Article can be accessed online at http://www.publishingindia.com

Text Analytics Framework using Apache Spark and Combination of Lexical and Machine Learning Techniques

Anuja Prakash Jain*, Padma Dandannavar**

Abstract

Today, we live in a 'data age'. The sudden increase in the amount of user-generated data on social media platforms like Twitter, has led to new opportunities and challenges for companies that strive hard to keep an eye on customer reviews and opinions about their products. Twitter is a huge fast emergent micro-blogging social networking platform for users to express their views about politics, products sports etc. These views are useful for businesses, government and individuals. Hence, tweets are used in this framework for mining public's opinion. Sentiment analysis is a process of naturally recognising whether a user-generated content expresses positive, negative or neutral opinion about an entity (i.e. product, people, topic, event etc). The traditional analytics tools are costly and are not built to handle Big data. Hadoop, though being a popular framework for data intensive applications, does not perform well on iterative process (like data analysis) due to the cost paid for data reloading from disk for each iteration. This paper proposes a text analysis framework for twitter data using Apache spark and hence is more flexible, fast, and scalable. The proposed framework is also domain independent as it uses a hybrid approach by combining supervised machine learning algorithms (Naïve Bayes and decision tree machine learning algorithms) and lexicon approach (pattern analyser) for sentiment classification thereby comparing various supervised learning models and using the one with highest accuracy for predicting sentiment.

Keywords: Sentiment Analysis, Machine Learning, Lexical Approach, Apache Spark, Natural Language Processing, Twitter

Introduction

Twitter is a huge, fast, emergent, popular, micro-blogging social networking platform for users to express their views about politics, products sports etc. Here, clients send messages (a.k.a., tweets) to a system of contacts from a wide assortment of gadgets or sites. A tweet is a content predicated post and just has 140 characters, which is around the length of a typical newspaper headline or subhead (Milestein, 2008). Twitter is a "what's-happening-right-now" social network and hence tweets are valuable sources for businesses, government, and individuals to determine public's opinion or sentiment about an entity (product, people, topic, event etc). But, the volume of tweets produced by Twitter everyday is very vast (21 million tweets per hour, as measured in 2015). Hence there is a need to automate the process of sentiment analysis so as to ease the tasks of determining public's opinions without having to read millions of tweets manually. This process of analysing and summarising user's views on a particular entity is usually called Sentiment Analysis or Opinion Mining which is an extremely fascinating and prominent space for analysts these days.

Text analysis includes data retrieval, lexical analysis to study word recurrence appropriations, pattern recognition, labeling/annotation, data extraction, data mining techniques, visualisation, and predictive analytics. The general objective is, basically, to transform content into information for investigation, by means of use of natural language processing (NLP) and analytical methods. Sentiment analysis is a process of automatically

* MTech Student, Computer Science and Engineering, Visvesvaraya Technological University, Belgaum, Karnataka, India. Email: jainanu04@gmail.com

** Asst. Prof, Computer Science and Engineering, Gogte Institute of Technology, Belgaum, Karnataka, India. Email: padmad@git.edu

identifying whether a user-generated content expresses positive, negative or neutral opinion about an entity (i.e. product, people, topic, event etc.). Sentiment classification can be done at Document level, Sentence level and Aspect or Feature level (Vohra & Teraiya, 2013). At document level the whole document is used as a basic information unit to classify it either into positive or negative class. Sentence level sentiment classification classifies each sentence first as subjective or objective and then classifies into positive, negative or neutral class. There is no much difference between the above two methods as sentence is just a short document. Aspect or Feature level sentiment classification deals with identifying and extracting product features from the source data (Vohra & Teraiya, 2013).

