1. INTRODUCTION

Elections empower citizens to choose their leaders. It gives all an opportunity for equal voice and representation in our government. Democracy is government for the people, and by the people, which means government leaders are determined by participation in elections. As we approach the 2016 November presidential election, the public sentiment towards candidates will influence the future leader of our country.

We are interested in how the public views the top election candidates, namely Donald Trump, Hillary Clinton, and Bernie Sanders. Feelings towards candidates fluctuate quickly as interviews, debates, responses to global events, and other issues come to front. To achieve a large, diverse dataset of current public opinions on the candidates we focus on twitter for data analysis, where twitter is an online networking service that enables users to send and read short 140- character messages called "tweets". In addition to its publicity, twitter is accessible for unregistered users to read and monitor most tweets, unlike Facebook where users can control the privacy of their profiles. Twitter is also a large social networking micro blogging site. The massive information provided by twitter such as tweet messages, user profile information, and the number of followers/ followings in the network play a significant role in data analysis, which in return make most studies investigate and examine various analysis techniques to grasp the recent used technologies. We are analyzing recent tweets regarding the top candidates in the 2016 election to predict the public sentiments towards each candidate.

Our project attempts to solve the real-world problem of understanding the current sentiment towards election candidates based off the public's live opinions and emotions rather than off of smaller, localized polls typically done by mainstream media corporations.

1.1 Scope of the Project

To know the public views of the top election candidates for the USA in the year 2016, namely Donald Trump, Hillary Clinton and Bernie Sanders. Twython twitter API is used to collect tweets based on keywords and Python 3.6 is used for sentiment analysis and the results are graphically represented.

1.1.1 Collecting Tweets using Twitter API Twython

Twython is a library of python providing an easy (and up-to-date) way to access Twitter data in Python. With the help of Twython we collect tweets during election and these tweets are filtered using nltk (Natural Language toolkit) where unnecessary common words are removed. These filtered tweets are stored in text files which are then imported to Sentiment Analysis.

1.1.2 Sentiment Analysis of Tweets

Sentiment analysis aims to determine the attitude of a speaker, writer, or other subject with respect to some topic or the overall contextual polarity or emotional reaction to a document, interaction, or event. The attitude may be a judgment or evaluation (see appraisal theory), affective state (that is to say, the emotional state of the author or speaker), or the intended emotional communication. A basic task in sentiment analysis is classifying the polarity of a given text at the document, sentence, or feature/aspect level—whether the expressed opinion in a document, a sentence or an entity feature/aspect is positive, negative, or neutral.

Sentiment Analysis calculates the polarity and subjectivity of each tweet and using these values it classifies the tweets as positive, negative or neutral. The module TextBlob contains necessary methods to carry out sentiment analysis on text files.The main default sentiment calculation is defined in _text.py, which gives credit to the pattern library. The lexicon it refers to is in en-sentiment.xml, an XML document that includes the following four entries for thousands of words defined in the English dictionary.

Each word in the XML file is assigned a polarity, subjectivity, intensity and confidence values. Based on the context in which the word is used the values may change. TextBlob goes along finding words and phrases it can assign polarity and subjectivity to, and it averages them all together for longer text.

1.1.3 Output

The output is represented visually via tables, bar charts, and word clouds. For generating the tables and bar graphs plotly.py, an interactive, browser-based charting library for python, is used. In the table each row contains the name of the candidate, the number of positive tweets, the number of negative tweets, number of neutral tweets, percentage of positive tweets, percentage of negative tweets, percentage of neutral tweets, average polarity and average subjectivity. In the bar graph the positive, negative and neutral sentiments obtained are compared among the three candidates. The x-axis is the number of tweets and the y-axis is positive, negative or neutral.

A Word Cloud is a visual representation of text data, typically used to depict keyword metadata (tags) on websites, or to visualize free form text. Tags are usually single words, and the importance of each tag is shown with font size or color. This format is useful for quickly perceiving the most prominent terms and for locating a term alphabetically to determine its relative prominence. The WordCloud library is used to generate word clouds from text files in python.

2. LITERATURE SURVEY

2.1 Sentiment Analysis

The opinions of others have a significant influence in our daily decision-making process. These decisions range from buying a product such as a smart phone to making investments to choosing a school—all decisions that affect various aspects of our daily life. Before the Internet, people would seek opinions on products and services from sources such as friends, relatives, or consumer reports.

However, in the Internet era, it is much easier to collect diverse opinions from different people around the world. People look to review sites (e.g., CNET, Epinions.com), e-commerce sites (e.g., Amazon, eBay), online opinion sites (e.g., TripAdvisor, Rotten Tomatoes, Yelp) and social media (e.g., Facebook, Twitter) to get feedback on how a particular product or service may be perceived in the market.

Similarly, organizations use surveys, opinion polls, and social media as a mechanism to obtain feedback on their products and services. sentiment analysis oropinion mining is the computational study of opinions, sentiments, and emotions expressed in text. The use of sentiment analysis is becoming more widely leveraged because the information it yields can result in the monetization of products and services.

2.1.1 Converting Unstructured Text into Structured Opinions

Consumer sentiments are mainly expressed in an unstructured format. Text analysis involves taking unstructured data and finding ways to convert it into a more structured format that facilitates analysis of data and getting deeper insights into the sentiments. There are various techniques that can be used to convert unstructured data into a structured format.

2.1.2 Aspect Based Sentimental Analysis (ABSA)

ABSA is based on identifying aspects of given target entities and estimating the sentiment polarity for each mentioned aspect. This can be decomposed into two tasks: aspect extraction and aspect sentiment classification. Aspect extraction pertains to recognizing aspects of the entity, and more generally can be seen as an information extraction task. Aspect sentiment classification

determines whether the opinions on different aspects are positive, negative or neutral. While lexicon-based approaches use a list of aspect related sentiment phrases as the main resource, the key issue for learning methods is to determine the scope of each sentiment expression, if it covers the aspect in the sentence.

2.1.3Sentiment Classification Levels

Sentiment analysis can occur at different levels: document level, sentence level or aspect/feature level.

Document Level Classification: In this process, sentiment is extracted from the entire review, and a whole opinion is classified based on the overall sentiment of the opinion holder. The goal is to classify a review as positive, negative, or neutral.

Sentence Level Classification: This process usually involves two steps:

• Subjectivity classification of a sentence into one of two classes: objective and subjective.

• Sentiment classification of subjective sentences into two classes: positive and negative.

An objective sentence presents some factual information, while a subjective sentence expresses personal feelings, views, emotions, or beliefs. Subjective sentence identification can be achieved through different methods such as Naïve Bayesian classification[7]. However, just knowing that sentences have a positive or negative opinion is not sufficient. This is an intermediate step that helps filter out sentences with no opinions and helps determine to an extent if sentiments about entities and their aspects are positive or negative. A subjective sentence may contain multiple opinions and subjective and factual clauses[8].

2.2 Types of Sentiment Analysis

2.2.1 Corpus Analysis

First, we checked the distribution of words frequencies in the corpus. A plot of word frequencies is presented in Figure1.As we can see from the plot, the distribution of word frequencies follows Zipf's law, which confirms a proper characteristic of the collected corpus.

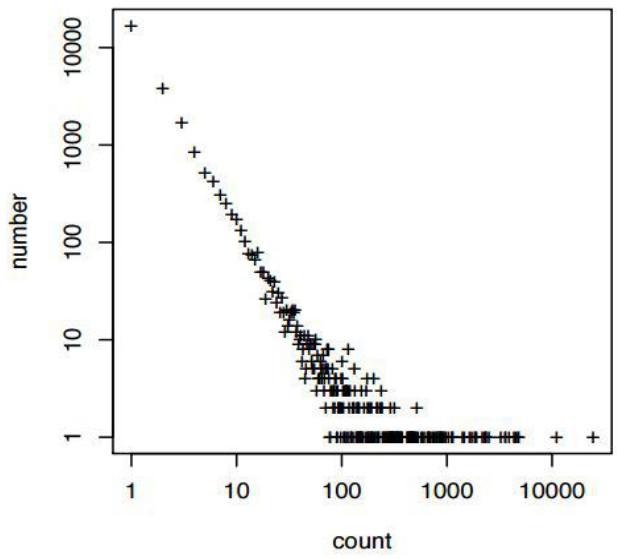

Figure 2.1: The distribution of the word frequencies follows Zipf's law

Next, we used Tree Tagger for English to tag all the posts in the corpus. We are interested in a difference of tags distributions between sets of texts (positive,negative, neutral). To perform a pairwise comparison of tags distributions, we calculated the following value for each tag and two sets (i.e. positive and negative posts):

$$P_{1,2}^{T} = \frac{N_1^T - N_2^T}{N_1^T + N_2^T}$$

where NT1 and NT2 are numbers of tag T occurrences in the first and second sets respectively.

