MGTA 415: Tweet Sentiment Extraction with NLP models

Shunsuke Iwata
Mao-Ho Wang
Yada Klueabvichit
Jiahui Zhang

Abstract

Capturing sentiment in language is important in these times where decisions and reactions are created and updated in seconds. By analyzing tweets, businesses can gain insights into how customers feel about their offerings and identify areas for improvement. Sentiment analysis can also be a useful tool for understanding investor sentiment and identifying potential areas of concern or opportunity. Thus, this project will provide a sentimental analysis of Twitter using several NLP techniques and compare their performance. We picked out a rule-based SentimentIntensity Analyzer as a baseline, the combination of Bag-of-words (tf-idf) and Logistic Regression, the combination of Elmo and Logistic Regression, and BERT. In the end, among these four models, BERT exhibited greater proficiency in this undertaking, achieving an F1 score of 0.79.

1. Introduction

Every day, social media platforms such as Twitter, Facebook, and Instagram generate billions of lines of text. This text is mostly unstructured data, making it difficult and time-consuming to interpret using traditional methods. However, text classification with machine learning can bridge this gap by allowing us to automatically categorize and analyze this data more efficiently. The Kaggle competition, Tweet Sentiment Extraction(https://www.kaggle.com/competitions/tweet-sentiment-extraction), offers an insightful opportunity to explore the practical application of Natural Language Processing (NLP). The competition entails picking out the part of the tweet (word or phrase) that reflects the sentiment. In this paper, instead, we will utilize NLP to classify tweets into three categories: "positive," "neutral," and "negative," using the text mining function. Using this dataset for NLP classification can enhance our understanding of the methods employed by algorithms to analyze, organize, and comprehend extensive amounts of textual data. Additionally, the competition allows for the evaluation of the performance of various machine learning models when applied to textual data.

2. Exploratory Data Analysis

This section delves into the project dataset sourced from Natural Language Processing with Tweets. After providing a general overview, the tweets contained in the training dataset are examined in detail. One of the challenges encountered pertains to the unprocessed nature of the tweets, necessitating the application of data cleansing techniques before delving further into the texts. Segmenting the data exploration results into positive, neutral, and negative categories yields an intriguing

comparison, forming the groundwork for further investigation.

2.1 Dataset Overview

The dataset used contains the text and sentiment of the tweet from Kaggle competition, Tweet Sentiment Extraction. This task includes one file which is train.csv. Each row in the training contains the text of a tweet and a sentiment label and also provided with a word or phrase selected from the tweet (selected_text) that encapsulates the provided sentiment.

The columns included in the dataset are as follows:

1) textID: A unique identifier for each piece of text
2) text: The text of the tweet
3) sentiment: The general sentiment of the tweet
4) selected_text: The text that supports the tweet's sentiment

This dataset contains a total of 27,481 example rows for the training set. In the training set, the data includes textID to identify for each piece of text, text of tweet, sentiment, which can be classified into negative, neutral, and positive, and select_text, which is the text that supports the tweet's sentiment and this column is only available in the training set.

After that, we split the data from the training set into three sets including training set, validation set, and test set by using a proportion of 80/10/10. By splitting the data, we can use the validation set to validate and tune the model to improve the accuracy. Table 1 shows the first five rows of the training dataset as samples. Figure 1 shows the most common five words of the training dataset as samples. This indicates that there are lots of stopwords which should be eliminated before the learning process.

Table 1: The first five rows of the training dataset

textID	text	selected_text	sentiment
cb774db0d1	I`d have responded, if I were going	I`d have responded, if I were going	neutral
549e992a42	Sooo SAD I will miss you here in San Diego!!!	Sooo SAD	negative
088c60f138	my boss is bullying me...	bullying me	negative
9642c003ef	what interview! leave me alone	leave me alone	negative
358bd9e861	Sons of ****, why couldn`t they put them on t...	Sons of ****,	negative

As Figure 1 shows, stop words have a high frequency of appearance. If we include stop words for the model training, the importance of the meaningful words will be diluted. Figure 2 shows the high frequency words after removing the stopwords, which makes meaningful words have higher importance.