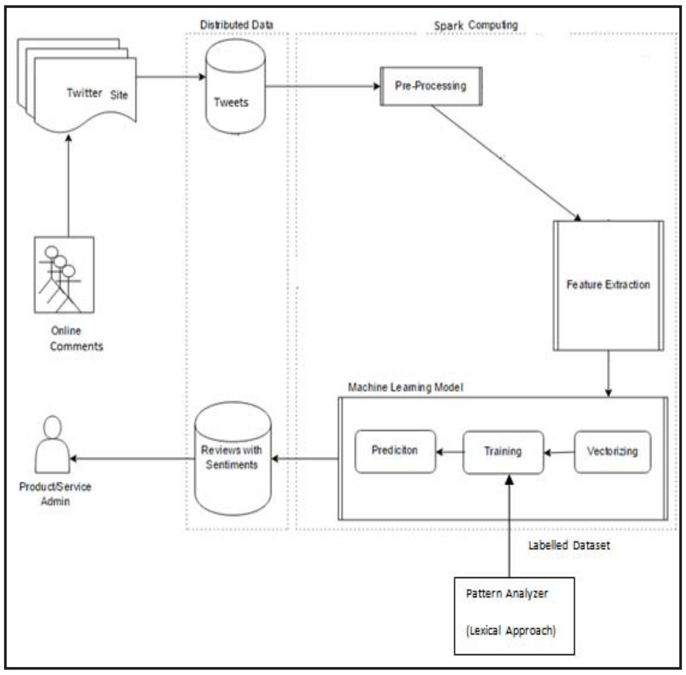
There are few methodologies for sentiment analysis. Machine learning based approach (ML) uses several machine learning algorithms (supervised or unsupervised algorithms) to classify data (Neethu & Rajashree, 2013). Here, two datasets are needed: training and test dataset. A supervised learning classifier utilises the training set to learn and train itself w.r.t the differentiating characteristics of text, and a test set is utilised to check the performance of the classifier. Lexicon based approach uses a dictionary containing positive and negative words to determine the sentiment polarity. It deals with counting the number of positive and negative words in the text. If the text consists of more positive words, the text is assigned a positive score. If there are more number of negative words the text is assigned a negative score. If the text contains equal number of positive and negative words then it is assigned a neutral score. To determine whether a word is positive or negative an opinion lexicon (positive and negative opinion words) is built. Hybrid based approach uses a combination of both ML and lexicon based approach for classification. The drawback of machine learning based approach is that they need a huge training data which is very difficult to obtain. Also, manually labeling the data is a very tedious job. Lexicon based approach faces a disadvantage that the strength of the sentiment classification depends on the size of the lexicon (dictionary). As the size of the lexicon increases this approach becomes more erroneous and time consuming. As mentioned in the paper by Zhang, Ghosh, Dekhil, Hsu and Liu (2011), lexicon based approach have high precision and low recall. Hence combining it with a machine learning classifier can improve the recall and accuracy of the algorithm. This paper focuses on the proposition of combining the two approaches into a hybrid model in order to mitigate the drawback of these two approaches by using the lexicon-based classifier for the task of annotating the training data for the learning-based classifier, this technique leverages the learning-based classifier's performance while taking advantage of the lexicon-based classifier's effortless setup process (Zang *et al.*, 2011)

The amount of user generated data available online is astronomically immense. Considering the span of information starting 2011, i.e. 30 million, we can anticipate that it will develop over billions by 2017. Consequently we have to guarantee that the framework is adaptable and fast to support any measure of information. This project develops a framework that is both scalable and fast. This is accomplished by using Apache Spark which is a quick and universally useful cluster computing system. Spark runs programs up to 100x speedier than Hadoop MR in memory, or 10x quicker on disk. Machine Learning Library (mllib) is available in Apache Spark (Bhuvan & Rao, 2015). With the help of this, various classification models are built and model with the highest accuracy is used to predict the user sentiment on twitter data.

The Proposed Technique

This section describes the proposed framework. Fig. 1 gives an architectural overview of the proposed technique for twitter sentiment analysis.

Fig. 1: Architecture of Twitter Sentiment Analysis using Apache Spark and Combination of Lexical and Machine Learning Technique



Data Collection

Twitter allows researchers to collect tweets by using a Twitter API. One must have a twitter account to obtain twitter credentials (i.e. API key, API secret, Access token and Access token secret) which can be obtained from twitter developer site. Then install a twitter library to connect to the Twitter API. We used Twitter Search API to extract tweets based on a search query (e.g. #Apple , "Apple Products") specified by the user. It works similar to the search function provided by twitter web client or twitter mobile. We used tweepy twitter library to extract tweets. The twitter data is obtained in Json form containing tweets and information about the users. The important fields are explained below:

1. Created_at: Time at which the user posted the tweet.
2. Entities: Several fields like URL, Hashtags, user_mentions parsed from text.
3. Id: A unique identifier for this tweet
4. Retweet_count: Indicates the number of times this tweet is retweeted.
5. Source: indicates the source of a tweet (device or website)
6. Text: shows the actual tweet as posted by the user containing external web links (e.g. http://amze.ly/8K4n0t) (Zang *et al.*, 2011), hashtags (e.g. #Apple , used to filter tweets based on a topic), user names (e.g. @user1 , indicates that the tweet is a reply to a user named user1) and the user comment.
7. User: containing information about the user like user id, user's profile image, description of the user etc.