Subjective vs. Objective: Objective texts tend to contain more common and proper nouns (NPS, NP, NNS), while authors of subjective texts use more often personal pronouns (PP,PP$)[8].Authors of subjective texts usually describe themselves(first person) or address the audience (second person)(VBP), while verbs in objective texts are usually in the third person (VBZ). As for the tense, subjective texts tend to use simple past tense (VBD) instead of the past participle(VBN). Also a base form of verbs (VB) is used often in subjective texts, which is explained by the frequent use of modal verbs (MD). Superlative adjectives (JJS) are used more often for expressing emotions and opinions, and comparative adjectives (JJR) are used for stating facts and providing information[7]. Adverbs (RB) are mostly used in subjective texts to give an emotional color to a verb.Another indicator of a positive text is superlative adverbs(RBS), such as "most" and "best". Positive texts are also characterized by the use of possessive ending

(POS).As opposite to the positive set, the negative set contains more often verbs in the past tense (VBN, VBD), because many authors express their negative sentiments about their loss or disappointment. Here is an example of the most frequent verbs: “missed”, “bored”, “gone”, “lost”, “stuck”,“taken”.

2.2.2 Hash tag-based Sentiment Analysis

The Twitter data set contains above 2.5 million different user-defined hash tags. Many tweets include more than a single tag and 3852 “frequent” tags appear in more than 1000 different tweets. Two human judges manually annotated these frequent tags into five different categories: 1 – strong sentiment (e.g #sucks), 2 – most likely sentiment (e.g., #notcute), 3 – context dependent sentiment (e.g., #shoutsout), 4 – focused sentiment (e.g., #tmobilesucks where the target of the sentiment is part of the hash tag), and 5 – no sentiment (e.g. #obama). Table 1 shows annotation results and the percentage of similarly assigned values for each category.

Category	# of tags	% agreement
Strong sentiment	52	87
Likely sentiment	70	66
Context-dependent	110	61
Focused	45	75
No sentiment	3564	99

Table 2.1: Annotation results (2 judges) for the 3852 most frequent tweeter tags. The second column displays the average number of tags, and the last column shows % of tags annotated similarly by two judges.

We selected 50 hash tags annotated “1” or “2” by both judges. For each of these tags we automatically sampled 1000 tweets resulting in 50000 labeled tweets. We avoided sampling tweets which include more than one of the sampled hash tags. As a no-sentiment dataset we randomly sampled 10000 tweets with no hash tags/smileys from the whole dataset assuming that such a random sample is unlikely to contain a significant amount of sentiment sentences[4].

2.2.3 Emoticons as Cues for Sentiment

In order to assess the role emoticons play in conveying the sentiment of a text, a qualitative analysis of a collection of 2,080 Dutch tweets and forum messages is shown. Also randomly sampled this content from search results from Twitter and Google discussion groups when

querying for brands like Vodafone, KLM, Kinect, etcetera is shown. First step is to hypothesize that emoticons have a rather local effect, i.e., they affect a paragraph or a sentence. Paragraphs typically address different points of view for a single topic or different topics, thus rendering the applicability of an emoticon in one paragraph to another paragraph rather unlikely. In sample collection, upon inspection, emoticons generally have a paragraph-level effect for paragraphs containing only one emoticon. [3]When a paragraph contains multiple emoticons, our sample shows that an emoticon is generally more likely to affect the sentence in which it occurs. Interestingly, in this sample given in below table, 84.0% of all emoticons are placed at the end of a paragraph, 9.0% are positioned somewhere in the middle of a paragraph, and 7.0% are used at the beginning of a paragraph. This positioning of emoticons suggests that it is typically not a single word, but rather a text segment that is affected by an emoticon. Additionally, these results imply that in case an emoticon is used in the middle of a paragraph with multiple emoticons, the emoticon is statistically more likely to be associated with the preceding text segment.

Sentence	How	Sentiment
I love my work :-D	Intensification	Positive
The movie was bad :-D	Negation	Positive
:-D I got a promotion	Only sentiment	Positive
-_- I love my work	Negation	Negative
The movie was bad -_-	Intensification	Negative
I got a promotion -_-	Only sentiment	Negative

Table 2.2: Typical examples of how emoticons can be used to convey sentiment.

Table 1 clearly shows that the sentiment associated with a sentence can differ when using different emoticons, i.e., the happy emoticon ":-D" and the "-_-" emoticon indicating extreme boredom or disagreement, irrespective of the position of the emoticons[3]. The sentiment carried by an emoticon is independent from its embedding text, rendering word sense disambiguation techniques not useful for emoticons. As such, the sentiment of emoticons appears to be dominating the sentiment carried by verbal cues in sentences, if any. In some cases, this may be a crucial property which can be exploited by automated sentiment analysis approaches. For instance, when an emoticon is the only cue in a sentence conveying sentiment, we are typically dealing with a phenomenon that we refer to as factual sentiment. For example, the sentence "I got a promotion" does nothing more than stating the fact that one got promoted. However, getting a promotion is usually linked to a positive emotion like happiness or pride. Therefore,

human interpreters could typically be inclined to acknowledge the implied sentiment and thus consider the factual statement to be a positive statement. This however requires an understanding of context and involves incorporating real-world knowledge into the process of sentiment analysis. For machines, this is a cumbersome task. In this light, emoticons can be valuable cues for deriving an author's intended sentiment.

2.2.4 SentiWordNet 3.0

SentiWordNet 3.0, an enhanced lexical resource explicitly devised for supporting sentiment classification and opinion mining applications[1]. SentiWordNet is the result of the automatic annotation of all the synsets of WORDNET according to the notions of "positivity", "negativity", and "neutrality".". Each synset s is associated to three numerical scores P os(s), Neg(s), and Obj(s) which indicate how positive, negative, and "objective" (i.e., neutral) the terms contained in the synset are. Different senses of the same term may thus have different opinion-related properties.

Generating SentiWordNet 3.0: We here summarize in more detail the automatic annotation process according to which SENTIWORDNET 3.0 is generated. This process consists of two steps, (1) a weak-supervision, semi-supervised learning step, and (2) a random-walk step.

1. The semi-supervised learning step:

The semi-supervised learning step is identical to the process used for generating SENTIWORDNET 1.0can then be consulted for more details on this process. The step consists in turn of four substeps: (1) seed set expansion, (2) classifier training, (3) synset classi- fication, and (4) classifier combination.

In Step (1), two small "seed" sets (one consisting of all the synsets containing 7 "paradigmatically positive" terms, and the other consisting of all the synsets containing 7 "paradigmatically negative" terms) are automatically expanded by traversing a number of WORDNET binary relations than can be taken to either preserve or invert the P os and Neg properties (i.e., connect synsets of a given polarity with other synsets either of the same polarity – e.g., the "also-see" relation – or of the opposite polarity – e.g., the "direct antonymy" relation), and by adding the synsets thus reached to the same seed set (for polarity-preserving relations) or to the other seed set (for polarity-inverting ones). This expansion can be performed with a certain "radius"; i.e., using radius k means adding to the seed sets all the synsets that are within distance k from the members of the original seed sets in the graph collectively resulting

from the binary relationships considered.k-supervision, semi-supervised learning step, and (2) a random-walk step.

In Step (2), the two sets of synsets generated in the previous step are used, along with another set of synsets assumed to have the Obj property, as training sets for training a ternary classifier (i.e. one that needs to classify a synset as P os, Neg, or Obj). The glosses of the synsets are used by the training module instead of the synsets themselves, which means that the resulting classifier is indeed a gloss (rather than a synset) classifier. SENTIWORDNET 1.0 uses a "bag of words" model, according to which the gloss is represented by the (frequency-weighted) set of words occurring in it. In SENTIWORDNET 3.0 we instead leverage on the manually disambiguated glosses available from the Princeton WordNet Gloss Corpus, according to which a gloss is actually a sequence of WORDNET synsets. Our gloss classifiers are thus based on what might be called a "bag of synsets" model.

In Step (3), all WORDNET synsets (including those added to the seed sets in Step (2)) are classified as belonging to either P os, Neg, or Obj via the classifier generated in Step (2).

In Step (4), the final P os (resp., Neg, Obj) value of a given synset is generated as its average P os (resp., Neg, Obj) value across the eight classifiers in the committee.

2. The random-walk step :

The random-walk step consists of viewing WORDNET 3.0 as a graph, and running an iterative, "random-walk" process in which the P os(s) and Neg(s) (and, consequently, Obj(s)) values, starting from those determined in the previous step, possibly change at each iteration. The random walk step terminates when the iterative process has converged. The graph used by the random-walk step is the one implicitly determined on WORDNET by the definiensdefiniendum binary relationship; in other words, we assume the existence of a directed link from synset s1 to synset s2 if and only if s1 (the definiens) occurs in the gloss of synset s2 (the definiendum). The basic intuition here is that, if most of the terms that are being used to define a given term are positive (resp., negative), then there is a high probability that the term being defined is positive (resp., negative) too. In other words, positivity and negativity are seen as "flowing through the graph", from the terms used in the definitions to the terms being defined.

