**1. Introduction**

**Sentiment Analysis is the process of determining whether a piece of writing is positive, negative or neutral. It’s also known as opinion mining, deriving the opinion or attitude of a speaker.** **A widespread use case for this technology is to discover how people feel about a topic.** **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 organisations across the world.** **Sentiment scoring allows a computer to consistently rate the positive or negative assertions that are associated with a document or entity. The scoring of sentiment (sometimes referred to as tone) from a document is a problem that was originally raised in the context of marketing and business intelligence, where being able to measure the public’s reaction to a new marketing campaign (or a corporate scandal) can have a measurable financial impact on the business.**

Humans seemingly have no trouble reading a sentence and mentally scoring the sentiment. We humans use a process of reading and understanding the descriptors placed on the subject of a sentence.

Consider these sentences:

• A horrible pitching performance resulted in another devastating loss.

• Sub-par pitching and superb hitting combined to cost us another close game.

They both have the same basic subject, the loss of a baseball game, but obviously (to you!) the first sentence is contextually much more negative. The keys that humans use to discern this are to focus on the emotive phrases “horrible pitching” and “devastating loss”. The sentiment analysis system does exactly the same thing. It may involve 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 healthcare materials for applications that range from marketing to customer service to clinical medicine.

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". 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 analysed 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. the target of SA is to find opinions, identify the sentiments they express, and then classify their polarity as shown in Fig. 1.

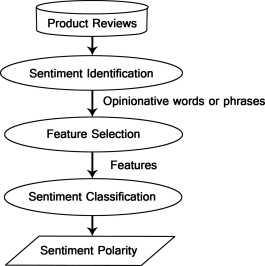


Fig.1 Sentiment analysis process on product reviews

In the past few years, it attracted a great deal of attentions from both academia and industry due to many challenging research problems and a wide range of applications. Opinions are important because whenever we need to make a decision we want to hear other’s opinions. This is not only true for individuals but also true for organizations. However, there was almost no computational study on opinions before the Web because there was little opinionated text available. In the past, when an individual needed to make a decision, he/she typically asked for opinions from friends and families. When an organization wanted to find opinions of the general public about its products and services, it conducted surveys and focus groups. However, with the explosive growth of the social media content on the Web in the past few years, the world has been transformed. People can now post reviews of products at merchant sites and express their views on almost anything in discussion forums and blogs, and at social network sites. Now if one wants to buy a product, one is no longer limited to asking one’s friends and families because there are many user reviews on the Web. For a company, it may no longer need to conduct surveys or focus groups in order to gather consumer opinions about its products and those of its competitors because there is a plenty of such information publicly available. However, finding opinion sites and monitoring them on the Web can still be a formidable task because there are a large number of diverse sites, and each site may also have a huge volume of opinionated text.

In many cases, opinions are hidden in long forum posts and blogs. It is difficult for a human reader to find relevant sites, extract related sentences with opinions, read them, summarize them, and organize them into usable forms. Automated opinion discovery and summarization systems are thus needed.

**1.1 The Problem of Sentiment Analysis**

The research in the field started with sentiment and subjectivity classification, which treated the problem as a text classification problem. Sentiment classification classifies whether an opinionated document (e.g., product reviews) or sentence expresses a positive or negative opinion. Subjectivity classification determines whether a sentence is subjective or objective. Many real-life applications, however, require more detailed analysis because the user often wants to know what the opinions have been expressed on. For example, from the review of a product, one wants to know what features of the product have been praised and criticized by consumers.

Let us use the following review segment on iPhone as an example to introduce the general problem (a number is associated with each sentence for easy reference)

“(1) I bought an iPhone 2 days ago. (2) It was such a nice phone. (3) The touch screen was really cool. (4) The voice quality was clear too. (5) However, my mother was mad with me as I did not tell her before I bought it. (6) She also thought the phone was too expensive, and wanted me to return it to the shop. … ”

The question is: what we want to extract from this review? The first thing that we may notice is that there are several opinions in this review. Sentences (2), (3) and (4) express three positive opinions, while sentences (5) and (6) express negative opinions. Then we also notice that the opinions all have some targets on which they are expressed. The opinion in sentence (2) is on iPhone as a whole, and the opinions in sentences (3) and (4) are on the “touch screen” and “voice quality” features of iPhone respectively. The opinion in sentence (6) is on the price of iPhone, but the opinion/emotion in sentence (5) is on “me”, not iPhone. This is an important point. In an application, the user may be interested in opinions on certain targets, but not on all (e.g., unlikely on “me”). Finally, we may also notice the sources or holders of opinions. The source or holder of the opinions in sentences (2), (3) and (4) is the author of the review (“I”), but in sentences (5) and (6) it is “my mother”. With this example in mind, we can define sentiment analysis or opinion mining [1, 4]. We start with the opinion target.

Object and feature: In general, opinions can be expressed on any target entity, e.g., a product, a service, an individual, an organization, or an event. We use the term object to denote the target entity that has been commented on. An object can have a set of components (or parts) and a set of attributes (or properties) [1, 4], which we collectively call the features of the object.

A particular brand of cellular phone is an object. It has a set of components (e.g., battery and screen), and also a set of attributes (e.g., voice quality and size), which are all called features. An opinion can be expressed on any feature of the object and also on the object itself. For example, in “I like iPhone. It has a great touch screen”, the first sentence expresses a positive opinion on “iPhone” itself, and the second sentence expresses a positive opinion on its “touch screen” feature.