Data Pre-processing

Before we perform the sentiment analysis on twitter data the data should be brought into proper form and sentiment relevant features need to be extracted. The text field of twitter data, as described above contains external links URLs and tweet. Processing a natural language is very difficult and trying to determine the user sentiment is even more difficult as users make sarcastic comments. The data pre-processing steps include the following:

1. **Removal of external links:** URLs and user names are not useful for determining the sentiment analysis.
2. **Removal of duplicate tweets:** Twitter data may contain redundant tweets and retweets which need to be removed.
3. **Spelling Correction:** Social media tweets may contain incorrect spellings. Spelling of the erroneous words can be rectified predicated on automated selection of more probable word.
4. **Case Conversion:** All words are transformed into lower case in order to eradicate the distinction between "Text" and "text" for further processing.
5. **Stop-words Removal:** The commonly used words like a, an, the, has, have etc which carry no meaning i.e. do not help in determining the sentiment of text while analysing should be removed from the input text.
6. **Punctuation Removal:** Punctuation marks such as comma or colon often carry no meaning for the textual analysis hence they can be removed from input text.
7. **Stemming:** Stemming customarily refers to a simple process that chops off the terminuses of words to abstract derivational affixes.
8. **Lemmatisation:** Deals with abstraction of inflectional terminuses only and to return the base or dictionary form of a word, which is termed as the lemma.

The proposed method makes use of NLTK (Natural Language Tool Kit) to carry out pre-processing steps.

Feature Extraction and Vectorizing

Once the tweets are pre-processed, we need to extract features relevant for sentiment analysis. This framework uses Hashing TF-IDF algorithm to extract features. TF-IDF (*term frequency-inverse document frequency)* is frequently utilised in information retrieval and text mining. This TF-IDF weight is a statistical measure used to assess the importance of a word in a document (tweet). The Hashing TF-IDF algorithm implemented in the proposed framework using Apache Spark is:

Algorithm: Hashing TF-IDF (tweet)
Input: RDD of tweet
Output: TF-IDF weight of each term in the tweet
hash(t) – hash function for term t
num_of_terms – total number of terms in a tweet
freq(*t*,*tweet*)–Calculates number of times t appears in a *tweet*
d(*t*)- Number of tweets that contain term *t*
m-Total number of tweets

Vector.sparse(arg)- Returns a Sparse Vector of arg

1. For each term *t* in *tweet*:
   a. Calculate index of *t*
      index = hash(*t*) % num_of_terms
   b. Frequency[index] = freq(*t*,*tweet*)
2. Vector = Vector.Sparse(num_of_terms, Frequency[])
3. Data = RDD of Vectors
4. Calculate IDF for each vector in Data using the formula:
   idf = log((m + 1) / (d(*t*) + 1))
5. Calculate TF-IDF weight using:
   TF-IDF = Frequency[*t*]*idf

This approach eschews the need to compute a global term-to-index map, which can be extravagant for a sizably voluminous corpus, but it suffers from hash collisions, where different raw features may become identically tantamount term after hashing. In order to avoid this, the number of buckets in the hash table are increased to 1,048,576.

Building a Training Dataset

The lexicon-based approach is used to build the training data. The training data consists of tweets labeled by lexical-based pattern analyzer (Zang *et al.*, 2011). Using training data provided by lexicon-based method has following advantages:

- Mitigating the labor-intensive and time consuming process of manually annotating training data.
- Lexicon based approach have low recall. By using hybrid approach we can achieve higher accuracy and recall (Zang *et al.*, 2011).
- The proposed framework is domain independent as the classifier is trained on the fly based on the output of lexical-based approach and not on a domain-specific manually labeled dataset.

