Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 203 (2022) 165–172

Procedia Computer Science

www.elsevier.com/locate/procedia

The 19th International Conference on Mobile Systems and Pervasive Computing (MobiSPC)
August 9-11, 2022, Niagara Falls, Canada

Live Sentiment Analysis Using Multiple Machine Learning and Text Processing Algorithms

Andrew Motz[a], Elizabeth Ranta[a], Adan Sierra Calderon[a], Quin Adam[a], Fadi Alzhouri[b], Dariush Ebrahimi*[a]

[a]*Thompson Rivers University, 805 TRU Way, Kamloops, British Columbia V2C 0C8, Canada*
[b]*Trent University, 1600 W Bank Dr, Peterborough, ON K9L 0G2, Canada*

Abstract

Due to the massive amount of data being generated on the platform, Twitter has been the subject of numerous sentiment analysis studies. Such social network services generate massive unstructured data streams which make working with them very challenging. The aim of this study is to reliably analyze the sentiment of trending tweets in the Twitter API data stream using a combination of different algorithms to achieve a consensus. The methods we implemented include Support-Vector Machine, Naive Bayes, Textblob, and Lexicon Approach. The hypothesis is that using these methods together would enable us to get more accurate results. Using a labeled dataset to test our model, the results show that the combination of these four algorithms all together performed best with an overall accuracy of 68.29%. We conclude that our combination method of analysis is suitable and fast enough for our data stream and also accurate for analyzing sentiment.

Keywords: Lexicons; Machine Learning; Twitter; Data Streams; Sentiment Analysis;

1. Introduction

Since the 1970s, data has been exploited in the business world with information systems such as dashboards or business analytics. Over time, the volume, size, and the complexity of modern data have made it necessary to implement other more powerful technologies such as big data and data science. This makes a large number of information sources available to companies, to which they previously had no access. This contributes to company management improvement and fosters its growth by having more information about potential clients, times of economic expansion

* Corresponding author. Tel.: +1-250-828-5053 ; fax: +0-000-000-0000.
E-mail address: debrahimi@tru.ca

1877-0509
10.1016/j.procs.2022.07.023

and recession, and relevant information for making projections about the future of the company in order to protect itself against adverse economic cycles.

One of the fundamental aspects of big data is sentiment analysis. It consists of being able to discern the feelings, opinions or attitudes towards elements such as products, individuals, organizations or services through documents of a varied nature. The use of sentiment analysis in documents is inevitably linked to machine learning, enabling technology for the extraction of new variables and relationships in data that enable advanced analysis of the extracted information. Due to the new links established, predictions can be presented to the business user that until now were not possible to show or were unreliable. The general objective of this paper is to analyze positive and negative sentiment on a live data stream by combining multiple analysis techniques. To develop the method so that it is fast enough to handle the incoming data. Whereas the specific objectives can be concentrated in the following:

- Determine if using multiple machine learning and lexicon based sentiment analysis techniques together gives a better result than using them individually.
- Analyze the viability of a solution combining multiple analysis techniques on live data with high velocity.

1.1. Value of Sentiment Analysis

The values of sentiment analysis can vary and are substantial. Different sectors such as companies, organizations, and governments around the world are the main stakeholders in advancing this field of research. Being able to know what people think about x products, rules, and policies at all times is a very desired tool, and under the right circumstances, it can offer a competitive edge that was unfathomable until a few years ago. In the same way, social media monitoring and sentiment analysis of certain issues can help with the detection and gestation of certain social events, such as strikes, seditions, revolts, etc. Some of the applications and problems we seek to solve with sentiment analysis could fall under any of the following:

Opinion Evaluation of products and services: This is probably the most practical and direct sentiment analysis. Through this technique, it is possible for companies to know the opinion of users about their products without having to carry out traditional studies such as satisfaction surveys. Thus, through the opinions expressed in forums, blogs, and social networks, it will be possible to know if the users like a certain product or not. In this way, companies can know at any time if their products are to the liking of users and, if not, be able to rethink strategies in the shortest possible time, thus granting competitive advantages.