Opinion holder: The holder of an opinion is the person or organization that expresses the opinion.

In the case of product reviews and blogs, opinion holders are usually the authors of the posts. Opinion holders are more important in news articles because they often explicitly state the person or organization that holds a particular opinion.

Opinion and orientation: An opinion on a feature f (or object o) is a positive or negative view or appraisal on f (or o) from an opinion holder. Positive and negative are called opinion orientations.

With these concepts in mind, we can define a model of an object, a model of an opinionated text, and the mining objective, which are collectively called the feature-based sentiment analysis model.

Model of an object: An object o is represented with a finite set of features, F = {f1, f2, …, fn}, which includes the object itself as a special feature. Each feature fi ∈ F can be expressed with any one of a finite set of words or phrases Wi ={wi1, wi2, …, wim}, which are synonyms of the feature.

Model of an opinionated document: A opinionated document d contains opinions on a set of objects {o1, o2, …, or} from a set of opinion holders {h1, h2, …, hp}. The opinions on each object oj are expressed on a subset Fj of features of oj. An opinion can be either one of the following two types:

Direct opinion: A direct opinion is a quintuple (oj, fjk, ooijkl, hi, tl), where oj is an object, fjk is a feature of the object oj, ooijkl is the orientation of the opinion on feature fjk of object oj, hi is the opinion holder and tl is the time when the opinion is expressed by hi. The opinion orientation ooijkl can be positive, negative or neutral.

Comparative opinion: A comparative opinion expresses a preference relation of two or more objects based on some of their shared features. It is usually conveyed using the comparative or superlative form of an adjective or adverb, e.g., “Coke tastes better than Pepsi”.

Objective of sentiment analysis on direct opinions: Given an opinionated document *d*,

1. Discover all opinion quintuples (*oj*, *fjk*, *ooijkl*, *hi*, *tl*) in *d*, and

2. Identify all synonyms (*Wjk*) of each feature *fjk* in *d*.

In practice not all five pieces of information in the quintuple need to be discovered for every application because some of them may be known or not needed. For example, in the context of online forums, the time when a post is submitted and the opinion holder are all known as the site typically displays such information.

Applications: A simple way to use the results is to produce a *feature-based summary* of opinions on an object or multiple competing objects [1, 4]. Figure 1 shows the summary of opinions on two competing cellular phones along different feature dimensions. In the figure, each bar above the *X*-axis in the middle shows the number of positive opinions on a feature (given at the top), and the bar below the *X*-axis shows the number of negative opinions on the same feature. We can clearly see how consumers view different features of each product. “PHONE” represents the phone itself. Phone 1 is clearly a better product.

**1.2 Technical Challenges**

The objective of opinion mining gives us a good clue of the main tasks involved and technical challenges. None of the problems is solved. Let us use a more complex example blog to discuss them:

“(1) *Yesterday, I bought a Nokia phone and my girlfriend bought a moto phone.* (2) *We called each other when we got home.* (3) *The voice on my phone was not clear.* (4) *The camera was good.* (5) *My girlfriend said the sound of her phone was clear.* (6) *I wanted a phone with good voice quality.* (7) *So I was satisfied and returned the phone to BestBuy yesterday*.”

Object identification: The objects to be discovered in this blog are “moto” (Motorola) and “Nokia”. This problem is important because without knowing the object on which an opinion has been expressed, the opinion is of little use. The issue is similar to the classic *named entity recognition* problem. However, there is a difference. In a typical opinion mining application, the user wants to find opinions on some competing objects (e.g., products). The system thus needs to separate relevant objects and irrelevant objects. For example, “BestBuy” is not a competing product name, but the name of a shop.

Feature extraction and synonym grouping: In the example above, the phone features are “voice”, “sound”, and “camera”. Although there were attempts to solve this problem, it remains to be a major challenge. Current research mainly finds nouns and noun phrases. Although the recall may be good, the precision can be low. Furthermore, verb features are common as well but harder to identify. To produce a summary similar to the one in Figure 1, we also need to group synonym features as people often use different words or phrases to describe the same feature (e.g., “voice” and “sound” refer to the same feature in the above example).

Opinion orientation classification: This task determines whether there is opinion on a feature in a sentence, and if so, whether it is positive or negative. Existing approaches are based on supervised and unsupervised methods. One of the key issues is to identify opinion words and phrases (e.g., *good*, *bad*, *poor*, *great*), which are instrumental to sentiment analysis. The problem is that there are seemly an unlimited number of expressions that people use to express opinions, and in different domains they can be significantly different. Even in the same domain, the same word may indicate different opinions in different contexts . For example, in the sentence, “*The battery life is long*” “long” indicates a positive opinion on the “battery life” feature. However, in the sentence, “*This camera takes a long time to focus*”, “long” indicates a negative opinion. Also, sentence (6) in our example blog above seemingly expresses a positive opinion, but it does not. There are still many problems that need to be solved.

Integration: Integrating the about tasks is also complex because we need to match the five pieces of information in the quintuple. That is, the opinion *ooijkl* must be given by opinion holder *hi* on feature *fjk* of object *oj* at time *tl*. To make matters worse, a sentence may not explicitly mention some pieces of information, but they are implied due to pronouns, language conventions, and the context.To deal with these problems, we need to apply NLP techniques in the opinion mining context, e.g., parsing, word-sense disambiguation, and coreference resolution. We use coreference resolution as an example to give a glimpse of the issues. For our example blog, to figure out what is “my phone” and what is “her phone” in sentences (3) and (5) is not a simple task. Sentence (4) does not mention any phone and does not have a pronoun. The question is which phone “the camera” belongs to. Coreference resolution is a classic problem in NLP.

