Assignment 3 - Group 18
SemEval 2017 Task 4: Sentiment Analysis in Twitter

Text Mining — Assignment paper

Brian Arnesto Sitorus
b.a.sitorus@umail.leidenuniv.nl
Leiden University
Leiden, Netherlands

Sudarshna Arya Patel
s.a.patel@umail.leidenuniv.nl
Leiden University
Leiden, Netherlands

ABSTRACT

The SemEval dataset consists of 11 datasets of tweets collected between years 2013 and 2016. This dataset is a bit noisy, and two of them aren't properly formatted with tab separators, causing some tweets to stack on top of each other. We decided to ignore these files and use only nine txt datasets in our experiments. In addition, we performed a couple of steps in preprocessing to make it easier to explore the dataset, such as parsing the dataset with spacy, removing @mention, lemmatizing each token with Spacy method '.lemma,' removing special characters, doing spell correction because sometimes users use repeating characters to exaggerate their argument, i.e (sooo bad) to imply that it was not just bad but it was already too bad, and so on. Furthermore, the dataset is unbalanced, which means that it is not evenly distributed among the three classes offered (neutral, positive, and negative). The disadvantage of training the model on an unbalanced dataset is that when applied to real-world data, the model will be biased in favor of the dominant class. This is a challenge for us when attempting to forecast the minority class. To address this issue, we used the oversampling and undersampling methods to balance the data and monitor performance (accuracy, recall, precision and f-1 score). The oversampling method, particularly when using SMOTE, demonstrated the highest accuracy. Furthermore, we are experimenting with different classifiers with SMOTE, and the results show that the Logistic Regression classifier outperforms the others.

ACM Reference Format:
Brian Arnesto Sitorus and Sudarshna Arya Patel. 2022. Assignment 3 - Group 18 SemEval 2017 Task 4: Sentiment Analysis in Twitter: Text Mining — Assignment paper. In *Proceedings of Text Mining 2021 (TM '21).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

There are 186 million Twitter users who are active every day, and approximately 500 millions tweets are sent each day. They send each other short messages called "tweets" on this social media site. People write about their thoughts and feelings in these messages, even though they're only 140 characters long. This gives them a lot of information. Consequently, automatic sentiment classification of tweets could be useful in a wide range of applications, from commercial to political purpose[16, 43, 44].

Micro-blogging sites such as Twitter that's discussed previously have become popular in recent years as a means of disseminating and receiving information. It's gaining traction as more researchers and organizations join the study. Despite the availability of tools that can extract text data from Twitter or any other micro blogging service for sentiment analysis, data extraction remains a problem for data workers. As the World Wide Web has grown in popularity, so has the use of social media platforms such as Twitter, which generates massive amounts of opinion text in the form of tweets that can be analyzed using sentiment analysis [19]. As a result, humans now have unprecedented access to data. As a result, they are unable to extract concepts from text, read and evaluate it tweet by tweet, or quickly summarize and organize it [19].

Additionally, on occasion, people will use the informal form or informal language of a word, which can result in a misinterpretation of the context. Informal language entails the use of colloquialisms and slang in communication, as well as spoken language standards like 'wouldn't' [10]. Analysis and decision-making processes may be hampered because not all systems recognize emotions in informal language.

People also used emoticons to communicate with one another. In the absence of body language and prosody, animated facial expressions (emoticons) assist the recipient in focusing on the tenor or temper of the sender's nominal verbal communication, thereby improving and altering perception [10, 12]. For example, the emoticon ☺ denotes a cheerful mental state. At the moment, the algorithms that are available do not have enough data to understand and interpret certain emoticons. Furthermore, when people are unable to communicate verbally, they frequently rely on emoticons to convey their feelings (emotion icons) [12]. This is becoming a problem because we can use this emoticon to gain insight or make decisions based on it.Even in text-messaging services, the abbreviated form is commonly used for ease of use (SMS). To keep Twitter's character limit at a minimum, short-form will be used more frequently. This is because Twitter has a 140-character character limit [6]. For example, the phrase "to be announced" is abbreviated as "Tba."

SemEval 2017 is a good place for researchers to learn about sentiment analysis on Twitter [37]. Our work mainly focuses on Task 4: Sentiment Analysis in Twitter Subtask A: Message Polarity Classification, experiments with various different classifier and their performance analysis. Using a three-point scale, it attempts to determine the overall polarisation of the tweet (i.e., Positive, Negative and Neutral).

TM '21, Master CS, Fall 2021, Leiden, the Netherlands

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

A task like sentiment orientation classification on tweets can be thought of as a sentence-level sentiment analysis task because of the limited character. In this study [24, 45, 46], they used a wide range of traditional natural language processing (NLP) features. These features include linguistic features (word n-grams, POS tags, and more), sentiment lexicon features (scores based on eight sentiment lexicons), and domain content features (e.g., the number of words that have the same number of POS tags) (e.g., emoticons, capital words, elongated words, etc)

Tweets contain a wealth of metadata, so we combed through them to find any features that might be present [47]. These methods included a variety of word embeddings, including those for general words as well as sentiment words. Experiments were conducted to see how each feature and supervised machine learning algorithm works.

2 RELATED WORK

Sentiment analysis which also frequently referred to as opinion mining is a subset of text analysis that focuses on detecting, extracting, and analyzing opinion-oriented text, differentiating between positive and negative opinions, and quantifying how an entity (such as a person or an organization, an event or a location, a product or a topic) is perceived positively or negatively by the general public. [30]. It is an enthralling research topic with the potential for a wide variety of real-world applications in which discovered opinion data can assist individuals, corporations, or organizations in making more informed decisions [31].

Numerous authors have written about the Sentiment Analysis in Twitter task in recent years, which has been run annually at SemEval since 2013 [22, 28, 38], with the 2015 task incorporating sentiment toward a specific topic [39] and the 2016 task incorporating tweet quantification and five-point ordinal classification [38].