However, it should be observed that, in "regular" WORDNET, the definiendum is a synset while the definiens is a non-disambiguated term, since glosses are sequences of non-disambiguated terms[11]. In order to carry out the random walk step, we need the glosses to be disambiguated

against WORDNET itself, i.e., we need them to be sequences of WORDNET synsets. While for carrying out the random walk step for SENTIWORDNET 2.0 we had used the automatically disambiguated glosses provided by EXTENDEDWORDNET, for SENTIWORDNET 3.0 we use the manually disambiguated glosses available from the above-mentioned Princeton WordNet Gloss Corpus[1].

2.3 Word Cloud

A word cloud is a visual representation of text data, typically used to depict keyword metadata (tags) on websites, or to visualize free form text. Tags are usually single words, and the importance of each tag is shown with font size or color. This format is useful for quickly perceiving the most prominent terms and for locating a term alphabetically to determine its relative prominence. When used as website navigation aids, the terms are hyper-linked to items associated with the tag.

2.3.1 Types of Word Cloud Applications

There are two main types of word cloud applications in social software, distinguished by their meaning rather than appearance. In the first type, there is a tag for the frequency of each item, then there are global word clouds where the frequencies are aggregated over all items and users. In the second type, the cloud contains categories, with size indicating number of subcategories.

The term keyword cloud is sometimes used as a search engine marketing (SEM) term that refers to a group of keywords that are relevant to a specific website. In recent years tag clouds have gained popularity because of their role in search engine optimization of Web pages as well as supporting the user in navigating the content in an information system efficiently. Tag clouds as a navigational tool make the resources of a website more connected, when crawled by a search engine spider, which may improve the site's search engine rank. From a user interface perspective they are often used to summarize search results to support the user in finding content in a particular information system more quickly.

2.3.2 Frequency

In the first type, size represents the number of times that tag has been applied to a single item. This is useful as a means of displaying metadata about an item that has been democratically

"voted" on and where precise results are not desired. Examples of such use include Last.fm (to indicate genres attributed to bands) and Library Thing (to indicate tags attributed to a book).

In the second step, more commonly used type, size represents the number of items to which a tag has been applied, as a presentation of each tag's popularity. Examples of this type of tag cloud are used on the image-hosting service Flickr, blog aggregator Technorati and on Google search results with DeeperWeb.

2.3.3 Categorization

In the second type, tags are used as a categorization method for content items. Tags are represented in a cloud where larger tags represent the quantity of content items in that category.

There are some approaches to construct tag clusters instead of word clouds, e.g., by applying tag co-occurrences in documents.

More generally, the same visual technique can be used to display non-tag data, as in a word cloud or a data cloud.

2.3.4 Visual appearance

Tag clouds are typically represented using inline HTML elements. The tags can appear in alphabetical order, in a random order, they can be sorted by weight, and so on. Sometimes, further visual properties are manipulated in addition to font size, such as the font color, intensity, or weight. Most popular is a rectangular tag arrangement with alphabetical sorting in a sequential line-by-line layout. The decision for an optimal layout should be driven by the expected user goals. Some prefer to cluster the tags semantically so that similar tags will appear near each other. Heuristics can be used to reduce the size of the tag cloud whether or not the purpose is to cluster the tags. There are ways of representing the word cloud they are: Data Clouds, Text Clouds and Collocate Clouds[11].

2.3.4.1 Data Cloud

A data cloud or cloud data is a data display which uses font size and/or color to indicate numerical values. It is similar to a tag cloud but instead of word count, displays data such as population or stock market prices.

2.3.4.2 Text Cloud

A text cloud or word cloud is a visualization of word frequency in a given text as a weighted list. The technique has recently been popularly used to visualize the topical content of political speeches.

2.3.4.3 Collocate Clouds

Extending the principles of a text cloud, a collocate cloud provides a more focused view of a document or corpus. Instead of summarizing an entire document, the collocate cloud examines the usage of a particular word. The resulting cloud contains the words which are often used in conjunction with the search word. These collocates are formatted to show frequency (as size) as well as collocational strength (as brightness). This provides interactive ways to browse and explore language.

2.4 Natural Language Toolkit

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet[1], along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries[4], and an active discussion forum.

The NLTK project began when Steven Bird was teaching CIS-530 at the University of Pennsylvania in 2001, and hired his star student, Edward Loper, from the previous offering of the course to be the teaching assistant (TA)[5]. They agreed a plan for developing software infrastructure for NLP teaching that could be easily maintained over time. Edward wrote up the plan, and both began work on it right away.

NLTK is open source software. The source code is distributed under the terms of the Apache License Version 2.0. The documentation is distributed under the terms of the Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 United States license. The corpora are distributed under various licenses, as documented in their respective README files.

Thanks to a hands-on guide introducing programming fundamentals alongside topics in computational linguistics, plus comprehensive API documentation, NLTK is suitable for linguists, engineers, students, educators, researchers, and industry users alike[8]. NLTK is available for Windows, Mac OS X, and Linux. Best of all, NLTK is a free, open source, community-driven project.

NLTK has been called “a wonderful tool for teaching, and working in, computational linguistics using Python,” and “an amazing library to play with natural language.”

Natural Language Processing with Python provides a practical introduction to programming for language processing[6]. Written by the creators of NLTK, it guides the reader through the fundamentals of writing Python programs, working with corpora, categorizing text, analyzing linguistic structure, and more.

Some simple things you can do with NLTK

- Tokenize and tag some text
- Display a parse tree
- Identify named entities

2.5 Unidecode

It often happens that you have text data in Unicode, but you need to represent it in ASCII. For example when integrating with legacy code that doesn’t support Unicode, or for ease of entry of non-Roman names on a US keyboard, or when constructing ASCII machine identifiers from human-readable Unicode strings that should still be somewhat intelligible (a popular example of this is when making an URL slug from an article title).

In most of these examples you could represent Unicode characters as ??? or \15BA\15A0\1610, to mention two extreme cases. But that’s nearly useless to someone who actually wants to read what the text says.

What Unidecode provides is a middle road: function unidecode() takes Unicode data and tries to represent it in ASCII characters (i.e., the universally displayable characters between 0x00 and 0x7F), where the compromises taken when mapping between two character sets are chosen to be near what a human with a US keyboard would choose.

The quality of resulting ASCII representation varies. For languages of western origin it should be between perfect and good. On the other hand transliteration (i.e., conveying, in Roman letters,

the pronunciation expressed by the text in some other writing system) of languages like Chinese, Japanese or Korean is a very complex issue and this library does not even attempt to address it. It draws the line at context-free character-by-character mapping. So a good rule of thumb is that the further the script you are transliterating is from Latin alphabet, the worse the transliteration will be.

This module generally produces better results than simply stripping accents from characters (which can be done in Python with built-in functions). It is based on hand-tuned character mappings that for example also contain ASCII approximations for symbols and non-Latin alphabets.

2.6 TextBlob

TextBlob is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging[8], noun phrase extraction, sentiment analysis, classification, translation, and more.

Features

- Noun phrase extraction
- Part-of-speech tagging
- Sentiment analysis
- Classification (Naive Bayes, Decision Tree)
- Language translation and detection powered by Google Translate
- Tokenization (splitting text into words and sentences)
- Word and phrase frequencies
- Parsing
- n-grams
- Word inflection (pluralization and singularization) and lemmatization
- Spelling correction
- Add new models or languages through extensions
- WordNet integration.

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

To find how people feel about a particular topic we have to devote enormous budgets and countless man-hours to conducting surveys and cold calling. This costs a lot of precious time and money[2]. The sample size for such methods is also relatively low consisting of a few hundred people.

These types of analysis usually are done by some organizations, or news reporters etc. Since people from different regions think or feel different from those of other regions, to get people opinions or sentiments at every region is very laborious and costly work.

The most important thing is that people aren't usually truthful in person which means his/her's opinion fake[9]. People fear to give truthful opinions sometimes to avoid conflict with others because their opinion or thinking is different from other group of people, so asking their opinion in person for sentiment analysis, can make this analysis go wrong.

3.1.1 DISADVANTAGES OF EXISTING SYSTEM

- This system is very costly because we need numerous workers to be deployed in every region to collect people's opinion or sentiment.
- It costs countless man-hours, collecting sentiments from every person will take countless hours.
- Cold calling is also not worth it, it takes many hours and it is very costly.
- Even after collecting number of opinions or sentiments from people, there can be human errors in analysis of these opinions.
- Taking opinions from people in person cannot be assured as truthful opinion.

3.2 PROPOSED SYSTEM

By using social media sites like Twitter where millions of people ranging from regular users to celebrities, company representatives, politicians, and even country presidents, we can obtain a large sample size that can be used for analytics[10]. The analysis process takes a few minutes to complete hence saving time and money. Also, the entire process can be carried out at the comfort of one's home.

This analysis process is very cheap because we are collecting opinions or sentiments through tweets which is very easy to collect. Twitter allows its users to collect tweets, to collect tweets from Twitter we make use of Twitter API Twython and by executing code in python we can collect approx 10,000 tweets in just 30 min which means we have collected 10,000 opinions just by giving a query[1].

