SENTIMENTAL ANALYSIS OF ONLINE CONTENT: A PRACTICAL APPROACH

A THESIS SUBMITTED

In Fulfillment of Requirements for the Degree

of

DOCTOR OF PHILOSOPHY

by

S.YAMINI
[Reg. No.: D12IT001]

To the

Department of Information Technology

Under the Supervision of

Dr. V. KHANAA

Faculty of Engineering and Technology

Bharath Institute of Higher Education and Research

(Deemed University)

173, Agaram Road, Selaiyur, Chennai – 600 073. India.

AUGUST 2016

DECLARATION

I declare that the thesis, entitled " Sentimental Analysis Of Online Content: A Practical Approach" , submitted by me for the degree of Doctor of Philosophy (Ph.D) is the record of research work carried out by me during the period from **July 2012 to June 2016** under the guidance of **DR.V.KHANAA** and has not formed the basis for the award of any degree, diploma associateship, fellowship or titles in this or any other University or any other similar institution of higher learning.

Signature of the candidate

Date:

Place:

CERTIFICATE

I certify that the thesis, entitled "Sentimental Analysis Of Online Content: A Practical Approach " submitted for the Degree of Doctor of Philosophy by **Ms.S.YAMINI** is the record of research work carried out by her during the period from **JULY 2012 to June 2016** under my guidance and supervision and that this research work has not formed the basis for the award of any degree , diploma, associateship, fellowship or other similar titles in this University or any other University or institution.

Signature of the Guide

Date:

Place:

ACKNOWLEDGEMENT

I express my sincere gratitude to beloved Founder **Dr.S.Jagathratchagan**, and Chancellor **Dr.J.Sundeep Aanand** and also Managing Director **Dr.Swetha Sundeep Aanand** for providing me the necessary facilities for the completion of my Synopsis.

I take great pleasure in expressing sincere thanks to Vice-Chancellor **Dr.M.Ponnavaikko** and Pro Vice-Chancellor **Dr.K.P.Thooyamani** for backing us in this synopsis.

I thank our Director **Prof.S.Theagarajan** and Dean Engineering **Dr.J.Hameed Hussain** for providing sufficient facilities for the completion of this Synopsis.

I take great pleasure in expressing sincere thanks to Dean-Research and Development **Dr.M.Sundarrajan** for backing us in this synopsis.

I express sincere thanks to our Guide **Dr.V.Khanaa**, for providing sufficient facilities for the completion and the support, encouragement of this Synopsis.

I express sincere thanks to **Dr.A.Kumaravel**, Professor and Head, Department of Information Technology,**Dr.K.P.Kaliyamurthie**, Professor and Head, Department of Computer Science Engineering ,**Dr.C.Lakshmi**, Professor and Head of Department of Software Engineering, SRM University, Chennai and **Dr.Krishna Mohanta**,Professor, Department of CSE,Sri Ramanujar College of Engineering, Chennai for their support and encouragement.

I also thank people who directly or indirectly gave me encouragement and support throughout the synopsis. I offer my heartfelt thanks to my friends, and other staff members and well-wishers for their continuous encouragement throughout my career. Finally, I would like to thank my family members and all my well wishers whose good wishes have brought me where I am and would continue to take me forward in the right direction.

ABSTRACT

Sentiment analysis (or) opinion mining is an area of research that analyzes opinions, sentiments, evaluations, attitudes, and emotions from a written text. The feelings of others have a key effect in our day by day process. Directed ways to deal with sentiment order neglects to fulfill execution when moving to different areas. These choices range from purchasing an item, to making interests in purchasing a property or to watch a film in a theater and so forth…. prior, individuals would look for assessments on items and administrations from sources, for example, known persons, neighbours or online networking. The web has a gigantic measure of opinionated data, as web journals, surveys, the unsupervised methodologies thrive. Ideas use reviews, assessment surveys, and online networking as a device to obtain criticism on their items and services. The web is the impetus for these progressions. More than eighty percent of data on the Internet is unstructured. It so happens that more than eighty rate of information on the Internet is unstructured and is accessible from input fields in review, sites, wikis etc. Since the execution is subject to the alternatives of the data, numerous studies dedicate on building capable information accessible with cautious designing. In this, we address the outline of systems, ways and means which are steady and set apart as the fundamental field in the area of Sentiment Analysis. Many pioneering research works have been proposed in the past with regards to this research area. This immense volume of information may forces potential beneficial business related data, which when separated keenly and spoke to sensibly, can be a mine of gold for administrations Research &Development (R&D), attempting to add library an item in light of prominent popular opinion.Our work aims at forfeiting these researches with greater accuracy and potential to excel in both retail and corporate forums.

Keywords: Sentiment analysis, opinion mining, text mining, sentiment extraction , information retrieval….

ஆய்வுசுருக்கம்

உணர்வு ஆய்வு (அல்லது) கருத்து சுரங்க கருத்துக்களை, உணர்வுகளை, மதிப்பீடுகள், மனப்பாங்கு, மற்றும் உணர்ச்சிகளை ஒரு எழுதப்பட்ட உரை இருந்து ஆய்வு ஆராய்ச்சி ஒரு பகுதி உள்ளது. மற்றவர்கள் உணர்வுகளை நாள் செயல்முறை நம்முடைய நாளில் ஒரு முக்கிய விளைவை. உணர்வு பொருட்டு சமாளிக்க இயக்கிய வழிகளில் பல்வேறு இடங்களில் நகரும் போது நிறைவேற்ற மறந்துவிட்டது. இந்த தேர்வுகள் ஒரு பொருளை வாங்கும் இருந்து, ஒரு சொத்து வாங்கும் நலன்களை செய்யும் அல்லது முன்னும் பின்னுமாக ஒரு தியேட்டரில் மற்றும் ஒரு படம் பார்க்க ... வரை காணப்படுகின்றன. முன்னதாக, தனிநபர்கள் உதாரணமாக, என நபர்களாக, அண்டை அல்லது ஆன்லைன் நெட்வொர்க்கிங், ஆதாரங்களில் இருந்து பொருட்களை மதிப்பீடுகள் மற்றும் நிர்வாகங்கள் தேடுவார். இணைய வலை பத்திரிகைகள், ஆய்வுகள் என, கவனிக்கப்படாத செயல்முறையியல்களில் செழித்து, கருத்துக்களில் தரவு ஒரு பிரம்மாண்டமான நடவடிக்கை உள்ளது. ஆலோசனைகள் தங்கள் பொருட்களை மற்றும் சேவைகள் மீதான விமர்சனத்தை பெற ஒரு சாதனமாக விமர்சனங்களை, மதிப்பீடு ஆய்வுகள், மற்றும் ஆன்லைன் வலைப்பின்னல் பயன்படுத்த. வலை இந்த

விருத்தி உத்வேகம் உள்ளது. இணையத்தில் தரவு மேற்பட்ட எண்பது சதவீதம் கட்டமைப்பற்ற உள்ளது. இணையதளத்தில் தகவல் எண்பதுக்கும் மேற்பட்ட விகிதம் கட்டமைப்பற்ற மற்றும் மரணதண்டனை தரவு மாற்று உட்பட்டது என்பதால் விமர்சனம், தளங்கள், விக்கிகள் முதலியன உள்ளீடு துறைகள் அணுக என்று அது எப்படியென்றால், ஏராளமான ஆய்வுகள் கொண்டு அணுக திறன் தகவல் கட்டி சமர்ப்பிக்கிறேன் எச்சரிக்கையாக வடிவமைத்தல். இந்த, நாம் அமைப்புகள், வழிகள் மற்றும் நிலையான மற்றும் உணர்வு பகுப்பாய்வு பகுதியில் அடிப்படை துறையில் தவிர அமைக்க அதாவது எல்லைக்கோட்டை உரையாற்ற. பல முன்னோடி ஆராய்ச்சி படைப்புகள் இந்த ஆய்வுப் பகுதியில் குறித்து கடந்த காலத்தில் முன்மொழியப்பட்டுள்ளன. தகவல் இந்த மகத்தான தொகுதி முனைப்போடு பிரிக்கப்பட்ட மற்றும் புத்தியுடன் சொன்ன போது படைகள் சாத்தியமான நன்மை வணிக தொடர்பான தரவு, முக்கிய பிரபலமான கருத்து.எங்கள் வெளிச்சத்தில் நூலகம் உருப்படி சேர்க்க முயற்சிக்கும், நிர்வாகங்கள் ஆராய்ச்சி மற்றும் அபிவிருத்தி (R & D) பொன் ஒரு சுரங்கத்தில் இருக்க முடியும் இருக்கலாம் வேலை சில்லறை மற்றும் நிறுவன மன்றங்கள் ஊக்குவிப்பதுடன் அதிக துல்லியமாகவும் திறன் கொண்ட இந்த ஆய்வுகள் பந்தையத்தில் நோக்கம்.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATION

KEYWORDS	ABBREVIATION
SE	Sentiment Extraction
MLT	Machine Learning Techniques
SVM	Support Vector Machines
PMI-IR	Point wise Mutual Information- Information Retrieval
GI	General Inquirer
KB	Knowledge Base
SA	SentimentAnalysis

CHAPTER 1

INTRODUCTION

1.1 SENTIMENT EXTRACTION:

Sentiment analysis – otherwise known as opinion mining .In essence, it is the process of determining the emotional tone behind a series of words, used to gain an understanding of the attitudes, opinions and emotions expressed within an online mention. Sentiment analysis is extremely useful in social media monitoring as it allows us to gain an overview of the wider public opinion behind certain topics. The applications of sentiment analysis are broad and powerful. The ability to extract insights from social data is a practice that is being widely adopted by organizations across the world.

Shifts in sentiment on social media have been shown to correlate with shifts in the stock market. The ability to quickly understand consumer attitudes and react accordingly but that is not to say that sentiment analysis is a perfect science at all. The human language is complex. Teaching a machine to analyze the various grammatical nuances, cultural variations, slang and misspellings that occur in online mentions is a difficult process. Teaching a machine to understand how context can affect tone is even more difficult. Humans are fairly intuitive when it comes to interpreting the tone of a piece of writing.

Consider the following sentence: “My flight’s been delayed. Brilliant!”

Most humans would be able to quickly interpret that the person was being sarcastic. We know that for most people having a delayed flight is not a good experience (unless there’s a free bar as recompense involved). By applying this contextual understanding to the sentence, we can easily identify the sentiment as

negative. Without contextual understanding, a machine looking at the sentence above might see the word “brilliant” and categories it as positive.

That’s not entirely dissimilar to how a linguist expert would teach a machine how to conduct basic sentiment analysis. As language evolves, the dictionary that machines use to comprehend sentiment will continue to expand. With the use of social media, language is evolving faster than ever before. 140 character limits, the need to be succinct and other prevailing memes have transformed the ways we talk to each other online. This of course brings with it many challenges.

We take all the words and phrases that imply positive or negative sentiment and apply rules that consider how context might affect the tone of the content. Carefully crafted rules help our software know the first sentence below is positive and the second is negative.

The above examples show how sentiment analysis has its limitations and is not to be used as a 100% accurate marker. As with any automated process, it is prone to error and often needs a human eye to watch over it. Sentiment Extraction (SE) deals with the retrieval of the opinion or mood conveyed in a block of unstructured text in relation to the domain of the document being analyzed.

The extraction is performed in steps:

At the lowest level, we have rating words such as adjectives or adverbs that play a key role in determining polarity of a sentence. Examples of positive rating words include “good”,”awesome”,”excellent” and soon. At the other end of the spectrum such as “bad”, “poor”, “abomination” and so on.

1) At the next level, we have contextual polarity of the rating word that takes into account local modifiers that precede or succeed the rating word.

2) At the highest level, these rating words are attached to some entity, typically the subject of some discussion.

As a complete example, in the sentence “The gouda was abysmal”, the entity is “gouda” and a negative sentiment is being expressed about this entity.

Unfortunately, many vendors haven’t been doing it as long as we have, so you need to be wary of over-reaching claims and confusing descriptions by the vendors rolling out early sentiment analysis engines. This paper will demystify sentiment scoring and explain how the Lexalytics sentiment analysis engine works. This includes a discussion of how and why we have extended the basic concept of document sentiment to the paragraph and entity level, and how this technology is being further extended to measure other indicators within content, including the assessment of threat, customer satisfaction and many other contextual indicators.

1.2. STRUCTURED TEXT:

It came from the name for a common language used to access database called Structured Query Language (SQL).SQL provides ways to manage data in database. In general structured mean orderly form. Eg: Excel.

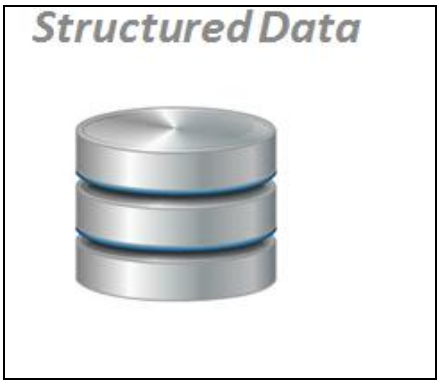


Fig 1.2.1 : Structured Data

Dependent data is facts, commonly textual content files, displayed in titled columns and rows that could without difficulty be ordered and processed by means of records mining equipment. This can be visualized as a superbly organized file wherever everything is known, labeled and straightforward to access. Most groups are possibly to be familiar with this form of records and already using it effectively.

1.3. UNSTRUCTURED TEXT

The World Wide Web has been dominated by unstructured content, and searching the web has primarily been based on techniques from Information Retrieval .It represents 80% of the data, which includes text and multimedia content. Eg: e-mail messages, videos, photos, audio files. They may have internal structure but neatly they don't fit the database. Unstructured text usually it may be either human or machine generated. Database systems are islands of structure in a sea of unstructured data sources.

Fig 1.3.1 : Unstructured Data

Several real world applications now need to create bridges for smooth integration of semi structured sources with existing structured databases for seamless querying.

Human generated:

- Mobile data- It gives text and the location.
- Social media data- YouTube, FB, Flickr, Twitter, LinkedIn.
- Text internal of the company- Documents, logos, survey, results, e-mails.

Machine generated:

- Satellite images- Information about weather data. Eg: Google earth.
- Radar or sonar- Vehicular and oceanic information.
- Scientific data- Atmospheric data and seismic image.

1.4 WHY SENTIMENT ANALYSIS?

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) Everyday enormous amount of data is created from social networks, blogs and other media and diffused in to the World Wide Web.

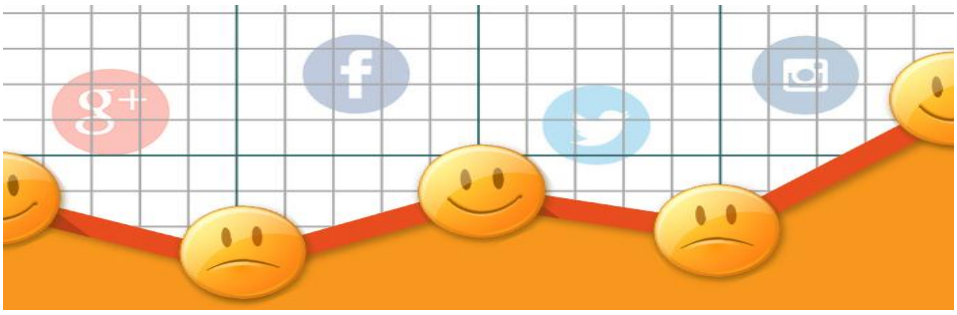


Fig 1.4.1 : Sentiment Analysis –for reviews

This huge data contains very crucial opinion related information that can be used to benefit businesses and other aspects of commercial and scientific industries. Manual tracking and extraction of this useful information is not possible, thus, Sentiment analysis is required. Sentiment Analysis is the phenomenon of extracting sentiments or opinions from reviews expressed by users over a particular subject, area or product online. It is an application of natural language processing, computational linguistics, and text analytics to identify subjective information from source data. It clubs the sentiments in to categories like positive or negative. Thus, it determines the general attitude of the speaker or a writer with respect to the topic in context.

1.4.1. APPLICATIONS OF SENTIMENT ANALYSIS IN REAL WORLD

(a) Product and service reviews:

Reviews of consumer products and service. Many automated websites provides feedback. Eg: Google product search.

(b) Reputation Monitoring:

Monitoring reputation of a specific brand.Eg: Twitter, Facebook.

(c) Result Prediction:

By analyzing sentiments from different sources one can predict the outcome of the event.Eg: Election. It enables managers to

track how voters feel about different issues, how they relate to speeches and actions of the candidate.

(d) Decision making:

Sentiment analysis uses these various sources to find the articles that discuss the aggregate the score. Eg: The Stock Sonar graphically shows positive and negative sentiment of each stock.

1.5. TECHNIQUES FOR SENTIMENT EXTRACTION:

There are two main techniques for sentiment classification: symbolic techniques and machine learning techniques. The symbolic approach uses manually crafted rules and lexicons, where the machine learning approach uses unsupervised, weakly supervised or fully supervised learning to construct a model from a large training corpus. We proposed a system which uses machine learning techniques instead of symbolic techniques to provide the polarity for sentences present in the World Wide Web.

1.5.1. MACHINE LEARNING TECHNIQUES

a. Supervised Methods:

Supervised learning is the machine learning task of deriving a function from marked training data. The training data consist of a set of training examples. In supervised learning, every case analyse an information object (normally a vector) and coveted yield esteem (likewise called the supervisory sign). A supervised learning calculation breaks down the preparation information and produces a function, which can be utilized for mapping new cases. An ideal situation will take into consideration the calculation to accurately decide the class names for unknown occurrences. This

requires the taking in calculation to sum up from the preparation information to unknown circumstances in a "sensible" manner.

In order to train a classifier for sentiment recognition in text classic supervised learning techniques (e.g Support Vector Machines, naïve Bayes Multinomial, Hidden Markov Model) can be used. A supervised approach entails the use of a labeled training corpus to learn classification function. The method that in the literature often yields the highest accuracy regards a Support Vector Machine classifier. They are the ones we used in our experiments described below.

b. Unsupervised Methods:

Unsupervised learning is the machine learning undertaking of inducing a capacity to hide structure from unlabeled information. Since the cases given to the learner are unlabeled, there is no mistake or reward sign to assess a potential arrangement. Unsupervised learning is firmly identified with the issue of thickness estimation in measurements. However unsupervised adapting additionally envelops numerous different systems that look to condense and clarify key elements of the information.

c. Reinforcement Learning:

Reinforcement learning varies from standard managed learning in that right information/yield sets are never exhibited, nor imperfect activities explicitly adjusted. Further, there is an attention on-line execution, which includes finding a harmony between investigation (of unknown region) and exploitation (of current information).

1.6 THE NETWORK: TWITTER

Twitter is an online social networking service and micro blogging service that enables its users to send and read text-based messages called \tweets". Tweets are publicly visible by default, but senders can restrict the message delivery to a limited crowd. Twitter is one of the largest microblogging service having over 500 million registered users as of 2012. Statistics revealed by the Infographics Labs5 suggest that back in the year 2012, on a daily basis 175 million tweets were communicated.