**2. Literature Review**

Sentiment analysis has attracted a great deal of attention in recent years due to the rapid growth of e-commerce and social media services (Liu 2012; Pang and Lee 2008). There exists an extensive body of work on major tasks like aspect extraction (Hu and Liu 2004; Chen, Mukherjee, and Liu 2014; Popescu, Nguyen, and Etzioni 2005; Xu et al. 2013; Fang, Huang, and Zhu 2013), opinion and polarity identification e.g., (Hu and Liu 2004; Pang and Lee 2008; Wilson, Wiebe, and Hwa 2004; Yu, Kaufmann, and Diermeier 2008; Jiang et al. 2011) and subjectivity analysis (Hatzivassiloglou and Wiebe 2000). The task of discovering phrasal opinions has also been studied extensively. For example, Wilson et.al. (2009) investigate phrasal opinions with opinion lexicons.

Fei, Chen and Liu (2014) used topic models to discover noun phrases. Zhang and Liu (2011) identify noun phrases implying inexplicit opinions.

YAO Tian-fang et al, and HOU Feng et al. approached from the perspective of opinion mining, WANG Hui et al., Ren Hongjuan and Zhang Zhiqiang took the method of viewpoint mining, ZHOU Li-zhu et al., Li Gang et al., ZHAO Yan- Yan et al., and WEI Wei et al, however, adopted the sentiment analysis method. Li Xiaojun et al. followed the route of the sentiment polarity. All the afore-mentioned research makes distinctive summary and exploration of the language and technology support in this field. Kaiser et al. proposed a warning system for online market research which allows the identification of critical situations in online opinion formation. Hasan and Adjeroh proposed three proximity-based features, namely, proximity distribution, mutual information between proximity types, and proximity patterns.

Machine Learning techniques use a training set and a test set for classification. Training set contains input feature vectors and their corresponding class labels. Using this training set, a classification model is developed which tries to classify the input feature vectors into corresponding class labels. Then a test set is used to validate the model by predicting the class labels of unseen feature vectors.

A number of machine learning techniques like Naive Bayes (NB), Maximum Entropy (ME), and Support Vector Machines (SVM) are used to classify reviews. Some of the features that can be used for sentiment classification are Term Presence, Term Frequency, negation, n-grams and Part-of-Speech. These features can be used to find out the semantic orientation of words, phrases, sentences and that of documents. Semantic orientation is the polarity which may be either positive or negative.

Domingos et al. found that Naive Bayes works well for certain problems with highly dependent features. This is surprising as the basic assumption of Naive Bayes is that the features are independent. Zhen Niu et al. introduced a new model in which efficient approaches are used for feature selection, weight computation and classification. The new model is based on Bayesian algorithm. Here weights of the classifier are adjusted by making use of representative feature and unique feature. 'Representative feature’ is the information that represents a class and ‘ Unique feature’ is the information that helps in distinguishing classes. Using those weights, they calculated the probability of each classification and thus improved the Bayesian algorithm.

Barbosa et al. designed a 2-step automatic sentiment analysis method for classifying tweets. They used a noisy training set to reduce the labeling effort in developing classifiers. Firstly, they classified tweets into subjective and objective tweets. After that, subjective tweets are classified as positive and negative tweets. Celikyilmaz et al. developed a pronunciation based word clustering method for normalizing noisy tweets. In pronunciation based word clustering, words having similar pronunciation are clustered and assigned common tokens. They also used text processing techniques like assigning similar tokens for numbers, html links, user identifiers, and target organization names for normalization. After doing normalization, they used probabilistic models to identify polarity lexicons. They performed classification using the BoosTexter classifier with these polarity lexicons as features and obtained a reduced error rate.

Wu et al. proposed a influence probability model for twitter sentiment analysis. If @username is found in the body of a tweet, it is influencing action and it contributes to influencing probability. Any tweet that begins with @username is a retweet that represents an influenced action and it contributes to influenced probability. They observed that there is a strong correlation between these probabilities.

Pak et al. created a twitter corpus by automatically collecting tweets using Twitter API and automatically annotating those using emoticons. Using that corpus, they built a sentiment classifier based on the multinomial Naive Bayes classifier that uses N-gram and POS-tags as features. In that method, there is a chance of error since emotions of tweets in training set are labeled solely based on the polarity of emoticons. The training set is also less efficient since it contains only tweets having emoticons.

Xia et al. used an ensemble framework for sentiment classification. Ensemble framework is obtained by combining various feature sets and classification techniques. In that work, they used two types of feature sets and three base classifiers to form the ensemble framework. Two types of feature sets are created using Part-of-speech information and Word-relations. Naive Bayes, Maximum Entropy and Support Vector Machines are selected as base classifiers. They applied different ensemble methods like Fixed combination, Weighted combination and Meta-classifier combination for sentiment classification and obtained better accuracy.

Certain attempts are made by some researches to identify the public opinion about movies, news etc from the twitter posts. V.M. Kiran et al. utilized the information from other publicly available databases like IMDB and Blippr after proper modifications to aid twitter sentiment analysis in movie domain.