Opinion correction: It is common for users to express their opinion through online shopping sites. Commonly, in addition to a review, many sites offer the option to rate a product with a score, for instance, one to five stars. It may happen that, by mistake, the user does not correctly indicate the said score. Thus, a sentiment analysis system could analyze the user's words and correct that score automatically.

Product recommendation systems improvement: Based on user opinions, an online store may prioritize the products it offers based on these opinions, for instance not recommending products whose general opinion is negative.

Online Advertising Positioning: Advertisers of certain products may require that their ads be published only on websites where positive concepts are expressed to avoid pages where the texts express negative feelings.

Political reputation: Sentiment analysis shows strong potential for finding out people's opinions about many political concerns. For instance, a particular political party, a candidate, the reception of a particular political party, certain policies implemented by the Government, as well as their variations over time.

Analysis of the financial market: Based on the information contained on web pages, forums and social networks about a specific company, it is possible to predict its evolution in the market from the added value of the polarity of all the conflicting opinions.

1.2. Related Work

Working with unstructured data such as the Twitter data stream comes with its own unique set of challenges. In [1], the authors during the study found that Part-Of-Speech features did not give good results with Twitter data. Whether this was due to the nature of Twitter and tweeting, or due to an issue with their methods, they were not sure. In [2], some of the complications of dealing with streams of data were discussed. The authors in their proposed model used the "Twitter Streaming API", "R-Project", and open-source packages "rTweet", "stringr" and "glue".

For sentiment analysis, they used a Lexicon approach. Ultimately, they proved that their analysis was able to detect meaningful sentiment changes across a hashtag's lifetime, and was efficient enough to handle the Twitter API data stream. The authors in [3] aimed to create improved methods for Twitter trend analysis. They were able to perform rapid analysis on tweets from the Twitter API, and by using Apache SPARK the model became very fast and was able to detect real-time Twitter trends. Many researchers have looked at combining different methods and algorithms to gain better results. A couple of papers have used results from lexicons to train ML methods. Kolchyna et al. [4] used sentiment data made with lexicons as an input feature for training their classifiers. Similarly, the authors in [5] also used results obtained from lexical analysis to train ML methods. A combination approach can be seen in [6], where supervised classification and unsupervised clustering are used together to create a more efficient method. Luiz et al. [7] achieved improved results when using Support-Vector-Machine and cluster ensembles [8] together. From the literature review we found that we can get promising results from combining algorithms, for example, in [4] the authors gain 7% increased accuracy by using lexicon results to train their machine learning methods. In [9], authors combined 4 lexicons into a single, improved one called Senti-Merge. To analyze the success of the combination, they used majority-voting of the 4 lexicons against Senti-Merge. The majority of the work above focuses on datasets. Whereas, in this paper, our aim is to combine results from different machine learning methods and lexicons to analyze Online data streams. While working with a data stream, it is necessary for methods to be efficient enough to handle the velocity of incoming data. The uniqueness of our work comes from both the analysis of the feasibility of live data and the way that we combine our algorithms to reach a solution. While many more complicated solutions focus on integrating intermediate results and machine learning methods together, our approach is simple and uses each algorithm's opinion to reach a consensus. We find this has gained us improvements despite being a simpler approach.

2. Initial Proposed Model

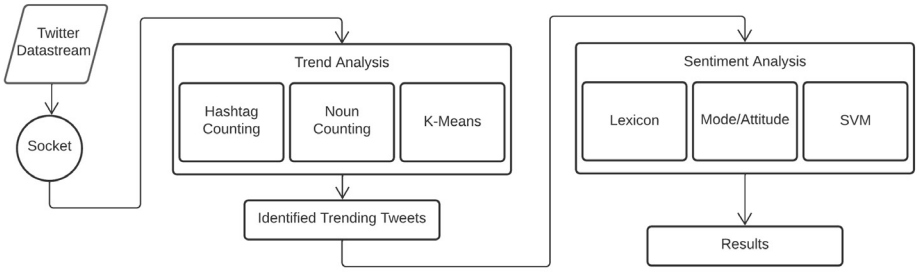

Fig. 1: Initial proposed model, to be updated in Section 3.