There is a large mass of people using twitter to express sentiments, which makes it an interesting and challenging choice for sentiment analysis. When so much attention is being paid to twitter, why not monitor and cultivate methods to analyze these sentiments. Twitter has been selected with the following purposes in mind.

Fig 1.6.1 : Networking on twitter

Twitter is an Open access social network.

- Twitter is an Ocean of sentiments (limited within 140 characters, i.e. high sentiment density).
- Twitter provides user friendly API making it easier to mine sentiments in real-time.

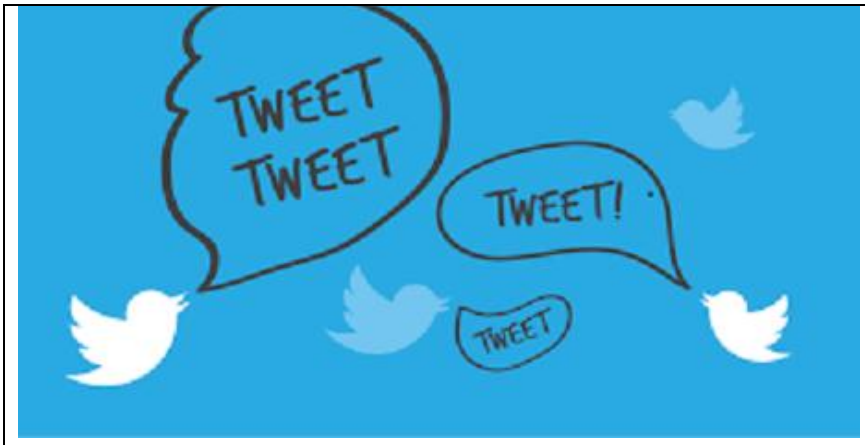


Fig 1.6.2 : Posting a message in twitter

CHAPTER 2

LITERATURE REVIEW

2.1.1 INTRODUCTION:

This section includes the explanation on concepts to be identified, the existing techniques and method how it can be incorporated in the research specified.

2.1.1 AUTOMATIC SENTIMENT ANALYSIS IN ON-LINE TEXT

Erik Boiy; Pieter Hens; Koen Deschacht; Marie-Francine Moens

. The idea of automatic sentiment analysis is important for marketing research, where companies wish to find out what the world thinks of their product; for monitoring newsgroups and forums, where fast and automatic detection of flaming is necessary; for analysis of customer feedback; or as informative augmentation for search engines.

The automatic analysis of sentiments on data found on the Web is useful for any company or institution caring about quality control. For the moment, getting user feedback means bothering him or her with surveys on every aspect the company is interested in. The problems with this approach are making a survey for each product or feature; the format, distribution and timing of the survey (asking to send a form right after purchase might not be very informative); and the reliance on the goodwill of people to take the survey. This method can be made obsolete by gathering such information automatically from the World Wide Web, where the large amount of available data creates the opportunity to do so.

One of the sources are blogs (short for “web logs”), a medium through which the blog owner makes commentaries about a certain subject or talks about his or her personal experiences, inviting readers to provide their own comments. Another source is the electronic discussions boards, where people can discuss all kinds of topics, or ask

for other people's opinions. We define a topic as the subject matter of a conversation or discussion, e.g. an event in the media or a new model of car, towards which the writer can express his or her views.

Emotions urge for actions and prompt for plans: an emotion gives priority for one or a few kind of *actions* to which it gives a sense of urgency. We use the term "action" to denote all mental or physical actions (that are the result of an emotion). This includes actions such as moving away from a negative event, mental processes, such as worrying about the event, and other effects that are direct result of the emotion, such as crying or going pale. A lot of linguistic scholars agree on the three dimensions of Osgood and al. [1], who investigated how the meaning of words can be mapped in a semantic space. Factor analysis extracted 3 major dimensions:

(1) Positive or negative evaluation

(2) A power, control or potency dimension and

(3) An activity, arousal or intensity dimension.

Although these dimensions are originally proposed as the dimensions of a semantic space, they can also be used to organize linguistic categories of emotion or for the automatic detection of emotions. Most research is devoted towards the appraisal component of emotions, and we will look into it a bit deeper by briefly going over Osgood's dimensions, giving some examples along the way.

(1) Evaluation (positive/negative)

The evaluation dimension is fairly straightforward; it contains all choices of words, parts of speech, word organization patterns, conversational techniques, and discourse strategies that express the orientation of the writer to the current topic. Evaluation is often expressed by using adjectives. e.g. “It was an amazing show.”

(2) Potency (powerful/unpowerful)

This dimension contains all elements that general express whether the writer identifies and commits himself towards the meaning of the sentence or whether he dissociates himself. From a psychological standpoint these phenomena are related to approach and avoidance behavior. This dimension consists of 3 sub-dimensions: proximity, specificity and certainty.

(2.1) Proximity (near/far)

This category contains all linguistic elements that indicate the 'distance' between the writer and the topic. The proximity from the writer to the current topic expresses whether the writer identifies himself with the topic or distances himself from it. e.g. “I'd like you to meet John.” versus “I'd like you to meet Mr. Adams.” (Social proximity)

(2.2) Specificity (clear/vague)

Specificity is the extent to which a conceptualized object is referred to by name in a direct, clear way; or is only implied, suggested, alluded to, generalized, or otherwise hinted at. e.g. “I left my / a book in your office.” (particular vs. general reference)

(2.3) Certainty (confident/doubtful)

This dimension expresses the certainty of the writer towards the expressed content. A stronger certainty indicates that the writer is entirely convinced about the truth of his writings and possibly indicates a stronger emotion. e.g. “It supposedly is a great movie.” versus “It definitely is a great movie.”

(3) Intensifiers (more/less)

When expressing emotions, a lot of the emotional words used do not express an emotion, but modify the strength of the expressed emotion. These words, the intensifiers, can be used to strengthen or weaken both positive and negative emotions.

e.g. “This is simply the best movie.” (Adverb)

“He had cuts all over.” (Quantifier)

“Where the hell have you been?” (Swearing)

To estimate the magnitude of a dimension of appraisal for a particular word, they compare the MPL of that word towards the positive and towards the negative end of that dimension. Both ends of a dimension are represented by prototype-words. The positive end of the evaluative dimension is represented by the word "good" and the negative end is represented by the word "bad". The prototypes for the potency dimension are respectively "strong" and "weak" and for the activity dimension "active" and "passive".

Feature	SVM	NBM	Maxen
Unigrams	85.45	81.45	84.80
Unigrams & Bigrams	86.35	83.95	87.40
Bigrams	85.35	83.15	85.40
Adjectives	75.85	82.00	80.30

Table 2.1.1.1: Results in terms of accuracy on the movie review corpus for different machine learning methods using a selection of features (and processing)

The Methods involved in this paper are:

Symbolic Techniques:

- Lexicon Based Techniques:

  -> Collection of words without considering relationship.

  -> Using aggregation function, web search, Wordnet.

- Sentiment of Sentences:

-> Grammatical, affective meaning is intensified and propagated towards a target.

Machine Learning Techniques:

Supervised Methods:

- Support Vector Machines(SVM),
- Naïve Bayes Multinomial(NBM),
- Maximum Entropy(Maxent),
- Unsupervised and Weakly-supervised Methods.

Feature	Baseline NBM	Our latest
accuracy %	84.25	90.25
precision/recall % for	64.52/49.93	74.39/75.62
precision/recall % for	88.48/72.96	87.43/82.70

Table 2.1.1.2 : Results on the blog corpus, comparing the results of the baseline approach and those of our latest methods

Advantages:

(i) The people who share their views usually have more pronounced opinions than average, which are additionally influencing others reading them, leading to so called word-of-mouth marketing. Extracting these opinions is thus extra valuable.

(ii)Opinions are extracted in real-time, allowing for quicker response times to market changes and for detailed time-based statistics that make it possible to plot trends over time.

Disadvantages:

(i)A downside is the feature vector size, which is substantially (over 5 times for unigrams) larger e.g. than when only adjectives are included. For the machine learning method we see a more substantial difference between NBM and both SVM and Maxent.

(ii)While many of the methods show encouraging results, there are still challenges to be overcome when applying them to data gathered from the World Wide Web, especially from blogs.

2.1.2.THUMBS UP OR THUMBS DOWN? SEMANTIC ORIENTATION APPLIED TO UNSUPERVISED CLASSIFICATION OF REVIEWS

Peter D. Turney *Institute for Information Technology National Research Council of Canada Ottawa, Ontario, Canada.*

This paper presents a simple unsupervised learning algorithm for classifying reviews as recommended (thumbs up) or not recommended (thumbs down). The classification of a review is predicted by the average semantic orientation of the phrases in the review that contain adjectives or adverbs. A phrase has a positive semantic orientation when it has good associations (e.g., "subtle nuances") and a negative semantic orientation when it has bad associations (e.g., "very cavalier").

In this paper, the semantic orientation of a phrase is calculated as the mutual information between the given phrase and the word "excellent" minus the mutual information between the given phrase and the word "poor". A review is classified as recommended if the average semantic orientation of its phrases is positive. The algorithm achieves an average accuracy of 74% when evaluated on 410 reviews from Epinions, sampled from four different domains (reviews of automobiles, banks, movies,

and travel destinations). The accuracy ranges from 84% for automobile reviews to 66% for movie reviews.

If you are considering a vacation in Akumal, Mexico, you might go to a search engine and enter the query "Akumal travel review". However, in this case, Google1 reports about 5,000 matches. It would be useful to know what fraction of these matches recommend Akumal as a travel destination.

With an algorithm for automatically classifying a review as "thumbs up" or "thumbs down", it would be possible for a search engine to report such summary statistics. This is the motivation for the research described here. Other potential applications include recognizing "flames" (abusive newsgroup messages) (Spertus, 1997) and developing new kinds of search tools (Hearst, 1992).

The algorithm takes a written review as input and produces a classification as output. The first step is to use a part-of-speech tagger to identify phrases in the input text that contain adjectives or adverbs (Brill, 1994). The second step is to estimate the semantic orientation of each extracted phrase (Hatzivassiloglou & McKeown, 1997). A phrase has a positive semantic orientation when it has good associations (e.g., "romantic ambience") and a negative semantic orientation when it has bad associations (e.g., "horrific events"). The third step is to assign the given review to a class, recommended or not recommended, based on the average semantic orientation of the phrases extracted from the review. If the average is positive, the prediction is that the review recommends the item it discusses. Otherwise, the prediction is that the item is not recommended.

The Techniques used in this paper are:

- PMI-IR(Point wise Mutual Information and Information Retrieval)

  PMI (word1, word2) = log2 [p (word1 & word2)/ p (word1) p(word2)]

- Semantic Orientation of Phrase.

SO (phrase) = PMI (phrase, “excellent”) – PMI (phrase, “poor”).

The PMI-IR algorithm is employed to estimate the semantic orientation of a phrase (Turney, 2001). PMI-IR uses Point wise Mutual Information (PMI) and Information Retrieval (IR) to measure the similarity of pairs of words or phrases. The se-mantic orientation of a given phrase is calculated by comparing its similarity to a positive reference word (“excellent”) with its similarity to a negative reference word (“poor”).

More specifically, a phrase is assigned a numerical rating by taking the mutual information between the given phrase and

the word “excellent” and subtracting the mutual information between the given phrase and the word “poor”. In addition to determining the direction of the phrase’s semantic orientation (positive or negative, based on the sign of the rating), this numerical rating also indicates the strength of the semantic orientation (based on the magnitude of the number).

The classification algorithm is evaluated on 410 reviews from Epinions2, randomly sampled from four different domains: reviews of automobiles, banks, movies, and travel destinations. Reviews at Epinions are not written by professional writers; any person with a Web browser can become a member of Epinions and contribute a review. Each of these 410 reviews was written by a different author. Of these reviews, 170 are not recommended and the remaining 240 are recommended (these classifications are given by the authors). Always guessing the majority class would yield an accuracy of 59%. The algorithm achieves an average accuracy of 74%, ranging from 84% for automobile reviews to 66% for movie reviews.

As an example, they present the following three sentences (Hatzivassiloglou & McKeown, 1997):

1. The tax proposal was simple and well received by the public.

2. The tax proposal was simplistic but well received by the public.

3. (*) The tax proposal was simplistic and well-received by the public.

The third sentence is incorrect, because we use “and” with adjectives that have the same semantic orientation (“simple” and “well-received” are both positive), but we use “but” with adjectives that have different semantic orientations (“simplistic” is negative).

Feature	Author’s Classification		
Average Semantic Orientation	Thumbs Up	Thumbs Down	Sum
Positive	28.33 %	12.50 %	40.83 %
Negative	21.67 %	37.50 %	59.17 %
Sum	50.00 %	50.00 %	100.00 %

Table 2.1.2.1 : Review for movie classification

Hatzivassiloglou and McKeown (1997) use a four-step supervised learning algorithm to infer the semantic orientation of adjectives from constraints on conjunctions:

1. All conjunctions of adjectives are extracted from the given corpus.

2. A supervised learning algorithm combines multiple sources of evidence to label pairs of adjectives as having the same semantic orientation or different semantic orientations. The result is a graph where the nodes are adjectives and links indicate sameness or difference of semantic orientation.

3. A clustering algorithm processes the graph structure to produce two subsets of adjectives, such that links across the two subsets are mainly different orientation links, and links inside a subset are mainly same-orientation links.

4. Since it is known that positive adjectives tend to be used more frequently than negative adjectives, the cluster with the higher average frequency is classified as having positive semantic orientation.

This algorithm classifies adjectives with accuracies ranging from 78% to 92%, depending on the amount of training data that is available.

Advantages:

The algorithm has three steps:

(1) Extract phrases containing adjectives or adverbs,

(2) Estimate the semantic orientation of each phrase, and

(3) Classify the review based on the average semantic orientation of the phrases.

The core of the algorithm is the second step, which uses PMI-IR to calculate semantic orientation (Turney, 2001). In experiments with 410 reviews from Epinions, the algorithm attains an average accuracy of 74%. It appears that movie reviews are difficult to classify, because the whole is not necessarily the sum of the parts; thus the accuracy on movie reviews is about 66%. On the other hand, for banks and automobiles, it seems that the whole is the sum of the parts, and the accuracy is 80% to 84%.

Drawback:

- The time required for queries and, for some applications, the level of accuracy that was achieved.
- The former difficulty will be eliminated by progress in hardware. The latter difficulty might be addressed by using semantic orientation combined with other features in a supervised classification algorithm.

2.1.3. OPINION MINING OF CUSTOMER FEEDBACK DATA ON THE WEB.,

Dongjoo Lee School of Computer Science and Engineering, Seoul National University Seoul 151-742, Republic of Korea.

The World Wide Web is growing at an alarming rate not only in size but also in the types of services and contents provided. Individual users are participating more actively and are generating vast amount of new data. These new Web contents include customer reviews and blogs that express opinions on products and services – which are collectively referred to as customer feedback data on the Web. As customer feedback on the Web influences other customer's decisions, these feedbacks have become an important source of information for businesses to take into account when developing marketing and product development plans.

The importance of automatically extracting actionable knowledge from customer feedback data on the Web, "opinion mining (OM)" has become a significant subject of research in the field of data mining. The ultimate goal of OM is to extract customer opinions (feedback) n products and present the information in the most effective way that serves the chosen objectives.

Let us consider an example of customer feedback.

“This camera is my first digital one and was super easy to learn to use. The picture looks great and it’s simple to get the correct exposure. The memory card that comes with the camera has a very small capacity though, (it holds about 4 photos) so a separate memory card is a necessity. I’m not very happy with the memory card.

In this example, we can extract several phrases such as ‘super easy to learn to use’, ‘the picture looks great’, ‘simple to get the correct exposure’, ‘very small capacity‘, and ‘not very happy with the memory card’, which convey customer’s opinion rather than facts. In particular, subjective words such as ‘super easy’, ‘looks great’, ‘simple’, ‘very small’, and ‘not very happy’ are used to express customer’s positive/negative sentiment regarding the product features, which are referred by ‘learn to use’, ‘picture’, ‘exposure’, ‘capacity’, and ‘photo’. Although information gathered from multiple reviews are more reliable compared to information from only one review, manually sorting through large amounts of review one by one requires a lot of time and cost for both businesses and customers. Therefore it is more efficient to automatically process the various reviews and provide the necessary information in a summarized form.ard.”

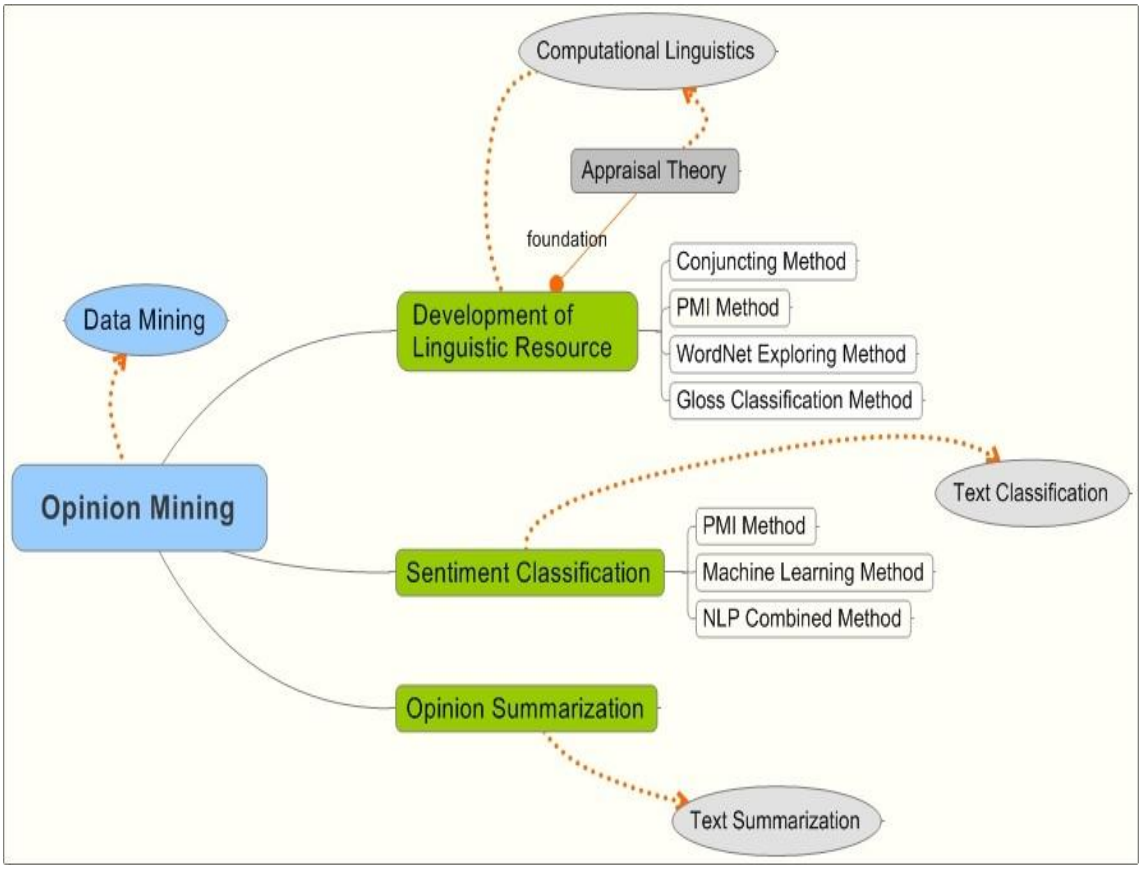


Figure 2.1.3.1 . Tasks for opinion mining and its relationship with related areas.