Machine learning algorithms require huge training data for better performance. Suppose, on an average if a human takes around 10 seconds to classify one tweet, then he would take 15000 seconds (4 hours) to classify 1500 tweets. Though a machine learning approach performs slightly better than the proposed hybrid approach, the difference in performance might no longer be worth the inconvenience of acquiring training data, making the hybrid model an appealing alternative with a more beneficial trade-off between performance and convenience (Fredrick, 2015). Nowadays, large organisations are desperately in need of fast approximate results rather than accurate results to take important decisions faster. Hence, the proposed framework fulfils this need by making use of hybrid approach and Apache Spark framework.

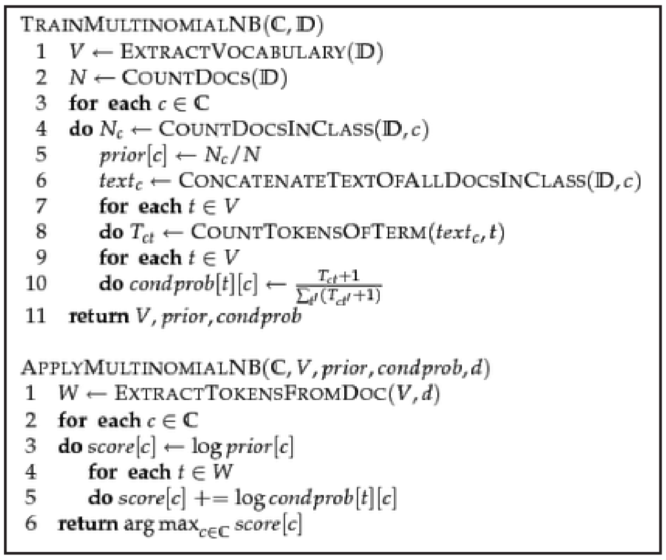
Predicting User Sentiment

Machine learning classifiers are trained on the training data obtained from lexicon approach and then they are tested on test dataset. The machine learning model classifies the tweets as positive, negative and neutral. Their performances are compared based on various performance measures like accuracy, precision, recall and F1-score. The machine learning algorithms used in this framework are explained in the next section.

Machine Learning Algorithms for Sentiment Classification

Multinomial Naïve Bayes

The Naive Bayes classifier is the easiest of all (as the name proposes) and extremely viable for text classification as it figures the posterior probability of a class, in view of the dispersion of the words (features) in the document. We use multinomial naive bayes as we classify a tweet into 3 different classes (positive, negative, neutral). Here, each feature is denoted as a term whose value is the frequency of the term. The multinomial naive bayes algorithm is as shown below:



```
TrainMultinomialNB($\mathbb{C}$, $\mathbb{D}$)
 $V \leftarrow$ ExtractVocabulary($\mathbb{D}$)
 $N \leftarrow$ CountDocs($\mathbb{D}$)
 for each $c \in \mathbb{C}$
 do $N_c \leftarrow$ CountDocsInClass($\mathbb{D}, c$)
     $prior[c] \leftarrow N_c / N$
     $text_c \leftarrow$ ConcatenateTextOfAllDocsInClass($\mathbb{D}, c$)
     for each $t \in V$
     do $T_{ct} \leftarrow$ CountTokensOfTerm($text_c, t$)
     for each $t \in V$
     do $condprob[t][c] \leftarrow \frac{T_{ct}+1}{\sum_{t'}(T_{ct'}+1)}$
 return $V, prior, condprob$

ApplyMultinomialNB($\mathbb{C}, V, prior, condprob, d$)
 $W \leftarrow$ ExtractTokensFromDoc($V, d$)
 for each $c \in \mathbb{C}$
 do $score[c] \leftarrow \log prior[c]$
     for each $t \in W$
     do $score[c] \mathrel{+}= \log condprob[t][c]$
 return $\arg\max_{c \in \mathbb{C}} score[c]$
```

Multinomial naive bayes perform better when trained on huge dataset.