The International Workshop on Semantic Evaluation, formerly known as SensEval, has been renamed SemEval. SIGLEX, the Association for Computational Linguistics' Special Interest Group on Lexicon, is currently conducting a review series on computational semantic analysis systems [37]. Other comparable problems at SemEval have included sentiment analysis of product reviews and their apsects [32–34], sentiment analysis of figurative language on Twitter [13], recognizing posture in tweets [23], determining the sentiment intensity of words and phrases out of context [18], and emotion detection [41]. Several of these research utilized non-English languages, such as Arabic [23, 32]; nevertheless, they did not analyze tweets or emotional responses to a topic.

In the majority of studies on sentiment analysis, machine learning is the most frequently used method [26]. According to [29], the primary disadvantage of this method is the difficulty in developing appropriate datasets, as trained classifiers are typically domain-specific.

Moreover, because the dataset provided is imbalanced, we researched how to address this issue. The authors of [40] advanced the concept of undersampling and are well-known in the field. It discards data points whose target class does not correspond to the majority of KNNs. The authors of [25] addressed a variety of issues relating to learning in a class distribution with a skewed distribution. Take into account the relationship between price-sensitive information and class scatterings, as well as the error frequency and accuracy constraints associated with evaluating model behavior.

The authors of [9] proposed resampling strategies in light of the difficulty in locating the minority target. They developed a novel resampling strategy based on class-specific sub-clustering that includes an equal sampling of excessively positive individuals while undersampling the majority of uninfected individuals. According to them, their novel resampling method outperformed the more traditional random resampling method.

Moreover, the author of [20] proposes a novel addition method that uses k-means grouping to reduce the number of majority examples in order to balance the imbalanced cases.The authors of [21] used an undersampling technique to exclude data points from majority cases based on their distance from the majority case.

For our work, we used the oversampling and undersampling methods to tackle class imbalances in the datasets and observed which method performed the better. Additionally, we worked with various machine learning classifiers and analysed their performance.

3 DATA

We worked with Twitter English tweet collection consists of year 2013 till 2016 which contains total of 50K tweets and we used this to perform sentimental analysis. Our dataset contains three columns *id, sentiment, tweet*. Sentiment column represents the sentiment of the tweet which means whether the text is of positive, negative, or neutral sentiment. A summary of datasets is given in table 1 that provide insights about data used.

After importing all 11 datasets into the pandas dataframe, with a total of 49467 tweets, there is a minor issue with two datasets that are not properly parsed. Twiiter has a character limit of 280, but both Twitter-2016train-A and Twitter-2016dev-A have more. These two datasets were then removed (twitter-2016train-A and twitter-2016dev-A), as only 9 datasets are required to train the model. There are a total of 41705 tweets. The dataset is not evenly distributed across the three classes (neutral, positive, and negative); the negative class has the fewest tweets (7713), while the positive class has the most (22182 tweets).

dataset	Total	Negative	Neutral	Positive
twitter-2013dev-A	1654	340	739	575
twitter-2013test-A	3547	559	1513	1475
twitter-2013train-A	9684	1458	4586	3640
twitter-2014sarcasm-A	49	22	7	20
twitter-2014test-A	1853	202	669	982
twitter-2015test-A	2390	365	987	1038
twitter-2015train-A	489	66	253	170
twitter-2016dev-A	1966	391	746	82 9
twitter-2016devtest-A	2000	325	681	994
twitter-2016train-A	5868	850	2001	3017
twitter-2016test-A	20632	3231	10342	7059

Table 1: Datasets used in training and testing

3.1 Data Cleaning

We preprocessed the tweets before extracting features for analysis. All punctuation, repeating characters, URLs, numeric numbers, and special characters were removed. To tokenize the tweet text, we removed the stop words and changed the text to lower case. Furthermore, we used the extensive list of contractions to return certain text to its original form, such as "you'd" to "you would," in order to understand the true meaning of the sentences. We also performed spelling correction, because some people use repeated characters to emphasize a point, such as "soooo delicious". If a character appears more than twice, it will be reduced to two. This may not be the best solution to solve this problem because it may cause confusion by changing the true meaning of the word. If "soo" isn't the correct spelling, this will help to reduce feature space by converting "sooooo", "soooo", and "sooo" to the same word for "soo".

4 METHODS

4.1 Text Representation

For our experiments, we used *tf-idf* (term frequency - inverse document frequency) representation of texts. The *tf–idf* weight is a frequently used weighting scheme in information retrieval and text mining. This weight is used to determine the utility of particular words within a collection or corpus. The more frequently a term occurs in a document, the more critical it becomes. On the other hand, the word's prominence in the corpus dilutes its significance. According to search engines, TF–IDF weighting algorithms are frequently used to rank and analyze the relevance of sites to specific user queries. This method can be used to ascertain the frequency with which particular document phrases occur [1]. We used *Scikit-learn* implementation of *tf-idf* to convert our raw data to a matrix of *tf-idf* features.

4.2 Classifiers

We used *Scikit-learn* implementation of several classifiers.

4.2.1 ***Logistic Regression Classifier***.

Logistic Regression is a linear classifier for binary classification tasks which predicts the probabilities of the classes. It uses a sigmoid function to calculate the probabilities. It uses one-vs-rest approach in case of multi-class problem.

The vast majority of logistic regression models return a binary value, such as true/false, yes/no, or something along those lines. Multinominal More than two discrete outcomes can be represented using logistic regression. By resolving classification issues, logistic regression can be used to determine whether or not a new sample belongs in a specific category [11].

4.2.2 ***Random Forest Classifier***.

Random Forests, alternatively called random decision forests, are an ensemble learning technique for classifying, regression, and other tasks that entails training a large number of decision trees and then generating a class that is the mode or mean prediction of those trees (regression) [2].

Random forest is used for both classification and regression tasks. It is built of several decision trees on different samples and it takes the majority of votes of these decision trees to perform classification.

4.2.3 ***K-Nearest Neighbors Classifier***.

K Nearest Neighbors is a simple classification algorithm which works by calculating the distance between the test query and the training examples and selecting the closest K examples and then getting the votes for most frequent label to classify.

This algorithm is non-parametric, which means it makes no assumptions about the data. This technique, also known as the "lazy learner algorithm," does not recognize the training data right away. Instead, it saves the data and then categorizes it. The KNN algorithm simply stores the data at the training point. It gathers and categorizes new information. Data from both the past and the present are grouped together in the same category [3].

4.2.4 ***SGD Classifier***.