Methodology:

- PMI method:

  ->Measure of association used in information theory and statistics.

- WordNet Exploring:

  ->Share same orientation- Synonyms

  ->Opposite orientation- Antonyms.

- Gloss Classification Method:

  ->Similar orientation – Oriented Glosses

  ->without orientation – Non-Oriented Glosses.

Conjunction Method:

The work presented is the first attempt to automatically develop linguistic resources for opinion mining. The approach relies on an analysis of textual corpora that correlates linguistic features or indicators with semantic orientation. The authors demonstrated that conjunctions between adjectives provide indirect information about orientation, based on the hypothesis that "The conjoined adjectives and conjunctions usually have similar orientation, though 'but' is used with opposite orientation."

Their system identifies and uses this indirect information in the following steps:

First, all conjunctions of adjectives are extracted from the corpus along with relevant morphological relations. And then, a log-linear regression model combines information from different conjunctions to determine if each of the two conjoined adjectives is of the same or different orientation. The result is a graph with hypothesized same- or different-orientation links between adjectives.

Here, clustering algorithm that separates the adjectives into two subjects of different orientation is applied. It places as many words of the same orientation as possible into the same subset. Finally, the average frequencies in each group are compared and the group with the higher frequency is labeled as positive.

PMI Method:

PMI is a measure of association used in information theory and statistics. This can be defined as the following equation 1 which the log of this ratio is the amount of information that we acquire about the presence of one of the words when we observe the other.

$$PMI(x,y)=\log_2 N(p(x,y)/p(x)p(y))$$

There are several works that tries to develop sentiment related properties of words by means of PMI.

WordNet Exploring Method:

Hu et al utilized the adjective synonym set and antonym set in WordNet to predict the semantic orientation of adjectives. In WordNet, adjectives are organized into bipolar clusters and share the same orientation of their synonyms and opposite orientation of their antonyms. To assign orientation of an adjective, the synset of the given adjective and the antonym set are searched. If a synonym/antonym has known orientation, then the orientation of the given adjective could be set correspondingly. As the synset of an adjective always contains a sense that links it to the head synset, the search range is rather large. Given enough seed adjectives with known orientations, the orientations of all the adjective words can be predicted.

Gloss Classification Method:

Esuli and Sebastiani tried to develop linguistic resources by classifying term glosses. They assume that terms with similar orientation have similar glosses and terms without orientation have non-oriented glosses. They use semi-supervised learning methods. By using the manually assigned initial seed sets and expanding them by using the synonyms and antonyms defined in the thesaurus, they formed a final training set which was used for processing the learning step for binary text classifiers such as Naïve Bayes Classifier, Support Vector Machine (SVM). This was then used to find the sentiment related properties of the words in the rest of the test set.

They have developed a publicly available linguistic resource, SentiWordNet, in which each synset of WordNet is associated to three numerical scores Obj(s), Pos(s) and

Neg(s), describing how objective, positive, or negative the terms contained in the synset. The assumption that underlies their switch from terms to synsets is that different senses of the same term may have different opinion-related properties. Each of the three scores ranges from 0.0 to 1.0, and their sum is 1.0 for each synset. Given that the sum of the opinion-related scores assigned to a synset is always 1.0, it is possible to display these values in a triangle whose vertices are the maximum possible values for the three dimensions observed.

Advantages:

Product features are extracted and then sentiment of each feature is assigned.

Disadvantage:

Used mainly in Linguistic Resources. No specified polarity given for classified text.

2.1.4 OPINION POLARITY DETECTION USING WORD SENSE DISAMBIGUATION TO DETERMINE THE POLARITY OF OPINIONS

Tamara Martín-Wanton, Aurora Pons-Porrata Center for Pattern Recognition and Data Mining, Universidad de Oriente, Patricio Lumumba s/n, Santiago de Cuba

An unsupervised method determines the polarity of opinions. It uses a word sense disambiguation algorithm to determine the correct sense of the words in the opinion. The method is also based on SentiWordNet and General Inquirer to determine the polarity of the senses. Due to the characteristics of these external resources, the proposed method does not depend on the knowledge domain and can be extended to other languages.

Opinion Mining (also known as sentiment classification or subjectivity analysis) refers to a broad area of Natural Language Processing and Text Mining. It is concerned not with the topic a document is about, but with the opinion it expresses, that is, its aim is to determine the attitude (feelings, emotions and subjectivities) of a speaker or a writer with respect to some topic. A major task of Opinion Mining is the classification of the opinion's polarity, which consists in determine whether the opinion is positive, negative or neutral with respect to the entity to which it is referring (e.g., a person, a product, a movie, etc.).

Used Resources:

- WordNet
- SentiWordNet
- Subset of General Inquirer.
- WordSense Disambiguation

WordNet:

WordNet (Miller et al., 1993) is a lexical database based on psycholinguistic theories about the mental lexicon. In WordNet the words are grouped into sets of synonyms (synsets). Each synset is provided with a glossary and can be connected to other synsets by semantic relations (e.g., hypernymy, hyponymy, antonym, etc.).

SentimentWordNet :

SentiWordNet (Esuli and Sebastiani, 2006) is a lexical resource for opinion mining. Each synset in WordNet has assigned three values of sentiment: positive, negative and objective, whose sum is 1. It was semi-automatically built so all the results were not manually validated and some resulting classifications can appear incorrect.

General Inquirer:

General Inquirer (GI) (Stone et al., 1966) is an English dictionary that contains information about the words. For the proposed method we use the words labelled as positives, negatives and negations (Positiv, Negativ and Negate categories in GI).

From the Positiv and Negativ categories, we build a list of positive and negative words respectively. From the Negate category we obtain a list of polarity shifters terms (also known as valence shifters).

Word Sense Disambiguation (WSD) :

It consists on selecting the appropriate meaning of a word given the context in which it occurs. For the disambiguation of the words, we use the method proposed in (Anaya-Sánchez et al., 2006), which relies on clustering as a way of identifying semantically related word senses.

	Acc.	Prec.	Rec.	F1
GI-based	31.2	31.18	66.38	42.43
WSD-MFS	42.8	36.73	71.22	48.46
Our Method	44.3	37.66	72.11	49.41

Table 2.1.4.1: The proposed method against the GI-based method.

In this WSD method, the senses are represented as signatures built from the repository of concepts of WordNet. The disambiguation process starts from a clustering distribution of all possible senses of the ambiguous words by applying the Extended

Star clustering algorithm (Gil-García et al., 2003). Such a clustering tries to identify cohesive groups of word senses, which are assumed to represent different meanings for the set of words. Then, clusters that match the best with the context are selected. If the selected clusters disambiguate all words, the process stops and the senses belonging to the selected clusters are interpreted as the disambiguating ones.

Otherwise, the clustering are performed again (regarding the remaining senses) until a complete disambiguation is achieved. Once the correct sense for each word on the opinion is obtained, the method determines its polarity regarding the sentiment values for this sense in SentiWordNet and the membership of the word to the Positiv and Negativ categories in GI. It is important to mention that the polarity of a word is forced into the opposite class if it is preceded by a valence shifter (obtained from the Negate category in GI).

Finally, the polarity of the opinion is determined from the scores of positive and negative words it contains.

The GI-based method only takes into account the lists of positive and negative words of GI and handles valence shifters to determine the polarity of the headlines. Notice that, in this case, no disambiguation is carried out. The number of positive and negative words in the headline was calculated. If the number of positive words is greater than the number of negative words, then the headline is positive. On the contrary, if the number of positive words is less than that of negative words, the headline is negative. Finally, if there are neither positive nor negative words, then the headline is neutral.

The second method only differs from the proposed method in that it uses to disambiguate the MFS baseline. In WordNet, senses of a same word are ranked based on the frequency of occurrence of each sense in the SemCor corpus; the baseline is simply to assign as correct sense to each word its first sense in WordNet.

	Acc.	Prec.	Rec.	F1
ClaC	55.10	61.42	9.20	16.00
UPAR7	55.00	57.54	8.78	15.24
Our method	44.3	37.66	72.11	49.41
SWAT	53.20	45.71	3.42	6.36
CLaC-NB	31.20	31.18	66.38	42.43
SICS	29.00	28.41	60.17	38.60

Table 2.1.4.2 : Results of the valence annotation.

Advantages:

A new unsupervised method to opinion polarity detection has been introduced. Its most important novelty is the use of word sense disambiguation together with standard external resources for determining the polarity of the opinions. These resources allow the method to be extended to other languages and be independent of the knowledge domain.

Disadvantages:

The use of SentiWordNet leads to wrong annotations in different knowledge domain. In many cases our approach fails because the wrong annotations of SentiWordNet.

2.1.5. Semantic Orientation Applied to Unsupervised Classification of Reviews.

Peter D Turney Institute for Information Technology National Research Council of Canada Ottawa, Ontario, Canada, K1A 0R6 peter.turney@nrc.ca

The first step is to use a part-of-speech tagger to identify phrases in the input text that contain adjectives or adverbs The PMI-IR algorithm is employed to estimate the semantic orientation of a phrase (Turney, 2001). PMI-IR uses Point wise Mutual Information (PMI) and Information Retrieval (IR) to measure the similarity of pairs of words or phrases.

The Computational Linguistics (ACL), Philadelphia, July 2002, pp. 417-424. Proceedings of the 40th Annual Meeting of the Association for mantic orientation of a given phrase is calculated by comparing its similarity to a positive reference word ("excellent") with its similarity to a negative reference word ("poor"). More specifically, a phrase is assigned a numerical rating by taking the mutual information between the given phrase and the word "excellent" and subtracting the mutual information between the given phrase and the word "poor". In addition to determining the direction of the phrase's semantic orientation (positive or negative, based on the sign of the rating), this numerical rating also indicates the strength of the semantic orientation (based on the magnitude of the number).

Reviews at Epinions are not written by professional writers; any person with a Web browser can become a member of Epinions and contribute a review. In several of these applications, the first step is to recognize that the text is subjective and then the natural second step is to determine the semantic orientation of the subjective text.

For example, a flame detector cannot merely detect that a newsgroup message is subjective, it must further detect that the message has a negative semantic orientation;

otherwise a message of praise could be classified as a flame. Hearst (1992) observes that most search engines focus on finding documents on a given topic, but do not allow the user to specify the directionality of the documents (e.g., is the author in favor of, neutral, or opposed to the event or item discussed in the document?).

The directionality of a document is determined by its deep argumentative structure, rather than a shallow analysis of its adjectives. Sentences are interpreted metaphorically in terms of agents exerting force, resisting force, and overcoming resistance. It seems likely that there could be some benefit to combining shallow and deep analysis of the text. This work is most closely related to Hatzivassiloglou and McKeown's (1997) work on predicting the semantic orientation of adjectives.

This algorithm classifies adjectives with accuracies ranging from 78% to 92%, depending on the amount of training data that is available. The algorithm can go beyond a binary positive-negative distinction, because the clustering algorithm can produce a "goodness-of-fit" measure that indicates how well an adjective fits in its assigned cluster. Although they do not consider the task of classifying reviews, it seems their algorithm could be plugged into the classification algorithm presented, where it would replace PMI-IR and equation in the second step. However, PMI-IR is conceptually simpler, easier to implement, and it can handle phrases and adverbs, in addition to isolated adjectives.

The directionality of a document is determined by its deep argumentative structure, rather than a shallow analysis of its adjectives. Sentences are interpreted metaphorically in terms of agents exerting force, resisting force, and overcoming resistance. It seems likely that there could be some benefit to combining shallow and deep analysis of the text. send queries to AltaVista. Inspection of Equation shows that it takes four queries to calculate the semantic orientation of a phrase.

Advantages:

This system tracks online discussions about movies and displays a plot of the number of positive sentiment and negative sentiment messages over time. Messages are classified by looking for specific phrases that indicate the sentiment of the author towards the movie (e.g., "great acting", "wonderful visuals", "terrible score", "uneven editing"). Each phrase must be manually added to a special lexicon and manually tagged as indicating positive or negative sentiment. The lexicon is specific to the domain (e.g., movies) and must be built anew for each new domain. The company Mindfuleye7 offers a technology called Lexant™ that appears similar to Tong's (2001) system.

Disadvantages:

This will reduce the processing time to less than one second per review. The limitations of this work include the time required for queries and, for some applications , the level of accuracy that was achieved. The former difficulty will be eliminated by progress in hardware. The latter difficulty might be addressed by using semantic orientation combined with other features in a supervised classification algorithm.

CHAPTER 3

NEED FOR STUDY

3.1 Existing system

The Existing System consists of Sentiment Analysis or opinion mining aims to use automated tools to detect subjective information such as opinions, attitudes and feelings expressed in text.

The most important problem is that the process of sentiment extraction is not generic but highly domain specific. The vocabulary of a person, language ,resources used should be domain relevant in order to get meaningful results.

There exists a problem of subjectivity and neutral texts. A major problem lies in quantifying the polarities of the rating words, intensifiers, negators and the computed sentiment. The scale of polarity adapted and the mathematical results that follow from computations have to be mapped to something significant and tangible to the end user.

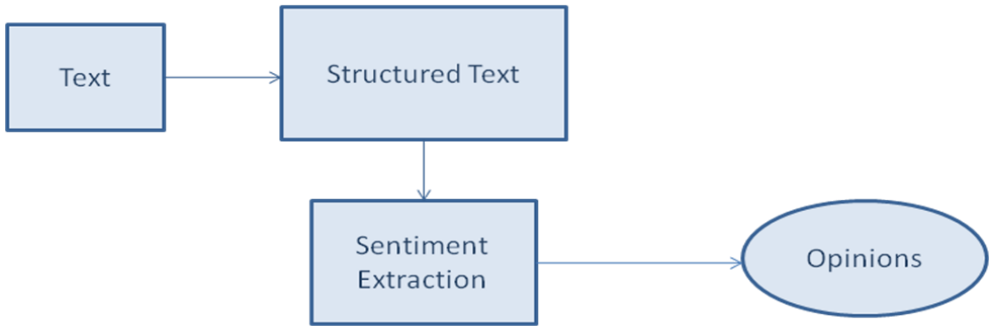


Fig 3.1.1. Existing System Architecture

3.2 PROPOSED SYSTEM

It consists of two basic components: word sense disambiguation and determination of polarity. The first, given an opinion, determines the correct senses of

its terms and the second, for each word sense determines its polarity, and from them gets the polarity of the opinion.

A preprocessing of the text is carried out including sentence recognizing, stop-word removing, part-of-speech tagging and word stemming by using the Tree Tagger tool.

Word Sense Disambiguation (WSD) works by selecting the appropriate meaning of a word given in the context. For the disambiguation of the words, use the method proposed in, which relies on clustering as a way of identifying semantically related word senses.

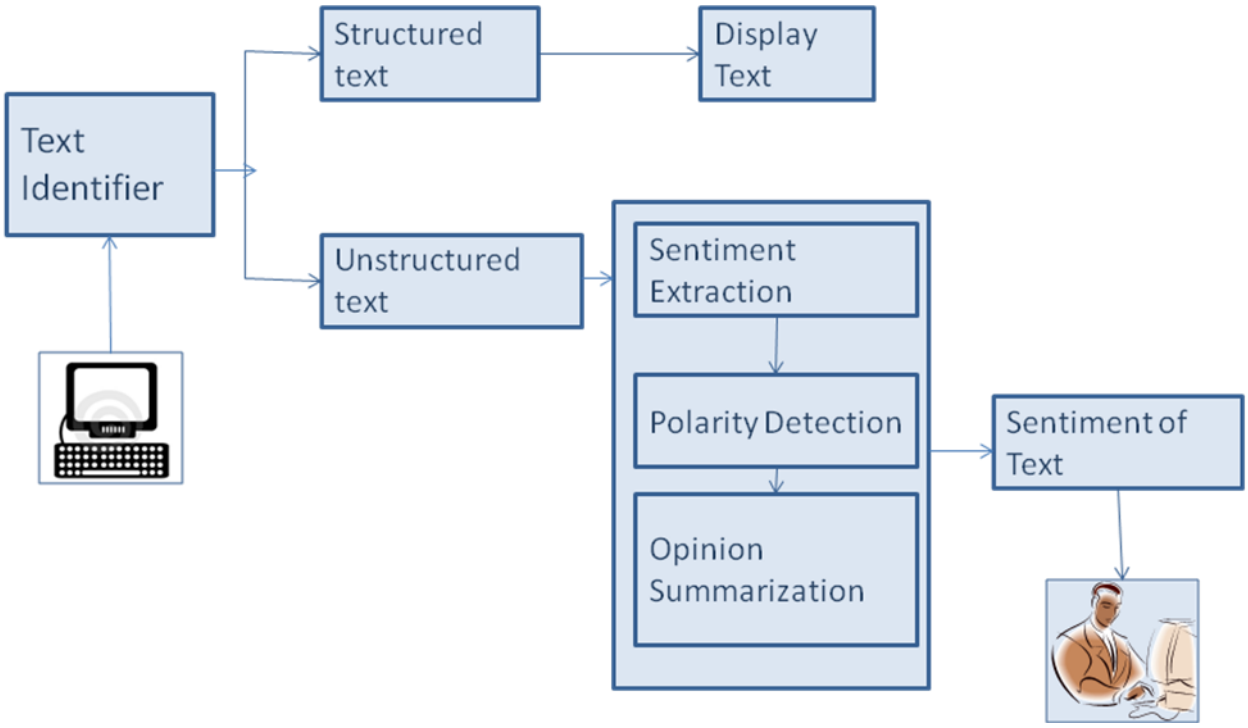


Fig.3.2.1. Proposed System Architecture.

The above architecture depicts the major functionality of the system. The input is given via system. Initially the Text identifier is used to differentiate whether the

entered text is structured or unstructured. Naive Bayes Classifier method is used for the process. If the entered text is structured then the text is displayed as such.

If the entered text is Unstructured then it enters into the process. The process is classified into three major Sub-processes. 1) Sentiment Extraction

2) Polarity Detection

3) Opinion Summarization.

In the first sub process the sentiments are extracted from the various users of the system. The users will provide various Sentiments about the text. Once the sentiments are gathered it moves to the next sub - process.

The Second process is to detect the polarity for the given sentiments. The polarity is based on three criteria: “Good”, “Poor”, “Neutral”. Based on the polarity the rating is given to the sentiments. Finally the opinions are summarized and the results are provided. The outcome of this process is the Sentiment of Text and it is provided to the user of the system.