**3. System Requirements**

* 1. **Hardware Requirements**
* RAM : 8 GB
* Processor: Intel i5
* Processor Speed : >2.6 GHz
* Hard Disk: 1 TB

**3.2 Software Requirements**

* Operating Systems : Windows 10 or above
* Platform: R 3.3.2

**4. Text Classification**

Text classification (a.k.a. text categorization) is the task of assigning predefined categories to free-text documents. It can provide conceptual views of document collections and has important applications in the real world. For example, news stories are typically organized by subject categories (*topics*) or geographical codes; academic papers are often classified by technical domains and sub-domains; patient reports in health-care organizations are often indexed from multiple aspects, using taxonomies of disease categories, types of surgical procedures, insurance reimbursement codes and so on. Another widespread application of text categorization is spam filtering, where email messages are classified into the two categories of *spam* and *non-spam*, respectively.

In text classification, we are given a description of a document , where  is the *document space*;

and a fixed set of *classes* . Classes are also called *categories* or *labels* . Typically, the document space  is some type of high-dimensional space, and the classes are human defined for the needs of an application, as in the examples China and documents that talk about multicore computer chips above. We are given a *training set*   of labeled documents , where  . For example:

for the one-sentence document ‘The buttons of the keyboard wear out really fast on frequent use’ and the class (or label) Negative.

Using a *learning method* or *learning algorithm* , we then wish to learn a classifier or *classification function*   that maps documents to classes:

This type of learning is called *supervised learning* because a supervisor (the human who defines the classes and labels training documents) serves as a teacher directing the learning process.

**4.1 Text Mining**

Text mining, also referred to as text data mining, roughly equivalent to text analytics, is the process of deriving high-quality information from text. High-quality information is typically derived through the devising of patterns and trends through means such as statistical pattern learning. Text mining usually involves the process of structuring the input text (usually parsing, along with the addition of some derived linguistic features and the removal of others, and subsequent insertion into a database), deriving patterns within the structured data, and finally evaluation and interpretation of the output. 'High quality' in text mining usually refers to some combination of relevance, novelty, and interestingness

**4.2 Parts of Speech**

In traditional grammar, a part of speech (abbreviated form: PoS or POS) is a category of words (or, more generally, of lexical items) which have similar grammatical properties. Words that are assigned to the same word part of speech generally display similar behavior in terms of syntax—they play similar roles within the grammatical structure of sentences—and sometimes in terms of morphology, in that they undergo inflection for similar properties. Commonly listed English parts of speech are noun, verb, adjective, adverb, pronoun, preposition, conjunction, interjection, and sometimes numeral, article or determiner.

A part of speech—particularly in more modern classifications, which often make more precise distinctions than the traditional scheme does—may also be called a word class, lexical class, or lexical category, although the term lexical category refers in some contexts to a particular type of syntactic category, and may thus exclude parts of speech that are considered to be functional, such as pronouns. The term form class is also used, although this has various conflicting definitions. Word classes may be classified as open or closed: open classes (like nouns, verbs and adjectives) acquire new members constantly, while closed classes (such as pronouns and conjunctions) acquire new members infrequently, if at all.

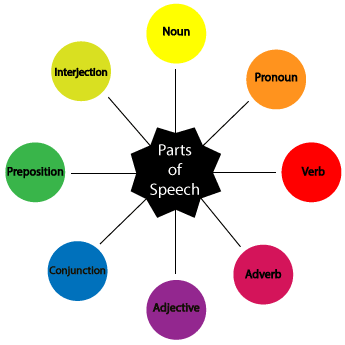


Fig.2 Parts of Speech

**4.2.1 Noun**

This part of a speech refers to words that are used to name persons, things, animals, places, ideas, or events. Nouns are the simplest among the 8 parts of speech, which is why they are the first ones taught to students in primary school.

Examples: *Tom Hanks* is very versatile; *Dog*s can be extremely cut; It is my *birthday*.

There are different types of nouns namely:

* Proper– proper nouns always start with a capital letter and refers to specific names of persons, places, or things. Eg: Volkswagen Beetle, Shakey’s Pizza, Game of Thrones
* Common– common nouns are the opposite of proper nouns. These are just generic names of persons, things, or places. Eg: car, pizza parlor, TV series
* Concrete– this kind refers to nouns which you can perceive through your five senses. Eg: folder, sand, board
* Abstract- unlike concrete nouns, abstract nouns are those which you can’t perceive through your five senses. Eg: happiness, grudge, bravery
* Count– it refers to anything that is countable, and has a singular and plural form. Eg:  kitten, video, ball
* Mass– this is the opposite of count nouns. Mass nouns are also called non-countable nouns, and they need to have “counters” to quantify them.
* Collective– refers to a group of persons, animals, or things. Eg: faculty (group of teachers), class (group of students), pride (group of lions)

**4.2.2 Pronoun**

A pronoun is a part of a speech which functions as a replacement for a noun. Some examples of pronouns are: *I*, *it, he, she, mine, his, hers, we, they, theirs,*and *ours.*

Sample Sentences:

* Janice is a very stubborn child. *She* just stared at me and when I told her to stop.
* The largest slice is *mine*.
* *We* are number one.

**4.2.3 Adjective**

This part of a speech is used to describe a noun or a pronoun. Adjectives can specify the quality, the size, and the number of nouns or pronouns.

