SemEval 2017 Task 4: Sentiment Analysis in Twitter SubTask A

Parthipan Ramakrishnan and Sarp Önal

Leiden University, The Netherlands

Abstract. In this paper, we present 5 models that we used to predict the sentiment of a tweet. We created a pre-processing function which performs tokenization, word normalization, segmentation, removal of URLs, Hashtags, mentions. We got a best accuracy of 68% using the RoBERTa Model and it has the best precision, recall scores for the positive and neutral classes.

Keywords: Sentiment Analysis · NLP · Twitter · BERT · RoBERTa.

1 Introduction

Sentiment analysis on Twitter involves using natural language processing techniques to determine the sentiment of tweets on the platform. This can be useful for businesses looking to understand how people are feeling about their brand, products, or services on Twitter. By analyzing the sentiment of tweets, businesses can get a better sense of what people are saying about them and take appropriate action to address any concerns or issues that may arise. Sentiment analysis can also be used to identify trends and patterns in the way people are talking about a particular topic on Twitter. This can help businesses stay on top of emerging issues and opportunities on the platform.

Problem Statement : In Sentiment Analysis in Twitter, given a tweet we have to decide whether the tweet expresses 'Positive', 'Neutral' or 'Negative' sentiment. The goal is to analyse the sentiment of the tweets. Since, Twitter is a rich source of information about people's opinions and emotions, and sentiment analysis can help businesses and organizations make more informed decisions by providing insight into the attitudes and emotions of Twitter users. Also, sentiment analysis on Twitter is a challenging task due to the informal and often abbreviated nature of tweets, as well as the presence of slang, emoticons, Sarcasms, and other forms of informal language. To design a model that will tackle the sarcasms, slang and misspelled words are the key challenges of the sentiment analysis. In order to do so, we have to do a lot of pre-processing on the data and then build a model to predict the sentiment.

2 Related Work

1. Twitter Sentiment Analysis[5] : In this paper the authors used the combination of NLP, Case-Based Reasoning and the Artificial Neural Network to find the sentiment of a text/tweet.
2. Bayesian Multinomial Naïve Bayes Classifier to Text Classification[7] : One NB classifier version that is frequently employed as a starting point in text categorization is the multinomial naive Bayes (NB) classifier. Multinomial NB classifier is not entirely Bayesian, though. In this study, a Bayesian version NB classifier is proposed.
3. BERT[3] : Pre-training of BERT, a brand-new language representation model, or Deep Bidirectional Transformers for Language Understanding. By concurrently conditioning on both left and right context in all layers, BERT is aimed to pre-train deep bidirectional representations from unlabeled text, in contrast to recent language representation models. As a result, without making significant task-specific architecture alterations, the pre-trained BERT model may be improved with just one additional output layer to produce cutting-edge models for a variety of tasks, including question answering and language inference.
4. Finetuning BERT for Text Classification[6] : Transfer Learning is proven to be very useful in text mining tasks. A state of the art transformer model BERT outperformed traditional methods in many language modelling tasks. A pretrained BERT model needs to be finetuned in downstream tasks in order to achieve good performances. This paper investigates different fine_tuning approaches of BERT to fully explore its potential in text classification. Pretrained BERT model is finetuned in three steps: more pretraining on domain data, finetuning on related tasks and finetuning on the target task.
5. RoBERTa[4] : Robustly optimized BERT approach (RoBERTa), is based on BERT model with modified pretraining procedures to improve performance. RoBERTa, diverges from BERT on 4 main issues: training with larger batches, larger text encoding, dynamic masking and different input format.

3 Data

We used the data provided for SemEval-4 Task4A. SemEval-4 Task 4A is a natural language processing task that was organized as part of the Fourth International Workshop on Semantic Evaluation (SemEval-4). The task involves detecting and classifying the sentiment of English language tweets. The task is a sub-task of SemEval-4 Task 4, which focuses on detecting and classifying the sentiment of English language tweets. The task is designed to evaluate the ability of natural language processing systems to automatically identify and classify the sentiment of tweets as either positive, negative, or neutral. For the sentiment analysis, we have used the train, dev dataset to evaluate our models and test dataset for the testing of the models, the dataset are provided by the SemEval-2017 competition. We have used the datasets from 2013 to 2016. We combined the train, dev and test dataset from 2013 to 2016, so that we will have a single train, dev and test dataset. A summary of the datasets is given in Table 1.