SGD is a linear classifier that uses stochastic gradient descent to learn classifications. Each sample's gradient of loss is computed separately, and the model is updated as the process proceeds. The Stochastic Gradient Descent Classifier (SGDC) can be extremely effective when dealing with a large number of attributes and data. SGD Classifier and SGD Regressor fit linear models with log losses using different loss functions for classification and regression. SGD Classifier fits Logistic Regression, Linear Support Vector Machine with loss of hinge, and Logistic Regression with loss of hinge. SGD Classifier generates a regularized linear model using SGD [42].

4.2.5 ***Perceptron Classifier***.

The Perceptron is a linear classification algorithm. It is a type of neural network mode. The input, which is frequently represented as a sequence of vectors, is examined to determine which of several categories it belongs to. A perceptron is a single-layer neural network. These algorithms are made up of four major components: input values, weights and biases, a net sum, and an activation function. A perceptron is a single-layer neural network, whereas a neural network is a multi-layer perceptron. A linear classifier is an example of a perceptron classifier (binary). In addition, this strategy is used in supervised learning. It makes it easier to classify input data. A perceptron is a mathematical device that divides data into two distinct segments. Following that, it is described using a Linear Binary Classifier [15].

5 RESULTS

5.1 Imbalanced Learning

As previously stated, this dataset is not evenly distributed, with the neutral class being three times larger than the negative class and the positive class being 2.5 times larger than the negative class. In this experiment, we attempted to fit the model to various types of data (original, down sampling, oversampling). Following that, we evaluated the performance of each sampling method.

Furthermore, we used the logistic linear with Tf-IDF Vectorizer to compare each sampling method. TFIDF is an abbreviation for Term Frequency-Inverse Document Frequency, and it is yet another method for converting textual data to numerical form. Combining these two terms, TF and IDF, yields the vector value. For validation, the K-fold Cross Validation (KCV) technique was used. The K-fold Cross Validation technique is a popular machine learning technique for model selection and error estimation in classifiers. A subset of the subsets is used to train the model iteratively, while

the remaining subsets are used to assess its performance [4]. Due to the small size of the dataset (41705 tweets), we did not split it for testing and validation, and thus risked missing out on interesting insights.

5.2 Original Imbalanced Data

We evaluated the performance with the original imbalanced data by using a linear regression model with TF-Idf. Figure 1 depicts the outcome. Without applying any resampling to the dataset, we noticed that the overall precision is greater than the recall. The negative class has the low recall, with around 34-36%, and high precision 58-66%. A low recall value indicates that the classifier used has a high number of False Negatives as a result of an imbalanced dataset. The neutral class has the highest recall and f1 score among the others, with 78-79 % recall and 70-71 % f1 score for each fold. In general with the imbalanced dataset we have high precision but low recall.

```
              negative    neutral     positive
precision: [0.65132497 0.64671039 0.70624784]
recall:    [0.36006168 0.78505393 0.64952381]
f1 score:  [0.46375372 0.70919847 0.67669919]
--------------------------------------------------
              negative    neutral     positive
precision: [0.58661417 0.64597066 0.72461752]
recall:    [0.34464148 0.78037503 0.66137734]
f1 score:  [0.43419136 0.70684039 0.69155467]
--------------------------------------------------
              negative    neutral     positive
precision: [0.63932107 0.65095137 0.70764463]
recall:    [0.34849653 0.79090676 0.65217391]
f1 score:  [0.4510978  0.71413661 0.67877787]
--------------------------------------------------
              negative    neutral     positive
precision: [0.66248257 0.64938896 0.72168172]
recall:    [0.36622976 0.79167737 0.65915582]
f1 score:  [0.47169811 0.71350851 0.68900315]
--------------------------------------------------
              negative    neutral     positive
precision: [0.63224894 0.64549266 0.72101955]
recall:    [0.34464148 0.79090676 0.65534751]
f1 score:  [0.44610778 0.7108392  0.68661679]
--------------------------------------------------
accuracy: 67.02% (+/- 0.28%)
precision: 66.61% (+/- 0.81%)
recall: 59.87% (+/- 0.36%)
f1 score: 61.63% (+/- 0.46%)
```

Figure 1: Original Imbalanced Data with Linear Regression

5.3 Oversampling

Oversampling with the imbalanced class resulted in an increase in the number of observations that are simply duplicates of previous samples drawn at random from a larger pool of data. We require a sufficient number of samples in order to conduct experiments effectively. There are numerous oversampling techniques available; in this experiment, we will use two of them:

(1) RandomOverSampling
(2) SMOTE (Synthetic Minority Over-Sampling Technique)

When using oversampled data for cross validation, we must consider the possibility of "overfitting". The phrase "overfitting" refers to a model that accurately models the "training data". Overfitting occurs when a model learns so much detail and noise from training data that it lowers the model's performance on new data. Overfitting happen in this case because we are leaking the information that already occur in the training set so it's like training with duplicate dataset. When the "lr_cv()" cross validation function in the source code is utilized, the pipeline will only be fitted with data from the training set after it has been cross-validated. As a result, no validation set information will be leaked to the model.

5.3.1 RandomOverSampling.

RandomOverSampling compensates for the minority class's lack of data points by generating additional data points from the majority class. Because both classes are equally represented, the classifier's decision function can reach well-informed conclusions. Oversampling occurs as a result of the replication process and increases the weight of the minority class. Oversampling does not add any new information to the dataset, but by increasing the number of samples used, it increases the likelihood of overfitting [5] .

