COMP90049 – Assignment 3

The Effect of Pre-processing in Sentiment Analysis

Anonymous

1 Introduction

Sentiment Analysis (SA) is a study of how language can be expressed by views, attitudes, emoticons, and viewpoints. Due to the rapid expansion of social networks and the rise of big data, SA has been used in several areas to address practical issues, such as analysing customer feedback, brand research, public opinion, financial prediction, and so on. As a result, SA has emerged as a significant and popular study area, attracting a huge number of researchers from the fields of machine learning, data mining, and natural language processing (NLP).

Twitter is among the most significantly and widely used social network platforms in the digital world. Twitter users' tweet about almost every topic, each tweet is limited to 140 characters, which encourages the usage of abbreviations, irregular phases, emoji/emoticons, and uncommon terms. This phenomenon increases data sparsity, which has an impact on the effectiveness of Twitter sentiment classifiers [1]. As a results, makes Twitter an even better source of sentiment data, because users may express their thoughts or views about anything as it happens.

When applying SA on a set of twitter data, there is a set of processes to be followed, such as pre-processing, baseline classifier implementation, training classifiers on training data, and tuning classifiers to produce best predictions.

Pre-processing is a crucial step in sentiment analysis, as described in multiple papers, given that they can increase the accuracy of an overall prediction [2] [3]. This paper will focus on the importance of pre-processing and the effects of multiple pre-processing techniques. These techniques include but not limited to the replacement of emoji & emoticon with their respective description (emo), emoji & emoticon removal (no_emo), and stop-words filtering. In addition, three data representations are evaluated when vectorised: unigram, bigrams, and 1-to-3 grams.

The twitter data provided for this research paper are provided from one large set of data split into three parts using a holdout strategy: training, development, and test. A set of pre-processed data are supplied however, due to this paper's focus being the effect of processing, the provided data will not be used, and only full data set will be used. Once the data is pre-processed using variety of techniques mentioned above, the following phases of SA will be conducted, first, training phase where the classifier is trained using training data. Second, development phase where the performance of the trained classifiers will be observed. Third, by combining the best pre-process technique and n-gram representation, the test data is predicted with the best performing classifier from development phase. Finally, the predicted target label is submitted onto an online Kaggle competition.

The rest of the paper is arranged as follows. First, literature review on related Twitter SA papers, with an emphasis on the approaches employed for pre-processing. Following that, datasets used for this paper. Then, the pre-processing techniques, and ML models employed. Finally, a discussion about the outcomes of the experiment and share the conclusions and future work.

2 Literature Review

Following is a literature reviews to be acquainted on the underlying problem and to identify any gaps that can be filled by this study.

2.1 Literature 1

"EXPLORE THE EFFECTS OF EMOTICONS ON TWITTER SENTIMENT ANALYSIS" **[4]**

In this literature, Wegrzyn-Wolska et al [4], analyse the sentiment polarity of the tweets by comparing three emoticon pre-processing methods.

The three pre-processing methods consist of the following:

- Emoticon deletion (emoDel)
  Delete all emoticons in each tweet processed
- Emoticons 2-valued translation (emo2label)
  Give all emoticons a 2-valued label of Negative or Positive
- Emoticon explanation (emo2explanation)
  Replacing the emoticon by their description, for example, ':)' replaced by 'smile' after pre-processing

Furthermore, an emoticon-weight lexicon method is proposed, where every emoticon is given a weight or either 1 or -1 based on their meaning. For example, (:), 1), (:(, -1).

The multiple pre-processing methods are then experimented based on Naive Bayes classifier. The findings suggest that using an emoticon-weight lexicon model enhances the performance of the NB model on the TSA test. However, between the first three pre-process methods, it's shown for a training size of 768 tweets, emoDel produces better accuracy than with emo2label or emo2explanation. While emo2explanation has slight a better accuracy than emo2label.

For this study, a comparison between emoDel and emo2explanation will be conducted.

2.2 Literature 2

"The Effects of Emoji in Sentiment Analysis" **[5]**

While this study may be similar to literature 1, Ayvaz et al investigates the usage of emojis and their effects on SA. Tweets were gathered from various positive and negative worldwide events to investigate the role of Emoji characters in sentiment analysis. Within their research, they discovered that using Emoji characters in sentiment analysis leads in better sentiment ratings. Furthermore, they discovered that using Emoji characters in sentiment analysis tended to have a greater influence on total feelings of good opinions than negative thoughts.