Table 1. Dataset used for Sentiment analysis in Twitter

Dataset	Size	Positive	Neutral	Negative
Twitter-2013 - Train-A	9684	3640	4586	1458
Twitter-2013 - Dev-A	1654	575	739	340
Twitter-2013 - Test-A	3547	1475	1513	559
Twitter-2014 - Sarcasm-A	49	20	7	22
Twitter-2014 - Test-A	1853	982	669	202
Twitter-2015 - Train-A	489	170	253	66
Twitter-2015 - Test-A	2390	1038	987	365
Twitter-2016 - Train-A	5868	3017	2001	850
Twitter-2016 - Dev-A	1966	829	746	391
Twitter-2016 - Devtest-A	2000	994	681	325
Twitter-2016 - Test-A	20632	7059	10342	3231
Combined Dataset				
Train Dataset	16041	6827	6840	2374
Dev Dataset	3620	1404	1485	731
Test Dataset	30422	11548	14192	4682
Total	50083	19779	22517	7787

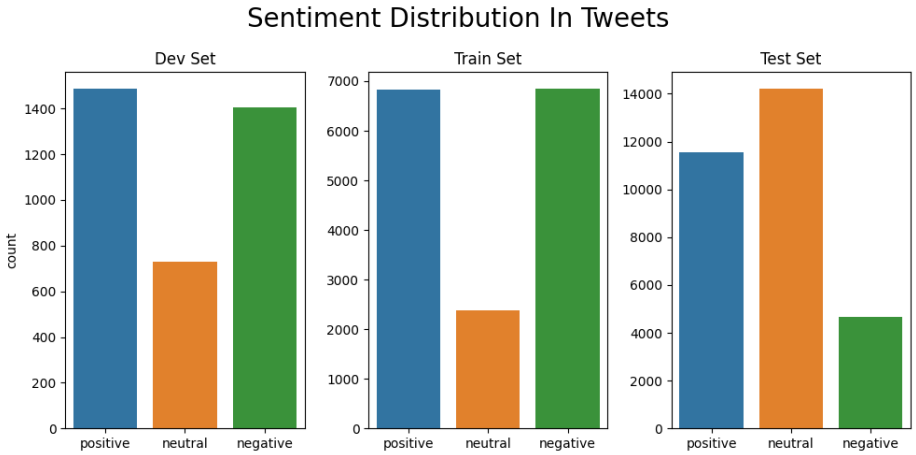

Fig. 1. Sentiment Distribution in the combined dataset

4 Method

Data Pre-Processing : The tweets in the dataset we got have URLs, mobile numbers, emoticons, Hashtags, twitter mentions. So, we removed those unwanted characters from the tweets. And also some of the dataset has unicode characters in them, so we decoded those unicode character before starting the processing. After that we tokenized the data, this will make it easier to analyze. We also removed the stopwords using the list of stopwords from NLTK package. We also used word lemmatizer to group the words and also normalised(lower case) the data. This helps in reducing the dataset and improve the accuracy of the sentiment analysis. We also created a list of positive, neutral and negative words, that will help us in determining the overall sentiment of a tweet/text.

Data Split : We have train, dev and test dataset separately, to fit it onto a model, but the classes are not balance properly, you can infer from the figure 1. Only the positive and negative classes are balanced on both Dev and the Train dataset. So, we used **RandomOverSampler**[2] to balance the imbalanced classification on Train, Dev and Test dataset and we used this balanced dataset to fit on our models to get better results.

We build 5 different models for this task, including a Naive Bayes model(baseline), BERT, BERT + NB, BERT with Fine Tuning and Roberta(state-of-the-art). There are two different ways to use BERT in a downstream task: fine tuning and using BERT for feature extraction for a classifier [1]. We used feature based approach in model 3 and fine tuning in model 4.

Model 1 :- Naive Bayes(Baseline) : We created a Naive Bayes model with a pipeline object that applies a series of transformations to the input data before passing it to a MultinomialNB classifier. The pipeline consists of three steps:

- The **CountVectorizer** step converts the input text into a matrix of token counts. It tokenizes the input text, meaning it breaks the text down into individual words or tokens, and then counts the number of occurrences of each token in the text.
- The **TfidfTransformer** step applies Term Frequency-Inverse Document Frequency (TF-IDF) scaling to the token counts. TF-IDF is a way of weighing the importance of each token in a document by considering how often it appears in the document relative to how often it appears in the entire corpus of documents. The use_idf parameter controls whether to use the inverse document frequency component of the scaling.
- The **MultinomialNB** step is the classifier itself, which takes the transformed input data and uses it to make predictions about the class of the input.In the MultinomialNB algorithm, the classifier is trained on a set of labeled input data, where each input is a document represented as a bag of words. The classifier uses this training data to learn the conditional probabilities of each feature given each class. During the prediction phase, the

classifier uses these probabilities to calculate the probability of each class given a new input document, and selects the class with the highest probability as the prediction. One advantage of the MultinomialNB algorithm is that it is relatively simple and easy to implement.