Sample Sentences:

* The carvings are *intricate*.
* I have *two* hamsters.
* The italicized word “two,” is an adjective which describes the number of the noun “hamsters.”
* Wow! That doughnut is *huge*!
* The italicized word is an adjective which describes the size of the noun “doughnut.”

**4.2.4 Verb**

This is the most important part of a speech, for without a verb, a sentence would not exist. Simply put, this is a word that shows an action (physical or mental) or state of being of the subject in a sentence.

Examples of “State of Being Verbs” : *am*, *is*, *was*, *are*, and *were*

Sample Sentences:

* As usual, the Stormtroopers*missed* their shot.
* The italicized word expresses the action of the subject “Stormtroopers.”
* They are always prepared in emergencies.
* The verb “are” refers to the state of being of the pronoun “they,” which is the subject in the sentence.

**4.2.5 Adverb**

Just like adjectives, adverbs are also used to describe words, but the difference is that adverbs describe adjectives, verbs, or another adverb.

The different types of adverbs are:

* Adverb of Manner– this refers to how something happens or how an action is done.
* Example: Annie *danced* gracefully.
* The word “gracefully” tells how Annie *danced*.
* Adverb of Time- this states “when” something happens or “when” it is done.
* Example: She came *yesterday*.
* Adverb of Place– this tells something about “where” something happens or ”where” something is done.
* Example:  Of course, I looked everywhere!
* The adverb “everywhere” tells where I “looked.”
* Adverb of Degree – this states the intensity or the degree to which a specific thing happens or is done.
* Example: The child is *very* talented.
* The italicized adverb answers the question, “To what degree is the child talented?”

**4.2.6 Preposition**

This part of a speech basically refers to words that specify location or a location in time.

Examples of Prepositions: *above, below, throughout, outside, before, near,*and *since*

Sample Sentences:

* Micah is hiding*under* the bed.
* The italicized preposition introduces the prepositional phrase “under the bed,” and tells where Micah is hiding.
* *During* the game, the audience never stopped cheering for their team.
* The italicized preposition introduces the prepositional phrase “during the game,” and tells when the audience cheered.

**4.3 Tokenizer**

In sentiment analysis, tokenization is the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens. The list of tokens becomes input for further processing such as parsing or text mining. Tokenization is useful both in linguistics (where it is a form of text segmentation), and in computer science, where it forms part of lexical analysis. Typically, tokenization occurs at the word level. However, it is sometimes difficult to define what is meant by a "word". Often a tokenizer relies on simple heuristics. Tokens are separated by whitespace characters, such as a space or line break, or by punctuation characters.

In languages that use inter-word spaces (such as most that use the Latin alphabet, and most programming languages), this approach is fairly straightforward. However, even here there are many edge cases such as contractions, hyphenated words, emoticons, and larger constructs such as URIs (which for some purposes may count as single tokens). A classic example is "New York-based", which a naive tokenizer may break at the space even though the better break is (arguably) at the hyphen. Tokenization is particularly difficult for languages written in scriptio continua which exhibit no word boundaries such as Ancient Greek, Chinese. Agglutinative languages, such as Korean, also make tokenization tasks complicated. Some ways to address the more difficult problems include developing more complex heuristics, querying a table of common special-cases, or fitting the tokens to a language model that identifies collocations in a later processing step.

**5. Extraction of Expressions**

The definitions of words are not enough to characterize the semantic content of a text.“Bag of words” model which analyzes text as a package of words without connections can be sufficient in certain cases. It is the approach that Internet-based search engines use to index web pages.

Any semantic analyzer must at least be able to extract “noun phrases” automatically from text, i.e. expressions made up of 2 more or words.For instance, in texts related to health care: *health insurance, marketplace, insurance company, private insurance, affordable coverage, pre-existing condition*. These expression are much more precise than simple words (*health, marketplace, coverage, condition)*. In particular in this example, the word *condition*is extremely vague, but the phrase *pre-existing condition*clearly refers to the medical definition of the noun *condition.*At Synomia (and linguistics), these groups are called *phrases*.

To extract noun phrases from text, semantic engines use a standard technology to identify all sequences of words that match valid sequences of word classes. These are, for example:

An adjective followed by a noun (*private, insurance, affordable coverage)*

A noun followed by a preposition followed by a noun *(enrollment for coverage)*

A noun followed by a noun followed *(health, insurance, insurance company)*

This technology is too primitive to extract noun phrases from a text with sufficient coverage and accuracy. In particular, it generates a lot of noise. For instance, in the sentence:

*"It holds insurance companies accountable for unjustified premium increases."*

A standard engine will extract *unjustified premium*(adjective followed by noun) which is an error because it is the *increases*which are *unjustified,*not the *premium.* And in the sentence:

*"If you're eligible, enroll in a Marketplace health plan."*

A standard engine will extract *Marketplace health*(noun followed by noun) which is also a mistake.