These collected tweets are then filtered, the filtration process requires python libraries. The filtration on the basis of retrieving the meaningful parts of the tweet and removing of unnecessary text from the tweet such as url tags, @tags, hashtags, emoticons,trailing whitespaces and newlines is done.

Then final step of this process is to take in the filtered tweets and perform analysis and generate word cloud, group bar graph or table etc to visualize the results. This process requires python library called plotly to generate word clouds, group bar graphs or tables.

3.2.1 ADVANTAGES OF PROPOSED SYSTEM

- Easy to collect people's sentiment, requires very less man-hours.
- This process is completely free, collecting tweets from Twitter is free and for analysis we use open source python libraries.
- This process requires does not need any numerous number of people instead it can be done by a group of five people within one day.

3.3 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.Three key considerations involved in the feasibility analysis are

3.3.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

3.3.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

3.3.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.4 Time and Cost Estimation

COCOMO, Constructive Cost Model is Static Single Variable Model. Barry Boehm introduced this COCOMO Models. COCOMO is only one of the most open and well-documented cost estimation models.COCOMO can be applied to the following software project's categories.

3.4.1 Organic Mode

These projects are simple and have small team size. The team has a good application experience Work to a set of less than rigid requirements. A thermal analysis program developed for heat transfer is a good example.

3.4.2 Semi-Detached Mode

These are intermediate in size and complexity. Here the team has mixed experience to meet a mix of rigid and less than rigid requirements. A transaction processing system with fixed requirements for terminal hardware and database software is an example of this.A transaction processing system with fixed requirements for terminal hardware and database software is an example of this.

3.4.3 Embedded Mode

Software projects that must be developed within a set of tight hardware, software,and operational constraints. For example, flight control software for aircraft.The basic COCOMO equations take the form

$$E=a_b(KLOC)^{b}_{b}$$

$$D=c_b(E)^{d}_{b}$$

$$P=E/D$$

where E is the effort applied in person-months, D is the development time in chronological months, KLOC is the estimated number of delivered lines of code for the project (expressed in

thousands), and P is the number of people required. The coefficients ab, bb, cb and db are given in the following table.

Table 3.1: Basic COCOMO

Software project	a_b	b_b	c_b	d_b
Organic	**2.4**	**1.05**	**2.5**	**0.38**
Semi-detached	**3.0**	**1.12**	**2.5**	**0.35**
Embedded	**3.6**	**1.20**	**2.5**	**0.32**

Basic COCOMO is good for quick, early, rough order of magnitude estimates of software costs, but it does not account for differences in hardware constraints, personnel quality and experience, use of modern tools and techniques, and other project attributes known to have a significant influence on software costs, which limits its accuracy.

Effort = $a_b (KLOC)^{b}_{b}$

$=3.0(1.84)^{1.12}= 6$

Duration = $c_b(E)^{d}_{b}$

$=2.5(6)^{0.35} = 2$ Months

Number of people = E/D = 6/2 = 3 people

So according to the estimation, it takes 3 employees working for 2 months for the project to complete.If the salary of each employee is 15000 per month, then total cost of the project becomes

COST = 3 * 2 * 15000 = Rs.90, 000

3.5 SYSTEM REQUIREMENTS

3.5.1 HARDWARE REQUIREMENTS

- System : Pentium IV 2.4 GHz. Or Higher
- Hard Disk : 500 GB.
- Ram : 4Gb.

3.5.2 SOFTWARE REQUIREMENTS

- Operating system : Windows XP/7/8.
- Coding Language : Python 3.7
- Data Base : SQLyog
- Tools Used : Anaconda Navigator

4. SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

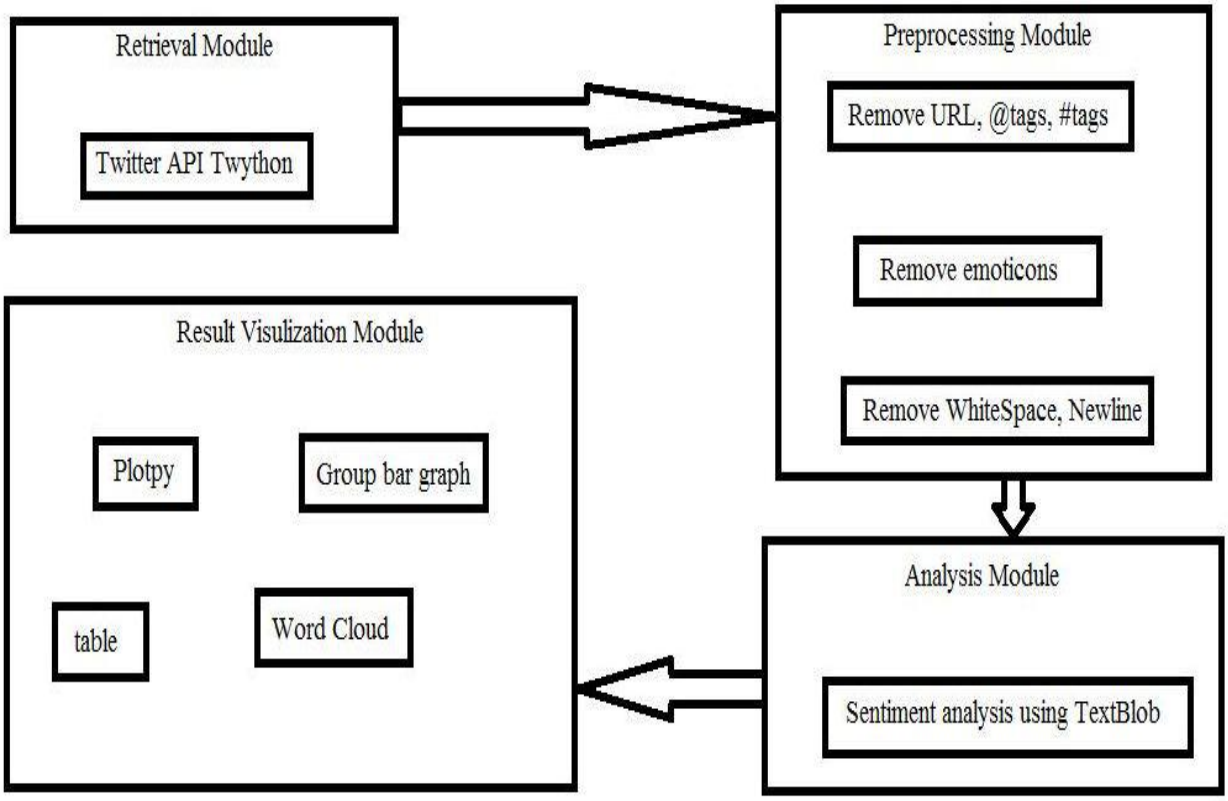

Figure 4.1: Proposed Architecture

4.2 UML DIAGRAMS

4.2.1 USE CASE DIAGRAM

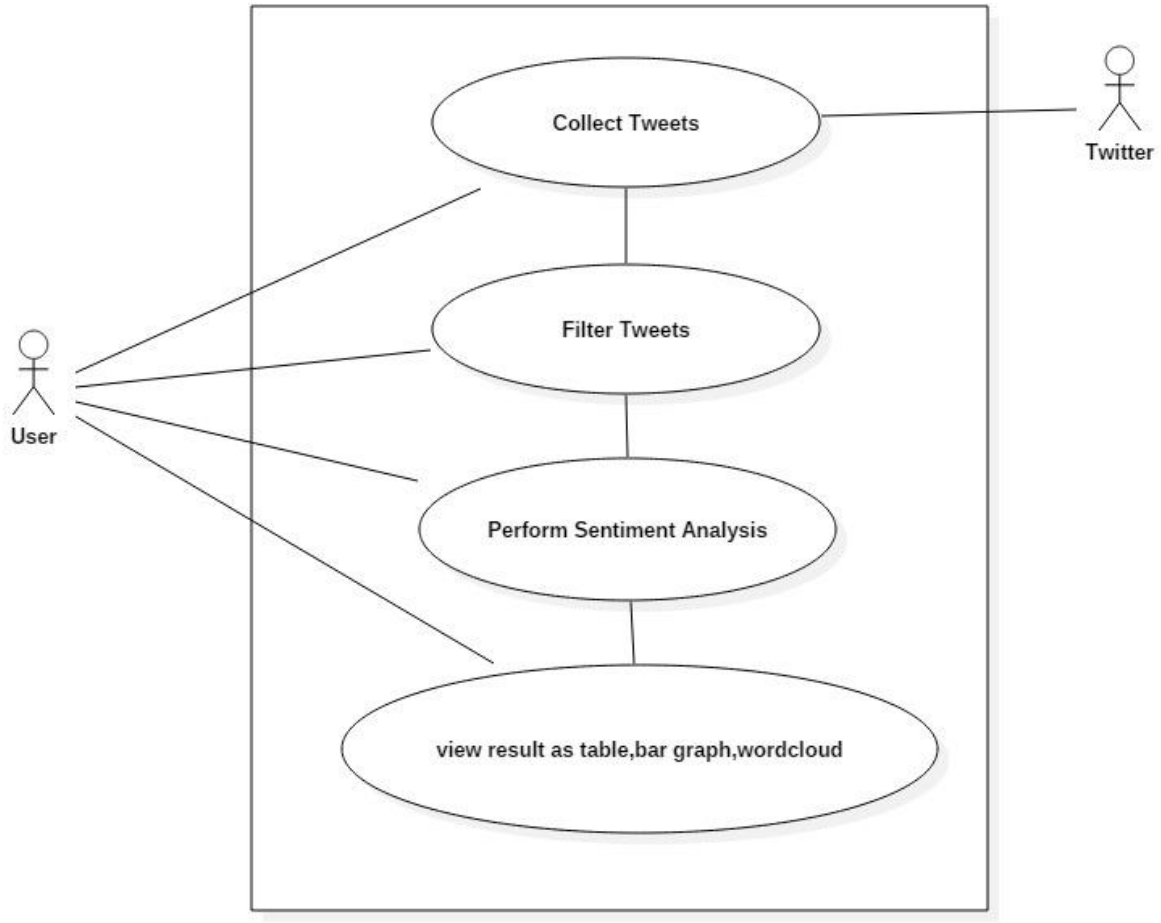

Figure 4.2: Use Case Diagram 1

4.2.2 CLASS DIAGRAM

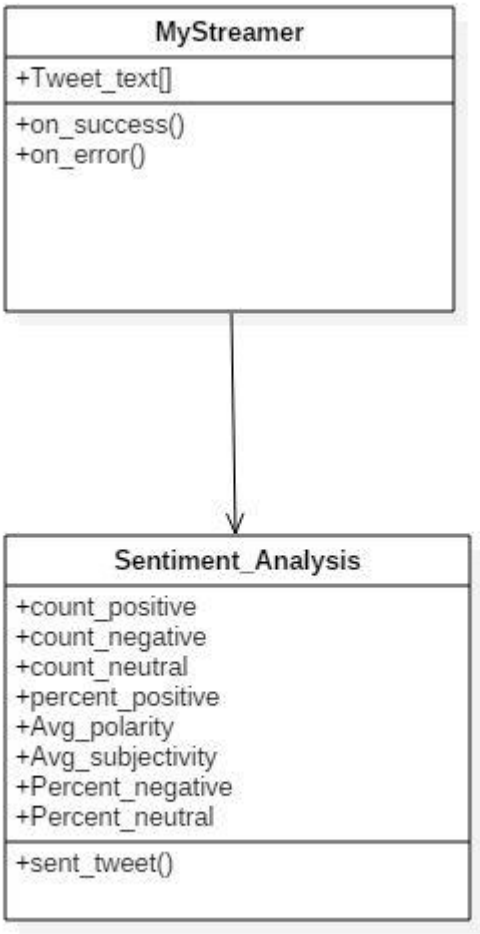

Figure 4.3: Class Diagram

4.2.3 ACTIVITY DIAGRAM

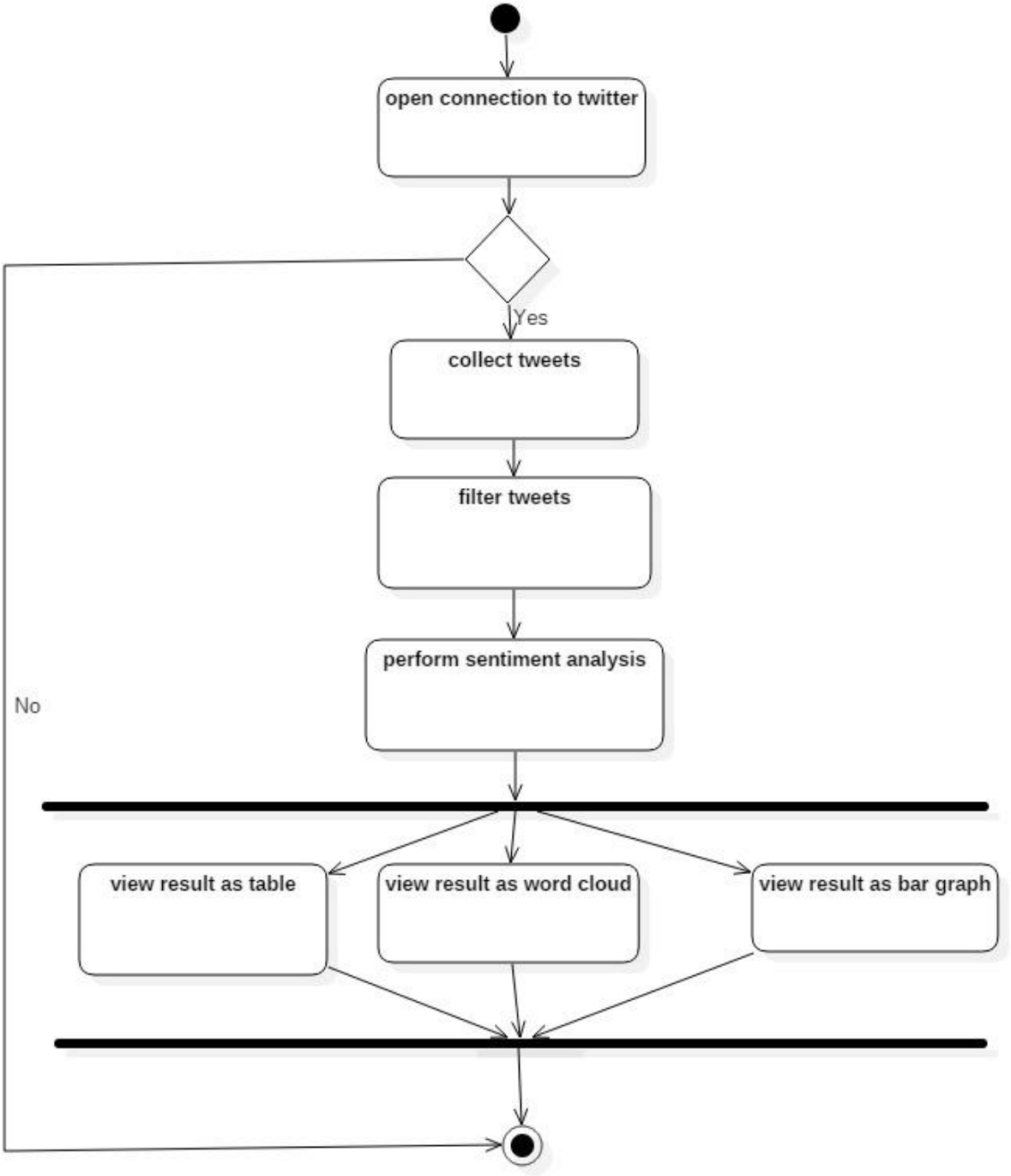

Figure 4.4: Activity Diagram

5. IMPLEMENTATION

5.1 Sentiment Analysis

Sentiment analysis (sometimes known as opinion mining or emotion AI) refers to the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information. Sentiment analysis is widely applied to voice of the customer materials such as reviews and survey responses, online and social media, and health care materials for applications that range from marketing to customer service to clinical medicine.

Generally speaking, sentiment analysis aims to determine the attitude of a speaker, writer, or other subject with respect to some topic or the overall contextual polarity or emotional reaction to a document, interaction, or event[4]. The attitude may be a judgment or evaluation (see appraisal theory), affective state (that is to say, the emotional state of the author or speaker), or the intended emotional communication (that is to say, the emotional effect intended by the author or interlocutor).

5.1.1 Types

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

Early work in that area includes Turney and Pang who applied different methods for detecting the polarity of product reviews and movie reviews respectively. This work is at the document level. One can also classify a document's polarity on a multi-way scale, which was attempted by Pang and Snyder among others: Pang and Lee expanded the basSic task of classifying a movie review as either positive or negative to predict star ratings on either a 3 or a 4 star scale, while Snyder performed an in-depth analysis of restaurant reviews, predicting ratings for various aspects of the given restaurant, such as the food and atmosphere (on a five-star scale). Even though in most statistical classification methods, the neutral class is ignored under the assumption that neutral texts lie near the boundary of the binary classifier, several researchers suggest that, as in every polarity problem, three categories must be identified. Moreover, it can be proven that specific classifiers such as the Max Entropy and the SVMs can benefit from the introduction of a neutral class and improve the overall accuracy of the classification. There are in principle two ways for operating with a neutral class. Either, the algorithm proceeds by first identifying the neutral language, filtering it out and then assessing the rest in terms of positive and negative sentiments, or it builds a three way classification in one step. This second approach often involves estimating a probability distribution over all categories (e.g. naive Bayes classifiers as implemented by Python's NLTK kit). Whether and how to use a neutral class

depends on the nature of the data: if the data is clearly clustered into neutral, negative and positive language, it makes sense to filter the neutral language out and focus on the polarity between positive and negative sentiments. If, in contrast, the data is mostly neutral with small deviations towards positive and negative affect, this strategy would make it harder to clearly distinguish between the two poles.