Machine Learning techniques play the vital role for classification and retrieval process. Supervised methods are followed for the process. Support Vector Machine (SVM) is used for the classification process. PMI-IR method is used for information retrieval and polarity detection.

When a new user enters into the system they can view which text is related for their search and which doesn’t suit for their search.

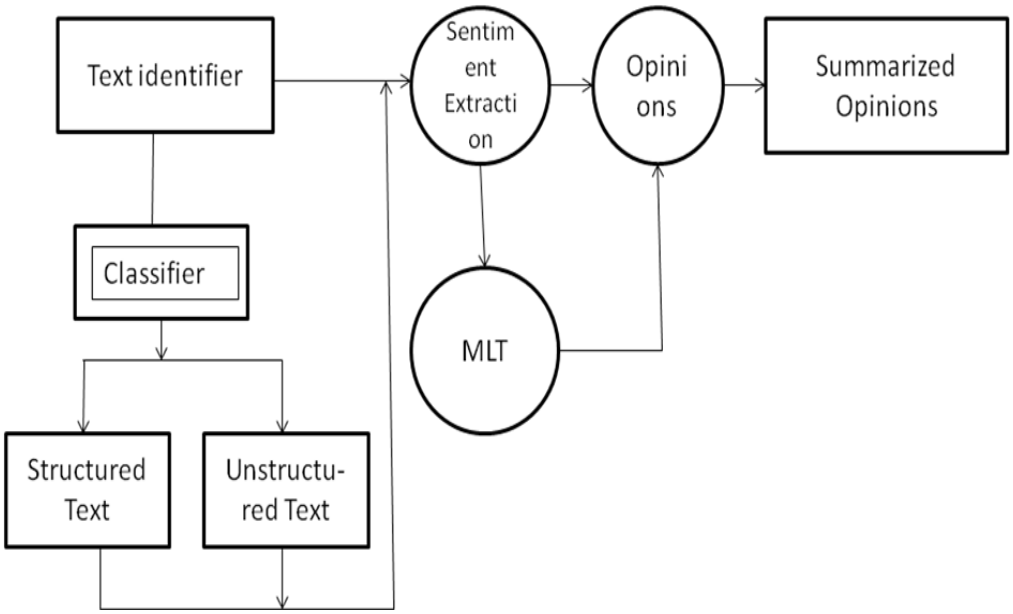


Fig 3.2.2: Given text is identified as structured or unstructured and the opinions are extracted

Text Classification:

In this module describe the classification of the entered text. It identifies whether the text belongs to structured or unstructured. It uses the classifier method for identification process.

Sentiment Extraction:

In this module, it describes the extraction of sentiments from various user's of the system. The sentiments are gathered from different user's based on the unstructured text. SE plays the major role in the proposed system. It is used for the retrival of sentiments from the previous viewers.

Opinion Detection:

This module describes the collection of opinions from the various user's and the polarity detection for the different opinions. The opinions are obtained from the SE.The polarity is given based on "positive" , "negative" and "neutral".

Opinion summarization:

This is the final module which describes summarized opinions from the sentiments. It classifies the opinions based on the MLT mechanism using the support vector machines and PMI-IR methods. The final result is viewed by the user.

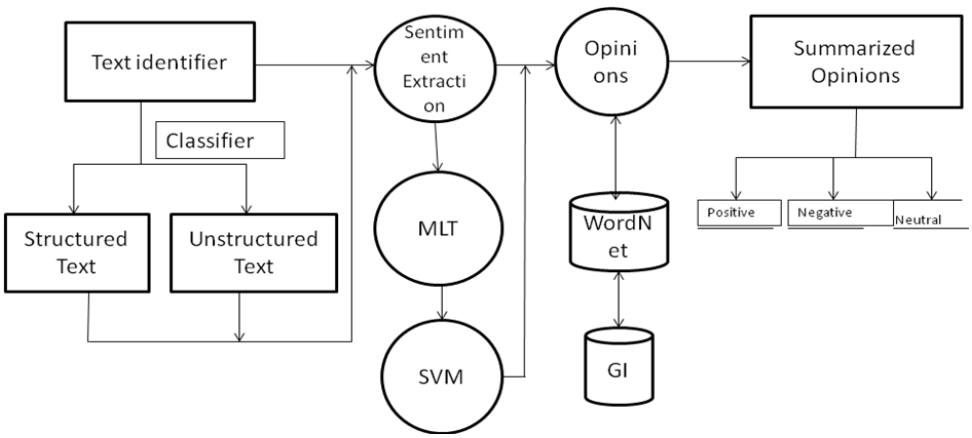


Fig 3.2.3 : Includes sentiment classification and summarizes positive, negative and neutral

NATURAL LANGUAGE PROCESSING (NLP)

It is a field of computer science where Artificial Intelligence is concerned with the interaction between computer and human languages.NLP is related to the human machine interaction.

The main *challenges* are

- Natural language understanding.
- Deriving the meaning from human (or) Natural Language input.

Applications:

Spelling and grammar checking,information retrieval,document classification such as filtering and routing,clustering,information extraction,question

answering,summarization,text segmentation,text segmentation,e-mail understanding and machine translation.

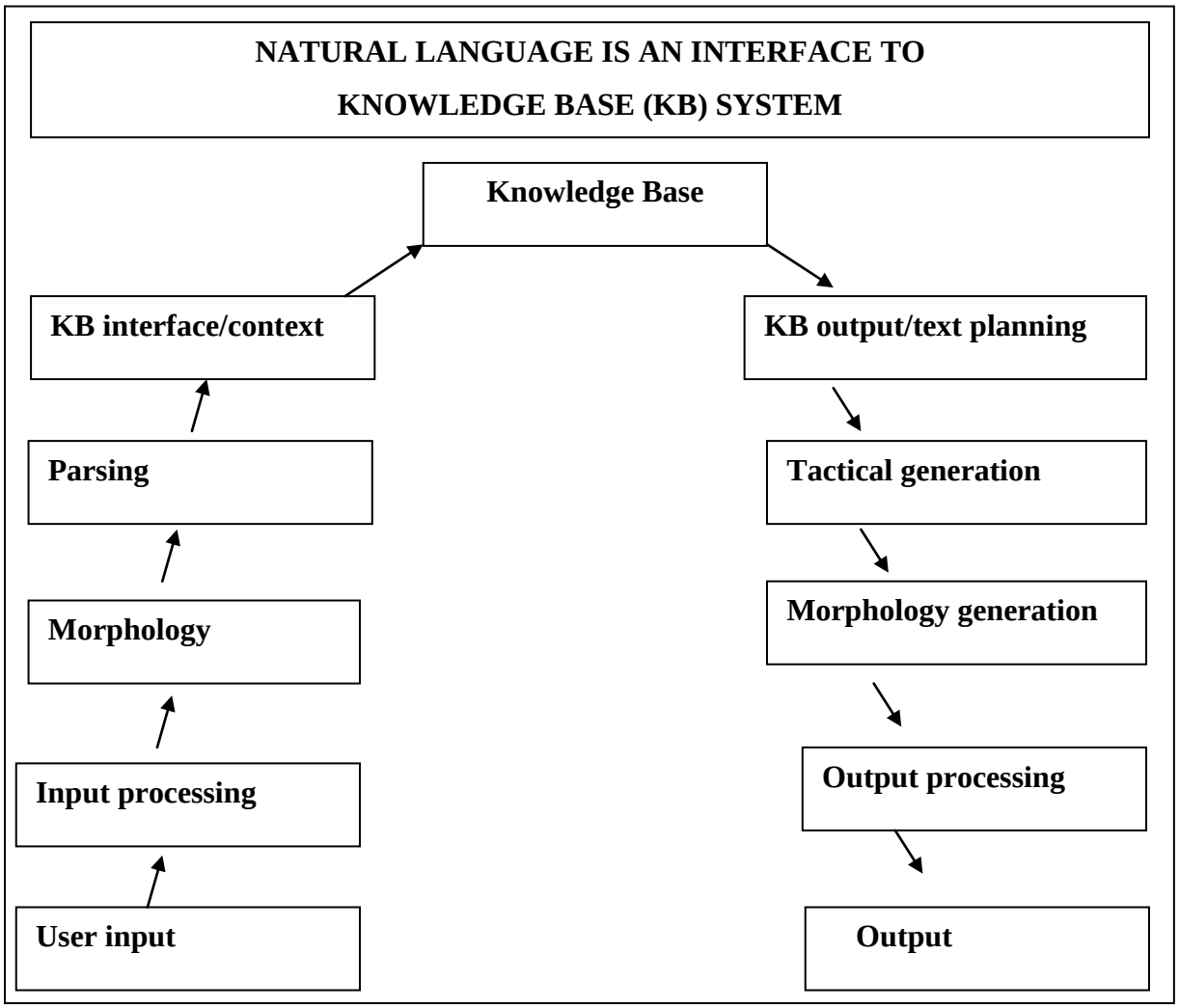


Fig 3.2.4: NL interface to knowledge base.

Evaluation Metrics of Information Retrieval

$$\text{Precision} = \frac{\text{No: of relevant documents retrieved}}{\text{Total no: of documents retrieved}}$$

$$\text{Recall} = \frac{\text{No: of relevant documents retrieved}}{\text{Total no: of relevant documents}}$$

Precision gives the relevant retrieved documents and recall gives the documents that are retrieved. Eg: Search engine returns 50 pages in that only 30 pages are relevant and it also failed to return 40 additional relevant pages.

Precision=30/50	**Recall=30/70**

CHAPTER 4

OBJECTIVE

- To propose many research strategies in NLP and is in commercial deployment in various field.
- To provide the accuracy and reliability of the content.
- To propose a unique strategy for mining high dimensional text data using sentimental analysis technique.

The main focus of the research is on minimizing the outline time as well as to keep up a viable record of the final plan parameters. The design of machine elements involves proceeding through several specific steps based on design criterion, evaluating results and returning to an earlier stage in the design process if the design constraints are not satisfied. There are various iterations need to be performed before finalizing a design strategies for a particular machine element, and a record of the various iterations. Thus, it is an iterative process and the decisions taken regarding the effectiveness of the design is dependent on the designer’s perspective. The objective of this research is to is to give the creator an algorithm that guides the configuration procedure by giving a method for organizing every one of the assets relating to the outline of a machine component without smothering the planner's inventiveness. Using mining techniques for text data using sentimental analysis to provide the accuracy and the reliability of the content.

CHAPTER 5

METHODOLOGY

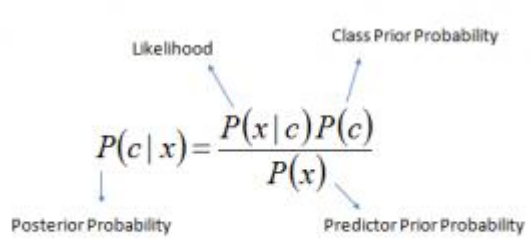
5.1. ALGORITHM

5.1.1 Naive Bayes Classifier:

We have hundreds of thousands of data points and quite a few variables in our training data set. We would use 'Naive Bayes', which can be extremely fast relative to other classification algorithms. It works on Bayes theorem of probability to predict the class of unknown data set.

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naive'. Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:



$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Above,

- P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes).
- P(c) is the prior probability of class.
- P(x|c) is the likelihood which is the probability of predictor given class.
- P(x) is the prior probability of predictor.

Below I have a training data set of weather and corresponding target variable 'Play' (suggesting possibilities of playing). Now, we need to classify whether players will play or not based on weather condition. Let's follow the below steps to perform it.

Step 1: Convert the data set into a frequency table

Step 2: Create Likelihood table by finding the probabilities like Overcast probability= 0.29 and probability of playing is 0.64.

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

Step 3: Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

Problem: Players will play if weather is sunny. Is this statement is correct?

We can solve it using above discussed method of posterior probability.

P(Yes | Sunny) = P(Sunny | Yes) * P(Yes) / P (Sunny)

Here, we have P (Sunny |Yes) = 3/9 = 0.33, P(Sunny) = 5/14 = 0.36, P(Yes)= 9/14 = 0.64

Now, P (Yes | Sunny) = 0.33 * 0.64 / 0.36 = 0.60, which has higher probability.

Naive Bayes uses a similar method to predict the probability of different class based on various attributes. This algorithm is mostly used in text classification and with problems having multiple classes.

Advantages:

- It is easy and fast to predict class of test data set. It also perform well in multi class prediction
- When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
- It performs well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

Disadvantages:

- On the other side naive Bayes is also known as a bad estimator, so the probability outputs from predict_proba are not to be taken too seriously.
- Another limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

Applications:

- Real time Prediction: Naive Bayes is an eager learning classifier and it is sure fast. Thus, it could be used for making predictions in real time.
- Multi class Prediction: This algorithm is also well known for multi class prediction feature. Here we can predict the probability of multiple classes of target variable.
- Text classification/ Spam Filtering/ Sentiment Analysis: Naive Bayes classifiers mostly used in text classification (due to better result in multi class problems and independence rule) have higher success rate as compared to other algorithms. As a result, it is widely used in Spam filtering (identify spam e-mail) and Sentiment Analysis (in social media analysis, to identify positive and negative customer sentiments)

5.1.2 PMI-IR :

Mutual Information (MI) measures the mutual dependence of two random variables.

- The higher it is, the more dependent the two random variables are with each other.
- Its value is always positive.

$$MI(X;Y) = \sum_{x,y} p(x,y) \log\left(\frac{p(x,y)}{p(x)\,p(y)}\right)$$

For example, say a discrete random variable *X* represents visibility at a certain moment in time and random variable *Y* represents wind speed at that moment.

The mutual information between *X* and *Y:*

$$MI(X;Y) = p(X = good, Y = high)\log(\frac{p(X = good, Y = high)}{p(X = good)\,p(Y = high)}) +$$
$$p(X = bad, Y = high)\log(\frac{p(X = bad, Y = high)}{p(X = bad)\,p(Y = high)}) +$$
$$p(X = good, Y = low)\log(\frac{p(X = good, Y = low)}{p(X = good)\,p(Y = low)}) +$$
$$p(X = bad, Y = low)\log(\frac{p(X = bad, Y = low)}{p(X = bad)\,p(Y = low)})$$

Pointwise Mutual Information (PMI) measures the mutual dependence of between two instances or realizations of random variables.

- Positive => high correlated
- Zeros => no information (independence)
- Negative => opposite correlated
- The equation of PMI:

$$PMI(X = x, Y = y) = \log(\frac{p(X = x, Y = y)}{p(X = x)\,p(Y = y)})$$

The Pointwise Mutual Information (PMI) between two words, *word*1 and *word*2, is defined as follows (Church & Hanks, 1989):

PMI(word1, word2) = log2 (p(word1 & word2) / p(word1) p(word2))

Here, p(word1 & word2) is the probability that *word*1 and *word*2 co occur. If the words are statistically independent, then the probability that they co-occur is given by the product p(word1) p(word2). The ratio between p(word1 & word2) and p(word1) p(word2) is thus a measure of the degree of statistical dependence between the words.

The log of this ratio is the amount of information that we acquire about the presence of one of the words when we observe the other.

word 1	**word 2**	**count word 1**	**count word 2**	**count of co-occurrences**	**PMI**
puerto	rico	1938	1311	1159	10.0349081703
hong	kong	2438	2694	2205	9.72831972408
los	angeles	3501	2808	2791	9.56067615065
carbon	dioxide	4265	1353	1032	9.09852946116
prize	laureate	5131	1676	1210	8.85870710982
san	francisco	5237	2477	1779	8.83305176711
nobel	prize	4098	5131	2498	8.68948811416
ice	hockey	5607	3002	1933	8.6555759741
star	trek	8264	1594	1489	8.63974676575
car	driver	5578	2749	1384	8.41470768304
it	the	283891	3293296	3347	-1.72037278119

are	of	234458	1761436	1019	-2.09254205335
this	the	199882	3293296	1211	-2.38612756961
is	of	565679	1761436	1562	-2.54614706831
and	of	1375396	1761436	2949	-2.79911817902
a	and	984442	1375396	1457	-2.92239510038
in	and	1187652	1375396	1537	-3.05660070757
to	and	1025659	1375396	1286	-3.08825363041
to	in	1025659	1187652	1066	-3.12911348956
of	and	1761436	1375396	1190	-3.70663100173

Great collocation sets have high PMI in light of the fact that the likelihood of co-event is just somewhat lower than the probabilities of event of every word. On the other hand, a couple of words whose probabilities of event are impressively higher than their likelihood of co-event gets a little PMI score.

5.1.3 NGram Extractor :

A N-Gram extractor and is a next sequence of 'n' items from a given sequence of text or speech. The items can be letters, words or base pairs.N-Grams are collected from a text or speech Corpus. N-Grams of text are used in text mining and NLP tasks.A set of co-occuring words and when computing we move one word forward.

Eg. The cow jumps over the moon.

If N=2 (bigrams) then n-grams would be (i) the cow (ii) cow jumps (iii) jumps over (iv)over the (v) the moon. So, we have 5 n=grams

The->cow to cow→jumps to jumps→ over to over etc… Move one word forward.

If n=3(trigrams) (i) the cow jumps (ii) cow jumps over (iii) jumps over the (iv) over the moon. We have 4 n-grams.

If n=1 then unigrams, n=2 then bi-grams, n=3 then tri-grams, n>3 then four grams, five grams etc..

How many n-grams in a sentence?

A = Number of words in a given sentence B.N-grams for a sentence would be

$$N\ grams_B = A-(N-1)$$

Applications:

Predicting next item in a sequence in the form (n-1) order. Widely used in probability, computational linguistics, searching in text documents and data compression.

Advantages:

Simplicity: With larger ‘n’

Scalability: Store more context and scale up efficiently.

N-Gram is a java based library containing two types of N-grams based applications. Its major focus is to provide robust and state of art language recognition or language guessing.

5.1.4 WORDNET:

WordNet is a lexical database for the English language. It groups English words into sets of synonyms called synsets, provides short definitions and usage examples, and records a number of relations among these synonym sets or their members. The main usage of WordNet is to determine the similarity between words.

WordNet externally looks like a thesaurus, in that it bunches words together in light of their implications. In any case, there are some critical refinements. In the first place, WordNet interlinks word shapes—series of letters—as well as particular senses of words. Accordingly, words that are found in close closeness to each other in the system are semantically disambiguated. Second, WordNet names the semantic relations among words, while the grouping of words in a thesaurus does not take after any unequivocal example other than significance closeness.