As seen in Fig. 1, our initial proposed model has two main components; trend analysis and sentiment analysis. By identifying trends first and only passing trending topics to the sentiment analysis algorithms, we would only be performing processing on relevant data. This initial proposed model allows us to use it in a modular way to add, remove, and combine various algorithms easily. This model has various limitations which we will modify in the next section. The main issue is that hashtags are not as commonly used as originally speculated and that detecting trends using them is not reliable. This model also fails to take data cleaning into account, which is an integral part of any analysis of unstructured and user-generated data.

3. Updated System Model

Our final system model can be seen outlined in Fig. 2. Firstly, tweets are taken in with the Twitter Application Programming Interface[10]. We then filter out tweets in languages other than English and do the first stage of text cleaning by removing non-Latin characters such as emojis. These cleaned tweets are then compared with Twitter's trending API [10] to find trending topics in the tweets. After this, the trending tweets are passed to the analysis process.

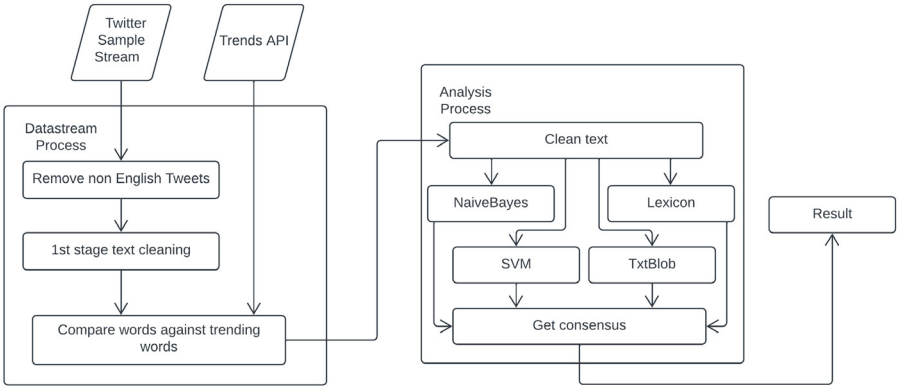

Fig. 2: Updated System Model

A second-stage text cleaning process takes place, which removes stop words and performs lemmatization on the text. Finally, the clean text is passed through to each of our implemented algorithms NaiveBayes, Lexicon, Support-Vector-Machine, and TextBlob. Each of these algorithms works independently to find whether the sentiment is positive or negative, then finally a majority vote is taken to reach a consensus on the sentiment of the tweet. Each of the methods are weighted equally, so each algorithm gets one vote as to whether the tweet is positive or negative. If there is a tie in the number of positives and negatives, (For instance, 2 labelling positive and 2 labelling negative) is broken by labeling it as negative. We found during testing that this gave us the best results.

3.1. Data stream

We will be using the Twitter Application Programming Interface (API) to retrieve our data stream [10]. The API allows us to subscribe to a random data stream where approximately 1% of Twitter's traffic will be sampled. This gives us data that users generate in real-time and allows us to potentially discover topics never seen before. The large amount of data coming in from this data stream would confirm the model's capability of operating in such environments.

3.2. Testing Dataset

Since the Twitter API [10] does not have labeled data, we can't verify our solution using it. To test the accuracy of our solution, we used the Sentiment140 dataset made by Alec Go, Richa Bhayani, and Lei Huang [11]. This allows us to compare our results against the labeled data and perform testing on combinations of the algorithms. We randomly sampled 30,000 lines from the 1.6 million Tweets available for testing purposes.

3.3. Twitter Trends API

After trying different ML methods to identify trending topics, we found that it makes more sense to retrieve the data from Twitter directly. To identify trending topics, we used Twitter's trending API [10] which returns a list of trending hashtags and words for a given location. These locations are already predefined using the "where on earth identifier" (WOEID) [10]. Each request returns a list of 50 trending topics for that given location. There is a limit of 75 of these requests per 15 minutes, so depending on how many places we are interested in at a time, this will limit how often we can get trending topics. We decided to use Twitter's API instead of discovering trends ourselves because a trend-identifying algorithm would take a large amount of time. Since Twitter's API has access to all tweet data, using their API gives us the ability to be much more accurate in identifying trending topics.