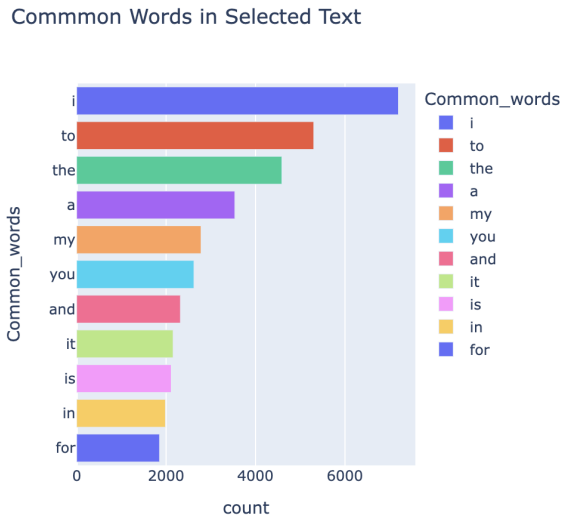

Figure 1 : Number of words for each frequency of appearance before omitting the stopwords

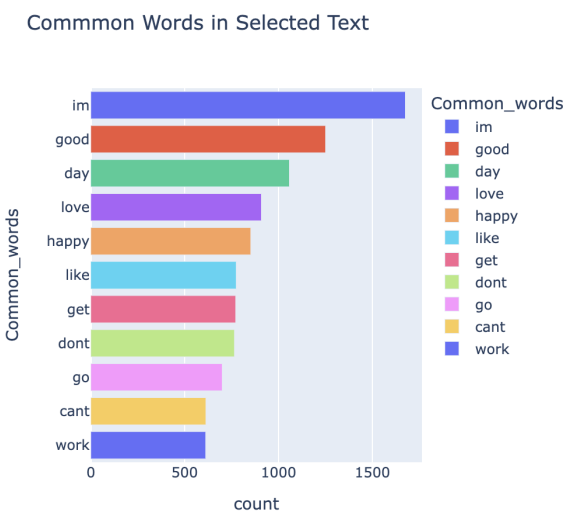

Figure 2 : Number of words for each frequency of appearance after omitting the stopwords

Figure 3, 4 and 5 saw the result of word clouds showing negative, neutral and positive respectively. The outcomes are very clear, and how these words are captured is related to the performance of the model.

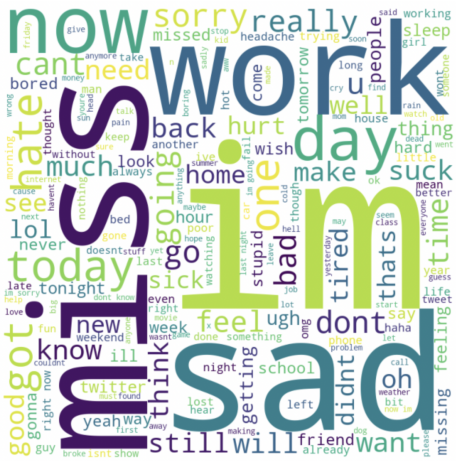

Figure 3: The word cloud of negative tweets

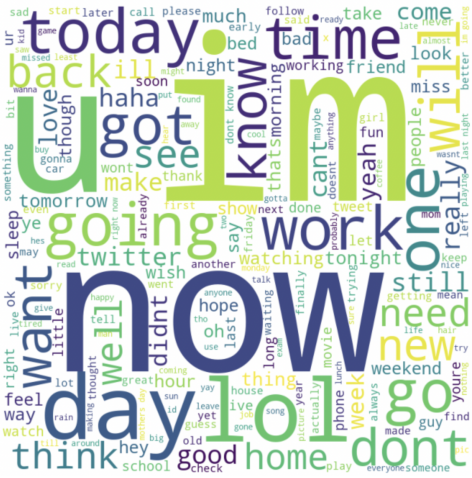

Figure 4: The word cloud of neutral tweets

Figure 5: The word cloud of positive tweets

2.2 Dataset Cleaning

Before data analysis, we conducted data cleaning to transform raw data into a consistent and standardized format that is suitable for further processing. We began pre-processing the data by tackling null values in the dataset. Here we decided to remove any rows with missing values.

