi Patil, Ms. Varsha Galande, Mr. Vedant Kekan3, Ms. Kalpana Dange, **“Sentiment Analysis Using Support Vector Machine”,** International Journal of Innovative Research in Computer and Communication Engineering, Vol. 2, Issue 1, January 2014, SSN(Online): 2320-9801, ISSN (Print): 2320-9798

[19] Aamera Z. H. Khan, Dr. Mohammad Atique, Dr. V. M. Thakare, **“Sentiment Analysis Using Support Vector Machine”**, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 5, Issue 4, April 2015, ISSN: 2277 128X

[20] Vivek Narayanan, Ishan Arora, Arjun Bhatia, **“Fast and accurate sentiment classification using an enhanced Naive Bayes model.”**,

[21] Pablo Gamallo, Marcos Garcia, **“Citius: A Naive-Bayes Strategy for Sentiment Analysis on English Tweets”** , 8th International Workshop on Semantic Evaluation (SemEval 2014), pages 171–175, Dublin, Ireland, August 23-24 2014.

[22] Ankush Sharma, Aakanksha, “**A Comparative Study of Sentiments Analysis Using Rule Based and Support Vector Machine”,** International Journal of Advanced Research in Computer and Communication Engineering, Vol. 3, Issue 3, March 2014, ISSN (Online) : 2278-1021, ISSN (Print) : 2319-5940,

[23]. Preety, Sunny Dahiya, **“SENTIMENT ANALYSIS USING SVM AND NAÏVE BAYES ALGORITHM”**, International Journal of Computer Science and Mobile Computing, ISSN 2320–088X, IJCSMC, Vol. 4, Issue. 9, September 2015, pg.212 – 219

[24] Lina L. Dhande and Dr. Prof. Girish K. Patnaik, **“Analyzing Sentiment of Movie Review Data using Naive Bayes Neural Classifier”**, International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), Volume 3, Issue 4 July-August 2014, ISSN 2278-6856

[25] Ashish Shukla, Rahul Misra, **“Sentiment Classification and Analysis Using Modified K-Means and Naïve Bayes Algorithm”**, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 5, Issue 8, August 2015, ISSN: 2277 128X