Model 2 :- BERT : We use Bidirectional Encoder Representations from Transformers (BERT) algorithm in order to classify the Tweets as positive negative or neutral. BERT algorithm is trained on *English Wikipedia* and *Books Corpus* which have more than 800 million words in total. There are two pre-trained BERT variations in [3], a relatively small network and a larger network. Smaller network has 12 hidden layers with 100 million parameters and larger network has 24 hidden layers with 340 million parameters. We use the relatively small pre-trained BERT. Training procedures of BERT algorithm is represented in figure 2. Since the BERT network has 100 million parameters, we make use of GPU acceleration of Google Colab to train it on the Twitter dataset. We use BERT base uncased model with zero-shot learning. However, this model was pretrained with two objectives: Masked language modelling and next sentence prediction. Therefore we changed the output layer to make a multiclass classification.

$$\mathbf{BERT_{Base}} : L = 12, H = 768, A = 12, TotalParameters = 110M$$

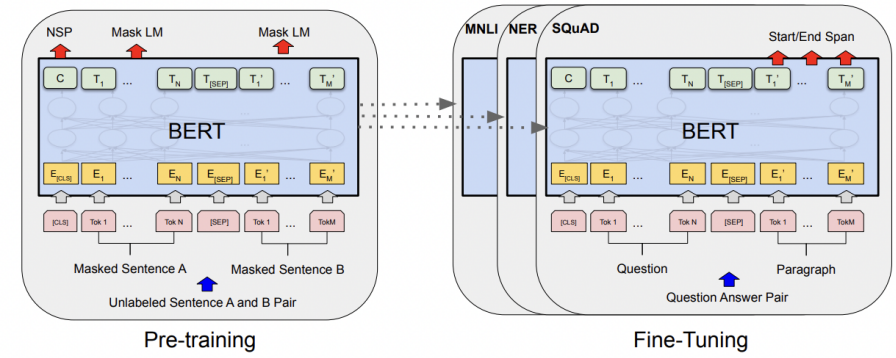

Fig. 2. Pre training and fine tuning procedures of BERT [3]

Model 3 :- BERT + NB : We use the same uncased base BERT as in the previous model. However, instead of changing the output layer, we add a Naive Bayes classifier at the end of the BERT. Because the BERT learns an inner representation of the English Language which we can use to extract features for a different task [3]. Naive Bayes uses the outputs of BERT model as features. By using the features produced by BERT, Multinomial_NB is trained on the training set and used to classify the test set.

Model 4 :- Fine Tuned BERT : Instead of using a zero shot learning of the base uncased BERT model, we do fine tuning. We use the *BERT tokenizer* to add special tokens, add padding and create attention mask. Padding is used in order to make input of the BERT same length. Maximum allowed sequence length is 512, because of computational complexity of BERT. Attention mask is a binary list, decides if the token is required or not when learning the representations. Fine tuning of the model is needed, since the base BERT model is pre-trained on a different task. We changed the output layer as 3 class classification layer, and weights of this layer needs to be trained with a labeled data. Therefore, we fine-tuned the model on the training set and used it to classify the sentences in test set.

Model 5 :- RoBERTa : RoBERTa (Robustly Optimized BERT Pretraining Approach) is a variant of the popular BERT (Bidirectional Encoder Representations from Transformers) model for natural language processing tasks. The RoBERTa architecture is based on the transformer architecture, which is composed of a series of self-attention layers and feedforward layers. Like BERT, RoBERTa is a "masked language model", which means that it is trained to predict the masked words in a sentence given the context provided by the other words. This allows the model to learn contextual relationships between words, which is useful for tasks such as language translation, question answering, and text classification.

Table 2. RoBERTa Model HyperParams[4]

Hyperparams	$RoBERTa_{BASE}$
Number of Layers	12
Hidden size	768
FFN inner hidden size	3072
Attention heads	12
Attention head size	64
Dropout	0.1
Attention Dropout	0.1
Warmup Steps	24k
Peak Learning Rate	6.00E-04
Batch Size	8k
Weight Decay	0.01
Max Steps	500k
Learning Rate Decay	linear
Adam ϵ	1.00E-06
Adam β_1	0.9
Adam β_2	0.98
Gradient Clipping	0

5 Results

Table 3. Results of all the models we used for Sentiment analysis in our dataset.