For this study, both emoticon and emoji's will be categorised as lexicons

2.3 Literature 3

"On Stopwords, Filtering and Data Sparsity for Sentiment Analysis of Twitter" **[6]**

In this literature, Saif et al, investigates the efficiency of Twitter sentiment classification by removing stopwords. Stopwords, by definition, are meaningless words with little discriminating power [7]. Six different stopwords identification methods are applied over six twitter datasets, they're as follows:

- Obama-McCain Debate dataset (OMD) [8].
- The Health Care Reform data (HCR) [9].
- The STS-Gold dataset [10].
- Two datasets from the Dialogue Earth project (GAS, WAB) [11].
- The SemEval dataset [12].

Due to the word limit only two stopword removal method will be summarised.
First method used is the classic method, by removing stopwords obtained from a pre-compiled list.
Another method used is Mutual Information MI, where characteristics that do not assist towards accurate classification are deemed stopwords and are subsequently deleted from the feature space.

Their results show that the use of pre-compiled lists of stopwords has a detrimental influence on the performance of Twitter sentiment classification. Whereas dynamically created stopword lists, appears to be the optimum technique for retaining good classification performance while lowering data sparsity and significantly downsizing the feature space.

For this study, a pre-compiled list of stopwords will be used to compare against emoji/emoticon replacement.

3 Data Sets

The twitter dataset provided were derived from the resources published in Vadicamo et al. [13] and Go et al [14]. Table 1 displays the number of tweets in each dataset and the number of positive, negative, and neutral tweets.

Dataset	# Tweets	# Pos Tweets	# Neg Tweets	# Neu Tweets
Train	159253	62447	64872	31934
Dev	19906	7916	8095	3895
Test	19906	?	?	?

Table 1: Statistics of 3 datasets used in this paper

The emoji library used is spacymoji from a public library within spaCy [15]. SpaCy is a dedicated python NLP library like NLTK. Code Snippet 1 displays the usage of spacymoji to show the description of an emoji within a tweet. The advantage of using spaCy compared to the NLTK TweetTokenizer is that spaCy's tokenizer will token emoji with skin tone's whereas nltk's tokenizer separates the emoji and skin tone that has harmful effects during pre-processing stage. However, this library is still limited to only emoji lookup.

```
import spacy
nlp = spacy.load("en_core_web_sm")
nlp.add_pipe("emoji", first=True)
doc = nlp("This is a test 😻 👍")
assert doc._.has_emoji is True
assert doc[4]._.is_emoji is True
assert
doc[5]._.emoji_desc == "thumbs up dark
skin tone"
```

Code Snippet 1: spacymoji library

The emoticon data used for this study is a python dictionary object derived from the python library emot [16]. This dictionary features over 200 emoticons commonly used and their description. Code Snippet 2 shows how to lookup the description of an emoticon ':)'.

```
>>> from emot.emo_unicode import EMOTICONS_EMO
>>> print(EMOTICONS_EMO[':)'])
Happy face or smiley
```

Code Snippet 2: Searching the description of emoticon

Lastly the stopwords list used for filtering are directly downloaded from the NLTK library, shown in figure 1 is the list of stopwords.

```
>>> from nltk.corpus import stopwords
>>> stop_words = stopwords.words('english')
>>> stop_words
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', '
you', "you're", "you've", "you'll", "you'd", 'your', 'yours',
yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she',
"she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself
', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'wh
ich', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those
', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'ha
ve', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a',
'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until
', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against
', 'between', 'into', 'through', 'during', 'before', 'after', 'a
bove', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on',
'off', 'over', 'under', 'again', 'further', 'then', 'once', 'he
re', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'bot
h', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no
', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'v
ery', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should'
, "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ai
n', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", '
doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven',
"haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn'
, "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "
shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't
", 'wouldn', "wouldn't"]
```

Figure 1: List of Stopwords from nltk python library

4 Pre-processing Techniques

The full twitter data is noisy in its raw form, they will be processed to reduce the number of features. The study by Symeonidis et al [17], recommends the following combination of pre-process for sentiment analysis: "replace URLs and user mentions, replace Contractions, remove Numbers, replace repetitions of punctuation, and lemmatizing" [17].

Taking these recommendation into consideration, three stages of pre-process can be formulated.