**5.1 Extraction of Verb Expressions**

The first step in our task is to extract candidate verb expressions that may imply opinions. Phrase extraction has been studied by many researchers, e.g., (Kim et al. 2010) and (Zhao et al. 2011). However the phrases they extract are mostly frequent noun phrases in a corpus. In contrast, our verb expressions are different because they are verb oriented and need not to be frequent. In our settings, we define a verb expression to be a sequence of syntactically correlated phrases involving one or more verbs and the identification of the boundary of each phrase is modeled as a sequence labeling problem. The boundary detection task is often called chunking. In our work, we use the OpenNLP chunker to parse the sentence. A chunker parses a sentence into a sequence of semantically correlated phrasal chunks. The following example shows the chunk output of a sentence from the chunker:

[NP Windows and Linux][VP do not respond] [PP to][NP its scroll button]

**5.1.1 Verb Extraction Algorithm**

**Input** : a sentence sent, the maximum number words in any verb expression K

**Output**: a list of verb expressions V E represented by their starting and ending positions

1: chunks ← Chunker.chunk(sent)

2: VE ←ø

3: **for** each chunki ϵ chunks s.t i ϵ [1, chunks.length]**do**

4: **if** chunki.label==VP chunki  contains verbs other than be **then**

5: // extraction begins from VP with be verbs included

6: chunkCnt ← 1

7: start ← chunki.start

8: end ← chunki.end

9: // add optional NP to the end of chunki

10: **if** i > 1 and chunki-1.label==NP **then**

11: start ← chunki-1.start

12: chunkCnt ← chunkCnt+1

13: **end if**

14: //add optional PP, PRT, ADJP, ADVP, NP to the right

15: **for** j= i+1 to chunks.length **do**

16: **if**  chunkCnt == K **then //** maximum number of tokens in verb is reached

17: break

18: **end if**

19: **if** chunkj.label ϵ{PP, PRT, ADJP, ADVP, NP} **then**

20: end ← chunkj.end

21: chunkCnt = chunkCnt+1

22: **end if**

23: **end for**

24: V E ← V E U < start, end >

25: **end if**

**5.2 Extraction of Adverb Expressions**

We also need to extract candidate adverb expressions that may imply opinions. Like verbs, even adverb phrases are not something frequently extracted. Nouns are the most frequently extracted

Phrases. . In contrast, our adverb expressions are different because they are adverb oriented and need not to be frequent. In our settings, we define an adverb expression to be a sequence of syntactically correlated phrases involving one or more adverbs and the identification of the boundary of each phrase is modeled as a sequence labeling problem. Again, OpenNLP chunker is

Used to parse the sentence to get a sequence of semantically correlated phrasal chunks.

**5.2.1 Adverb Extraction Algorithm**

**Input** : a sentence sent, the maximum number words in any adverb expression K

**Output**: a list of verb expressions ADVE represented by their starting and ending positions

1: chunks ← Chunker.chunk(sent)

2: AVE ←ø

3: **for** each chunki ϵ chunks s.t i ϵ [1, chunks.length]**do**

4: **if** chunki.label==VP or chunki label==ADVP **then**

5: // extraction begins from ADVP with be verbs included

6: chunkCnt ← 1

7: start ← chunki.start

8: end ← chunki.end

9: // add optional NP to the end of chunki

15: **for** j= i+1 to chunks.length **do**

16: **if**  chunkCnt == K **then //** maximum number of tokens in adverb is reached

17: break

18: **end if**

19: **if** chunkj.label ϵ{ ADJP, ADVP} **then**

20: end ← chunkj.end

21: chunkCnt = chunkCnt+1

22: **end if**

23: **end for**

24: AVE ← AVE U < start, end >

25: **end if**

26: **end for**

**6. Training Models**

**6.1 Naïve Bayes**

In machine learning, naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

Naive Bayes has been studied extensively since the 1950s. It was introduced under a different name into the text retrieval community in the early 1960s and remains a popular (baseline) method for text categorization, the problem of judging documents as belonging to one category or the other (such as spam or legitimate, sports or politics, etc.) with word frequencies as the features. With appropriate pre-processing, it is competitive in this domain with more advanced methods including support vector machines. It also finds application in automatic medical diagnosis.

Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

**6.1.1 Representation Used by Naive Bayes Models**

The representation for naive Bayes is probabilities. A list of probabilities are stored to file for a learned naive Bayes model. This includes:

Class Probabilities: The probabilities of each class in the training dataset.

Conditional Probabilities: The conditional probabilities of each input value given each class value.

**6.1.2 Learn a Naive Bayes Model from Data**

Learning a naive Bayes model from your training data is fast. Training is fast because only the probability of each class and the probability of each class given different input (x) values need to be calculated. No coefficients need to be fitted by optimization procedures.

**6.1.3 Calculating Probabilities:**

Class Probability:

The class probabilities are simply the frequency of instances that belong to each class divided by the total number of instances.

For example in a binary classification the probability of an instance belonging to class 1 would be calculated as:

P(class=1) = count(class=1) / (count(class=0) + count(class=1))

In the simplest case each class would have the probability of 0.5 or 50% for a binary classification problem with the same number of instances in each class.

Conditional Probabilities:

The conditional probabilities are the frequency of each attribute value for a given class value divided by the frequency of instances with that class value.

**6.1.4 Make Predictions with a Naive Bayes Model**

Given a naive Bayes model, you can make predictions for new data using Bayes theorem.