A different method for determining sentiment is the use of a scaling system whereby words commonly associated with having a negative, neutral or positive sentiment with them are given an associated number on a −10 to +10 scale (most negative up to most positive). This makes it possible to adjust the sentiment of a given term relative to its environment (usually on the level of the sentence). When a piece of unstructured text is analyzed using natural language processing, each concept in the specified environment is given a score based on the way sentiment words relate to the concept and its associated score. This allows movement to a more sophisticated understanding of sentiment, because it is now possible to adjust the sentiment value of a concept relative to modifications that may surround it. Words, for example, that intensify, relax or negate the sentiment expressed by the concept can affect its score. Alternatively, texts can be given a positive and negative sentiment strength score if the goal is to determine the sentiment in a text rather than the overall polarity and strength of the text.

5.1.1.1 Subjectivity/objectivity identification

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

5.1.1.2 Feature/aspect-based

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

5.1.2 Methods and features

Existing approaches to sentiment analysis can be grouped into three main categories: knowledge-based techniques, statistical methods, and hybrid approaches. Knowledge-based techniques classify text by affect categories based on the presence of unambiguous affect words such as happy, sad, afraid, and bored. Some knowledge bases not only list obvious affect words, but also assign arbitrary words a probable "affinity" to particular emotions. Statistical methods leverage on elements from machine learning such as latent semantic analysis, support vector machines, "bag of words" and Semantic Orientation — Pointwise Mutual Information (See Peter Turney's work in this area). More sophisticated methods try to detect the holder of a sentiment (i.e., the person who maintains that affective state) and the target (i.e., the entity about which the affect is felt)[7]. To mine the opinion in context and get the feature which has been opinionated, the grammatical relationships of words are used. Grammatical dependency relations are obtained by deep parsing of the text. Hybrid approaches leverage on both machine learning and elements from knowledge representation such as ontologies and semantic networks in order to detect semantics that are expressed in a subtle manner, e.g., through the analysis of concepts that do not explicitly convey relevant information, but which are implicitly linked to other concepts that do so.

Open source software tools deploy machine learning, statistics, and natural language processing techniques to automate sentiment analysis on large collections of texts, including web pages, online news, internet discussion groups, online reviews, web blogs, and social media. Knowledge-based systems, on the other hand, make use of publicly available resources, to extract the semantic and affective information associated with natural language concepts. Sentiment analysis can also be performed on visual content, i.e., images and videos. One of the first approach in this direction is SentiBank utilizing an adjective noun pair representation of visual content.

A human analysis component is required in sentiment analysis, as automated systems are not able to analyze historical tendencies of the individual commenter, or the platform and are often classified incorrectly in their expressed sentiment. Automation impacts approximately 23% of comments that are correctly classified by humans. However, also humans often disagree, and it is argued that the inter-human agreement provides an upper bound that automated sentiment classifiers can eventually reach.

Sometimes, the structure of sentiments and topics is fairly complex. Also, the problem of sentiment analysis is non-monotonic in respect to sentence extension and stop-word substitution (compare THEY would not let my dog stay in this hotel vs I would not let my dog stay in this hotel). To address this issue a number of rule-based and reasoning-based approaches have been applied to sentiment analysis, including defeasible logic programming. Also, there is a number of

tree traversal rules applied to syntactic parse tree to extract the topicality of sentiment in open domain setting.

5.1.3 Evaluation

The accuracy of a sentiment analysis system is, in principle, how well it agrees with human judgments. This is usually measured by precision and recall. However, according to research human raters typically agree 79% of the time (see Inter-rater reliability).

Thus, a 70% accurate program is doing nearly as well as humans, even though such accuracy may not sound impressive. If a program were "right" 100% of the time, humans would still disagree with it about 20% of the time, since they disagree that much about any answer . More sophisticated measures can be applied, but evaluation of sentiment analysis systems remains a complex matter. For sentiment analysis tasks returning a scale rather than a binary judgement, correlation is a better measure than precision because it takes into account how close the predicted value is to the target value.

5.1.4 Web 2.0

The rise of social media such as blogs and social networks has fueled interest in sentiment analysis. With the proliferation of reviews, ratings, recommendations and other forms of online expression, online opinion has turned into a kind of virtual currency for businesses looking to market their products, identify new opportunities and manage their reputations. As businesses look to automate the process of filtering out the noise, understanding the conversations, identifying the relevant content and actioning it appropriately, many are now looking to the field of sentiment analysis. Further complicating the matter, is the rise of anonymous social media platforms such as 4chan and Reddit. If web 2.0 was all about democratizing publishing, then the next stage of the web may well be based on democratizing data mining of all the content that is getting published.

One step towards this aim is accomplished in research. Several research teams in universities around the world currently focus on understanding the dynamics of sentiment in e-communities through sentiment analysis. The CyberEmotions project, for instance, recently identified the role of negative emotions in driving social networks discussions.

The problem is that most sentiment analysis algorithms use simple terms to express sentiment about a product or service. However, cultural factors, linguistic nuances and differing contexts make it extremely difficult to turn a string of written text into a simple pro or con sentiment. The fact that humans often disagree on the sentiment of text illustrates how big a task it is for computers to get this right. The shorter the string of text, the harder it becomes.

Even though short text strings might be a problem, sentiment analysis within microblogging has shown that Twitter can be seen as a valid online indicator of political sentiment. [1]Tweets'

political sentiment demonstrates close correspondence to parties' and politicians' political positions, indicating that the content of Twitter messages plausibly reflects the offline political landscape.

5.2 Source Code

5.2.1 Retrieving tweets

```
from twython import TwythonStreamer

from unidecode import unidecode

import sys


tweet_text = []

num=20


CONSUMER_KEY = '1XlO7bDLYARB1tTjZ4cspV1Q2'

CONSUMER_SECRET = 'othIx7WctmMQ38NEwXja2hKGsamKpVeohvAGgQfuxXoZlyFoZ3'

ACCESS_TOKEN = '846796640811933697-ttfVB5ekh6zLLh73DdUjFWy5ZpEjYWH'

ACCESS_TOKEN_SECRET = 'UJqdU8ZrjR7FmcsJlReJMmI2KcMKFyVxz4d5eLk0F33qA'


tweet_locations = {

   'dfw': "-97.48,32.31,-96.49,33.26,",

   'jfk': "-74,40,-73,41",

   'sfo': "-122.75,36.8,-121.75,37.8",

   'lax': "-118.43,33.73,-117.93,34.21",

   'ord': "-88.47,41.46,-87.16,42.35",

   'dca': "-77.26,38.73,-76.87,39.10",

   'atl': "-84.54,33.62,-84.18,33.96"
```

```
}


class MyStreamer(TwythonStreamer):
    # overriding
    def on_success(self, data):
        if data['lang'] == 'en':


            # filter tweet
            twitter_words                                                                    =
[u'http',u'https',u'RT','https','http',u's://t.co','rt','amp',u'amp',u'u2026',u'u2019']
            text=unidecode(data['text'])
            for word in twitter_words:
                text=text.replace(word,'')
            text=text.strip()
            text=text.replace('\n','')  #to remove newline
            tweet_text.append(text)


        if len(tweet_text) >= num:
            self.disconnect()


    # overriding
    def on_error(self, status_code, data):
        print(status_code, data)
        self.disconnect()
```

```
# download
keyword = ‘Trump’
location = 'sfo'
bounds = tweet_locations[location]
stream = MyStreamer(CONSUMER_KEY, CONSUMER_SECRET, ACCESS_TOKEN, ACCESS_TOKEN_SECRET)
stream.statuses.filter(track=keyword,locations=bounds)
f=open(Trump/sfo.txt','w')
for i in range(num):
    f.write(tweet_text[i]+'\n')
f.close()
print(len(tweet_text))
```

5.2.2 Sentiment analysis on tweets

```
from textblob import TextBlob
class Sem_Analysis:
    count_positive = 0
    count_negative = 0
    count_neutral = 0
    Average_polarity = 0
    Average_subjectivity = 0
    percent_positive=''
    percent_negative=''
    percent_neutral=''
    def sent_tweet(self,fname):
```

```
count_positive = 0
count_negative = 0
count_neutral = 0
Average_polarity = 0
Average_subjectivity = 0
total_tweets=0
lst1=[]
lst2=[]
f = open(fname,'r')
f1= f.readlines()
for line in f1:
    sent1 = TextBlob(line)
    tb1 = sent1.sentiment.polarity
    lst1.append(tb1)
    tb2 = sent1.sentiment.subjectivity
    lst2.append(tb2)
for item in lst1:
    if item>0:
        count_positive += 1
    elif item<0:
        count_negative +=1
    else:
        count_neutral +=1
self.count_positive = count_positive
self.count_negative = count_negative
```

```
        self.count_neutral = count_neutral
        total_tweets = count_positive + count_negative + count_neutral
        self.percent_positive = str(round(count_positive/total_tweets*100 ,4)) + "%"
        self.percent_negative = str(round(count_negative/total_tweets*100 ,4)) +"%"
        self.percent_neutral = str(round(count_neutral/total_tweets*100 ,4)) +"%"

        Average_polarity = sum(lst1)/len(lst1)
        self.Average_polarity = round(Average_polarity,4)
        Average_subjectivity = sum(lst2)/len(lst2)
        self.Average_subjectivity = round(Average_subjectivity,4)
```