1. "Clean" Tweets
   - Remove URL & hyperlinks
   - Remove user @ mentions
   - Remove standalone #
   - Remove numbers/ numeric terms
2. Tokenise Tweets
3. Apply pre-processor
   - Replace emoji with meaning (optional)
   - Replace emoticons with meaning (optional)
   - Remove stopwords (optional)
   - Remove punctuations
   - Lemmatise tweet
   - Reduce words with repeated character endings such as 'happyyy' to 'happy'

Lemmatisation is the process of grouping many words to one, by morphologically examining a word and eliminating its inflectional ending, resulting in the base form or lemma as found in a dictionary [17].

4.1 Normalization

Once the tweets have been pre-processed, they will need to be normalised as part of data preparation for machine learning.

The chosen method is count or "Bag of Words" that maps each word to a unique id, and represented in a list of (ID, word_count) tuple [18]. For example, the string "im feeling so happy today, so very happy" will be mapped to {im:0, feeling:1, so:2, happy:3, today:4, very:5} and figure 2 displays how it's represented.

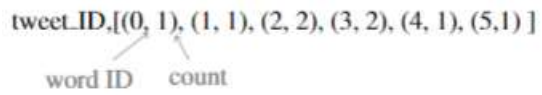

Figure 2: Count Representation of a string [18]

The represented tweet above in figure 2 is unigram considering only one word at a time. Similarly, bigram (using two words at a time, for example im feeling, feeling so, so happy, happy today and so on…), trigram (using three words at a time, for example im feeling so, feeling so happy, so happy today, and so on…), ngram (using combination of unigram, bigram, trigram, and n grams).

The data can be normalised and transformed into a count vector matrix by utilising the CountVectorizer function from sklearn library. Additionally, CountVectorizer allows ngram setting modification.

Referring to Krouska et al [2], unigram and 1-to3-gram representations outperform the others, and feature extraction increases classification performance. For this study , unigram, bigram, and 1-to-3-gram will be used to compare results.

5 SA Classifiers

Naïve Bayes is chosen as the baseline ML model for this study which is a suitable choice as a baseline model described by Copestake et al because it's an simpler model to implement [19].

The baseline model will use Multinomial Naïve Bayes with Count Vectors of unigram to classify the tweets. All three pre-process techniques will be used; emoji/emoticon removal, emoji/emoticon replacement, and stop word filtering.

To evaluate the effect of emoji and emoticons in sentiment classification, a well-known classifiers, Maximum Entropy (MaxEnt) and a less-known linear classifier with SGD (stochastic gradient descent) training from scikit-learn[20] have been chosen.

5.1 Maximum Entropy (MaxEnt)

Maximum Entropy classification (MaxEnt) is called Logistic Regression (LR) in the sklearn library, despite its name, logistic regression is a classification model rather than a regression model [21].

A logistic function is used to describe the probability defining the various outcomes of a single trial in this model.

5.2 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is a quick and easy way to fit linear classifiers to convex loss functions like (linear) Support Vector Machines and Logistic Regression [22].

This estimator uses SGD learning to build regularised linear models: the gradient of the loss is estimated one sample at a time, and the model is updated along the way using a decreasing strength schedule (aka learning rate) [23].

The accuracy and average F-measure performance of the baseline NB, Logistic Regression and SGD classifiers using the following pre-process methods:

- Emoji/Emoticon Replacement – emo2desc
- Emoji/Emoticon Removal – no_emo
- Stopwords filtering – sw / stopwords

and the following ngram settings will be reported:

- Unigram
- Bigram
- 1-to-3 Gram – 1_3gram

6 Results

6.1 Baseline NB Results

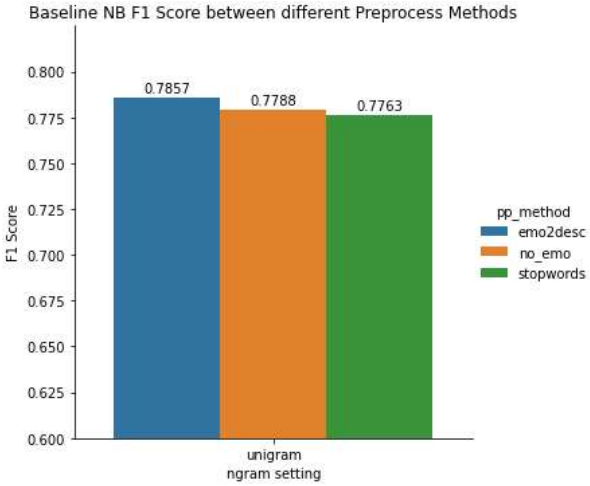

Figure 3: Baseline Accuracy Scores

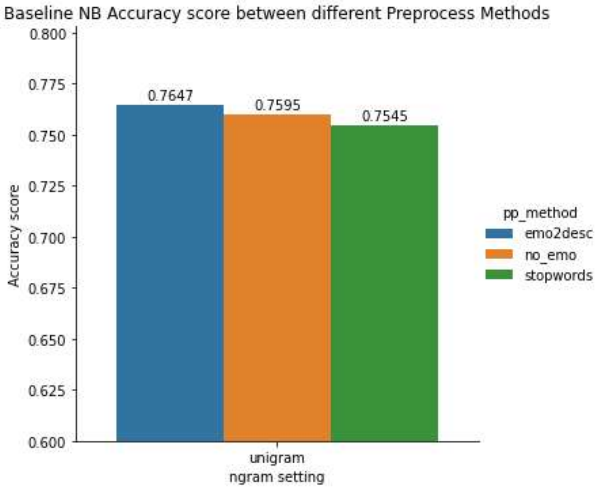

Figure 4: Baseline NB F1 Score

Based on the baseline NB model using all three pre-processing techniques and unigram, the accuracy and macro-level F1 scores are shown on Figure 3 and Figure 4.

6.2 Logistic Regression Results

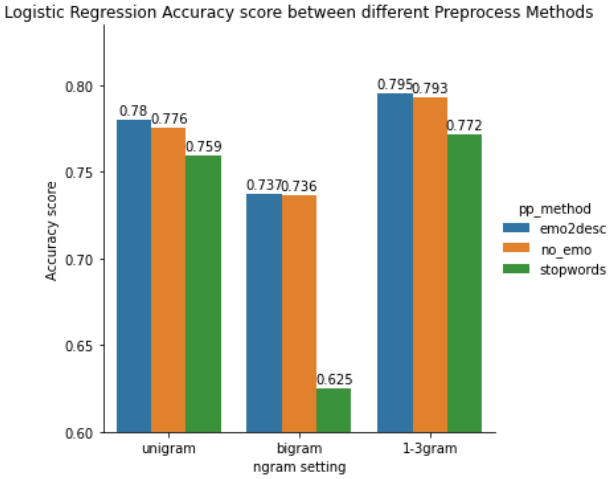

Figure 5: Logistic Regression Accuracy Score using different pre-process techniques and ngrams

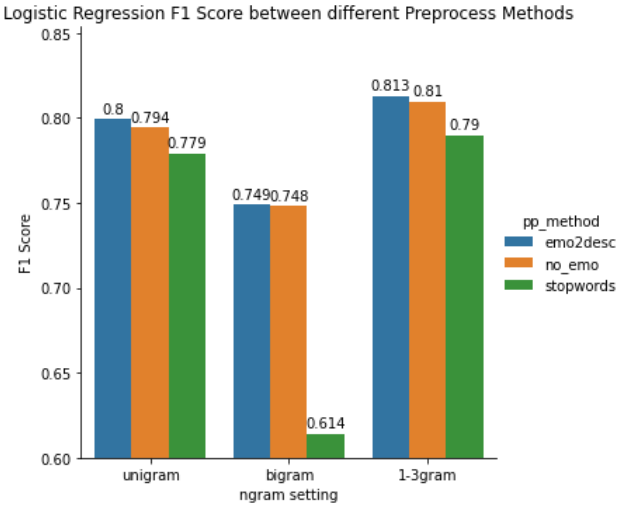

Figure 6: Logistic Regression F1 Score using different pre-process techniques and ngrams

Figure 5 shows that between the three pre-processing techniques, emo2desc performed the best with an accuracy score of 0.795 with 1-3 grams. While stopwords performed the worst with an accuracy score of 0.625 with bigram. Furthermore, there is a significant drop in accuracy and F1 when using stopwords filtering and bigram compared to unigram and 1-3 gram as shown in Figure 5 and Figure 6.