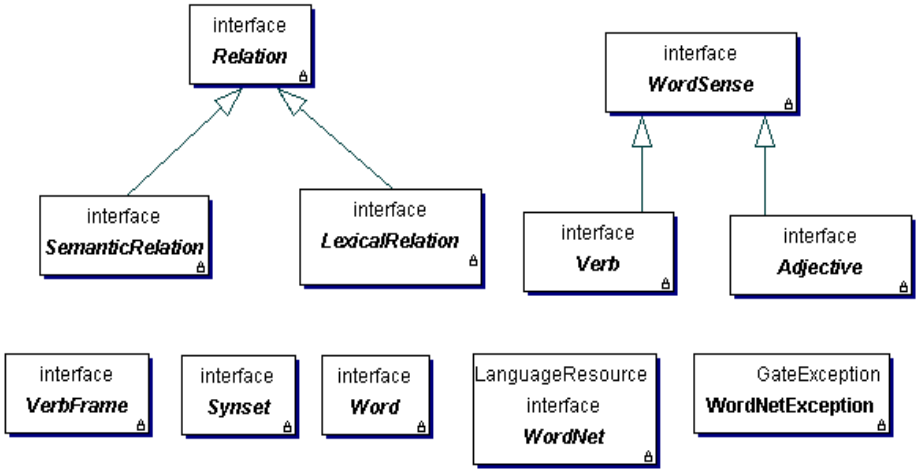


Fig 5.1.4.1: Hierarchy of interface between a wordsense and relation

WordNet has been used for a number of different purposes in information systems, including word-sense disambiguation, information retrieval, automatic text classification, automatic text summarization, machine translation and even automatic crossword puzzle generation.

The main relation among words in WordNet is synonymy, as between the words shut and close or car and automobile. Synonyms--words that denote the same concept and are interchangeable in many contexts--are grouped into unordered sets (synsets). WordNet states that the category furniture includes bed, which in turn includes bunkbed; conversely, concepts like bed and bunkbed make up the category furniture. All noun hierarchies ultimately go up the root node {entity}. Verb synsets are arranged into hierarchies as well; verbs towards the bottom of the trees (troponyms) express increasingly specific manners characterizing an event, as in {communicate}-{talk}-{whisper}. The specific manner expressed depends on the semantic field; volume (as in the example above) is just one dimension along which verbs can be elaborated.

- **S;** (noun) **apple** (fruit with red or yellow or green skin and sweet to tart crisp whitish flesh)
  - *Hyponyms*
  - *Direct hypernyms*
  - *Indirect hypernyms*
    - edible_fruit
      - fruit
        - reproductive_structure
          - plant_organ
            - plant_part
              - natural_object
                - whole
                  - object
                    - physical_entity
                      - entity

Fig 5.1.4.2: Classification for a fruit apple using wordnet.

- S; (noun) **orange** (round yellow to orange fruit of any of several citrus trees)
- S; (noun) **orange**, orangeness (orange color or pigment; any of a range of colors between red and yellow)
- S; (noun) **orange**, orange tree (any citrus tree bearing oranges)
- **S;** (noun) **orange** (any pigment producing the orange color)
  - *Direct hypernyms*
  - *Indirect hypernyms*
    - pigment
      - coloring_material
        - material
          - substance
            - matter
              - physical_entity
                - entity
            - part
              - relation
                - abstraction
                  - entity

Fig 5.1.4.3: Classification for a fruit orange using wordnet.

Adjectives are organized in terms of antonyms. Pairs of "direct" antonyms like wet-dry and young-old reflect the strong semantic contract of their members. There are only few adverbs in WordNet (hardly, mostly, really, etc.) as the majority of English adverbs are straightforwardly derived from adjectives via morphological affixation (surprisingly, strangely, etc.)

A well known illustration is to decide the feeling of pen in the accompanying entry (Bar-Hillel 1960):

Little John was searching for his toy box. At last he discovered it. The crate was in the pen. John was happy. WordNet records five faculties for the word pen:

(i) pen — a written work actualize with a point from which ink streams.

(ii) pen — a fenced in area for limiting animals.

(iii)playpen, pen — a compact fenced in area in which infants might be left to play.

(iv)prison, pen — a remedial foundation for those sentenced significant wrongdoings.

(v) pen — female swan.

5.1.5 Word Sense Disambiguation(WSD):

It consists on selecting the appropriate meaning of a word given the context in which it occurs. In this WSD method, the senses are represented as signatures built from the repository of concepts of WordNet. The disambiguation process starts from a clustering distribution of all possible senses of the ambiguous words

In natural language processing, **word sense disambiguation** (WSD) is the problem of determining which "sense" (meaning) of a word is activated by the use of the word in a particular context, a process which appears to be largely unconscious in people. WSD is a natural classification problem: Given a word and its possible senses, as defined by a dictionary, classify an occurrence of the word in context into one or more of its sense classes.

The features of the context (such as neighboring words) provide the evidence for classification. WSD is required for lexical decision in MT for words that have distinctive interpretations for various senses. The vocabulary is frequently pre-disambiguated for a given area, or hand-created principles are formulated, or WSD is collapsed into a factual interpretation model, where words are deciphered inside

expressions which in this manner give connection. Ambiguity must be determined in a few questions. For example, given the question "dejection" ought to the framework return reports about sickness, climate frameworks, or financial aspects? Flow IR frameworks, (for example, web search tools), like MT, don't utilize a WSD module; they depend on the client writing enough setting in the inquiry to just recover reports important to the planned sense.

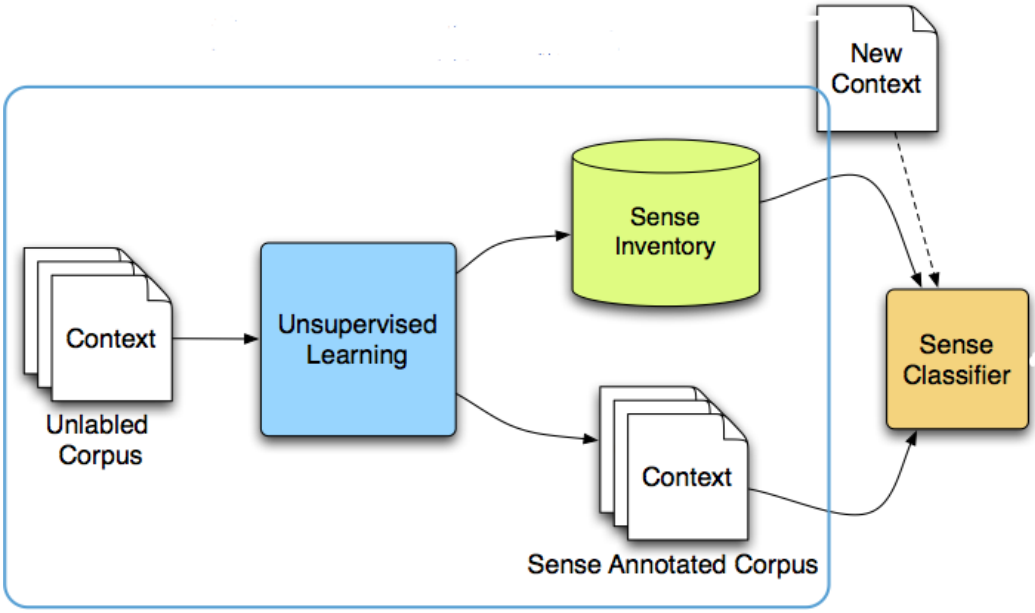


Fig 5.1.5.1 : Finding the relevant context

There are four conventional approaches to WSD:

- **Dictionary- and knowledge-based methods:** These rely primarily on dictionaries, thesauri, and lexical knowledge bases, without using any corpus evidence.
- **Supervised methods:** These make use of sense-annotated corpora to train from.
- **Semi-supervised or minimally-supervised methods:** These make use of a secondary source of knowledge such as a small annotated corpus as seed data in a bootstrapping process, or a word-aligned bilingual corpus.

- **Unsupervised methods:** These eschew (almost) completely external information and work directly from raw unannotated corpora. These methods are also known under the name of *word sense discrimination*.

The evaluation of WSD systems requires a test corpus hand-annotated with the target or correct senses, and assumes that such a corpus can be constructed. Two main performance measures are used:

- **Precision:** the fraction of system assignments made that are correct
- **Recall:** the fraction of total word instances correctly assigned by a system

If a system makes an assignment for every word, then precision and recall are the same, and can be called **accuracy**. This model has been extended to take into account systems that return a set of senses with weights for each occurrence.

There are two kinds of test corpora:

- **Lexical sample:** the occurrences of a small sample of target words need to be disambiguated, and
- **All-words:** all the words in a piece of running text need to be disambiguated.

The latter is deemed a more realistic form of evaluation, but the corpus is more expensive to produce because human annotators have to read the definitions for each word in the sequence every time they need to make a tagging judgement, rather than once for a block of instances for the same target word.

5.1.6 SYSTEM REQUIREMENTS

5.1.6.1 HARDWARE REQUIREMENT:

Processor	:	Intel Pentium IV
Clock speed	:	1.8 GHz
RAM	:	256 MB
HDD	:	80 GB
FDD	:	1.44 MB
CD Drive	:	52x Reader
Pointing device	:	Scroll Mouse
Keyboard	:	101 Standard Key-board
Peripherals	:	Printer

5.1.6.2 SOFTWARE REQUIREMENT:

Language	:	Java
IDE	:	Net beans
Database	:	MYSQL
Operating System	:	Windows XP.

CHAPTER 6

CONCLUSION

This research introduces the new innovative method for polarity detection and opinion summarization. It improves the searching process and also provides the user for easy access of text. The current state of the art techniques in SE typically employ classification algorithms from Data Mining and Machine Learning in order to anchor sentiments of unstructured text. Typical approaches are either lexicon based (and hence domain specific with little scalability) or learning based (and hence domain independent but performance is contingent upon availability of quality training data).

We obtained an overall classification accuracy of 88.8% on the test set of 25,000 reviews of the product. The running time of our algorithm is O (n+V log V) for training and O(n) for testing. When 'n' is the number of words in the document and V is the size of the reduced vocabulary. It is much faster than other machine learning algorithms like Support Vector Machine (SVM) which take a log time to cover the optimal set of weights. The accuracy is comparable to that of the current state of the art algorithms used for sentiment classification on reviews. It achieved a better or similar accuracy when compared to more complicated models like SVM, auto encoder etc..

Feature added	**Accuracy on test set**
SVM	73.77 %
Bi-grams & Tri-grams	85.20 %
Handling negation	82.80 %
Feature selection	88.80 %

6.1 BENEFITS OF THE RESEARCH:

In this research, a new method for Sentiment Extraction of Unstructured Text was introduced to determine the polarity and summarize the text efficiently. It's most important novelty is the use of WordNet and Word Sense Disambiguation tools together with standard external resources for determining the polarity of the opinions. These resources allow the method to be extended to other languages and be independent of the knowledge domain.

CHAPTER 7

REFERENCES

1.Automatic Sentiment Analysis in On-line Text Erik Boiy; Pieter Hens; Koen Deschacht; Marie-Francine Moens Katholieke Universities Leuven, Tiensestraat 41 B-3000 Leuven, Belgium

2. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews Peter D.Turney Institute for Information Technology National Research Council of Canada Ottawa, Ontario, Canada.

3. Opinion Polarity Detection Using Word Sense Disambiguation to Determine the Polarity of Opinions Tamara Martin-Wanton, Aurora Pons-Porrata Center for Pattern Recognition and Data Mining, Universidad de Orient, Patricio Lumumba s/n, Santiago de Cuba.

4. OSGOOD, C. E.; SUCI, G. J; TANNENBAUM, P. H. The Measurement of Meaning. University of Illinois Press, 1971 [1957].

5. BIBER, D; FINEGAN, E. Styles of stance in English: Lexical and grammatical marketing of evidentiality and affect. Text 9, 1989, pp. 93-124.

6. WALLACE, A. F. C.; CARSON, M. T., Sharing and diversity in emotion terminology. Ethos 1 (1), 1973, pp. 1-29.

7. HATZIVASSILOGLOU, V.; WIEBE, J., Effects of adjective orientation and gradability on sentence subjectivity, Proceedings of the 18th International Conference on Computational Linguistics, ACL, New Brunswick, NJ, 2000.

8. FELLBAUM, C. (ed.), Wordnet: An electronic lexical database, Language, Speech, and Communication Series, MIT Press, Cambridge, 1998.

9. KAMPS, J.; MARX, M.; MOKKEN, R. J.; DE RIJKE, M., Using WordNet to measure semantic orientation of adjectives. LREC 2004, volume IV, pp. 1115—1118.

10. MULDER, M.; NIJHOLT, A.; DEN UYL, M.; TERPSTRA, P., A lexical grammatical implementation of affect, Proceedings of TSD-04, the 7th International Conference Text, Speech and Dialogue, Lecture Notes in Computer Science, vol. 3206, SpringerVerlag, Brno, CZ, 2004, pp. 171–178.

11. DAVE, K.; LAWRENCE, S.; PENNOCK, D. M. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In Proceedings of WWW-03, 12th International Conference on the World Wide Web, ACM Press, Budapest, HU, 2003, pp. 519–528.

12. PEDERSEN, T. A decision tree of bigrams is an accurate predictor of word sense. In Proceedings of the Second Annual Meeting of the North American Chapter of the Association for Computational Linguistics, 2001, pp. 79–86.

13. Opinion mining of customer feedback data on theweb. Dongjoo LEE1, OK-RAN JEONG2, SANG-GOO LEE

14. Thumbs up? Sentiment Classification using Machine Learning techniques, BO PANG AND LILLIAN LEE, SHIVAKUMAR VAITHYANATHAN

15. Improving performance of Naive Bayes c lassification, Yirong Shen and Jing Jiang

16. Sentiment Extraction in unstructured text using Tabu-search enhanced Markov blanket, XUE BAI, REMA PADMAN, EDOARDO AIROLDI

17. Sentiment Analysis: A new approach for effective use of linguis- tic knowledge and exploiting similarities in a set of documents to be classified, ALEKH AGARWAL AND PUSHPAK BATTACHARYA.

18. Sentiment Analysis: Capturing favourability using Natural Lan- guage Processing, TETSUYA NASUKAWA, JEONGHEE YI

19. Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis, THERESA WILSON, JANYCE WIEBE, PAUL HOFFMANN

20. Mining and Summarizing Customer Reviews, BING LIU, MIN- QUING HU

21. Opinion Mining and Sentiment Analysis, BO PANG AND LILLIAN LEE

22. Sentiment Analyzer: Extracting sentiments about a given topic using NLP techniques; JEONGHEE YI, TETSUYA NASUKAWA, RAZVAN BUNESCU, WAYNE NIBLACK.