Decision Tree

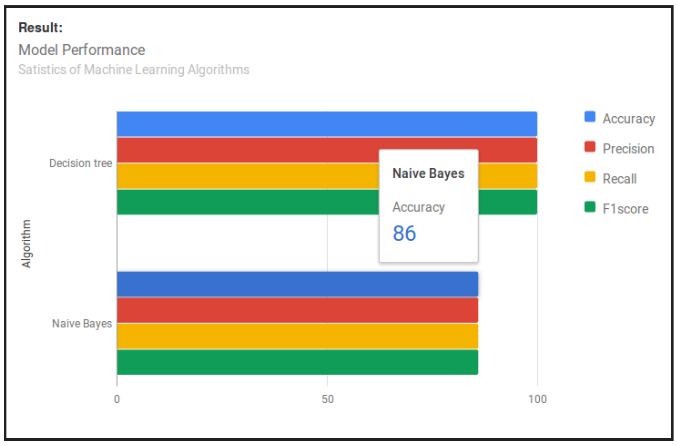
Decision trees in Spark MLlib are greedy algorithms and scale gracefully to distributed setting. A decision tree model is trained using training dataset and model builds a top-down tree (hierarchical if-else statements) which can then be used to predict unseen data. The decision tree performs binary partition of the feature space recursively. The tree avariciously picks every segment by selecting the best split from an arrangement of conceivable parts, with a specific end goal to expand the data pick up at a node of tree. We use Gini Impurity to measure the homogeneity of labels at each node.

These two algorithms are tested on test data and their performances are compared based on accuracy, precision, recall, F1-score.

Results

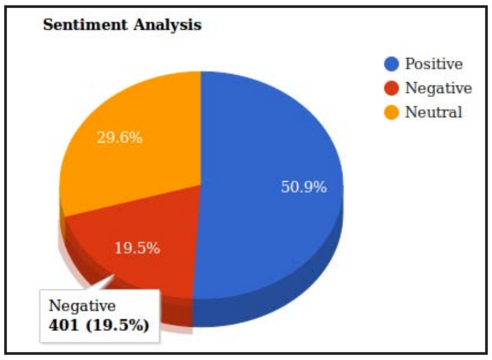
We used a dataset containing 4000 tweets for search query "Apple Products". 70% of the dataset was used for training while 30% was utilised for testing the performance of Naïve Bayes and Decision tree algorithms. The performance measured in terms of precision, recall, F1Score, accuracy is shown in Fig. 2. The figure shows that Decision tree algorithm has an accuracy of 100 percent and Naïve Bayes has an accuracy of 86 percent. The figure also shows precision, recall and F1Score.

Fig. 1: Performance of Naïve Bayes and Decision Tree



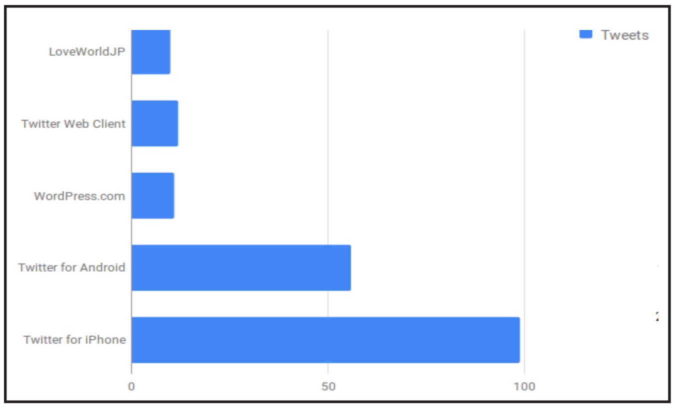
The result of sentiment analysis is shown in the form of pie chart (Fig. 3) which shows that about approx 20% tweets are negative for the search query "Apple Products".

Fig. 3: Sentiment Analysis Using Number of Positive, Negative and Neutral Tweets for Query "Apple Products"



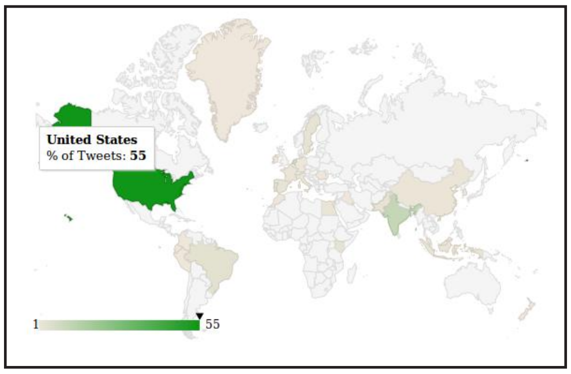
The framework also shows the sources of tweets (Fig. 4) which are obtained from the 'source' key in twitter json.