6.3 SGD Classifier Results

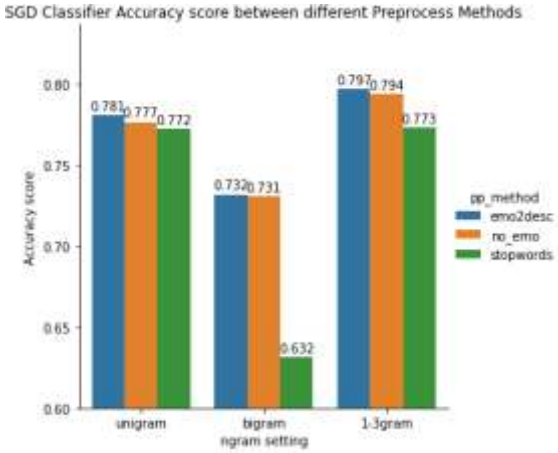

Figure 7: SGD Classifier Accuracy Score using different pre-process techniques and ngrams

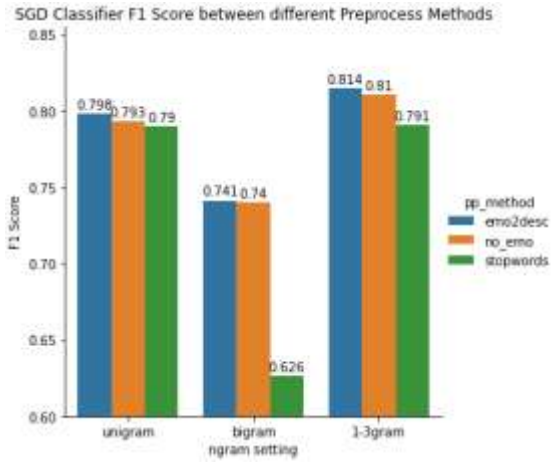

Figure 8: SGD Classifier F1 Score using different pre-process techniques and ngrams

Figure 7 shows that between the three pre-processing techniques, emo2desc performed the best with an accuracy score of 0.797 with 1-3 grams. While stopword filtering performed the closely the similar between unigram and 1-3 gram.

6.4 Test Results

The overall best performer is SGD with emo2desc and 1-3 gram. Additionally, LR's best performer used the same pre-processing techniques. Test labels can be predicted using both classifiers with the same techniques and submitted to Kaggle.

ML Model	Accuracy Score
LR	0.81015
SGD	0.80680

Table 2: Kaggle Submission Results

Table 2 display the results from Kaggle on the test data predicted via LR and SGD using emo2desc and 1-3 grams.

7 Discussion

While the results recorded between the two techniques of emoji/emoticon removal and replacement have a slight performance difference, a degree of 1% accuracy difference can be major improvement when running through a big dataset. For example, if a company were to focus on a dataset of 100,000 tweets related to a product they produce. Running SA with 78% accuracy compared to 79% accuracy would result having 10,000 tweets with correct sentiment classification. This difference can benefit the company's strategy on their next product release.

Another interesting observation from the result is that both models had similar performance among the different pre-process techniques. This is due to LR and SGD both classifiers are quite similar implementing a linear model. Furthermore, during the training phase SGD showed a varying number of results on the development data as SGD takes a random sample during each increment of its fitting.

A significant amount of performance cut was seen with bigram and stopword filtering. This was an unexpected result as the only related work with similar findings used an KNN classifier (nonlinear model) on one dataset[2]. Yet, for certain datasets bigram performed better than other classifiers.

8 Conclusion

With the importance of sentiment analysis becoming acknowledged and the use of emojis and emoticons on social media increasing, their meaning in SA cannot be overlooked. Moreover, stopword filtering can alter the performance of SA classifier by changing ngram selection significantly.

This study has revealed the importance of the preliminary phase of pre-processing and the effects it has on the sentiment classification performance. In particular, the performance improved by replacing emojis and emoticons with their respective descriptions and with 1-to-3 ngram.

Even though this study used a single dataset, it'll be interesting to see the difference while keeping the same pre-processing techniques on domain specific datasets. Furthermore, to study the performance of implementing a more

complex pre-process technique such as mutual information method of stop word filtering and emoticon-weight lexicon model.

References

[1] H. Saif, Y. He, and H. Alani, 'Alleviating data sparsity for Twitter sentiment analysis', in *2nd Workshop on Making Sense of Microposts (#MSM2012): Big things come in small packages at the 21st International Conference on theWorld Wide Web (WWW'12)*, 2012, pp. 2–9. [Online]. Available: http://oro.open.ac.uk/38501/

[2] A. Krouska, C. Troussas, and M. Virvou, 'The effect of preprocessing techniques on Twitter sentiment analysis', in *2016 7th International Conference on Information, Intelligence, Systems Applications (IISA)*, Jul. 2016, pp. 1–5. doi: 10.1109/IISA.2016.7785373.