3.4. Algorithms Used

Support-Vector Machine: Our Support-Vector Machine (SVM) was implemented using the scikit-learn library [12], and trained on data from the Natural-Language Toolkit (NLTK) library [13]. Since an SVM takes in vectors, the tweets first need to be converted into vectors. To do this, scikit-learn's Term Frequency - Inverse Document Frequency (TF-IDF) Vectorizer is used. This function takes in a dataset, and assigns each word its own index, up to a maximum number of words. In our case, we choose the top 5000 most common words. Then, each tweet in the dataset is turned into a sparse vector, where each entry in the vector is a TF-IDF score for every word that is recognized by the TF-IDF Vectorizer. The TF-IDF score is calculated by multiplying the term frequency, by the inverse document frequency. In other words, the number of times a word shows up in a tweet is multiplied by the total number of tweets, then divided by the number of tweets that contain the word. The intuition behind this is that the more a word shows up in a particular tweet, the more important it is. However, it should be noted that if a word shows up in almost every tweet (for example, the word "the"), the less important it is. It is also to be noted that in order to make small value changes more intensive and effective, a natural log (ln) must be taken from all the values before final multiplications.Consequently, the TF-IDF score is obtained using (1). Each sparse vector is then sent to the SVM for training along with its label (i.e., Positive or Negative). The SVM at first treats all dimensions with a value of zero.

$$Score = ln(1 + count(word, tweet)) \times ln(\frac{length(all\_tweets)}{count(word, all\_tweets)}) \tag{1}$$

Naive Bayes: Naive Bayes is a classifying algorithm that uses probability to predict which class a set of features belongs to [14]. Our Naive Bayes implementation was implemented using the NLTK library [13]. As described in [13], the algorithm uses the Bayes rule (2) to express P($label|features$) in terms of P($label$) and P($features|label$). This implementation then takes the "naive" approach and assumes all features are independent given the label shown by (3).

$$P(L|F) = \frac{P(L) * P(F|L)}{P(F)} \tag{2}$$

$$P(L|F) = \frac{P(L) * P(f1|L) * ... * P(fn|L)}{P(F)} \tag{3}$$

In the context of this paper, a tweet is labeled as positive or negative and the features are the text for a given tweet. Using the naive approach and assuming the features are independent, the order of the words does not matter from word $f1$ to word fn, which means the relation and context between words are lost. Naive Bayes must be trained to make predictions. Labelled training data provided by the NLTK library was used to train our model.

TextBlob (NLTK + pattern): TextBlob [15] is a text processing library for Python that allows us to perform Natural Language Processing tasks such as morphological analysis, entity extraction, sentiment analysis, machine translation, etc. It is built on top of two other very famous Python libraries: NLTK [13] and pattern [16]. The main advantage of TextBlob is that it allows us to combine the use of the two previous tools in a simpler interface.

Lexicon Approach: There are two main approaches to Lexicons, dictionary-based and corpus-based [17]. The main difference between the two is that dictionary-based have a collection of words, with sentiment data for each word. Corpus-based approaches try to assign the sentiment by also taking context into account, and there are statistical and semantic approaches to these. Lexicons don't always have to be static either - they can also be generated using any data. In very simple terms, we could look up keywords from our Tweets in the lexicon, and use the lexicon sentiment info to assign a value to our data. There are several open-source data packages for lexicons like [18] and [19]. We decided to use the NRC Word-Emotion Association Lexicon [20]. It is a very famous and widely used lexicon for its ability to analyze sentiment. It is composed of 27,000 words, each associated with 10 values. These values include positivity, negativity, as well as 8 other emotions. These other 8 include fear, anger, anticipation, trust, surprise, sadness, disgust, and joy. The values for each word are scored on a scale of 0 to 1, with 1 being a strong

association and zero having no association. We implemented this lexicon using a python library called NRCLex [21]. We analyze the incoming tweets and return positive or negative, based on the average values of the words contained in the tweet. Values of 0 indicate that the tweet is neutral, or is not leaning towards positive or negative. In our model, we classify these tweets as positive for a few reasons; the main reason is that the labeled dataset and other algorithms only have values of positive and negative, so introducing neutral tweets will reduce the accuracy of the lexicon. If we are to use a dataset with neutral values for testing, we would expect the accuracy to improve.