To illustrate how oversampling works, suppose we have 100 samples in our minority class and 1000 samples in the majority class. It will randomly select 100 samples from the minority class and place them in the minority class's dataset. As an example, we'll start at row 89 and work our way up the page, repeating the process until the majority class has the most points. The random oversampling algorithm works in this manner.

```
              negative    neutral     positive
precision: [0.50468457 0.69066667 0.70635452]
recall:    [0.6229761  0.66512583 0.67047619]
f1 score:  [0.55762595 0.67765568 0.68794788]
--------------------------------------------------
              negative    neutral     positive
precision: [0.49165673 0.69737562 0.71148087]
recall:    [0.63608327 0.65527871 0.67851476]
f1 score:  [0.55462185 0.6756721  0.69460689]
--------------------------------------------------
              negative    neutral     positive
precision: [0.50428922 0.69771198 0.70474282]
recall:    [0.63454125 0.66581043 0.66962869]
f1 score:  [0.56196654 0.68138801 0.68673718]
--------------------------------------------------
              negative    neutral     positive
precision: [0.51537979 0.69749671 0.72265493]
recall:    [0.63299923 0.67993835 0.67724532]
f1 score:  [0.56816609 0.68860562 0.69921363]
--------------------------------------------------
              negative    neutral     positive
precision: [0.4996873  0.69023034 0.71905565]
recall:    [0.61603701 0.6696635  0.6766106 ]
f1 score:  [0.55179558 0.6797914  0.6971877 ]
--------------------------------------------------
accuracy: 66.39% (+/- 0.40%)
precision: 63.69% (+/- 0.43%)
recall: 65.67% (+/- 0.36%)
f1 score: 64.42% (+/- 0.40%)
```

Figure 2: RandomOverSampling with Linear Regression

We noticed an improvement in the recall score in this experiment when compared to the previous experiment as in figure 2. Previously, the recall score of the negative class was only 34-36 percent; however, after implementing the RandomOverSampling, the recall score increased to around 61-62 percent. In contrast, the precision score for the negative class decreased from 58-66 percent to 49-51 percent. This experiment showed high recall but low precision, implying a high false positive rate. Some texts were incorrectly classified as negative when, in fact, some were negative but a large number were not.

Overall, RandomOverSampling increases the overall f1 score, which is the weighted average of Precision and Recall, from 61.63% to 64.42% when the data is imbalanced. However, accuracy is slightly lower with RandomOverSampling at 64.42%, while it is 3% higher with imlabanced data at 67.02%.

5.3.2 *SMOTE (Synthetic Minority Oversampling Technique).*

Oversampling can be achieved by simply reproducing the minority class's existing elements on the training set, as previously mentioned in the RandomOverSampling section. This technique, however, is well known to be prone to overfitting [8, 27]. To mitigate this risk, additional samples could be generated artificially by accounting for the distribution of the minority class. One method is to use the Synthetic Minority Over-sampling Technique (SMOTE).

```
              negative    neutral     positive
precision: [0.50944584 0.69469496 0.7083473 ]
recall:    [0.62374711 0.67257319 0.67079365]
f1 score:  [0.56083189 0.68345511 0.68905919]
--------------------------------------------------
              negative    neutral     positive
precision: [0.49629172 0.69697787 0.71627444]
recall:    [0.61912105 0.66349859 0.68581403]
f1 score:  [0.5509434  0.67982629 0.70071336]
--------------------------------------------------
              negative    neutral     positive
precision: [0.50381194 0.69754405 0.70109235]
recall:    [0.61141095 0.67120473 0.67216757]
f1 score:  [0.55242076 0.68412096 0.68632534]
--------------------------------------------------
              negative    neutral     positive
precision: [0.52451613 0.69104633 0.71888776]
recall:    [0.62683115 0.68199332 0.67280228]
f1 score:  [0.5711275  0.68648998 0.69508197]
--------------------------------------------------
              negative    neutral     positive
precision: [0.50124069 0.69045093 0.71916188]
recall:    [0.6229761  0.66863601 0.67534116]
f1 score:  [0.55551736 0.67936839 0.69656301]
--------------------------------------------------
accuracy: 66.51% (+/- 0.26%)
precision: 63.80% (+/- 0.36%)
recall: 65.59% (+/- 0.28%)
f1 score: 64.48% (+/- 0.33%)
```

Figure 3: SMOTE (Synthetic Minority Oversampling Technique)

The following procedure is followed to create synthetic samples: Assume that there is a difference between the feature vector under examination (sample) and the feature vector of its nearest neighbor. This difference should then be multiplied by a random value between 0 and 1 and added to the corresponding feature vector. This action selects a random location along the line segment connecting two distinct features. By employing this strategy, the minority class's decision region is effectively compelled to become more diverse." This means that when SMOTE generates new synthetic data, it will choose one data to duplicate and examine its k nearest neighbors before creating the new synthetic data. It will generate random values in feature space between the initial sample and its neighbors, and this process will be repeated until the feature space is exhausted [7]. In comparison to RandomOverSampling, as shown in figure 3 SMOTE performed better in all metric scores, with slightly higher accuracy as well as rest precision, recall, and f1 score. SMOTE sampling appears to have the same problem as RandomOverSampling, resulting in high recall but low precision for the negative.

Overall, when testing with oversampling methods, SMOTE outperformed RandomOverSampling.

5.4 Downsampling

Down-sampling is the process of randomly selecting and discarding observations from the majority class in order to prevent the majority class signal from dominating the learning algorithm during the learning phase [35]. The most frequently used heuristic for this is resampling without replacement.

This is analogous to the previous procedure known as oversampling. The steps are as follows: To begin, we will separate observations into DataFrames for each class we will analyze. In the following step, the majority class will be resampled without replacement with a sample size equal to that of the minority class. Finally, we'll connect the downsampled majority class DataFrame to the original minority class DataFrame to complete the transformation.

5.4.1 *RandomUnderSampler.*

RandomUnderSampler eliminates the majority of class occurrences at random to create a balanced data set [36]. As a result, the adjusted version of the training dataset reduces the number of examples in the majority class by a factor of two. This technique can be repeated as many times as needed to achieve the desired class distribution, such as an equal number of samples in each of the three classes [14].

This strategy may be more appropriate for datasets with a class imbalance as we encounter in this study case; however, if the minority class has a sufficient number of examples, a useful model can be fit using this method. Undersampling has the disadvantage of removing samples from the majority class that could be useful, significant, or even critical in fitting a robust decision boundary to a given decision boundary model. Because examples are deleted at random, there is no way to identify or save "good" or more information-rich examples from the majority class, which is a problem because they are deleted at random [14].