CHAPTER 8

APPENDIX 1

```
package sentimentanalysis;

import sentimentanalysis.view.SentimentAnalysisView;

import org.jdesktop.application.Application;

import org.jdesktop.application.SingleFrameApplication;

import sentimentanalysis.endpoint.Server;

public class SentimentAnalysisApp extends SingleFrameApplication

{

private  SentimentAnalysisView   view;

private  Server  endpoint;

private static  SentimentAnalysisApp    instance;

Override protected void startup()

{

assert(instance == null);

view        = new SentimentAnalysisView(this);

endpoint    = new Server();

show(view);

endpoint.start();

instance    = this;

}

public static SentimentAnalysisApp getApplication()

{

return instance;

}
```

```
public static void setStatus (String message)
{
if (getView() != null)
getView().setStatus(message);
}
public static SentimentAnalysisView getView ()
{
if (instance != null) return instance.view;
return null;
}
public static void main(String[] args)
{
launch(SentimentAnalysisApp.class, args);
}
}
```

```
package sentimentanalysis.algorithmtestingsuite;
public class Main implements Constants
{
public static void main(String[] args) throws Exception
{
TesterSuite ts =  new TesterSuite();
ts.doTesting(USE_BAYES, "nike");
System.out.println(TesterSuite.TP);
System.out.println(TesterSuite.TN);
```

```
System.out.println(TesterSuite.FP);
System.out.println(TesterSuite.FN);
System.out.println("precision:"+TesterSuite.getPrecision()+"
recall:"+TesterSuite.getRecall());
}
}
package sentimentanalysis.preprocessing;
import java.util.ArrayList;
import java.util.List;
public class NGramExtractor
{
private static List<String> ngrams(int n, String str)
{
List<String> ngrams = new ArrayList<String>();
String[] words = str.split(" ");
for (int i = 0; i < words.length - n + 1; i++)
ngrams.add(concat(words, i, i+n));
return ngrams;
}
private static String concat(String[] words, int start, int end)
{
StringBuilder sb = new StringBuilder();
for (int i = start; i < end; i++)
sb.append((i > start ? " " : "") + words[i]);
return sb.toString();
```

```
}
public static ArrayList<String> getNGrams(String s,int minNgram, int maxNgram)
{
 ArrayList<String> ngrams = new ArrayList<String>();
 for (int n = minNgram; n <= maxNgram; n++)
{
 for (String ngram : ngrams(n, s))
 {
ngrams.add(ngram);
}
}
return ngrams;
}
public static void main(String[] args)
{
ArrayList<String> test = getNGrams("hello world, I am so happy today!",2,2);
for (String ngram : test )
System.out.println(ngram);
}
}
package sentimentanalysis.algorithmtestingsuite;
public interface Constants
{
final static int  NORMALIZE_TEXT = 10;
final static int  REMOVE_DUPLICATES = 11;
```

```
final static int  LOWER_CASE = 12;

final static int  USE_DISTANCE = 13;

final static int  USE_SMILEYS = 14;

final static int  USE_NEGATIONS = 15;

final static int  USE_MODIFIERS = 16;

final static int USE_BAYES = 17;

}


package sentimentanalysis.preprocessing;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileReader;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.util.Enumeration;

import java.util.Iterator;

import java.util.LinkedList;

import java.util.logging.Level;

import java.util.logging.Logger;

import sentimentanalysis.common.LocalTweet;

import sentimentanalysis.database.Operator;

import sentimentanalysis.twitterapi.TweetOperator;

import twitter4j.TwitterException;

import weka.classifiers.Classifier;
```

```
import weka.classifiers.Evaluation;
import weka.classifiers.bayes.NaiveBayes;
import weka.classifiers.bayes.NaiveBayesMultinomial;
import weka.core.*;
import weka.core.converters.ArffSaver;
import weka.filters.Filter;
import weka.filters.unsupervised.attribute.StringToWordVector;
public class NBayesCalculator
{
private static void constructModel()
{
Try
{
BufferedReader reader = new BufferedReader(new FileReader("./data/train2data.arff"));
Instances data = new Instances(reader);
reader.close();
setting class attribute
data.setClassIndex(data.numAttributes() - 1);
NaiveBayes classifier = new NaiveBayes();
data.setClassIndex(0);
classifier.buildClassifier(data);
serialize(classifier);
Evaluation eval = new Evaluation(data);
eval.evaluateModel(classifier, data);
```

```
System.out.println(eval.toSummaryString("\nResults\n======\n",
false));System.out.println("\n\nClassifier model:\n\n" + classifier);
}
catch (Exception ex)
{
Logger.getLogger(NBayesCalculator.class.getName()).log(Level.SEVERE, null, ex);
}
}
private static void serialize(Classifier object)
{
Try
{
weka.core.SerializationHelper.write("nBayes.model", object);
}
catch (Exception ex)
{
Logger.getLogger(NBayesCalculator.class.getName()).log(Level.SEVERE, null, ex);
}

}
public static double classifyNewTweet(String tw)
{
double score = 0;
try
{
```

```
Classifier cls = (Classifier) weka.core.SerializationHelper.read("nBayes.model");
FastVector atts;
atts = new FastVector();
atts.addElement(new Attribute("content", (FastVector) null));
FastVector fvClassVal = new FastVector(4);
fvClassVal.addElement("");
fvClassVal.addElement("neutral");
fvClassVal.addElement("negative");
fvClassVal.addElement("positive");
Attribute ClassAttribute = new Attribute("Class", fvClassVal);
atts.addElement(ClassAttribute);
Instances instdata = new Instances("testData", atts, 0);
Instance iInst = new Instance(2);
iInst.setValue((Attribute) atts.elementAt(0), tw);
iInst.setValue((Attribute) atts.elementAt(1), '?');
instdata.add(iInst);
StringToWordVector filter = new StringToWordVector();
instdata.setClassIndex(instdata.numAttributes() - 1);
filter.setInputFormat(instdata);
Instances newdata = Filter.useFilter(instdata, filter);
System.out.println(newdata.toString());
newdata.setClassIndex(0);
double clsLabel = cls.classifyInstance(instdata.instance(0));
int label = (int) clsLabel;
switch (label)
```

```
{
case 3:score = -0.5;break;
case 2:score = +0.5;break;
case 1:score = +0.0;break;
}
eTest.evaluateModel(cls, isTrainingSet);
}
catch (Exception ex)
{
Logger.getLogger(NBayesCalculator.class.getName()).log(Level.SEVERE, null, ex);
}
return score;
}
public static void evaluateModel(LinkedList<String> tw,LinkedList<String> classes)
{
try
{
Classifier cls = (Classifier) weka.core.SerializationHelper.read("nBayes.model");
BufferedReader reader = new BufferedReader(new FileReader("./data/train2data.arff"));
Instances traindata = new Instances(reader);
reader.close();
FastVector atts;
atts = new FastVector();
atts.addElement(new Attribute("content", (FastVector) null));
FastVector fvClassVal = new FastVector(3);
```

```
fvClassVal.addElement("");
fvClassVal.addElement("neutral");
fvClassVal.addElement("negative");
fvClassVal.addElement("positive");
Attribute ClassAttribute = new Attribute("Class", fvClassVal);
atts.addElement(ClassAttribute);
Instances instdata = new Instances("testData", atts, 0);
int i=0;
for(String tweet:tw)
{
Instance iInst = new Instance(2);
iInst.setValue((Attribute) atts.elementAt(0), tweet);
if(!classes.isEmpty())
{
iInst.setValue((Attribute) atts.elementAt(1), classes.get(i));
}
instdata.add(iInst);
i++;
}
StringToWordVector filter = new StringToWordVector();
instdata.setClassIndex(instdata.numAttributes() - 1);
filter.setInputFormat(instdata);
Instances newdata = Filter.useFilter(instdata, filter);
ArffSaver saver = new ArffSaver();
saver.setInstances(newdata);
```

```
saver.setFile(new File("../data/testdata2.arff"));
saver.writeBatch();
System.out.println(newdata.toString());
traindata.setClassIndex(0);
Evaluation eval = new Evaluation(traindata);
eval.evaluateModel(cls, newdata);
String[] options = new String[6];
options[0] = "-T";
options[1] = "../data/testdata2.arff";
options[2] = "-l";
options[3] = "nBayes.model";
options[4] = "-p";
options[5] = "0";
System.out.println(Evaluation.evaluateModel(cls, options));
System.out.println(eval.toSummaryString("\nResults\n======\n", false));
System.out.println("\n\nClassifier model:\n\n" + cls);
System.out.println(instdata.numClasses());
Enumeration br = newdata.enumerateInstances();
int ind=0;
while(br.hasMoreElements())
{
double clsLabel = cls.classifyInstance((Instance)br.nextElement());
String cl ="";
switch((int)clsLabel)
{
```

```
case 0: cl ="";break;

case 1: cl ="neutral"; break;

case 2:cl= "negative"; break;

case 3:cl = "positive"; break;

}

System.out.println("cont: "+tw.get(ind)+" pred_class: "+cl + " actual:

"+classes.get(ind));

ind++;

}

}

catch (Exception ex)

{

 Logger.getLogger(NBayesCalculator.class.getName ()).log(Level.SEVERE, null, ex);

}

}

public static void main(String[] args)

{

Operator db = Operator.getInstance();

LinkedList<String> tw = db.getTweetsForTesting();

evaluateModel(tw,db);

constructModel();

Operator db = Operator.getInstance();

LinkedList<String> tw = db.getTweetsForTesting();

evaluateModel(tw,null);

classifyNewTweet(null);
```

```
}
}
package sentimentanalysis.preprocessing;

import java.util.HashMap;
import java.util.StringTokenizer;
public class TextNormalizer
{
static String tweet = "";
public TextNormalizer(String tweet)
{
this.tweet = tweet;
}
public String getTweet()
{
return tweet;
}
public static String getTweetWithoutUrlsAnnotations(String Tweet)
{
StringTokenizer tokens = new StringTokenizer(Tweet, " ");
String newTweet = "";
while (tokens.hasMoreTokens())
{
String temp = tokens.nextToken();
if (!temp.contains("@") && !temp.contains("http"))
```

```
{
newTweet += temp + " ";
}
}
tweet = "";
tweet = newTweet;
return tweet;
}
public static double detectSmiley(String tweet, HashMap<String, Double> smileys)
{
double score = 0;
StringTokenizer toks = new StringTokenizer(tweet, " ");
while (toks.hasMoreTokens())
{
String token = toks.nextToken();
if (smileys.containsKey(token))
{
score = smileys.get(token);
return score;
}
}
return score;
}
public static String removeDuplicates(String s) {
StringBuilder noDupes = new StringBuilder();
```

```
int pos = 0 ;
int len = s.length();
int curLen = 0;
while (pos < len) {
if (pos == 0 || s.charAt(pos) != s.charAt(pos - 1)) {
curLen = 1;
}
else
{
++curLen;
}
if (curLen < 3)
{
noDupes.append(s.charAt(pos));
}
++pos;
}
return noDupes.toString();
}
public static String toLowerCase(String tw)
{
return tw.toLowerCase();
}
public static void main(String[] args)
{
```

```
String testStrings[] = {"GoodBoy", "greeeeaaat", "oooooo","yeeeaaah", "abcde",
"abcdeeee"};
for (String s : testStrings) {
System.out.println(s + "  ==>  " + removeDuplicates(s));
}
}
}
package sentimentanalysis.sentiment;
import java.io.IOException;
import java.util.HashMap;
import java.util.LinkedList;
import sentimentanalysis.common.LocalTweet;
import sentimentanalysis.database.Operator;
import sentimentanalysis.search.Search;
import twitter4j.Tweet;
import java.lang.Object;
public class Analysis
{
private Search          parent      = null;
private Tokenizer       tokenizer   = null;
private Lexicon         lexicon     = null;
private double[]        gaussian    = new double[6];
public Analysis (Search parent) throws Exception
{
this.parent   = parent;
```

```
this.tokenizer= new Tokenizer();
this.lexicon = new Lexicon();
double sentiment = 0.0;
double score = 0.0;
this.gaussian = CalculateDistance();
LinkedList<LocalTweet>  tweets  = parent.getTweets();
System.out.println("Analyzing sentiment for " + tweets.size() + " tweets.");
for (LocalTweet tweet : tweets) {
if (tweet.isFromDB())
{
sentiment += tweet.getSentimentScore(parent.getQuery().getRequest());
}
else {
score  = getSentiment(tweet);
sentiment += score;
tweet.addSentiment(parent.getQuery().getRequest(), score);
}
}
System.out.println("Total sentiment score: " + sentiment);
}
private double getSentiment (LocalTweet tweet) throws IOException {
LinkedList<String> tokens = tokenizer.getTokens(tweet);
Double score     = 0.0;
Double negation = 1.0;
Double modifier   = 1.0;
```

```
int range      = 0;
Double gauss    = 1.0;
int i       = 0;
for (String token : tokens) {
gauss = getDistance(parent,token,tokens,i,this.gaussian);
score += (negation) * (modifier) * (gauss) * lexicon.get(token);
if (isNegation(token)) {
negation = -1.0;
range = 2;
}
else if (range > 1)
range--;
else
negation = 1.0;
modifier = getModifierValue(token, negation);
i++;
}
return score;
}
private static boolean isNegation( String token)
{
HashMap<String, Double> negations = Operator.getInstance().getNegations();
return negations.containsKey(token);
}
private static double getModifierValue(String token, Double negation) {
```

```
HashMap<String, Double> modifiers = Operator.getInstance().getModifiers();
double m;
if (modifiers.containsKey(token)) {
m = modifiers.get(token);
if ((negation==-1.0)&&(m==2))
m = 0.8;
}
else
m = 1.0;
return m;
}
public double[] CalculateDistance() {
double sigma = 1.0;
double gaussian_max = Math.exp(-(1/(sigma*sigma*2))) / (sigma * Math.sqrt(2 *
Math.PI));
for (int i=0;i<=5;i++) {
gaussian[i] = (1/gaussian_max)*Math.exp(-(((i)/sigma)*((i)/sigma)) / 2) / (sigma *
Math.sqrt(2 * Math.PI));
}
return gaussian;
}
private double getDistance(Search parent, String token, LinkedList<String> tokens, int
i, double gaussian[]) {
String entity = parent.getQuery().getRequest();
int x = Math.abs(tokens.indexOf(entity));
```

```
double height = Math.abs(lexicon.get(token));
if (Math.abs(x-i)<=5)
return height*gaussian[Math.abs(x-i)];
else
return 0.0;
}
}

package sentimentanalysis.endpoint;
import java.util.LinkedList;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
import sentimentanalysis.SentimentAnalysisApp;
import sentimentanalysis.common.LocalTweet;
import sentimentanalysis.search.Search;
{
SentimentAnalysisApp.getView().setQuery(query);
Search search = new Search(query);
search.execute();
LinkedList<SoapTweet> result = new LinkedList<SoapTweet>();
try {
for (LocalTweet tweet : search.get())
result.push(new SoapTweet(tweet, query));
return result;
```

```
}
 catch (Exception e)
{
e.printStackTrace();
}
return null;
}
}

package sentimentanalysis.endpoint;
import sentimentanalysis.common.LocalTweet;
public class SoapTweet {
public String author;
public double sentiment;
public String date;
public double rank;
public String content;
public SoapTweet () {}
public SoapTweet (LocalTweet parent, String query) {
author      = parent.getFromUser();
sentiment   = parent.getSentimentScore(query);
rank        = parent.getPageRank();
date        = parent.getCreatedAt().toGMTString();
content     = parent.getText();
}
```

```
}
package sentimentanalysis.database;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.sql.SQLException;
import java.text.ParseException;
import java.util.Date;
import java.util.HashMap;
import java.util.LinkedList;
import java.util.logging.Level;
import java.util.logging.Logger;
import sentimentanalysis.common.LocalTweet;
import sentimentanalysis.common.Sentiment;
import sentimentanalysis.common.TweetRankInfo;
import sentimentanalysis.common.User;
import twitter4j.Tweet;
public class Operator implements OperatorInterface
{
private        Connector   con;
private        Connector   conLex;
private static  Operator    instance;
private Operator()
{
con     = new Connector("sentimentdb");
```

```
conLex  = new Connector("lexicondb");
}
public static Operator getInstance ()
{
if (instance == null)
instance = new Operator();
assert(instance != null);
return instance;
}
public void storeTweet(Tweet tw)
{
long id = tw.getId();
String content = tw.getText().replaceAll("'", "");
String author = tw.getFromUser();
Date date = (Date) tw.getCreatedAt();
java.text.SimpleDateFormat sdf = new java.text.SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
String currentTime = sdf.format(date);
Pagerank pgtw = new Pagerank(author);
try
{
String query = "INSERT INTO tweets(id, content, author, date) VALUES(?,?,?,?)";
PreparedStatement stmt = (PreparedStatement) con.getCon().prepareStatement(query);
stmt.clearParameters();
stmt.setLong(1, id);
```

```
stmt.setString(2, content);
stmt.setString(3, author);
stmt.setString(4, currentTime);
stmt.executeUpdate();
}
catch (SQLException ex)
{
Logger.getLogger(Operator.class.getName()).log(Level.SEVERE, null, ex);
}
public long getEntityId(String entity) {
long entityId = 0;
try {
String query = "SELECT id FROM entity " +
"WHERE value = ?";
PreparedStatement ps = (PreparedStatement) con.getCon().prepareStatement(query);
ps.setString(1, entity);
ResultSet rs = (ResultSet) ps.executeQuery();
if (rs.next()) {
entityId = rs.getLong(1);
System.out.println("The entity id is: " + entityId);
}
else
{
System.out.println("N-are");
query = "INSERT INTO entity(value, score) VALUES(?, ?)";
```

```
ps = con.getCon().prepareStatement(query,
Statement.RETURN_GENERATED_KEYS);
ps.setString(1, entity);
ps.setDouble(2, 0);
ps.executeUpdate();
rs = ps.getGeneratedKeys();
con.getCon().commit();
if (rs.next())
{
entityId = rs.getLong(1);
System.out.println("The entity id is: " + entityId);
}
else
{
System.out.println("Something screwey");
}
}
}
catch (SQLException ex)
{
L
ogger.getLogger(Operator.class.getName()).log(Level.SEVERE, null, ex);
}
return entityId;
}
```

```
public void storeTweetBatch(LinkedList<LocalTweet> list, String entity) {
String query = "INSERT IGNORE INTO  `tweets`                    "
+ "            (id, content, author, date, pagerank) "
+ "VALUES      (?,  ?,      ?,     ?,    ?)       ";
try
{
int count       = 0;
PreparedStatement statement = con.getCon().prepareStatement(query);
for (LocalTweet tweet : list) {
++count;
statement.setLong  (1, tweet.getId());
statement.setString(2, tweet.getText());
statement.setString(3, tweet.getFromUser());
statement.setString(4, tweet.getCreatedString());
statement.setDouble(5, tweet.getPageRank());
statement.addBatch();
if (count % 1000 == 0)
{
count = 0;
statement.executeBatch();
}
}
statement.executeBatch();
long entityId = getEntityId(entity);
query = "INSERT INTO sentiments(tweet_id, entity_id, score) VALUES(?,?,?)";
```

```
statement = con.getCon().prepareStatement(query);
count = 0;
for (LocalTweet tw : list)
{
++count;
long id = tw.getId();
double sentimentScore = tw.getSentimentScore(entity);
System.out.println(id + " " + entityId + " " + sentimentScore);
statement.setLong(1, id);
statement.setLong(2, entityId);
statement.setDouble(3, sentimentScore);
statement.addBatch();
if (count % 1000 == 0) {
count = 0;
statement.executeBatch();
}
}
System.out.println("count = " + count);
statement.executeBatch();
con.getCon().commit();
}
catch(SQLException sex)
{
System.out.println("It's not saving because the following error occurs ");
}
```

```
}
public LinkedList<String> getTweetsForTesting()
{
LinkedList<String> list = new LinkedList<String>();
try
{
String query = "SELECT content FROM tweets  ";
con.getCon().setAutoCommit(false);
PreparedStatement ps = (PreparedStatement) con.getCon().prepareStatement(query);
ResultSet rs = (ResultSet) ps.executeQuery();
while (rs.next())
{
list.add(rs.getString(1));
}
}
catch (SQLException ex)
{
Logger.getLogger(Operator.class.getName()).log(Level.SEVERE, null, ex);
}
return list;
}
public LinkedList<String> getTweets(String key)
{
LinkedList<String> list = new LinkedList<String>();
try
```

```
{
String query = "SELECT content FROM tweets "+ "WHERE MATCH (content) AGAINST ('" + key + "' IN BOOLEAN MODE);";
con.getCon().setAutoCommit(false);
PreparedStatement ps = (PreparedStatement) con.getCon().prepareStatement(query);
ResultSet rs = (ResultSet) ps.executeQuery();
while (rs.next())
{
list.add(rs.getString(1));
}
}
catch (SQLException ex)
{
Logger.getLogger(Operator.class.getName()).log(Level.SEVERE, null, ex);
}
return list;
}
public LinkedList<LocalTweet> getLocalTweets(String key) {
LinkedList<LocalTweet> list = new LinkedList<LocalTweet>();
try
{
String query = "SELECT tweets.id, author, content, date, pagerank, "
+ "  entity.value, sentiments.score, sentiments.id "
+ "FROM tweets "
+ "  JOIN sentiments ON (tweets.id = sentiments.tweet_id) "
```

```
+ "  JOIN entity ON (sentiments.entity_id = entity.id) "
+ "WHERE MATCH (content)  AGAINST ('" + key + "' IN BOOLEAN MODE)
ORDER BY tweets.id;";
System.err.println(query);
con.getCon().setAutoCommit(false);
PreparedStatement ps = (PreparedStatement) con.getCon().prepareStatement(query);
ResultSet rs = (ResultSet) ps.executeQuery();
while (rs.next())
{
Sentiment s = new Sentiment();
s.setEntity(rs.getString(6));
s.setScore(rs.getDouble(7));
s.setId(rs.getLong(8));
if (!list.isEmpty() && list.get(list.size() - 1).getId() == rs.getLong(1)) {
list.get(list.size() - 1).addSentiment(key, s);
}
else
{
LocalTweet tweet = new LocalTweet();
tweet.setId(rs.getLong(1));
tweet.setAuthor(rs.getString(2));
tweet.setContent(rs.getString(3));
try
{
```

```
tweet.setDate(newjava.text.SimpleDateFormat("yyyy-MM-dd
HH:mm:ss").parse(rs.getString(4)));
} catch (ParseException ex)
{
Logger.getLogger(Operator.class.getName()).log(Level.SEVERE, null, ex);
}
tweet.setPageRank(rs.getLong(5));
tweet.addSentiment(key, s);
list.add(tweet);
}
}
} catch (SQLException ex)
{
Logger.getLogger(Operator.class.getName()).log(Level.SEVERE, null, ex);
}
return list;
}
public void storeSentiment(long tweet_id, long entity_id, double sentiment, double
score)
{
try
{
String query = "INSERT INTO sentiments(tweet_id, entity_id, score) VALUES(?,?,?)";
PreparedStatement stmt = (PreparedStatement) con.getCon().prepareStatement(query);
stmt.clearParameters();
```

```
stmt.setLong(1, tweet_id);
stmt.setLong(2, entity_id);
stmt.setDouble(3, score);
stmt.executeUpdate();
}
catch (SQLException ex)
{
Logger.getLogger(Operator.class.getName()).log(Level.SEVERE, null, ex);
}
}
public void storeUser(User user) {
try {
if (getUser(user.getId()) != null) {
return;
}
String query = "INSERT INTO users(id, followers_count,"
+ "friends_count, statuses_count) VALUES(?,?,?,?)";
PreparedStatement statement = (PreparedStatement)
con.getCon().prepareStatement(query);
statement.clearParameters();
statement.setLong(1, user.getId());
statement.setInt(2, user.getFollowersCount());
statement.setInt(3, user.getFriendsCount());
statement.setInt(4, user.getStatusesCount());
statement.execute();
```

```
}
catch (SQLException ex) {
Logger.getLogger(Operator.class.getName()).log(Level.SEVERE,
"Error while storing user info", ex);
}
}
public User getUser(long id) {
User user = null;
try {
String query = "SELECT followers_count,friends_count,statuses_count FROM users "
+ "WHERE id = ?";
PreparedStatement ps = (PreparedStatement) con.getCon().prepareStatement(query);
ps.setLong(1, id);
ResultSet rs = (ResultSet) ps.executeQuery();
while (rs.next()) {
user = new User(id, rs.getInt(1), rs.getInt(2), rs.getInt(3));
}
}
catch (SQLException ex) {
Logger.getLogger(Operator.class.getName()).log(Level.SEVERE,
"Error while retrieving user with id " + id, ex);
}
return user;
}
public TweetRankInfo getTweetRankInfo(long id) {
```

```
TweetRankInfo tweetRankInfo = null;
try {
String query = "SELECT retweet_count,is_favorited FROM tweet_rank_info "
+ "WHERE id = ?";
PreparedStatement ps = (PreparedStatement) con.getCon().prepareStatement(query);
ps.setLong(1, id);
ResultSet rs = (ResultSet) ps.executeQuery();
while (rs.next()) {
tweetRankInfo = new TweetRankInfo(id, rs.getLong(1), rs.getBoolean(2));
}
} catch (SQLException ex) {
Logger.getLogger(Operator.class.getName()).log(Level.SEVERE,
"Error while retrieving tweet rank info with id " + id, ex);
}
return tweetRankInfo;
}
public void storeTweetRankInfo(TweetRankInfo tweetRankInfo) {
try {
if (getTweetRankInfo(tweetRankInfo.getId()) != null) {
return;
}
String query = "INSERT INTO tweet_rank_info(id, retweet_count,"
+ "is_favorited) VALUES(?,?,?)";
PreparedStatement statement = (PreparedStatement)
con.getCon().prepareStatement(query);
```

```
statement.clearParameters();
statement.setLong(1, tweetRankInfo.getId());
statement.setLong(2, tweetRankInfo.getRetweetCount());
statement.setBoolean(3, tweetRankInfo.isFavorited());
statement.execute();
}
catch (SQLException ex) {
Logger.getLogger(Operator.class.getName()).log(Level.SEVERE,
"Error while storing retweet rank info", ex);
}
}
public double getSentiment(long key)
{
double score = 0.0;
try
{
String query = "SELECT score FROM sentiments "
+ "WHERE tweet_id = ?";
PreparedStatement ps = (PreparedStatement) con.getCon().prepareStatement(query);
ps.setLong(1, key);
ResultSet rs = (ResultSet) ps.executeQuery();
while (rs.next())
{
score = rs.getDouble(1);
}
```

```
} catch (SQLException ex)
{
Logger.getLogger(Operator.class.getName()).log(Level.SEVERE, null, ex);
}
return score;
}
public void fillLexicon()
{
try
{
String filename1 = "C://Users//maryger//Documents//KTH//new IR project//documentation//lexicon//mpqa.lex";
String filename2 = "C://Users//maryger//Documents//KTH//new IR project//documentation//lexicon//inquirer.lex";
String filename3 = "C://Users//maryger//Documents//KTH//new IR project//documentation//lexicon//modifiers.txt";
String tablename = "lexicon";
String tablename2 = "modifiers";
Statement stmt = (Statement) conLex.getCon().createStatement();
stmt.executeUpdate("LOAD DATA INFILE '" + filename3 + "' INTO TABLE "
+ tablename2 + " FIELDS TERMINATED BY ':' LINES TERMINATED BY '\\r\\n' (modifier, value)");
}
catch (SQLException ex)
{
```

```
Logger.getLogger(Operator.class.getName()).log(Level.SEVERE, null, ex);
}
}
public HashMap<String,Double> getLexicon () throws SQLException
{ HashMap<String,Double> lexicon   = new HashMap<String,Double>();
String query     = "SELECT `word`, `sentiment` FROM `lexicon`";
PreparedStatement      statement = conLex.getCon().prepareStatement(query);
ResultSet              result    = statement.executeQuery();
while (result.next())
lexicon.put(result.getString(1), result.getDouble(2));
return lexicon;
}
public void storeEntity(String value, double score)
{
try
{
String query = "INSERT INTO entity(value,score) VALUES(?,?)";
PreparedStatement stmt = (PreparedStatement) con.getCon().prepareStatement(query);
stmt.clearParameters();
stmt.setString(1, value);
stmt.setDouble(2, score);
stmt.executeUpdate();
}
catch (SQLException ex)
{
```

```
Logger.getLogger(Operator.class.getName()).log(Level.SEVERE, null, ex);
}
}
public double getEntityScore(String key)
{
double score = 0.0;
return score;
}
public void storeEntityRelatedWord(String entity, String related_word, double score)
{
try
{
String query = "INSERT INTO entity_relatedword(entity, related_word, score) VALUES(?,?,?)";
PreparedStatement stmt = (PreparedStatement) con.getCon().prepareStatement(query);
stmt.clearParameters();
stmt.setString(1, entity);
stmt.setString(2, related_word);
stmt.setDouble(3, score);
stmt.executeUpdate();
}
catch (SQLException ex)
{
Logger.getLogger(Operator.class.getName()).log(Level.SEVERE, null, ex);
}
```

```
}
public HashMap<String,Double> getRelatedWord(String key_entity)
{
HashMap<String,Double> list = new HashMap<String,Double>();
try
{
String query = "SELECT related_word,relation_score FROM entity_relatedword "
+ "WHERE MATCH (entity)  AGAINST ('" + key_entity + "' IN BOOLEAN
MODE);";
con.getCon().setAutoCommit(false);
PreparedStatement ps = (PreparedStatement) con.getCon().prepareStatement(query);
ResultSet rs = (ResultSet) ps.executeQuery();
while (rs.next())
{
list.put(rs.getString(1),rs.getDouble(2));
}
}
catch (SQLException ex)
{
Logger.getLogger(Operator.class.getName()).log(Level.SEVERE, null, ex);
}
return list;
}
public HashMap<String,Double> getLexiconEntries(String key)
{
```

```
HashMap<String,Double> list = new HashMap<String,Double>();
try
{
String query = "SELECT word,sentiment FROM entity_relatedword "
+ "WHERE MATCH (word)  AGAINST ('" + key + "' IN BOOLEAN MODE);";
con.getCon().setAutoCommit(false);
PreparedStatement ps = (PreparedStatement) con.getCon().prepareStatement(query);
ResultSet rs = (ResultSet) ps.executeQuery();
while (rs.next())
{
list.put(rs.getString(1),rs.getDouble(2));
}
}
catch (SQLException ex)
{
Logger.getLogger(Operator.class.getName()).log(Level.SEVERE, null, ex);
}
return list;
}
public void closeCon()
{
con.closeCon();
}
public void closeConLex()
{
```

```
conLex.closeCon();
}
public HashMap<String, Double> getSmileys()
{
HashMap<String,Double> list = new HashMap<String,Double>();
try
{
String query = "SELECT smiley,value FROM smileys;";
con.getCon().setAutoCommit(false);
PreparedStatement ps = (PreparedStatement)
conLex.getCon().prepareStatement(query);
ResultSet rs = (ResultSet) ps.executeQuery();
while (rs.next())
{
list.put(rs.getString(1),rs.getDouble(2));
}
}
catch (SQLException ex)
{
Logger.getLogger(Operator.class.getName()).log(Level.SEVERE, null, ex);

return list;
}
public HashMap<String, Double> getModifiers()
{
```

```
HashMap<String,Double> list = new HashMap<String,Double>();
try
{
String query = "SELECT modifier,value FROM modifiers;";
con.getCon().setAutoCommit(false);
PreparedStatement ps = (PreparedStatement)
conLex.getCon().prepareStatement(query);
ResultSet rs = (ResultSet) ps.executeQuery();
while (rs.next())
{
list.put(rs.getString(1),rs.getDouble(2));
}
}
catch (SQLException ex)
{
Logger.getLogger(Operator.class.getName()).log(Level.SEVERE, null, ex);
}
return list;
}
public HashMap<String, Double> getNegations()
{
HashMap<String,Double> list = new HashMap<String,Double>();
try
{
String query = "SELECT negation,value FROM negations;";
```

```
con.getCon().setAutoCommit(false);
PreparedStatement ps = (PreparedStatement) conLex.getCon().prepareStatement(query);
ResultSet rs = (ResultSet) ps.executeQuery();
while (rs.next())
{
list.put(rs.getString(1),rs.getDouble(2));
}
}
catch (SQLException ex)
{
Logger.getLogger(Operator.class.getName()).log(Level.SEVERE, null, ex);
}
return list;
}
public long getIdOfLatestTweetForThisEntity(String entity) {
long result = 0;
try
{
String query = "SELECT MAX(tweet_id) FROM sentiments WHERE entity_id = ?";
PreparedStatement ps = (PreparedStatement) con.getCon().prepareStatement(query);
long entityId = getEntityId(entity);
ps.setLong(1, entityId);
ResultSet rs = ps.executeQuery();
if (rs.next()) {
```

```
result = rs.getLong(1);
}
}
catch (SQLException ex)
{
Logger.getLogger(Operator.class.getName()).log(Level.SEVERE, null, ex);
}
return result;
}
}
```