3.5. Multi-threading

We utilize multi-threading to increase the reliability and performance of our model. Two processes are used to ingest and run sentiment analysis on Twitter's data. One process is dedicated to receiving the incoming data from Twitter's sample stream. It also provides some basic text cleaning and filtering. Tweets are filtered by language using a flag on each of the incoming tweets; Tweets which are flagged as English are stored. After filtering out non-English tweets, a regular expression is used to remove certain Unicode characters from the Tweets text. Some of the characters that are removed include emojis, country flags, newline characters, and dingbats. After the basic filtering and cleaning is done, each word within the Tweet's text is compared against a list of known trending topics from Twitter's trends API. If any words match between the Tweet's text and known trending topics, those matching words are put into a list. This provides a list of trending words contained in the tweet, and the tweet itself. These two items are then en-queued into a shared thread-safe queue.

The second process, labeled in Fig. 2 as "Analysis Process", handles all the sentiment analysis. This process sits in an infinite loop to check if anything has been put into the queue. Once something is put into it, it will run the sentiment analysis. Whenever data is dequeued, further text cleaning is performed before the analysis is run. To further clean the text we remove all stop words and use a lemmatizer. Lemmatization involves breaking words down into their common base form. An example of this would be that "cars", "cars'" and "car's" can all be summarized as just "car". This is done to reduce noise from slight differentiation of words. Once the further cleaning is done, the cleaned text is sent to the analyzing process using the different ML algorithms (TextBlob, Naive-Bayes, SVM, and Lexicon algorithms. The result from each algorithm is counted and then given the highest consensus, the Tweet is classified as positive or negative.

4. Simulation Results

According to Deng and Luque [22], a confusion matrix is a concept of supervised learning, which contains information about real and predicted classes, made by a classification system. A confusion matrix has two dimensions, one dimension is indexed by the value of the actual class of an object, and the other is indexed by the class that the classifier predicts. The confusion matrix illustrates the performance metrics used for the classification algorithms: Accuracy, Precision, Recall, and F-1 Score. Precision measures the number of positive class predictions that actually belong to the positive class. Recall measures the number of positive predictions made out of all positive labels in the data. F-1 Score is a single score that balances recall and precision into one number. These statistics are commonly used to analyze classifiers and will serve us well to analyze the performance of our model. To test the accuracy of our model, we run our classifier against a list of labeled Tweets. We use the Sentiment140 dataset [11]. Since the dataset is large, we only test our classifier against 30,000 entries picked at random. To test whether our idea of using multiple algorithms to create one classification is better than just one of the algorithms, we test every combination of them. To do this we create a Python script that will enable or disable each combination of algorithms and test that combination against the same 30,000 for each run. The combination that has the best accuracy and the combination that has the best f-1 score is selected. The combination with the best accuracy is the combination of all four algorithms as displayed in Table 1. While the combination with the best F-1 Score, highlighted in Table 2, is the combination of analyzing using Naive-Bayes, TextBlob, and Lexicon algorithms.

To analyze the performance of each algorithm, we use a Python library called cProfile [23]. We run the profiler on the same test to determine our model's accuracy using the Sentiment140 dataset [11]. We captured the profile of the running analysis using all four algorithms on a set of 30,000 randomly selected Tweets. On a system with a i7-6700 and 16GB of RAM at 2133MHz, the total run time is 147 seconds. 57.1 seconds are spent cleaning the text

Table 1: Performance metrics results obtained by using a combination of the four algorithms, which results in the best accuracy.

		True		
		Positive	Negative	Total
Predicted	Positive	10,339	4,683	15,022
	Negative	4,830	10,148	14,978
	Total	15,169	14,831	30,000

Accuracy:	**0.6829**
Precision:	0.6815
Recall:	0.6882
F-1 Score:	0.6849

Table 2: Performance metrics results obtained by using a combination of the three algorithms: Naive-Bayes, TextBlob, and Lexicon, which results in the best F-1 score.