Further, we converted text data into more unified format by conducting the following steps:

1. Convert all text to lowercase
2. Remove text in square brackets
3. Remove links
4. Remove punctuation marks
5. Remove any HTML tags
6. Remove any words containing numbers
7. Remove any newlines

This method replaced the original text data with the cleaned version, making it ready for further analysis. On top of that, we eliminated the stopwords which are common words like "and," "the," "is," and "in." for SentimentIntensity Analyzer and tf-idf. However, Elmo and Bert do not require explicit removal of stopwords. This is because both models learn to understand the context and semantics of a sentence or text, and stopwords often play an essential role in maintaining the structure and meaning of sentences.

	textID	text	selec
0	cb774db0d1	id have responded if i were going	id have responded if i w
1	549e992a42	sooo sad i will miss you here in san diego	
2	088c60f138	my boss is bullying me	bu
3	9642c003ef	what interview leave me alone	leave
4	358bd9e861	sons of why couldnt they put them on the rel...	

Table 2 : The first five rows of the training dataset after preprocessing

3. Natural Language Processing Model Comparison

3.1 SentimentIntensity Analyzer

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a rule-based sentiment analysis library specifically designed for analyzing sentiments in social media text (Hutto & Gilbert, 2014). The SenrimentIntensity Analyzer object in VADER is a natural language processing tool used for sentimental analysis.

VADER uses a combination of a sentiment lexicon (a list of words with associated sentiment scores) and a set of linguistic rules to determine the sentiment of a given text. It considers the intensity of positive, negative, and neutral sentiments. The analyzer object can take text as inputs, and outputs a set of sentiment scores, including positive, negative, neutral, and a compound score, which represents the aggregated sentiment of the text.

Unlike other predictive models we utilized in this paper, VADER does not use machine learning algorithms for prediction. Its rule-based approach makes it efficient and effective for certain types of text, particularly short and informal text found on social media platforms.

We used this rule-based tool as a baseline for model comparison. While rule-based tools are built with immense human effort, the models presented hereafter are machine-learned and require no prior human effort.

3.2 Bag-of-words (tf-idf) and Logistic Regression

The bag-of-words (BoW) model is a popular technique in natural language processing (NLP) that represents text documents as a collection of individual words or tokens, ignoring grammar and word order. In this

model, the frequency of each word is used to generate a numerical vector representation of the document, where each element corresponds to a unique word in the vocabulary.

However, the BoW model does not take into account the fact that some words may be more informative than others in distinguishing one document from another. This is where tf-idf (term frequency-inverse document frequency) comes in. Tf-idf is a statistical weighting scheme that assigns each word in the BoW model a weight that reflects its importance in the document.

The term frequency (tf) component of tf-idf measures how often a word appears in a document, while the inverse document frequency (idf) component measures how common or rare a word is across all documents in the corpus. The product of these two components results in a weight for each word in the document.

By using tf-idf to weight the words in the BoW model, we can obtain a more informative representation of a document that captures the unique characteristics of its content. This can be useful for various NLP tasks such as text classification, information retrieval, and sentiment analysis.

3.3 ELMO and Logistic Regression

ELMo (Embeddings from Language Models) is a deep contextualized word embedding model. It was introduced in a 2018 paper titled “Deep Contextualized word representations,” and has since become a popular technique in the field of NLP. It is a type of language representation model that takes into account the context of each word in a sentence or document, allowing it to capture nuances in meaning and improve performance on a range of natural language processing tasks.

Unlike traditional word embeddings that represent each word as a fixed vector regardless of context, ELMo uses a bi-directional language model to generate a unique vector representation for each word in a sentence, based on its surrounding context. This makes ELMo embeddings particularly useful for tasks such as sentiment analysis, question answering, and named entity recognition.

ELMo has been used extensively in a wide range of natural language processing applications, and has been shown to outperform traditional word embeddings on a variety of benchmark datasets.

3.4 BERT

While word embedding addresses many of the shortcomings of bag-of-words, it is not without its own flaws. While the meaning of words varies with context in real language, embeddings are static. BERT is one of several new models that attempt to vary the embedding of a word depending on the sentence it is contained in (Devlin 2018).

Bidirectional Encoder Representations from Transformers was released in 2018 by a team working at Google. As the name implies, it is a development of the slightly older Transformer (Vaswani 2017). A highly complex model, Transformer combines "Encoder" and "Decoder" blocks which mix recurrent and feed-forward neural networks.

A "Self attention" layer within each block learns contextual relationships between words, proving even more adept than LSTM at managing long term dependencies.