APPENDIX 2

A.2 Screen shots:

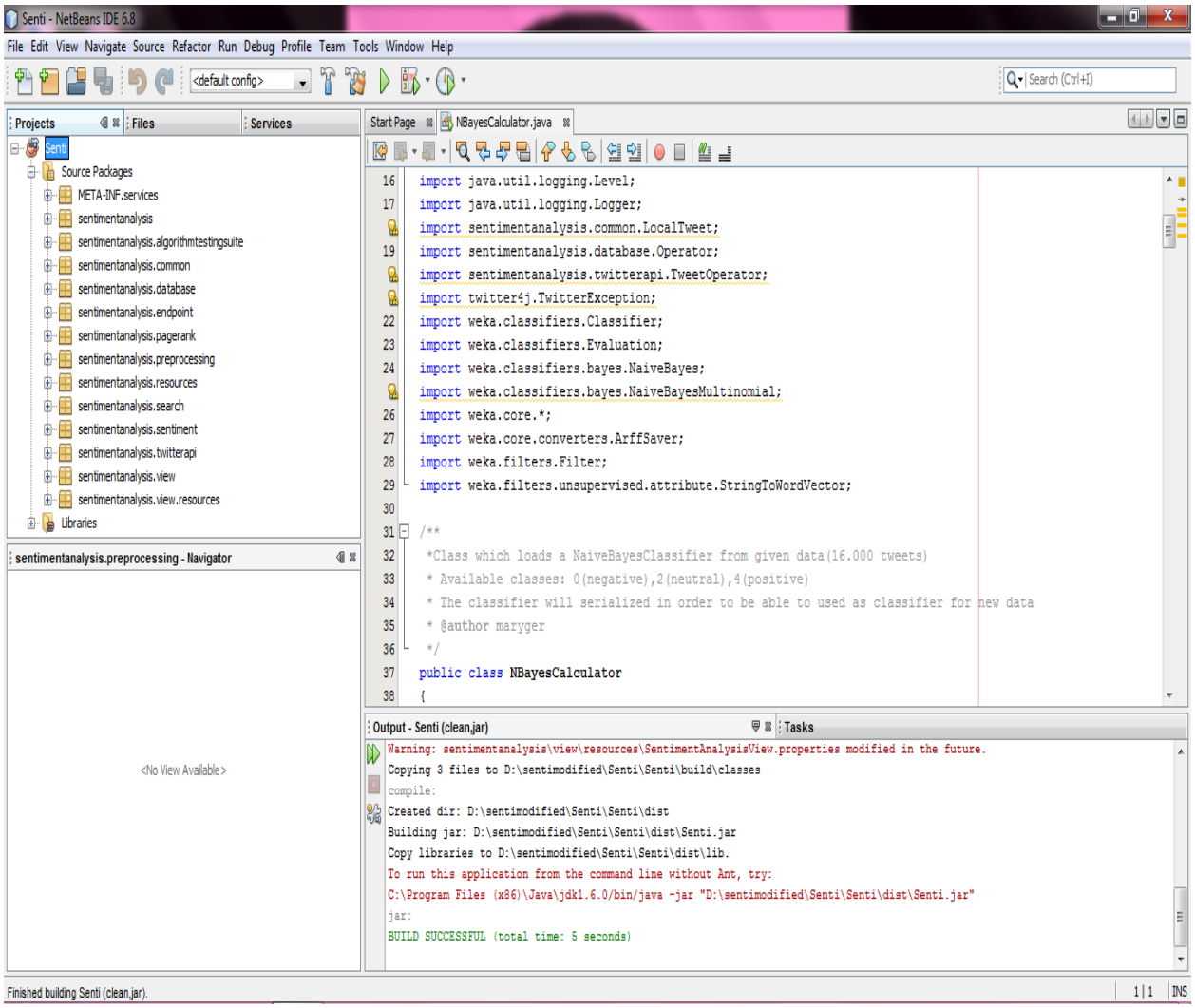


Fig., A.2. 1 Building the source code successfully.

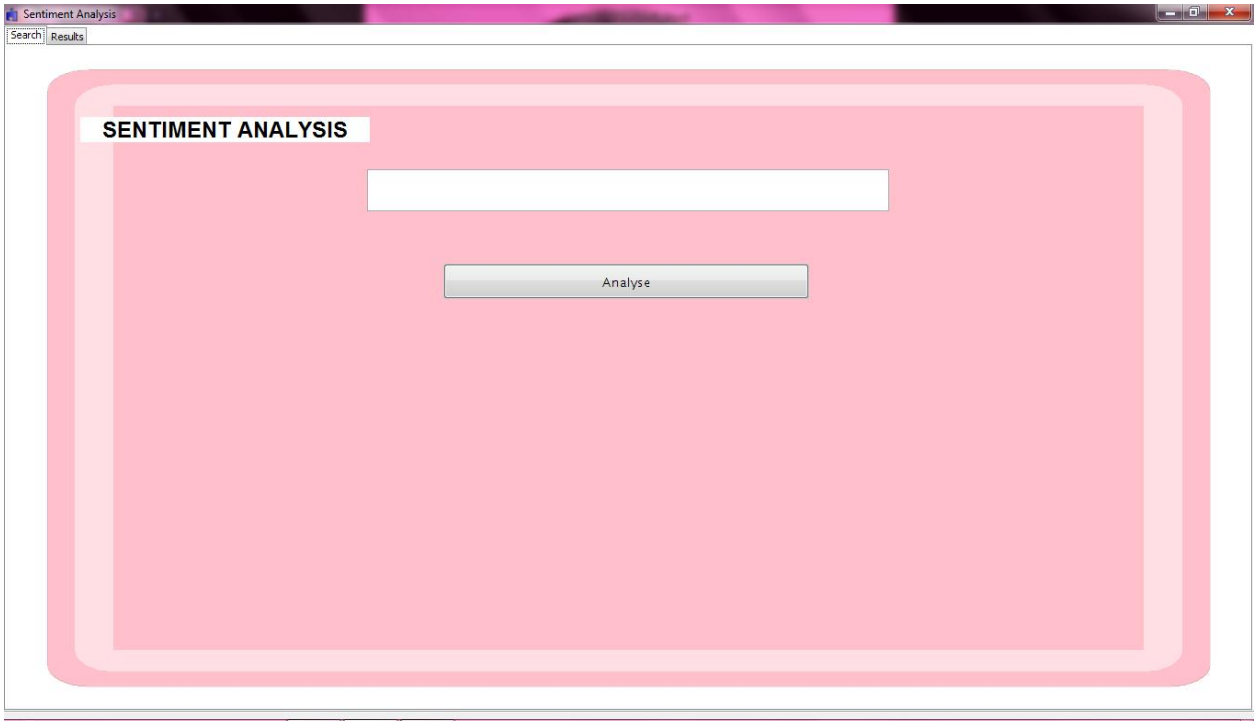


Fig: A.2.2 The initial page where the text should be given as an input.

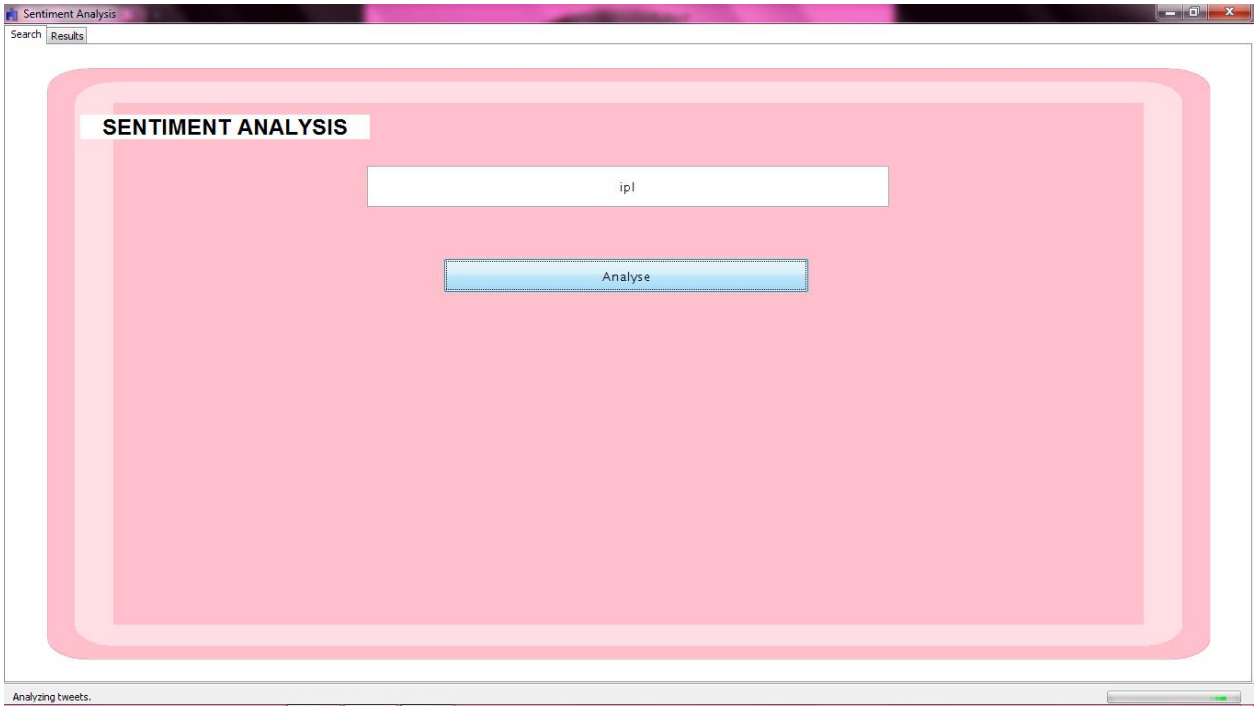


Fig., A.2.3 The input text is ipl

Fig., A.2.4 The result is displayed as sentiments in the form of "positive, negative & neutral" in graphical representation for the input ipl.