		True		
		Positive	Negative	Total
Predicted	Positive	11,666	3,356	15,022
	Negative	6,883	8,095	14,978
	Total	18,549	11,451	30,000

Accuracy:	0.6587
Precision:	0.6289
Recall:	0.7765
F-1 Score:	**0.6950**

while 87.8 seconds are actually spent performing the sentiment analysis. The "Other" category in Fig. 3 and Fig. 4 are basic operations like initialization, reading in data from the dataset, summing results, and comparing the answers. The performance of each of the algorithms within the 87.8 seconds spent analyzing is shown in Fig. 3. From the figure, we can illustrate that SVM has the largest runtime, whereas Naive-Bayes has the smallest runtime. Proportions of the entire run-time of 147 seconds are displayed in Fig. 4. So, on average it takes around 0.0058 seconds to clean and analyze a single Tweet. This gives us the capacity to handle a velocity of 172 Tweets per second. After filtering out non-English tweets and only sampling trending topics, our model does not exceed 172 tweets per second. Therefore, our implementation of using the four algorithms is feasible given the velocity of data coming into our model.

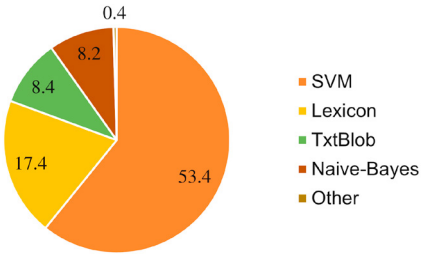

Fig. 3: Runtime in seconds

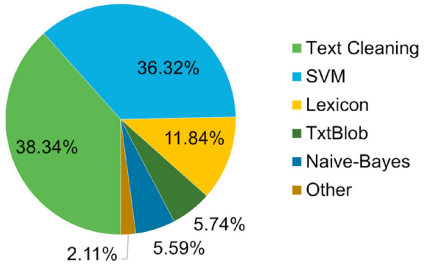

Fig. 4: Runtime proportions

5. Discussion

Interesting results have been achieved in the more experimental phases of the study, visualizing the recurring words or themes when searching through keywords places like Canada, U.S.A, U.K, Australia, New Zealand, Toronto and Vancouver. The sentiment classification of mentioned places through the aforementioned algorithms also showed interesting patterns. For instance, patterns of sports-related trends appearing on weekends, and sentiment related to Ukraine is positive. In sentiment analysis, a system is considered to have an accepted level of accuracy when it reaches 70% [24]. This is because not even people are capable of agreeing up to 30% of the time when classifying texts into categories. With this in mind, we are satisfied with the results that our approach has achieved.

6. Conclusion

This paper has set the objective of providing business value by studying the general sentiment behind some of the most trending topics on Twitter through performing different pre-processing methods to better extract patterns, as well as using Machine Learning techniques like Support Vector Machine and Naive Bayes classifiers to extract a sentiment value using Natural Language Processing procedures. Regarding implementation, we have developed cleaning techniques to deal with emoticons, flags, transport, and map symbols. We have also adjusted some abbreviations and words

with hyphens in order to provide our different algorithms with the cleanest samples possible, to better and with more confidence provide a classification of the text. In addition to these pre-processing techniques, we also implemented different classification algorithms: Naive Bayes based on the famous Bayes Theorem, Support Vector Machine, and two Lexicon-based algorithms. We did this intending to get a better sense of what works best. When we compare the results of each classification algorithm, we can conclude that there isn't a single algorithm that works best. When the four different algorithms were run in conjunction and a majority vote was taken, the best accuracy was achieved with an accuracy of 68.29%. While there is still much room for improvement and enormous possibilities in terms of combining algorithms' opinions, combining lexical and machine learning approaches, and sentiment analysis, our results showed that accuracy improved when combining multiple algorithms' opinions to form a result.

References

[1] E. Kouloumpis, T. Wilson, and J. Moore, "Twitter sentiment analysis: The good the bad and the omg!" in *Proceedings of the international AAAI conference on web and social media*, vol. 5, no. 1, 2011, pp. 538–541.