The overall accuracy is lower after implementing the RandomUnderSampler compared to the first experiment with the imbalanced data. The characteristics of low precision and high recall remain in this experiment, and the overall f1 score is significantly lower than in the previous.

```
              negative    neutral     positive
precision: [0.43133641 0.69519899 0.68186356]
recall:    [0.72166538 0.56522856 0.65047619]
f1 score:  [0.53994808 0.62351275 0.66580016]
--------------------------------------------------
              negative    neutral     positive
precision: [0.42179885 0.70188557 0.68783245]
recall:    [0.73400154 0.55458515 0.65661695]
f1 score:  [0.53573438 0.61960109 0.67186232]
--------------------------------------------------
              negative    neutral     positive
precision: [0.43217232 0.69839774 0.67977151]
recall:    [0.72706245 0.57102492 0.64201841]
f1 score:  [0.5421098  0.62832109 0.6603558 ]
--------------------------------------------------
              negative    neutral     positive
precision: [0.43567518 0.71061919 0.69195251]
recall:    [0.73631457 0.56896995 0.66582037]
f1 score:  [0.5474348  0.63195435 0.67863497]
--------------------------------------------------
              negative    neutral     positive
precision: [0.42614679 0.69604768 0.69492096]
recall:    [0.71626831 0.56999743 0.65566487]
f1 score:  [0.53436871 0.62674763 0.6747224 ]
--------------------------------------------------
accuracy: 62.43% (+/- 0.39%)
precision: 60.57% (+/- 0.37%)
recall: 64.90% (+/- 0.41%)
f1 score: 61.21% (+/- 0.38%)
```

Figure 4: Random Under Sampler with Linear Regression

5.4.2 NearMiss.

It is referred to as a near-miss algorithm, and it can be used to help balance an unbalanced dataset. It is a technique for data balancing that is classified as an undersampling algorithm. This is accomplished by analyzing the larger class's distribution and randomly removing samples from it. When two points in the distribution belong to different classes and are reasonably close to one another, this strategy eliminates the data point from the larger class in an attempt to restore the distribution's balance. In contrast to Random Undersampling, which takes samples from the majority class at random without respect for any rules, NearMiss incorporates a number of heuristics to ensure that samples from the majority class are representative [17].

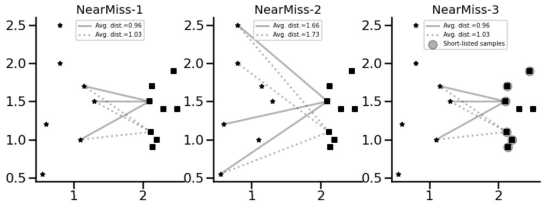

Figure 5: Three version of NearMiss[17]

The steps of this algorithm are as follows: first, the algorithm computes and divides the distance between all points in the larger and smaller classes. This may make the undersampling technique easier. Choose instances of the larger class that are the most similar to examples of the smaller class. This collection of n classes must be saved before they can be deleted. The function returns m*n instances of the larger class if the smaller class has m instances. There are three version of NearMiss as mention in figure 5.

3.4.2.1 Near Miss 1.

NearMiss-1 selects the majority class example with the shortest mean distance to the next N samples[17]. The NearMiss-1 keeps track of the majority-class locations that are the shortest mean distance from the N nearest points in the minority-class. In other words, it will keep all of the points of the majority class that are similar to those of the minority class.

When compared to random undersampling, the accuracy, F1 score, and precision appear to have declined as shown in figure 6. However, due to undersampling of the majority class, the recall score is still higher than the precision.

```
              negative    neutral     positive
precision: [0.41171171 0.69303136 0.65672101]
recall:    [0.70470316 0.51078582 0.67777778]
f1 score:  [0.51976116 0.58811354 0.66708327]
--------------------------------------------------
              negative    neutral     positive
precision: [0.40521114 0.70330056 0.66483012]
recall:    [0.69545104 0.51451323 0.68930498]
f1 score:  [0.51206358 0.59427385 0.67684637]
--------------------------------------------------
              negative    neutral     positive
precision: [0.41192412 0.69941197 0.65327565]
recall:    [0.70316114 0.51939378 0.67089813]
f1 score:  [0.51951011 0.59610849 0.66196963]
--------------------------------------------------
              negative    neutral     positive
precision: [0.41286863 0.70930638 0.66512059]
recall:    [0.71241326 0.52273311 0.68264043]
f1 score:  [0.52277228 0.60189293 0.67376664]
--------------------------------------------------
              negative    neutral     positive
precision: [0.4015887  0.69463315 0.67007346]
recall:    [0.70161912 0.52530182 0.66582037]
f1 score:  [0.5108055  0.59821559 0.66794015]
--------------------------------------------------
accuracy: 60.73% (+/- 0.31%)
precision: 59.02% (+/- 0.31%)
recall: 63.31% (+/- 0.32%)
f1 score: 59.41% (+/- 0.29%)
```

Figure 6: Near Miss 1 with Linear Regression

3.4.2.2 Near Miss 2.

NearMiss-2, in contrast to NearMiss-1, keeps the points from the majority class that have the shortest mean distance to the minority class's N furthest points. In other words, it will retain dominant-class characteristics that are diametrically opposed to minority-class characteristics.

As illustrated in Figure 7, NearMiss-2 performs the worst of the NearMiss variants. The accuracy has dropped slightly to 56.36 percent, and the other metric scores are also lower.

```
             negative    neutral     positive
precision: [0.33019786 0.6789511  0.71674877]
recall:    [0.7848882  0.49203903 0.55428571]
f1 score:  [0.46484018 0.57057772 0.62513426]
--------------------------------------------------
             negative    neutral     positive
precision: [0.33080974 0.67800772 0.73817292]
recall:    [0.77486507 0.49653224 0.57442082]
f1 score:  [0.46366782 0.5732503  0.64608246]
--------------------------------------------------
             negative    neutral     positive
precision: [0.3243071  0.68515797 0.72379032]
recall:    [0.7848882  0.47906499 0.56966043]
f1 score:  [0.45897205 0.56386999 0.63754218]
--------------------------------------------------
             negative    neutral     positive
precision: [0.32366845 0.68333333 0.74335548]
recall:    [0.79182729 0.48445929 0.56807363]
f1 score:  [0.45950783 0.56696227 0.64400072]
--------------------------------------------------
             negative    neutral     positive
precision: [0.32596336 0.68211441 0.74222959]
recall:    [0.79568234 0.48394554 0.56839099]
f1 score:  [0.46246919 0.56619083 0.64378145]
--------------------------------------------------
accuracy: 56.39% (+/- 0.30%)
precision: 58.05% (+/- 0.33%)
recall: 61.35% (+/- 0.23%)
f1 score: 55.65% (+/- 0.28%)
```