[3] E. Haddi, X. Liu, and Y. Shi, 'The Role of Text Pre-processing in Sentiment Analysis', *Procedia Comput. Sci.*, vol. 17, pp. 26–32, 2013, doi: https://doi.org/10.1016/j.procs.2013.05.005.

[4] K. Wegrzyn-Wolska, L. Bougueroua, H. Yu, and J. Zhong, 'Explore the Effects of Emoticons on Twitter Sentiment Analysis', Aug. 2016, pp. 65–77. doi: 10.5121/csit.2016.61006.

[5] S. Ayvaz and M. Shiha, 'The Effects of Emoji in Sentiment Analysis', *Int. J. Comput. Electr. Eng.*, vol. 9, pp. 360–369, Jan. 2017, doi: 10.17706/IJCEE.2017.9.1.360-369.

[6] H. Saif, M. Fernandez, Y. He, and H. Alani, 'On Stopwords, Filtering and Data Sparsity for Sentiment Analysis of Twitter', in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, May 2014, pp. 810–817. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2014/pdf/292_Paper.pdf

[7] R. T.-W. Lo, B. He, and I. Ounis, 'Automatically Building a Stopword List for an Information Retrieval System', *J Digit Inf Manag*, vol. 3, pp. 3–8, 2005.

[8] D. A. Shamma, L. Kennedy, and E. F. Churchill, 'Tweet the Debates: Understanding Community Annotation of Uncollected Sources', in *Proceedings of the First SIGMM Workshop on Social Media*, New York, NY, USA, 2009, pp. 3–10. doi: 10.1145/1631144.1631148.

[9] M. Speriosu, N. Sudan, S. Upadhyay, and J. Baldridge, 'Twitter Polarity Classification with Label Propagation over Lexical Links and the Follower Graph', in *Proceedings of the First Workshop on Unsupervised Learning in NLP*, USA, 2011, pp. 53–63.

[10] H. Saif, M. Fernandez, and H. Alani, *Evaluation Datasets for Twitter Sentiment Analysis. A survey and a new dataset, the STS-Gold*, vol. 1096. 2013.

[11] A. Asiaee T., M. Tepper, A. Banerjee, and G. Sapiro, 'If You Are Happy and You Know It... Tweet', in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, New York, NY, USA, 2012, pp. 1602–1606. doi: 10.1145/2396761.2398481.

[12] P. Nakov, S. Rosenthal, Z. Kozareva, V. Stoyanov, A. Ritter, and T. Wilson, 'SemEval-2013 Task 2: Sentiment Analysis in Twitter', in *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, Atlanta, Georgia, USA, Jun. 2013, pp. 312–320. [Online]. Available: https://aclanthology.org/S13-2052

[13] L. Vadicamo *et al.*, 'Cross-Media Learning for Image Sentiment Analysis in the Wild', Oct. 2017.

[14] D. Tsirogiannis, S. Harizopoulos, M. A. Shah, J. L. Wiener, and G. Graefe, 'Query Processing Techniques for Solid State Drives', in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, 2009, pp. 59–72. doi: 10.1145/1559845.1559854.

[15] I. Montani, 'spacymoji'. 2021. [Online]. Available: https://spacy.io/universe/project/spacymoji

[16] N. Shah and S. Rohilla, 'emot'. Aug. 2021. [Online]. Available: https://github.com/NeelShah18/emot

[17] S. Symeonidis, D. Effrosynidis, and A. Arampatzis, 'A comparative evaluation of pre-processing techniques and their interactions for twitter sentiment analysis', *Expert Syst. Appl.*, vol. 110, pp. 298–310, 2018, doi: https://doi.org/10.1016/j.eswa.2018.06.022.

[18] L. Rashidi, 'Assignment 3: Sentiment Classification of Tweets'. [Online]. Available:

https://canvas.lms.unimelb.edu.au/courses/105766/assignments/243520
[19] A. Copestake, H. Yannakoudakis, and P. Buttery, ‘Further notes on Naive Bayes’. [Online]. Available: https://www.cl.cam.ac.uk/teaching/1718/MLRD/handbook/nb.html
[20] scikit learn, ‘scikit-learn’. [Online]. Available: https://scikit-learn.org/stable/index.html
[21] scikit learn, ‘Logistic Regression’. [Online]. Available: https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
[22] scikit learn, ‘Stochastic Gradient Descent’. [Online]. Available: https://scikit-learn.org/stable/modules/sgd.html#sgd
[23] scikit learn, ‘SGDClassifier’. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html