BERT could be described as a highly pre-trained stack of transformer encoder blocks (Alammar 2018). By using only the encoder (12 in the Base, and 24 in the Large version) BERT is able to function bidirectionally, using the words "behind" and "ahead" of the target word to make its predictions. BERT was pre-trained on an enormous corpus, including 800M words from English language books, 2500M words from Wikipedia. This semi-supervised pre-training means that BERT only needs to be "fine tuned" to be applied to a specific supervised classification task. Like LSTM, BERT and Transformer both take word embeddings as their initial input.

BERT has been widely applied to tweet classification. Pota et al. (2021) studied the effect of pre-processing text on the efficacy of the BERT model, and found that normalizing and transforming mentions, URLs, emoticons and emojis in tweets leads to the best prediction.

While our project makes use only of "vanilla" BERT, it is worth noting that there have been some even more recent extensions of the BERT model. The most notable of these in the realm of tweet classification is RoBERTa, a robust optimization of the base model. Liu et al. (2019) find that BERT was significantly under trained in its original implementation, and that with improved training and hyperparameter choices is able to match or exceed the performance of every model published after it.

Barbieri et al. (2020) specifically adapt the RoBERTa model to the classification of twitter data. The authors propose TweetEval, a unified evaluation framework for natural language processing algorithms that attempt to parse twitter data. It contains defined criteria for "train/validation/test splits," and lays out seven benchmarks against which an algorithm can be measured: emotion recognition, emoji prediction, irony detection, hate speech detection, offensive language identification, sentiment analysis and stance detection. Making use of the RoBERTa language model the authors find that a model initialized with a broader corpus and then trained on twitter performed better in testing than models trained entirely on standard material or solely on twitter data.

4 Implementation

For each of the models, the training dataset was further partitioned into two distinct subsets, following a 80/10/10 ratio, designated for training, validation, and test purposes, respectively. The validation set was used for tuning parameters and the test set was used to compare model performance. To guarantee consistent and reproducible splitting, the random state was set to a fixed value of 42.

4.1 Implementation of SentimentIntensity Analyzer

For the baseline model, the technique applied is Sentiment Intensity Analyzer from VADER library. Since the analyzer is

pre-trained, the first step of implementation is to import the vaderSentiment package.

The workflow of our baseline analyzer proceeds as follows:

1. Clean the text data.

2. Called a SentimentIntensity Analyzer object

3. Obtain the polarity score for each row in the ‘text’ column from the analyzer object.

4. Determine the label based on the polarity score. If the score for positive is higher than both negative and neutral, the label is determined as ‘positive’. If the score for negative is the highest among the three scores, the label is decided to be ‘negative’. Otherwise, the label is ‘neutral’.

5. Calculate the F1- score.

Using the analyzer to determine the label for each row in the dataset, we obtained a F1-score equals to 0.55.

There was a column called ‘selected_text’ given especially to our dataset. ‘Selected_text’ contains sentences selected by humans from the ‘text’ column on the same row. Applying the same workflow described as above, we obtained a much better performance. The F1-score obtained was 0.75.

This indicates that SentimentIntensity Analyzer performs well with short and informative text, but within the relatively raw context, the analyzer needs improvement in accuracy.

4.2 Implementation of Bag-of-words (tf-idf)

For our first model, the technique used in natural language processing is bag-of-words (BOW) to represent text as a bag (multiset) of its words, disregarding grammar and word order but keeping track of frequency, and we made used of Tf-idf (term frequency-inverse document frequency) to reflects the importance of a term to a document in a collection or corpus.

To perform sentiment analysis, we decided to use logistic regression to predict the sentiment of tweets including positive, neutral, and negative. The workflow of our BOW model proceeds as follows:

1. Clean the text data.

2. The BOW (tf-idf) representation is created using the TfidfVectorizer() function to convert a collection of tweets into a matrix of TF-IDF features.

3. Logistic regression model is trained on the bag-of-words features. The parameters of the logistic regression model are chosen using hyperparameter tuning.

4. Calculate the F1 score.

The F1 score for our model was 0.666, indicating that the model has a moderate level of accuracy in predicting sentiment labels. This suggests that there may be room for improvement in the model, either by using a more complex model architecture, collecting additional data, or by fine-tuning the model's hyperparameters.

4.3 Implementation of ELMo

For our second model, we applied ELMo as a word embedding technique to represent a sequence of words as a corresponding sequence of vectors.