Figure 7: Near Miss 2 with Linear Regression

5.4.3 Near Miss 3.

The NearMiss-3 algorithm is a two-stage algorithm that combines the best features of the NearMiss-1 and NearMiss-2 algorithms. To begin, it keep the N nearest neighbors of each majority class sample. Finally, the majority-class examples are those with the greatest average distance between them and their N nearest neighbors [17].

```
             negative    neutral     positive
precision: [0.44832905 0.68180333 0.63628186]
recall:    [0.67232074 0.53595275 0.67365079]
f1 score:  [0.53793954 0.60014378 0.65443331]
--------------------------------------------------
             negative    neutral     positive
precision: [0.44078947 0.67963022 0.64807931]
recall:    [0.67154973 0.54764963 0.66391622]
f1 score:  [0.53223342 0.60654339 0.65590218]
--------------------------------------------------
             negative    neutral     positive
precision: [0.45724713 0.68039591 0.63050744]
recall:    [0.67617579 0.54739276 0.6585211 ]
f1 score:  [0.54556765 0.60669039 0.64420987]
--------------------------------------------------
             negative    neutral     positive
precision: [0.45594823 0.70577452 0.6819222 ]
recall:    [0.70624518 0.59337272 0.66201206]
f1 score:  [0.55414398 0.64471114 0.67181965]
--------------------------------------------------
             negative    neutral     positive
precision: [0.4340176  0.68777568 0.66933675]
recall:    [0.6846569  0.56075006 0.66296414]
f1 score:  [0.53125935 0.61780105 0.6661352 ]
--------------------------------------------------
accuracy: 61.70% (+/- 1.05%)
precision: 59.59% (+/- 0.98%)
recall: 63.45% (+/- 1.03%)
f1 score: 60.46% (+/- 0.98%)
```

Figure 8: Near Miss 3 with Linear Regression

Overall, as shown in figure 8, NearMiss 3 outperforms other types of NearMiss, but it still falls short of RandomUnderSampling in terms of accuracy, precision, recall, and f1 score.

	Before			After		
	Neutral	*Positive*	*Negative*	*Neutral*	*Positive*	*Negative*
NearMiss-1	19466	15754	6485	6485	6485	6485
NearMiss-2	19466	15754	6485	6485	6485	6485
NearMiss-3	19466	15754	6485	6485	6485	6485

Figure 9: Distribution before and after applying NearMiss-1, NearMiss-2, and NearMiss-3

Figure 9 shows that after implementing all variants of NearMiss, the class distribution becomes balanced for all three classes (neutral, positive, and negative).

6 DISCUSSION

Based on the results in figure 10, we can see that for data imbalance techniques that use either oversampling or downsampling, the SMOTE method has the highest accuracy among the others, as well as good precision, recall, and f1-score.

	accuracy	macro average precision	macro average recall	macro average f1 score
Original Imbalanced data	67.02% (+/- 0.28)	66.61% (+/- 0.81)	59.87% (+/- 0.36)	61.63% (+/- 0.46)
-oversampling-				
RandomOver Sampler	66.39% (+/- 0.40)	63.69% (+/- 0.43)	65.67% (+/- 0.36)	64.42% (+/- 0.40)
SMOTE	66.51% (+/- 0.26)	63.80% (+/- 0.36)	65.59% (+/- 0.28)	64.48% (+/- 0.33)
-downsampling-				
DownUnder Sampler	62.43% (+/- 0.39)	60.57% (+/- 0.37)	64.90% (+/- 0.41)	61.21% (+/- 0.38)
NearMiss-1	60.73% (+/- 0.31)	59.02% (+/- 0.31)	63.31% (+/- 0.32)	59.41% (+/- 0.29)
NearMiss-2	56.39% (+/- 0.30)	58.05% (+/- 0.33)	61.35% (+/- 0.23)	55.65% (+/- 0.28)
NearMiss-3	61.70% (+/- 1.05)	59.59% (+/- 0.98)	63.45% (+/- 1.03)	60.46% (+/- 0.98)

Figure 10: The results of a 5-fold cross validation experiment (Using Logistic Regression Without Tuning Hyperparameters)

Additionally, we evaluated the SMOTE method with various classifiers to determine how well it performed. Numerous previous publications have demonstrated that Logistic Regression performs

the best in this scenario, which is also supported by the results of this experiment, which are shown in Figure 11. Logistic regression is one of the most effective classifiers for text datasets.

	accuracy	macro average precision	macro average recall	macro average f1 score
Logistic Regression	*66.46% (+/- 0.69%)*	*63.77% (+/- 0.77%)*	*65.63% (+/- 0.86%)*	*64.48% (+/- 0.81%)*
Random Forest	53.25% (+/- 1.56%)	51.70% (+/- 1.71%)	50.97% (+/- 1.29%)	49.62% (+/- 1.47%)
KNN	42.68% (+/- 6.23%)	50.48% (+/- 2.49%)	39.66% (+/- 5.08%)	30.50% (+/- 3.99%)
Perceptron	63.59% (+/- 0.57%)	61.19% (+/- 0.66%)	59.89% (+/- 0.72%)	60.41% (+/- 0.64%)
Stochastic Gradient Descent	64.19% (+/- 0.44%)	62.19% (+/- 0.40%)	66.44% (+/- 0.41%)	62.88% (+/- 0.45%)

Figure 11: The results of a 5-fold cross validation experiment with SMOTE (Using Logistic Regression, Random Forest, KNN, Perceptron, Stochastic Gradient Descent)

Because the size of the dataset after implementing imbalanced learning is small for each class, it affects the performance of the classifiers being tested, particularly Random Forest and KNN. When the training dataset is large enough, this classifier performs well. Furthermore, after performing the preprocessing step, particularly when removing the repeating character, the result was still ambiguous. For example, removing the unnecessary 'o' character from the word 'soooo' and leaving it with only two 'o' characters. Because 'so' and 'soo' have slightly different meanings, this may cause confusion. Some of the datasets are not correctly parsed; for example, there are two datasets that are not properly separated using a separator, causing the tweets to stack on top of one another. In the end, we decided to disregard this dataset and work with the other nine. As a result, we have a slightly smaller dataset to process, which helps the model perform. Despite these constraints, we were able to overcome the high precision low recall problem that occurred with the original imbalanced data using the SMOTE method from oversampling. When combined with the Logistic Regression classifier, this method outperformed all other imbalanced learning methods.