MAP(h) = max(P(d|h) \* P(h))

Using our example above, if we had a new instance with the weather of sunny, we can calculate:

go-out = P(weather=sunny|class=go-out) \* P(class=go-out)

stay-home = P(weather=sunny|class=stay-home) \* P(class=stay-home)

We can choose the class that has the largest calculated value. We can turn these values into probabilities by normalizing them as follows:

P(go-out|weather=sunny) = go-out / (go-out + stay-home)

P(stay-home|weather=sunny) = stay-home / (go-out + stay-home

**6.2 Support Vector Machines**

In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyse data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

A support vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

Whereas the original problem may be stated in a finite dimensional space, it often happens that the sets to discriminate are not linearly separable in that space. For this reason, it was proposed that the original finite-dimensional space be mapped into a much higher-dimensional space, presumably making the separation easier in that space. To keep the computational load reasonable, the mappings used by SVM schemes are designed to ensure that dot products may be computed easily in terms of the variables in the original space, by defining them in terms of a kernel function {\displaystyle k(x,y)}selected to suit the problem. The hyperplanes in the higher-dimensional space are defined as the set of points whose dot product with a vector in that space is constant. The vectors defining the hyperplanes can be chosen to be linear combinations with parameters {\displaystyle \alpha \_{i}}of images of feature vectors {\displaystyle x\_{i}} that occur in the database.

**7. Implementation & Results**

The reviews crawled from the amazon electronics data range over multiple products. These products reviews for electronics devices are the baselines for the extraction algorithms. Training instances for our models are expressions extracted from titles of positive (5 stars) and negative (1 star) reviews. We observed that negative expressions are abundant in titles of reviews whose rating is 1 star, but both positive and neutral verb expressions can occur in 5-star review titles. So verb expressions in 1-star review titles are used as negative and those in 5-star review titles are used as non-negative. Test instances are verb expressions from both the titles and bodies of reviews whose ratings are 1 or 2 stars and they are labelled manually by three human judges.

The process of implementation has the following steps as represented in the following figure:

Fig.3 Steps of Implementation

The robust growth of web technology advances the prevailing idea of “User Generate Content” in internet era. Most of the website users are attracted by the blogs and microblogs. However, this form of commentaries is characterized as bulky, unformatted and full of spams. To facilitate a quick and effective analysis of the texts, the blog-based sentiment analysis has caught on rapidly. To access such database we accessed the amazon review dataset for electronic devices. These devices include reviews for the entire range of devices in the catalogue of amazon. Each device review is then accessed and filtered in order to separate the reviews in the categories of negative and non-negative. The negative reviews features are extracted with the title ranking 1 and the non-negative review are extracted with the title ranking 5. In the scope of review-specific sentiment analysis, feature extraction is an essential aspect of research for which our project brings necessary focus on verb expressions and adverb-adjective expressions implying negative expressions. These expressions have been used to train the probabilistic models of Naïve Bayes and support vector machines which form the baselines of training for the expressions extracted from the algorithms.

The models trained are then subjected to performance evaluation by subjecting the testing data set onto the trained models. The number of non-negative reviews and negative reviews used in training are 14207 and 4118 respectively, and the number of non-negative reviews and negative reviews used in testing are 6057 and 940 respectively. The sentiment classification obtained for the testing set are promising with high measurable accuracy of probabilistic classification in both non-negative and negative reviews are obtained. There have been certain variations in the probabilistic classification of the two training models which have been illustrated in the results table 1 for non-negative reviews and table 2 for negative reviews.

The text sentiment evaluation based on the non-negative reviews emanating on the two training models naïve bayes and support vector machines are given in tables.

|  |  |  |
| --- | --- | --- |
| **Non-negative Review Expression** | **SVM** | **Naïve Bayes** |
| This is a high quality Infrared filter. | 0.888616 | 0.846554 |
| Extras Needed for Use with HD Video Cameras | 0.909007 | 0.804656 |
| Master switch broke after 1 month... | 0.594941 | 0.786254 |
| Add the SD205 Switch to | 0.924196 | 0.765896 |
| Simply amazing lens . Sharp , great fast well | 0.992129 | 0.698566 |
| original is still the best | 0.957389 | 0.865455 |
| This is an absolute necessity | 0.863104 | 0.793568 |
| Buttons wear out really fast | 0.625454 | 0.667475 |
| keyboard does not recognize | 0.811254 | 0.672326 |
| I still get marginal signal | 0.75153 | 0.695312 |

Table.2 Non-Negative expressions model comparison

The non- negative expressions used in training the model is found to show high accuracy The probabilistic classification of the validated data to express as non- negative is as shown in the graph in figure 4, which showcases the variation of probabilistic classification of various non-negative expressions based on the two training models i.e. Naïve Bayes and Support vector machines.

Fig.4 Non-negative expression probabilities

The text sentiment evaluation based on the non-negative reviews emanating on the two training models naïve bayes and support vector machines are given in table.

|  |  |  |
| --- | --- | --- |
| **Negative Review Expressions** | **SVM** | **Naïve Bayes** |
| thing is a junk kids | 0.911225 | 0.892656 |
| Won't open ePub books and support | 0.746213 | 0.644845 |
| service is the worst that I ever | 0.686787 | 0.568356 |
| Did n't work on Win7 pc or Win8 | 0.873993 | 0.785255 |
| Does not work with my iPod Touch | 0.861629 | 0.852566 |
| Chinese made piece of junk | 0.970189 | 0.895665 |
| Do n't waste your time or money | 0.946069 | 0.842565 |
| service is the worst that I ever | 0.686787 | 0.647536 |
| Lasted a few months ... AVOID this | 0.563447 | 0.615489 |
| Beware of monitor power surge | 0.969761 | 0.945623 |

Table.2 Negative expressions model comparison

The negative expressions used in training the model is found to show high accuracy The probabilistic classification of the validated data to express as non- negative is as shown in the graph in figure 5, which showcases the variation of probabilistic classification of various non-negative expressions based on the two training models i.e. Naïve Bayes and Support vector machines.