5.2.3 Initialize plotly library in offline mode

```
import plotly
plotly.offline.init_notebook_mode()
```

5.2.4 Creating table

```
import plotly as py
import plotly.figure_factory as ff
import pandas as pd

trump = Sem_Analysis()
trump.sent_tweet("Trump/sfo.txt")
hillary = Sem_Analysis()
hillary.sent_tweet("Hillary/sfo.txt")
```

```
bernie = Sem_Analysis()

bernie.sent_tweet("tweetsbernie.txt")

data_matrix = [['Name', 'Number of<br>Positive<br>comments', 'Number of<br>Neutral<br>comments','Number of<br>Negative<br>comments',

              '% Positive ','% Neutral','% Negative ',

              'Average<br>Polarity','Average<br>Subjectivity'],

             ['Trump', trump.count_positive, trump.count_neutral,trump.count_negative,trump.percent_positive,

              trump.percent_neutral,trump.percent_negative,

              trump.Average_polarity, trump.Average_subjectivity],

             ['Hillary', hillary.count_positive, hillary.count_neutral, hillary.count_negative, hillary.percent_positive,

              hillary.percent_neutral,hillary.percent_negative,

              hillary.Average_polarity, hillary.Average_subjectivity],

             ['bernie', bernie.count_positive, bernie.count_neutral, bernie.count_negative,bernie.percent_positive,

              bernie.percent_neutral,bernie.percent_negative, bernie.Average_polarity, bernie.Average_subjectivity]

             ]


table = ff.create_table(data_matrix)

table.layout.width=1000 #width in pixels

py.offline.iplot(table, filename='jupyter/table')
```

5.2.5 Creating Bar Graph

```
import plotly as py
import plotly.graph_objs as go

trump = Sem_Analysis()
trump.sent_tweet("Trump/sfo.txt")
hillary = Sem_Analysis()
hillary.sent_tweet("tweetshillary.txt")
bernie = Sem_Analysis()
bernie.sent_tweet("tweetsbernie.txt")

trace1 = go.Bar(
    x=['Positive', 'Neutral', 'Negative'],
   y=[trump.count_positive, trump.count_neutral,trump.count_negative],
    name='Trump'
)
trace2 = go.Bar(
    x=['Positive', 'Neutral', 'Negative'],
    y=[hillary.count_positive, hillary.count_neutral, hillary.count_negative],
    name='Hillary'
)
trace3 = go.Bar(
    x=['Positive', 'Neutral', 'Negative'],
    y=[bernie.count_positive, bernie.count_neutral, bernie.count_negative],
    name='Bernie'
```

```
)

data = [trace1, trace2,trace3]
layout = go.Layout(
    barmode='group',
    bargroupgap=0.1,
    title='Comparing Sentiments in USA',
    xaxis=dict(
        title='Sentiment',
        titlefont=dict(
            size=16,
            color='rgb(107, 107, 107)'
        )),
    yaxis=dict(
        title='Number of tweets<br>',
        titlefont=dict(
            size=16,
            color='rgb(107, 107, 107)'
        )),
)

fig = go.Figure(data=data, layout=layout)
py.offline.iplot(fig, filename='grouped-bar-graph')
```