7 CONCLUSION

The provided dataset is unbalanced and noisy, which required extensive preprocessing. There are 11 datasets provided, but we removed two due to improper file formatting. Some entries appear to be incorrectly tab separated, causing some tweets to become stuck together and exceed the standard tweet length of 140 characters. The dataset is divided into three classes (neutral, positive, and negative), and the distribution of these classes is not balanced. Imbalanced data leads to overfitting, in which the model performs well in the majority class but poorly in the minority class. To address this issue, we used undersampling and oversampling with various methods available in both. We discovered that the SMOTE method from oversampling performs well and provides the highest accuracy among the others as mention in figure 11.

Moreover, previous research has shown that logistic regression performs well with SEMEVAL datasets, which is supported by the experiment that we conducted in this paper as well, as shown in figure 11.

8 CONTRIBUTIONS OF THE TEAM MEMBERS

Task 4 Subtask A of SEMEVAL 2017 focuses on Message Polarity Classification (positive, negative, or neutral sentiment). We implemented various imbalanced learning methods such as Oversampling (RandomOverSampling and SMOTE (Synthetic Minority Over-Sampling Technique) and Undersampling (RandomUnderSampling, NearMiss-1, NearMiss-2, and Near-Miss-3). In addition, we tested various classifiers such as Perceptron, Random Forest, Logistic Regression, and Stochastic Gradient Desent.

Both authors (as indicated by their names in the project title) contributed equally to the completion of this paper. In terms of the code, the first author focused on the pre-processing step, while the second author worked extensively on testing the model with various imbalanced learning methods, and the first author worked on testing with different classifiers after discovering that SMOTE performs the best among the other imbalanced learning methods. The first author is in charge of the paper's introduction, related work, data, and discussion, while the second author is in charge of the method and results section. The conclusion is written collaboratively by both authors.

REFERENCES

[1] Ajith Abraham, Jaime Lloret Mauri, John Buford, Junichi Suzuki, and Sabu M Thampi. 2011. *Advances in Computing and Communications, Part I: First International Conference, ACC 2011, Kochi, India, July 22-24, 2011. Proceedings*. Vol. 190. Springer Science & Business Media.

[2] Charu C Aggarwal et al. 2016. *Recommender systems*. Vol. 1. Springer.

[3] Syed Thouheed Ahmed, Syed Muzamil Basha, Sajeev Ram Arumugam, and Mallikarjun M Kodabagi. 2021. *Pattern Recognition: An Introduction*. MileStone Research Publications.

[4] Davide Anguita, Luca Ghelardoni, Alessandro Ghio, Luca Oneto, and Sandro Ridella. 2012. The 'K'in K-fold cross validation. In *20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*. i6doc. com publ, 441–446.

[5] Francis Effirim Botchey, Zhen Qin, and Kwesi Hughes-Lartey. 2020. Mobile Money Fraud Prediction—A Cross-Case Analysis on the Efficiency of Support Vector Machines, Gradient Boosted Decision Trees, and Naïve Bayes Algorithms. *Information* 11, 8 (2020), 383.

[6] Danah Boyd, Scott Golder, and Gilad Lotan. 2010. Tweet, tweet, retweet: Conversational aspects of retweeting on twitter. In *2010 43rd Hawaii international conference on system sciences*. IEEE, 1–10.

[7] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.

[8] Nitesh V Chawla, Nathalie Japkowicz, and Aleksander Kotcz. 2004. Special issue on learning from imbalanced data sets. *ACM SIGKDD explorations newsletter* 6, 1 (2004), 1–6.

[9] Gilles Cohen, Mélanie Hilario, Hugo Sax, Stéphane Hugonnet, and Antoine Geissbuhler. 2006. Learning from imbalanced data in surveillance of nosocomial infection. *Artificial intelligence in medicine* 37, 1 (2006), 7–18.

[10] Montserrat Comesaña, Ana Paula Soares, Manuel Perea, Ana P Piñeiro, Isabel Fraga, and Ana Pinheiro. 2013. ERP correlates of masked affective priming with emoticons. *Computers in Human Behavior* 29, 3 (2013), 588–595.

[11] Thomas Edgar and David Manz. 2017. *Research methods for cyber security*. Syngress.

[12] Rajesh Sinha Priyankar Sinha Adarsh Pradhan Eriq-Ur Rahman, Rituparna Sarma. 2018. A Survey on Twitter Sentiment Analysis. *International Journal of Computer Sciences and Engineering* 6 (11 2018), 644–648. Issue 11. https://doi.org/10.26438/ijcse/v6i11.644648

[13] Aniruddha Ghosh, Guofu Li, Tony Veale, Paolo Rosso, Ekaterina Shutova, John Barnden, and Antonio Reyes. 2015. Semeval-2015 task 11: Sentiment analysis of figurative language in twitter. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*. 470–478.

[14] Haibo He and Yunqian Ma. 2013. Imbalanced learning: foundations, algorithms, and applications. (2013).

[15] Kamal Kant Hiran, Ritesh Kumar Jain, Kamlesh Lakhwani, and Ruchi Doshi. 2021. *Machine Learning: Master Supervised and Unsupervised Learning Algorithms with Real Examples (English Edition)*. BPB Publications.

[16] Bernard J Jansen, Mimi Zhang, Kate Sobel, and Abdur Chowdury. 2009. Twitter power: Tweets as electronic word of mouth. *Journal of the American society for information science and technology* 60, 11 (2009), 2169–2188.

[17] Annie Kim. 2021. *Optimal Selection of Resampling Methods for Imbalanced Data with High Complexity*. Ph.D. Dissertation. Department of Biostatistics and Computing and the Graduate School of Yonsei University.