Fig.5 Negative expression probabilities

**8. Conclusion**

In this project, we dealt with the problem of discovering verb and adverb-adjective expressions that imply negative and non-negative opinions. Such expressions usually describe product issues. Our work differs from other works as it emphasizes the role of verbs and adverb-adjective, and their correlations with other words. We proposed algorithms to extract such verb and adverb-adjective expressions. The input to the algorithms were title reviews which contained title rankings of 1 implying negative reviews and 5 implying non-negative reviews. These expressions were then used to train the employed machine learning models naïve bayes and support vector machine. These models were trained on negative and non-negative expressions to solve the problem of sentiment classification and analysis. Experimental results showed that our model can effectively find the objective expressions under consideration that prevail in reviews indicating critical product issues. The implications of verb and adverb-adjective expressions in classifying sentiment and indicating key issues with a certain product were highly successful. Since our training data is obtained from titles of reviews whose labels are automatically inferred from review ratings, our approach can be easily applied to any of the review or product domain. This analysis is therefore beneficial for companies and business who would like to improve their products or service based on their users’ feedback. Companies can use the reviews and extract the negative opinions. These opinions can be utilised to identify key components in the issues regarding a product which should be focused on to know what is to be improved and can enhance the sales and market potential of the products and services.

**9. References**

1. Bies, A.; Ferguson, M.; Katz, K.; and MacIntyre, R. 1995. Bracketing Guidelines for Treebank II Style Penn Treebank Project. Technical report, Linguistic Data Consortium.

2. Breck, E.; Choi, Y.; and Cardie, C. 2007. Identifying Expressions of Opinion in Context. In IJCAI, 2683–2688.Chen, Z.; Mukherjee, A.; and Liu, B. 2014. Aspect extraction with automated prior knowledge learning. In ACL,347–358.

3. Choi, Y., and Cardie, C. 2010. Hierarchical Sequential Learning for Extracting Opinions and Their Attributes. In ACL, 269–274.

4. Fang, L.; Huang, M.; and Zhu, X. 2013. Exploring weakly supervised latent sentiment explanations for aspect-level review analysis. In CIKM, 1057–1066.

5. Fei, G.; Chen, Z.; and Liu, B. 2014. Review topic discovery with phrases using the p´olya urn model. In COLING, 667– 676.

6. Hatzivassiloglou, V., and Wiebe, J. M. 2000. Effects of adjective orientation and gradability on sentence subjectivity. In COLING, 299–305.

7. Hu, M., and Liu, B. 2004. Mining and summarizing customer reviews. In KDD, 168–177.

8. Jiang, L.; Yu, M.; Zhou, M.; Liu, X.; and Zhao, T. 2011. Target-dependent Twitter Sentiment Classification. In ACL,151–160.

9. Johansson, R., and Moschitti, A. 2011. Extracting opinion expressions and their polarities - exploration of pipelines and joint models. In ACL, 101–106.

10. Joshi, M., and Ros´e, C. P. 2009. Generalizing Dependency Features for Opinion Mining. In ACL, 313–316.

11. Kim, S.-M., and Hovy, E. 2006. Extracting opinions, opinion holders, and topics expressed in online news media text.

12.Kim, S. N.; Medelyan, O.; Kan, M.-Y.; and Baldwin, T. 2010. SemEval-2010 task 5: Automatic keyphrase extraction from scientific articles. In Proceedings of the 5th International Workshop on Semantic Evaluation, SemEval ’10, 21–26. Association for Computational Linguistics.

13. Y. Mejova, “Sentiment analysis: An overview,” Comprehensive exam paper, available on http://www. cs. uiowa. edu/˜ymejova/publications/CompsYelenaMejova. pdf [2010-02-03], 2009.

14. E. Boiy, P. Hens, K. Deschacht, and M.-F. Moens, “Automatic sentiment analysis in on-line text,” in Proceedings of the 11th International Conference on Electronic Publishing, pp. 349–360, 2007.

15. P. D. Turney, “Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews,” in Proceedings of the 40th annual meeting on association for computational linguistics, pp. 417–424, Association for Computational Linguistics, 2002

16. J. Kamps, M. Marx, R. J. Mokken, and M. De Rijke, “Using wordnet to measure semantic orientations of adjectives,” 2004.

17. C. Fellbaum, “Wordnet: An electronic lexical database (language, speech, and communication),” 1998.

18. D. Pucci, M. Baroni, F. Cutugno, and A. Lenci, “Unsupervised lexical substitution with a word space model,” in Proceedings of EVALITA workshop, 11th Congress of Italian Association for Artificial Intelligence, Citeseer, 2009.

19. A. Balahur, J. M. Hermida, and A. Montoyo, “Building and exploiting emotinet, a knowledge base for emotion detection based on the appraisal theory model,” Affective Computing, IEEE Transactions on, vol. 3, no. 1, pp. 88–101, 2012.

20. C. Strapparava and R. Mihalcea, “Semeval-2007 task 14: Affective text,” in Proceedings of the 4th International Workshop on Semantic Evaluations, pp. 70–74, Association for Computational Linguistics, 2007.