Fig 4: Sources of Tweets



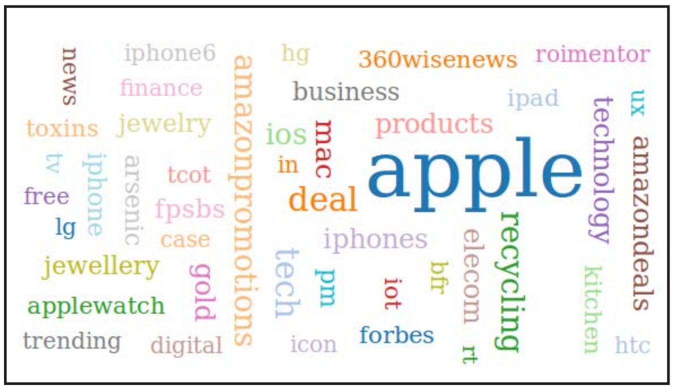
The country-wise location of tweets is shown using the world graph in Fig. 5. The graph shows that maximum people who are tweeting about Apple Products are from USA.

Fig. 5: Country-Wise Location of Tweets



It is also useful to know on what topic are people talking about. The framework gives an idea of the topic of discussion using a word cloud. The word cloud shows that most of the people are talking about "Apple" while few others talk about Mac and ios.

Fig. 6: WordCloud of Important Keywords



Conclusion

Sentiment analysis can be performed using lexicon based approach, machine learning based approach or hybrid approach. The lexicon-based approach is used to build the training data for mitigating the labor-intensive and time consuming process of manually annotating training data. The proposed framework is domain independent. The framework performs sentiment analysis using Naive Bayes and Decision tree algorithms. The results show that Decision tree performs extremely well showing 100% accuracy, precision, recall and F1Score. The framework also shows the sources and location of tweets along with the important keywords depicting the topic of discussion. The proposed text analytics framework is also real-time, fast, scalable, and reliable as we use Apache Spark framework.

References

Bhuvan, M. S., & Rao, V. D. (2015). *Semantic Sentiment Analysis Using Context Specific Grammar*. International Conference on Computing, Communication and Automation (ICCCA2015).

Cho, S., & Kang, H. (2012). *Text Sentiment Classification for SNS-based Marketing using Domain Sentiment Dictionary*. IEEE International Conference on Conference on Consumer Electronics (ICCE), (pp.717-718).

D'Andrea, A., & Ferri, F. (2015). Approaches, Tools and Applications for Sentiment Analysis Implementation. *International Journal of Computer Applications*, 125(3), 26-33.

Hassan, A., & Medhat, W. (2014). Sentiment analysis algorithms and applications: A survey. *Shams Engineering Journal*, 5(4), 1093-1113.

Kanakaraj, M., & Guddeti, R. M. R. (2015). *NLP Based Sentiment Analysis on Twitter Data using Ensemble Classifiers.* 3rd International Conference on Signal Processing, Communication and Networking (ICSCN).

Ko, E. H., & Klabjan, D. (2014). *Semantic Properties of Customer Sentiment in Tweets*. 28th International Conference on Advanced Information Networking and Applications Workshops.

Liu, B. (2012). *Sentiment analysis and opinion mining*. Morgan and Claypool Publishers, (pp.18-19).

Mane, S. B., Sawant, Y., Kazi, S., & Shinde, V. (2014). Real time sentiment analysis of twitter data using Hadoop. *International Journal of Computer Science and Information Technologies*, 5(3), 3098-3100.

Milstein, S., Chowdhury, A., Hochmuth, G., Lorica, B., & Magoulas, R. (2008). *Twitter and the micro-messaging revolution: Communication, connections*. An OReilly Radar Report. (pp. 54).

Neethu, M. S., & Rajashree R. (2013). *Sentiment analysis in twitter using machine learning techniques*. 4th International Conference on Computing, Communications and Networking Technologies (ICCCNT).

Pang, B., Lee, L., & Vaithyanathan, S. (2002). *Thumbs up? Sentiment classification using machine learning techniques*. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), (pp. 79).

Rajurkar, G. D., & Goudar, R. M. (2015). *A speedy data uploading approach for Twitter Trend and Sentiment Analysis using Hadoop*. International Conference on Computing Communication Control and Automation.

Turney, P. D. (2002). *Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews*. In Proceedings of 4th Annual Meetings for Computational Linguistics, (pp. 417-424).

Vohra, S. M., & Teraiya, J. B. (2013). A comparative study of sentiment analysis techniques. *Journal of Information, Knowledge and Research in Computer Engineering*, 2(2), 313-317.