5.2.6 Creating Wordcloud

```
#wordcloud
from os import path
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import nltk
stopwords = nltk.corpus.stopwords.words('english')
from nltk.stem.snowball import SnowballStemmer
ss = SnowballStemmer("english")
text = open('tweetshillary.txt').read()
text2 = ''
for word in text.split():
    if len(word) == 1 or word in stopwords:
        continue
    text2 += ' {}'.format(word)
ss.stem(text2)

wordcloud = WordCloud(max_font_size=150,width=800,height=400).generate(text)

plt.figure(figsize=(20,10))
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```

6. TESTING

6.1 Purpose

The purpose of testing is to assess product quality. It helps to strengthen and stabilize the architecture early in the development cycle. We can verify through testing, the various interactions, integration of components and the requirements which were implemented. It provides timely feedback to resolve the quality issues, in a timely and cost effective manner. The test work flow involves the following:

1. Verifying the interactions of components.

2. Verifying the proper integration of components.

3. Verifying that all requirements have been implemented correctly.

4. Identifying and ensuring that all discovered defects are addressed before the software is deployed.

6.2 Quality

The common usage of the term quality refers to a number of things: principally it means the absence of defects, but more importantly, a fitness for a desired purpose. The ultimate goal of testing is to assess the quality of the end product. Quality assessments often consider process quality and organizational factors as well as direct product quality.

6.3 Product Quality

The role of testing is not to assure quality, but to assess it, and to provide timely feedback so that quality issues can be resolved in a timely and cost-effective manner.

6.4 Testing in the iterative life cycle

Testing is not a single activity, nor is it a phase in the project during which we assess quality. If developers are to obtain timely feedback on evolving product quality,testing must occur throughout the life cycle, we can test the broad functionality of early prototypes, we can test the stability, coverage and performance of the architecture while there is still an opportunity to fix it; and we can test the final product to assess its readiness for delivery to customers.

6.5 Dimensions of testing

To assess product quality, different kinds of tests, each one with a different focus,are needed. These tests can be categorized by several dimensions:

1. Quality dimension: The major quality characteristic or attribute that is the focus of test.

2. Stage of testing: The point in the life cycle at which the test, usually limited to a single quality dimension.

3. Type of testing: The specific test objective for an individual test, usually limited to a single quality dimension.

6.6 Stages of testing

Testing is not a single activity, executed all at once. Testing is executed against different types of targets in different stages of the software development. Test stages progress from testing small elements of the system, such as components (unit testing), to testing completed systems.

6.7 Test Cases

Test Case ID: 1

Test Case Description: test case to check if sentiment analysis of specified tweet as expected.

Step	Description	Expected Result	Actual Result	Status
1	Tweet with positive connotation	Positive sentiment	Positive Sentiment	Success
2	Tweets with negative connotation	Negative Sentiment	Negative Sentiment	Success
3	Tweets with neutral connotation	Neutral Sentiment	Neutral Sentiment	Success

Table 6.1: Test Case

7. OUTPUT SCREENS

Name	Number of Positive comments	Number of Neutral comments	Number of Negative comments	% Positive	% Neutral	% Negative	Average Polarity	Average Subjectivity
Trump	17812	7853	24335	35.624	15.706	48.67	-0.0073	0.278
Hillary	23716	12475	13809	47.432	24.95	27.618	0.0482	0.2809
Bernie	20748	5741	23511	41.496	11.48	47.022	0.0282	0.3138

Table 7.1: visualize result in tabular form

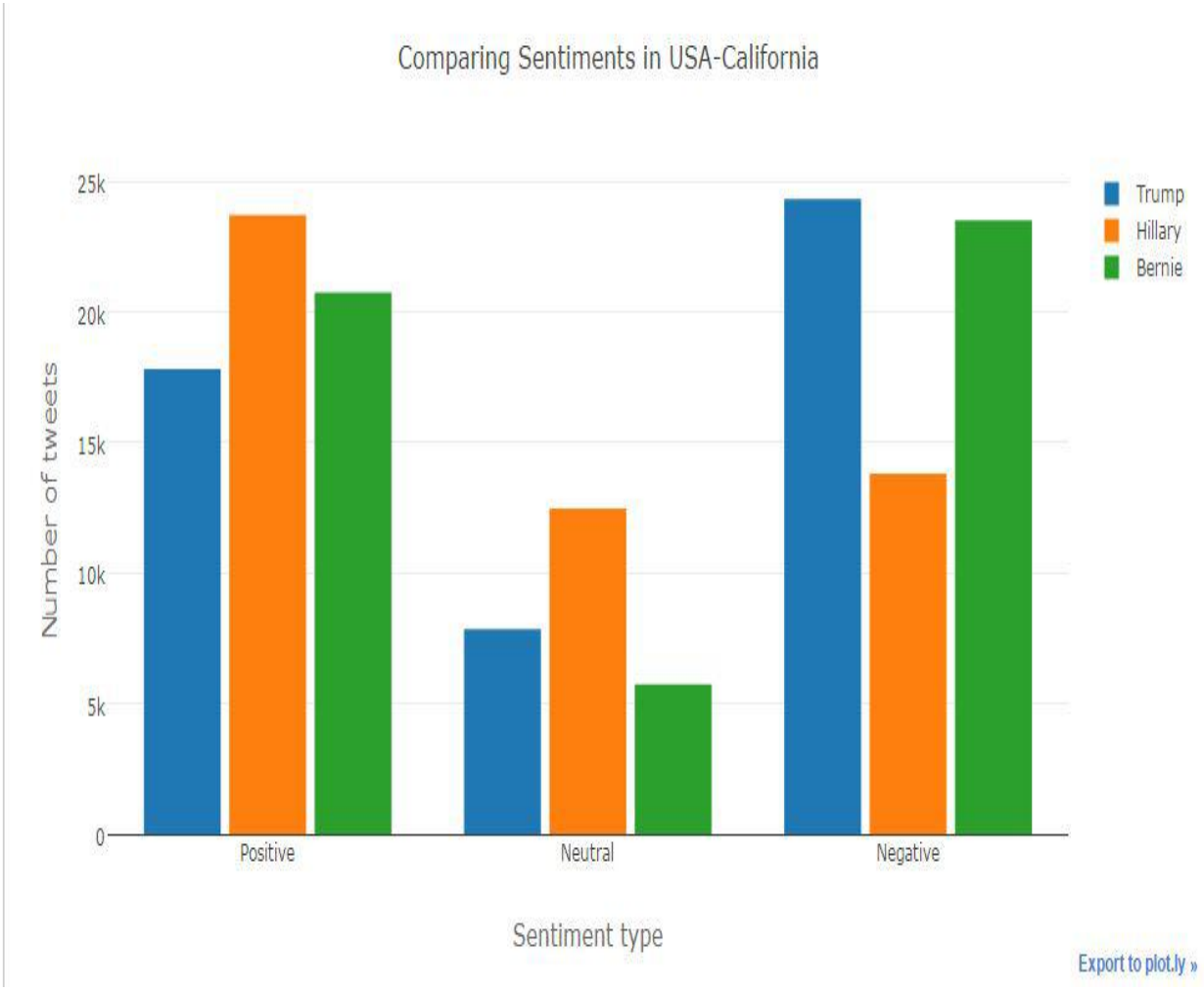

Figure 7.1: Group Bar Graph

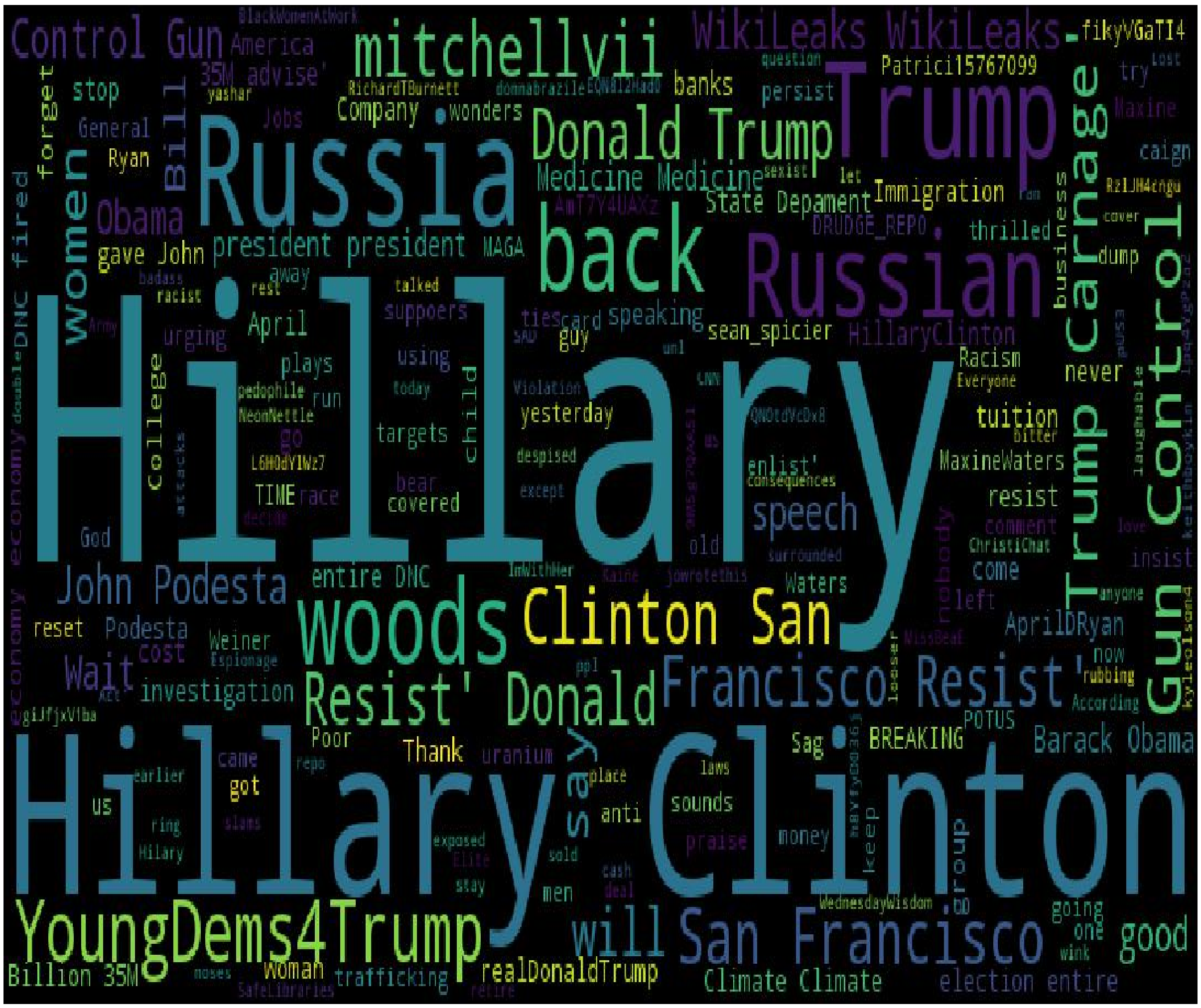

Figure 7.2: Word Cloud

CONCLUSION

Microblogging nowadays became one of the major types of the communication. The large amount of information contained in microblogging web-sites makes them an attractive source of data for opinion mining and sentiment analysis. In our project we have used a lexicon based sentiment analyzer that classifies tweets into positive, neutral and negative sentiments. By using the analyzer on tweets that were collected during the US 2016 presidential election, the public opinion on the three popular electoral candidates were found on seven different locations across America. Such data can be incredibly helpful for the candidates as they can know the locations wherein public opinion on them is negative and focus on directing their efforts to change such opinions.

FUTURE ENHANCEMENT

The project can further be enhanced by using machine learning techniques for classifying tweets in sentiment analysis which can prove to be more effective than a lexicon based approach. Also, a multilingual based sentiment analysis can be done to obtain public opinion from tweets in different languages.

REFERENCES

[1] J. Kamps, M. Marx, R. J. Mokken, and M. de Rijke, "Using WordNet to measure semantic orientation of adjectives," in Proceedings of LREC, 2004.

[2] R. Agrawal, S. Rajagopalan, R. Srikant, and Y. Xu, "Mining newsgroups using networks arising from social behavior," in Proceedings of WWW, pp. 529–535, 2003.

[3] C. O. Alm, D. Roth, and R. Sproat, "Emotions from text: Machine learning for text-based emotion prediction," in Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), 2005.

[4] A. Aue and M. Gamon, "Automatic identification of sentiment vocabulary: Exploiting low association with known sentiment terms," in Proceedings of the ACL Workshop on Feature Engineering for Machine Learning in Natural Language Processing, 2005.

[5] R. F. Bruce and J. M. Wiebe, "Recognizing subjectivity: A case study in manual tagging," Natural Language Engineering, vol. 5, 1999.

[6] C. Cardie, C. Farina, T. Bruce, and E. Wagner, "Using natural language processing to improve eRulemaking," in Proceedings of Digital Government Research (dg.o), 2006.

[7] P. Chaovalit and L. Zhou, "Movie review mining: A comparison between supervised and unsupervised classification approaches," in Proceedings of the Hawaii International Conference on System Sciences (HICSS), 2005.

[8] P. Chesley, B. Vincent, L. Xu, and R. Srihari, "Using verbs and adjectives to automatically classify blog sentiment," in AAAI Symposium on Computational Approaches to Analysing Weblogs (AAAI-CAAW), pp. 27–29, 2006.

[9] J. G. Conrad and F. Schilder, “Opinion mining in legal blogs,” in Proceedings of the International Conference on Artificial Intelligence and Law (ICAIL), pp. 231–236, New York, NY, USA: ACM, 2007.

[10] L. Dini and G. Mazzini, “Opinion classification through information extraction,” in Proceedings of the Conference on Data Mining Methods and Databases for Engineering, Finance and Other Fields (Data Mining), pp. 299–310, 2002.

[11] A. Esuli and F. Sebastiani, “SentiWordNet: A publicly available lexical resource for opinion mining,” in Proceedings of Language Resources and Evaluation (LREC), 2006.