Model	Class	Precision	Recall	F1-score	Accuracy
	negative	0.33	0.65	0.44	
Naive Bayes	neutral	0.64	0.41	0.50	53%
	positive	0.59	0.62	0.60	
	negative	0.00	0.00	0.00	
BERT(Zero Shot)	neutral	0.42	0.06	0.11	38%
	positive	0.38	**0.93**	0.54	
	negative	0.20	0.05	0.08	
BERT+NB	neutral	0.55	0.42	0.48	47%
	positive	0.44	0.69	0.54	
	negative	0.66	0.33	0.44	
FT BERT	neutral	0.62	0.76	0.69	65%
	positive	0.69	0.64	0.66	
	negative	0.61	0.63	0.62	
RoBERTa	neutral	0.64	0.82	**0.72**	**68%**
	positive	**0.84**	0.54	0.66	

It is possible to see from table 3 that, RoBERTa model performs the best and FineTuned BERT model performs little bit worse than it. On the other hand, Zero shot BERT model performs much worse compared the other models, as expected. Because the BERT model was pretrained for Masked language modelling and next sentence prediction tasks. In order to get good performance, it can be fine_tuned or a classifier can be implemented to use the output of the BERT model as features. We observe that both approaches improve the performance of the base BERT model, while finetuning does better.

Table 4. Weighted Average F1 scores of all the models.

Model	F1 Score
Naive Bayes	0.53
BERT(Zero Shot)	0.25
BERT+NB	0.44
FT BERT	0.64
RoBERTa	0.68

The table 4 shows the weighted average F1 scores of all the 5 models we trained for the sentiment analysis, and the F1 score for the Fine-tuned BERT and RoBERTa are has better scores compared to the other models.

We also implemented a traditional model, Naive Bayes, as a baseline model to have a benchmark for comparison of state-of-art models such as BERT and RoBERTa. It can be seen from table 3, Fine_tuned BERT and RoBERTa outperforms other models, however they require more computing power.

6 Discussion

In the experiments we conducted, we faced some limitations. Firstly, the data was noisy so we had to preprocess to remove the noise and normalize the data in order to improve the performance of the models we implemented. Secondly, computational power and time complexity was a problem since transformer models use self attention mechanism, which has complexity of $O(n^2)$ [3]. Which means it is quadratic with the sample size. To speed up the process, we used the GPU acceleration in Google Colab. Another problem occurred because of the maximum sequence length limit of BERT is 512 [3]. In order to overcome this problem, we used *padding* in the tokenization to make every input sequence the same length. The dataset also have sarcasms in it, these sarcasm texts are hard to classify.

7 Conclusion

In this paper, we introduced 5 models for the sentiment analysis on twitter data. The best model was the **RoBERTa** model which achieved an accuracy of **68%**. Also, BERT with fine Tuning got a close accuracy of 65%. The problem on analysis the sentiment of a text, is because of the noises in the data, we created a data processing function to remove the noises as much as possible from the data and also to normalize the data, this can improve the efficiency of the used algorithm. In future we want build other models like combining the RoBERTa model with the LSTM. We think it may produce better results.

8 Contribution

Coding :

- Ramakrishnan : Explanatory Data Analysis, Data Pre-Processing, Random Over Sampling, Train-Test Split, Naive Bayes Model.
- Sarp : BERT, BERT+NB, FT BERT, RoBERTa

References

1. Albanese, N.C.: Fine-tuning bert for text classification. https://towardsdatascience.com/fine-tuning-bert-for-text-classification-54e7df642894, accessed: 2022-12
2. Brownlee, J.: Random oversampling and undersampling for imbalanced classification (Jan 2021), https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/
3. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
4. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
5. Sarlan, A., Nadam, C., Basri, S.: Twitter sentiment analysis. pp. 212–216 (11 2014). https://doi.org/10.1109/ICIMU.2014.7066632
6. Sun, C., Qiu, X., Xu, Y., Huang, X.: How to fine-tune bert for text classification? In: China national conference on Chinese computational linguistics. pp. 194–206. Springer (2019)
7. Xu, S., Li, Y., Zheng, W.: Bayesian multinomial naïve bayes classifier to text classification. pp. 347–352 (05 2017). https://doi.org/10.1007/978-981-10-5041-1_57