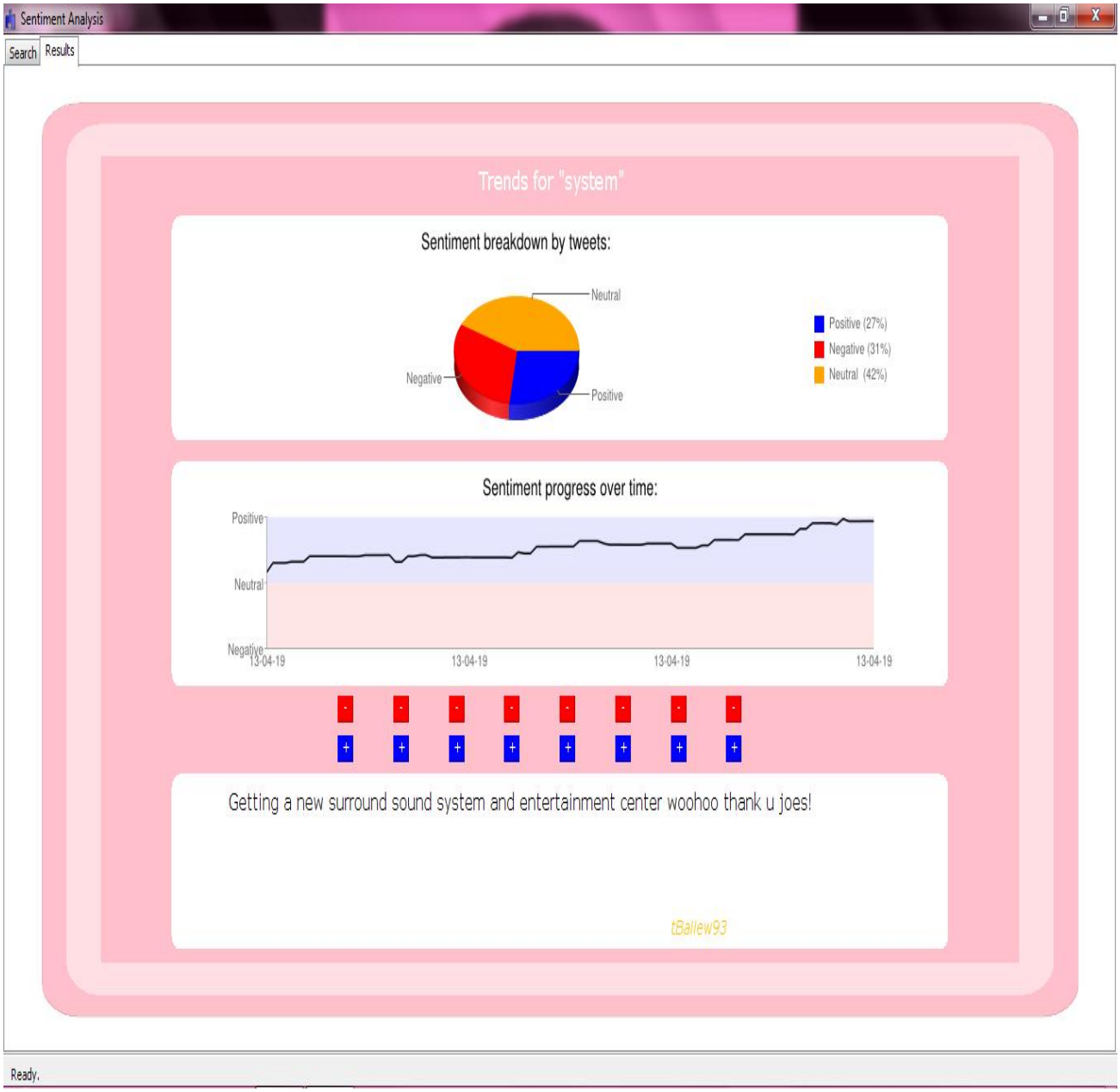


Fig., A.2.5 The result is displayed as sentiments in the form of "positive, negative & neutral" in graphical representation for the input system.

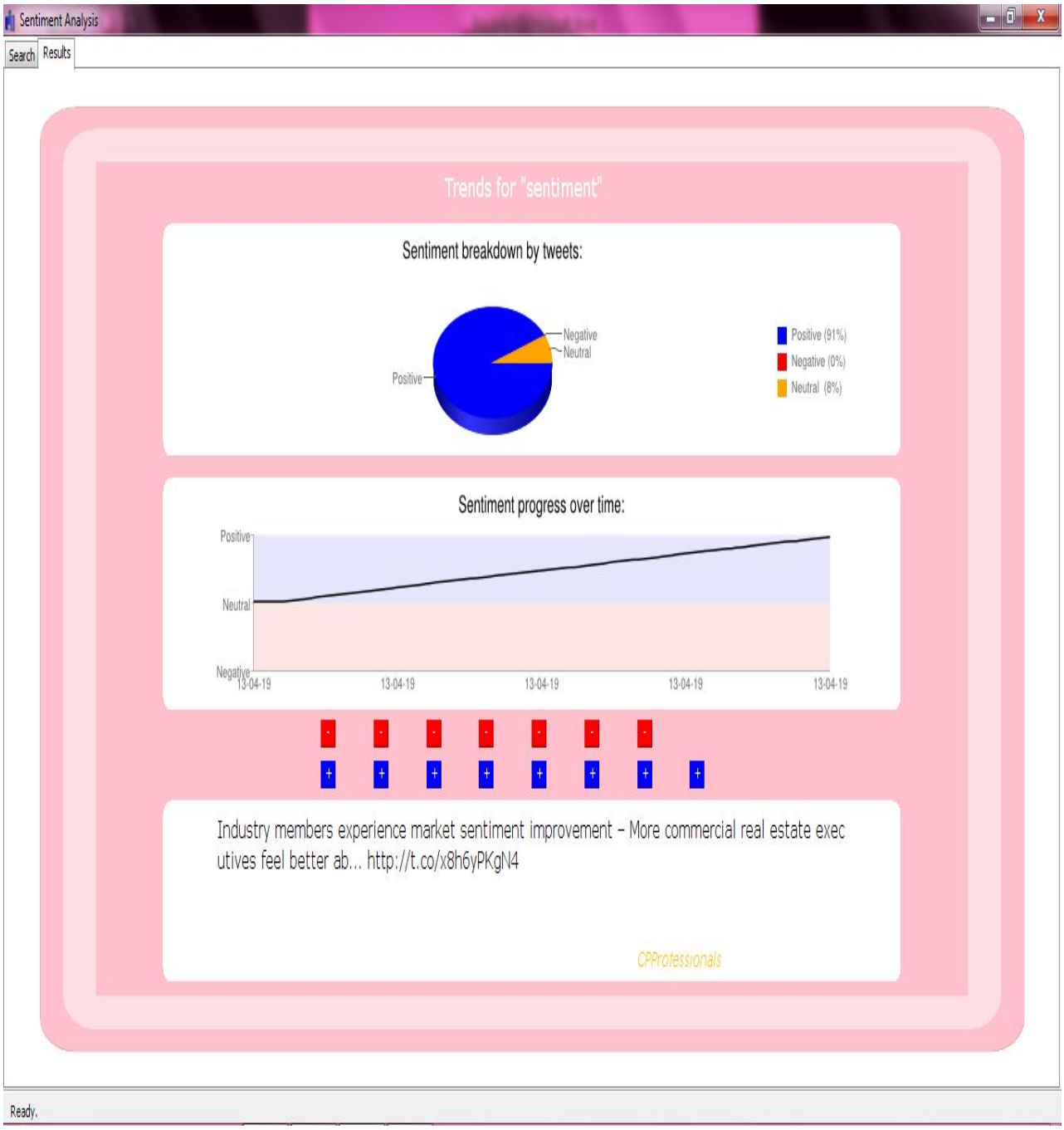


Fig., A.2.6 The result is displayed as sentiments in the form of "positive, negative & neutral" in graphical representation for the input sentiment.

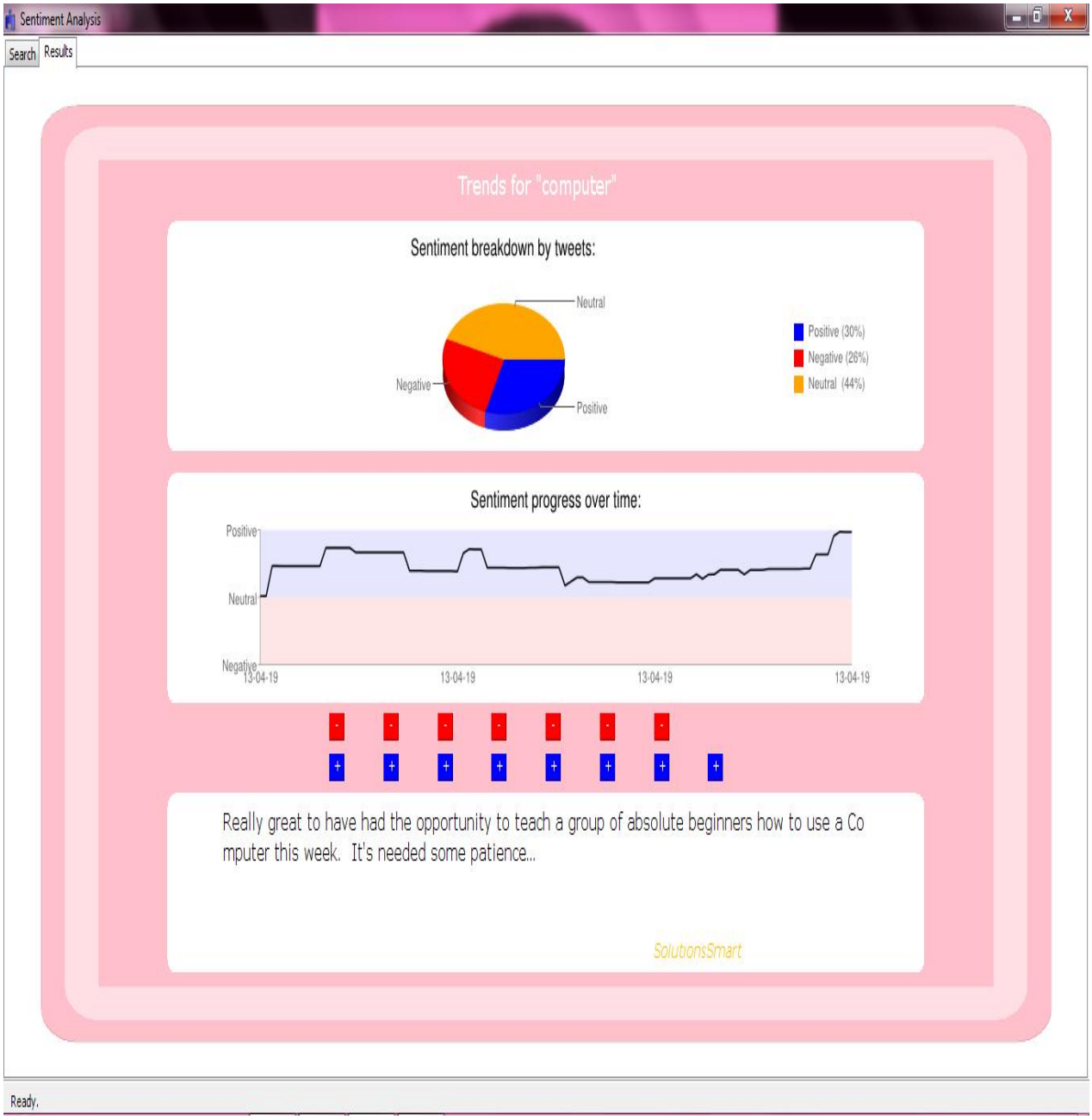


Fig.,A.2.7 The result is displayed as sentiments in the form of “positive, negative & neutral” in graphical representation for the input computer.

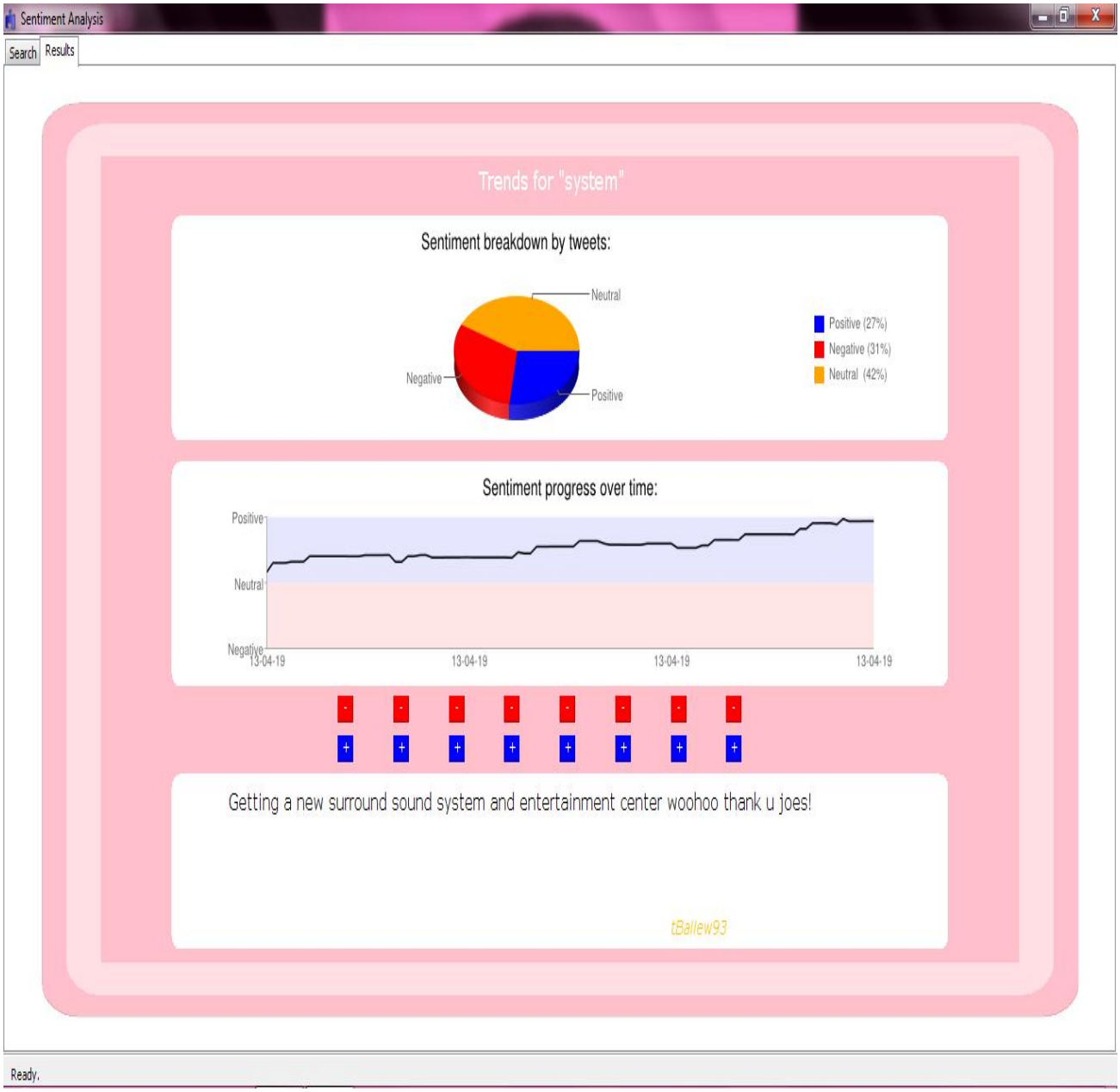


Fig.,A.2.8 The result is displayed as sentiments in the form of "positive, negative & neutral" in graphical representation for the input system.

CHAPTER 9

LIST OF PUBLICATIONS

9.1 INTERNATIONAL JOURNALS:

a) Ms.S.Yamini, Dr.V.Khanaa and Dr.Krishna Mohanta published a paper titled "Sentiment Analysis: Index Based On Empirical Study" in International Journal of Pharmacy and Technology (IJPT), Vol.8, Issue 2, ISSN: 0975-766X, PP: 12753-12761. **SCOPUS**-Indexed.

b) Ms.S.Yamini, Dr.V.Khanaa and Dr.Krishna Mohanta published a paper titled "Sentimental Analysis Of Online Content: A Practical Approach" in International Journal of Pharmacy and Technology (IJPT), Vol.8, Issue 2, ISSN: 0975-766X, PP: 13803-13814. **SCOPUS**-Indexed.

c) Ms.S.Yamini, Dr.V.Khanaa and Dr.Krishna Mohanta published a paper titled "A state of the art review on various data mining techniques" in International Journal of Innovative Research in Science, Engineering and Technology (IJIESET), Vol.5, Issue 3, March 2016.Impact factor- 5.482.

d) Ms.S.Yamini, Dr.V.Khanaa and Dr.Krishna Mohanta published a paper titled "Mining high dimensional web content with sentimental analysis: a proactive approach" in International Journals of Recent and Innovative Trends in Computing and Communication (IJRITCC) ,Vol.4,issue 2 , Feb 2016. ISSN: 2321-8169,
PP: 238 – 242.Impact factor- 5.837.

e) Ms.S.Yamini, Dr.V.Khanaa and Dr.Krishna Mohanta published a paper titled "Ensuring data storage in cloud computing" in International Journal of Advanced Research (IJAR) , Vol 1 , Issue 9 , 2013.PP: 415-420. Impact factor- 5.336.

(f) Ms.S.Yamini, Dr.V.Khanaa and Dr.Krishna Mohanta published a paper titled "Supervised approach to extract sentiments from unstructured text" in International Journal of Engineering Inventions (IJEI) Vol. 2, issue 10, June 2013.

Impact factor- 1.21.

9.2 WORKSHOPS PARTICIPATED

a) Ms.S.Yamini participated in TCS associated one day national level workshop on "Text Mining" organized by Department of CSE on February 2nd, 2015 at Easwari Engineering College, Chennai.

b) Ms.S.Yamini participated in the workshop on "Research Methodology" organized by Vel Tech University, Avadi, Chennai.

9.3 INTERNATIONAL CONFERENCE

a) Ms.S.Yamini,Dr.V.Khanaa presented a paper on "Sentiment extraction of unstructured text" in the International conference on innovations in communication, information and computing at Sasurie College of engineering on 9th, 10th, 11th January 2013.

b) Ms.S.Yamini presented a paper on "Supervised approach to extract sentiments from unstructured text" in the International conference on Electrical communication and computing organized by Tagore engineering college on 13th, 14th March 2014.

9.4 NATIONAL CONFERENCE:

a) Ms.S.Yamini presented a paper on "Sentiment extraction of unstructured text" in the National conference on Networking and computing at Vel tech multi tech engineering college on August 10th, 2012.

b) Dr.Krishna Mohanta,Ms.S.Yamini and Dr.V.Khanaa presented a paper on "Remote access of desktop through mobile" in the National conference on Networking and computing technology at Vel tech multi tech engineering college on August 12th, 2013.