[18] Svetlana Kiritchenko, Saif Mohammad, and Mohammad Salameh. 2016. Semeval-2016 task 7: Determining sentiment intensity of english and arabic phrases. In *Proceedings of the 10th international workshop on semantic evaluation (SEMEVAL-2016)*. 42–51.

[19] Patrick Lai. 2010. Extracting strong sentiment trends from Twitter.

[20] Guillaume Lemaître, Fernando Nogueira, and Christos K Aridas. 2017. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *The Journal of Machine Learning Research* 18, 1 (2017), 559–563.

[21] Inderjeet Mani and I Zhang. 2003. kNN approach to unbalanced data distributions: a case study involving information extraction. In *Proceedings of workshop on learning from imbalanced datasets*, Vol. 126. ICML United States.

[22] Yasuhide Miura, Shigeyuki Sakaki, Keigo Hattori, and Tomoko Ohkuma. 2014. TeamX: A sentiment analyzer with enhanced lexicon mapping and weighting scheme for unbalanced data. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. 628–632.

[23] Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*. 31–41.

[24] Saif M Mohammad. 2016. Sentiment analysis: Detecting valence, emotions, and other affectual states from text. In *Emotion measurement*. Elsevier, 201–237.

[25] Maria Carolina Monard and GEAPA Batista. 2002. Learning with skewed class distributions. *Advances in Logic, Artificial Intelligence and Robotics* 85 (2002), 173–180.

[26] Aminu Muhammad, Nirmalie Wiratunga, and Robert Lothian. 2014. A hybrid sentiment lexicon for social media mining. In *2014 IEEE 26th International Conference on Tools with Artificial Intelligence*. IEEE, 461–468.

[27] Iman Nekooeimehr and Susana K Lai-Yuen. 2016. Adaptive semi-unsupervised weighted oversampling (A-SUWO) for imbalanced datasets. *Expert Systems with Applications* 46 (2016), 405–416.

[28] Elisavet Palogiannidi, Athanasia Kolovou, Fenia Christopoulou, Filippos Kokkinos, Elias Iosif, Nikolaos Malandrakis, Harris Papageorgiou, Shrikanth Narayanan, and Alexandros Potamianos. 2016. Tweester at SemEval-2016 Task 4: Sentiment analysis in Twitter using semantic-affective model adaptation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. 155–163.

[29] Georgios Paltoglou, Stéphane Gobron, Marcin Skowron, Mike Thelwall, and Daniel Thalmann. 2010. Sentiment analysis of informal textual communication in cyberspace. *In Proc. Engage 2010, Springer LNCS State-of-the-Art Survey* (2010), 13–25.

[30] Georgios Paltoglou and Mike Thelwall. 2012. Twitter, MySpace, Digg: Unsupervised sentiment analysis in social media. *ACM Transactions on Intelligent Systems and Technology (TIST)* 3, 4 (2012), 1–19.

[31] Bo Pang and Lillian Lee. 2008. Opinion Mining and Sentiment Analysis. *Found. Trends Inf. Retr.* 2, 1–2 (jan 2008), 1–135. https://doi.org/10.1561/1500000011

[32] Maria Pontiki, Dimitrios Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad Al-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, et al. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. In *International workshop on semantic evaluation*. 19–30.

[33] Maria Pontiki, Dimitrios Galanis, Harris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. Semeval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*. 486–495.

[34] Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. SemEval-2014 Task 4: Aspect Based Sentiment Analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Association for Computational Linguistics, Dublin, Ireland, 27–35. https://doi.org/10.3115/v1/S14-2004

[35] Alireza Rahnama, Sam Clark, and Seetharaman Sridhar. 2018. Machine learning for predicting occurrence of interphase precipitation in HSLA steels. *Computational Materials Science* 154 (2018), 169–177.

[36] Farshid Rayhan, Sajid Ahmed, Asif Mahbub, Rafsan Jani, Swakkhar Shatabda, and Dewan Md Farid. 2017. Cusboost: Cluster-based under-sampling with boosting for imbalanced classification. In *2017 2nd International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS)*. IEEE, 1–5.

[37] Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th international workshop on semantic evaluation (SemEval-2017)*. 502–518.

[38] Sara Rosenthal and Kathleen McKeown. 2016. Social proof: The impact of author traits on influence detection. In *Proceedings of the First Workshop on NLP and Computational Social Science*. 27–36.

[39] Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif M Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. SemEval-2015 Task 10: Sentiment Analysis in Twitter. 451-463. In *Proc. of the 9th International Workshop on Semantic Evaluation*.

[40] Andrew P Sage, RM HARALICK, HE KOENIG, MH MICKLE, CP NEUMAN, HL OESTREICHER, and F DICESARE. 1979. IEEE Transactions on Systems, Man, and Cybernetics. (1979).

[41] Carlo Strapparava and Rada Mihalcea. 2007. Semeval-2007 task 14: Affective text. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*. 70–74.

[42] Meenakshi Tripathi and Sushant Upadhyaya. 2021. *Conference Proceedings of ICDLAIR2019*. Vol. 175. Springer Nature.

[43] Andranik Tumasjan, Timm Sprenger, Philipp Sandner, and Isabell Welpe. 2010. Predicting elections with twitter: What 140 characters reveal about political sentiment. In *Proceedings of the International AAAI Conference on Web and Social Media*, Vol. 4.

[44] Hao Wang, Doğan Can, Abe Kazemzadeh, François Bar, and Shrikanth Narayanan. 2012. A system for real-time twitter sentiment analysis of 2012 us presidential election cycle. In *Proceedings of the ACL 2012 system demonstrations*. 115–120.

[45] Sabih Bin Wasi, Rukhsar Neyaz, Houda Bouamor, and Behrang Mohit. 2014. CMUQ@ Qatar: Using rich lexical features for sentiment analysis on Twitter. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. 186–191.

[46] Zhihua Zhang, Guoshun Wu, and Man Lan. 2015. Ecnu: Multi-level sentiment analysis on twitter using traditional linguistic features and word embedding features. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. 561–567.

[47] Yunxiao Zhou, Man Lan, and Yuanbin Wu. 2017. ECNU at SemEval-2017 Task 4: Evaluating Effective Features on Machine Learning Methods for Twitter Message Polarity Classification. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. 812–816.