We also incorporate logistic regression to predict the sentiment of tweets based on the sequence of vectors. The workflow of our ELMo + Logistic Regression model proceeds as follows:

1. Perform cleaning and lemmatization on the text data

2. Load the pre-trained ELMo model from TensorFlow Hub

3. Apply ELMo model for sentence embeddings

4. Input sentence vectors into the logistic regression model and fine-tune the model.

5. Predict the testset labels and calculate the evaluation criterion

Our Logistic regression model used the maximum iteration: 10000. Since We observed the model performed better without regularization on the validation set, we didn' set penalty and regularization strength for the logistic regression model.

4.4 Implementation of BERT

For our third model, we made use of the pre-trained BERT large model, and transformers and pytorch-lightning for fine-tuning. Even though it merely consists of fine tuning the already pre-trained model, running BERT requires substantial computational resources (particularly GPU). We made the decision to use a kaggle notebook with GPU100, and run our BERT code there. The workflow of our BERT model proceeds as follows:

1. Load the pre-trained BERT large model

2. Fine-tune the pre-trained model for classifying the training data

Table3: Confusion Matrix

		Prediction Labels		
Model	**Actual Labels**	negative	neutral	positive
SentimentIntensity Analyzer	negative	425	292	65
	neutral	44	925	130
	positive	16	152	699
BOW & logistic regression	negative	442	258	48
	neutral	220	741	180
	positive	36	182	641
ELMo & Logistic Regression	negative	604	47	216
	neutral	51	496	235
	positive	134	177	788
BERT	negative	632	128	22
	neutral	161	826	112
	positive	21	129	717

3. Predict the labels by the trained model and calculate the evaluation criterion

Our BERT model used the following hyperparameters:

Learning rate: 1e-5; Epoch: 3; Batch size: 64 for training data, 128 for validation and test data;

Since we don't need to calculate the gradient of loss function for validation data, we used a bigger batch size for them than training data. Finally, we got a 0.791 F1 score from our BERT model.

5.Results

We have compared our results using the four models described so far. We prepared a confusion matrix and F1 score comparison results for discussion. In this step, 2748 test data which we already prepared from original training data set at 10% was used for the comparisons

5.1 Result

Table3 shows the labels predicted by each model and the actual labels. On top of that, F1 score results on table4 to compare the performance of the final model.

Table4. The result of F1 scores

Model	F1
SentimentIntensity Analyzer	0.55
BOW & Logistic regression	0.67
ELMo & Logistic Regression	0.69
BERT	0.79

5.2 Discussion

Observing the Confusion Matrix, we found that SentimentIntensity Analyzer tends to analyze text as neutral compared to other models. As a result, it has the greatest number of correct labels for neutral text. At the same time, it has more wrong neutral labels on texts that are actually negative/ positive. SentimentIntensity Analyzer is gambling more labels as neutral, while BERT has a more balanced matrix table. In spite of the labels it is predicting accurately, the wrong labels show a balanced pattern.

5.3 Conclusion

In conclusion, the latest bidirectional models such as ElMo and BERT were found to perform well without processing stopwords. These models, however, based on specifically deep learning architectures designed for natural language processing tasks took lots of time for training. So, It is also necessary to select the most appropriate method depending on the resource situation.

Reference

[1]Hutto, C.J., & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.

[2]shriram s, Tweet Sentiment Extraction, shriram s's Blog. https://medium.com/analytics-vidhya/tweet-sentiment-extraction-e5d242ff93c2

[3] Alammar, J. *The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)*. Jay Alammar's Blog. http://jalammar.github.io/illustrated-bert/

[4] Jurafsky, D., & Martin, J. H. (2020). Speech and Language Processing (3rd ed.). Pearson.

[5] Sebastiani, F. (2002). Machine learning in automated text categorization. ACM Computing Surveys, 34(1), 1-47.

[6] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations.

[7] Akbik, A., Blythe, D., & Vollgraf, R. (2019). Contextual string embeddings for sequence labeling.

[8] Devlin, J., Change, M.-W., Lee, K., & Toutanova, K. (2018) Bert: Pre-training of deep bidirectional transformers for language understanding.

[9] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach.

[10] Pota, M., Ventura, M., Fujita, H., & Esposito, M. (2021). Multilingual evaluation of pre-processing for BERT-based sentiment analysis of tweets. *Expert Systems with Applications*, 181, 115119.