[2] S. K. Tasoulis, A. G. Vrahatis, S. V. Georgakopoulos, and V. P. Plagianakos, "Real time sentiment change detection of twitter data streams," *arXiv preprint arXiv:1804.00482*, 2018.

[3] A. P. Rodrigues, R. Fernandes, A. Bhandary, A. C. Shenoy, A. Shetty, and M. Anisha, "Real-time twitter trend analysis using big data analytics and machine learning techniques," *Wireless Communications and Mobile Computing*, vol. 2021, 2021.

[4] O. Kolchyna, T. T. P. Souza, P. C. Treleaven, and T. Aste, "Twitter sentiment analysis," *CoRR*, vol. abs/1507.00955, 2015. [Online]. Available: http://arxiv.org/abs/1507.00955

[5] A. Mudinas, D. Zhang, and M. Levene, "Combining lexicon and learning based approaches for concept-level sentiment analysis," in *Proceedings of the First International Workshop on Issues of Sentiment Discovery and Opinion Mining*, ser. WISDOM '12. New York, NY, USA: Association for Computing Machinery, 2012. [Online]. Available: https://doi.org/10.1145/2346676.2346681

[6] Q. Qian, S. Chen, and W. Cai, "Simultaneous clustering and classification over cluster structure representation," *Pattern Recogn.*, vol. 45, no. 6, p. 2227–2236, jun 2012. [Online]. Available: https://doi.org/10.1016/j.patcog.2011.11.027

[7] L. F. Coletta, N. F. da Silva, E. R. Hruschka, and E. R. Hruschka, "Combining classification and clustering for tweet sentiment analysis," in *2014 Brazilian conference on intelligent systems*. IEEE, 2014, pp. 210–215.

[8] J. Ghosh and A. Acharya, "Cluster ensembles," *Wiley interdisciplinary reviews: Data mining and knowledge discovery*, vol. 1, no. 4, pp. 305–315, 2011.

[9] G. Emerson and T. Declerck, "Sentimerge: Combining sentiment lexicons in a bayesian framework," in *Proceedings of Workshop on Lexical and Grammatical Resources for Language Processing*, 2014, pp. 30–38.

[10] "Twitter api documentation - twitter developer platform," https://developer.twitter.com/en/docs/twitter-api, accessed: 2022-03-01.

[11] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," *CS224N project report, Stanford*, vol. 1, no. 12, p. 2009, 2009.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[13] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[14] J. H. M. Dan Jurafsky, *Speech and Language Processing*. Stanford CA 94305-2150: Standford, 2021, 3rd ed. draft.

[15] S. Loria, "textblob documentation," *Release 0.15*, vol. 2, 2018.

[16] T. D. Smedt, "Pattern," https://github.com/clips/pattern, 2018.

[17] N. Gupta and R. Agrawal, "Chapter 1 - application and techniques of opinion mining," pp. 1–23, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128186992000019

[18] M. Lisa and H. Bot, "My Research Software," 12 2017. [Online]. Available: https://github.com/github/linguist

[19] B. Pallier, Christophe & New, "Openlexicon," 2019. [Online]. Available: https://github.com/chrplr/openlexicon

[20] S. M. Mohammad and P. D. Turney, "Crowdsourcing a word-emotion association lexicon," *Computational Intelligence*, vol. 29, no. 3, pp. 436–465, 2013.

[21] M. M. Bailey, "Nrclex," https://github.com/metalcorebear/NRCLex, 2019.

[22] A. Luque, M. Mazzoleni, A. Carrasco, and A. Ferramosca, "Visualizing classification results: Confusion star and confusion gear," *IEEE Access*, 2021.

[23] "The python profilers." [Online]. Available: https://docs.python.org/3/library/profile.html

[24] K. Roebuck, *Sentiment Analysis: High-Impact Strategies - What You Need to Know: Definitions, Adoptions, Impact, Benefits, Maturity, Vendors*. Lightning Source, 2011. [Online]. Available: https://books.google.ca/books?id=5BGZZwEACAAJ