AML Challenge 3: Sentiment Analysis

Group 47

Jana JUROŠ Jana.Juros@eurecom.fr

Rea PINTAR Rea.Pintar@eurecom.fr

Ivona STOJANOVIĆ Ivona.Stojanovic@eurecom.fr

January 18, 2024

1 Problem setting

Within the scope of this task we will get familiarized with Natural Language Processing, and performing sentiment analysis of tweets[1] from Figure Eight's Data for Everyone platform. Sentiment analysis is a natural language processing technique that aims to determine the emotional tone or sentiment expressed in text, helping to identify whether the sentiment is positive, negative, or neutral. It involves analyzing and classifying text data to understand people's opinions, attitudes, or emotions towards a particular subject or entity. In this dataset, tweets will be classified in one of three classes - Positive, Neutral or Negative, as seen in the example:

- "*I love this ice-cream, it is so good ! :-)*" - Positive
- "*Last week I visited San Francisco*" - Neutral
- "*Disappointed by the last Avengers movie...*" - Negative

2 Chosen approach

First, we will explore the dataset we have at hand. This means that we'll illustrate the text data and also extract their features using tokenization, stopword removal and other natural language preprocessing techniques, in order to do some exploratory data analysis such as a word cloud and word frequency analysis.

We will be trying out 5 different models for our data, two of which are considered simple classifiers (and are given to us in the baseline) and three more popular, state-of-the-art approaches which we expect will perform better on our data than the baselines. The simple models we will be using are Naive Bayes text classifier and VADER (*Valence Aware Dictionary and sEntiment Reasoner*). The more complex models inlcude two versions of RoBERTa (*Robustly Optimized BERT Pre-training Approach*), one of which which we will use and fine-tune ourselves, and another already fine-tuned version from the source Hugging Face. Finally, we will explore SBERT (*Sentence-BERT*), a sentence embedding technique known for its proficiency in capturing semantic meaning, which is actually the predecessor to RoBERTa. We will explain the principles of these models when we come to model selection.

In the model selection process we will be using metrics such as Accuracy and F1-score, in order to measure model performance. We expect our version of fine-tuned RoBERTa to be the best performing model, based on its complexity and recent popularity in use.

3 Data analysis and pre-processing

The original dataset contains 24732 tweets and has 4 columns: *textID*, *text* (the full text of the tweet), *sentiment* (one of the following classes: {Positive, Neutral, Negative}) and *selected_text* (the words selected from *text* to lead to the classification stored in *sentiment*).

In order to do some exploratory data analysis, first some pre-processing must be done. For us, this involved some text processing steps such as:

- Removing contractions - e.g. *it's* will become *it is*
- Tokenizing - breaking the sentence down into smaller units, called tokens
- Converting all letters to lowercase
- Removing punctuation

[1] Tweet - a post on the app Twitter (twitter.com) containing up to 280 characters, so no more than a couple of short sentences.

- Removing stopwords - words such as *the, is, and,* etc.
- Lemmatization - breaking words down to their root meaning to identify similarities (e.g. *dogs* will become *dog*)

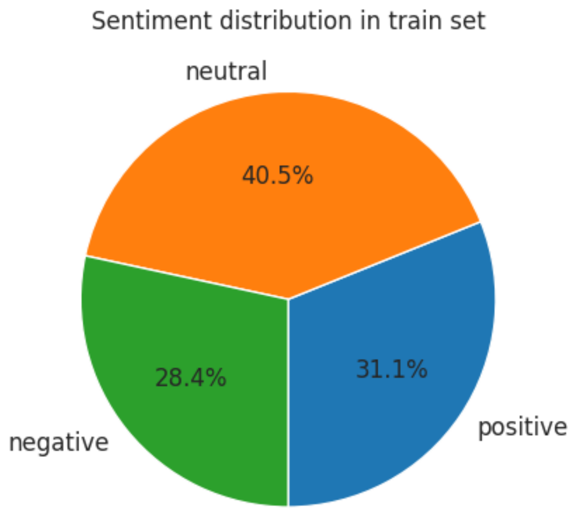

Figure 1: Distribution of sentiment labels (Positive, Neutral, Negative) within the training set.

Additionally, we will split the training set further into a smaller training set (90% of the original dataset) and a validation set (10%) as it was already suggested in the baseline. A pie chart that shows how the sentiment labels are distributed in the smaller training set can be found in Figure 1. We can see that the classes are more or less balanced, but that the majority of tweets is Neutral, which tells us that there is no need for techniques like over-/undersampling, data augmentation, etc. One of the mandatory data processing steps included converting the labels (sentiments; Positive, Neutral and Negative) into integers, using the mapping: 1 for Positive, 0 for Neutral and -1 for Negative.

Once we have finished the pre-processing steps, we can perform word frequency analysis and other types of data analysis. We analyzed the most frequent words in each of the classes and found that for class Positive (1) the 5 most common words are: *day, good, love, happy, get* whereas the 5 most common words in the class Negative (-1) are: *go, get, miss, work, like.* The words *go* and *get* are the most frequent words in the entire dataset, so it's no surprise they both appear in the Top 5 of each class (same for class Neutral).

Additionally, we can represent the frequency of words visually using a Word Cloud. A Word Cloud is a visual representation of text data where words are displayed in different sizes based on their frequency or importance within a given context. It provides a quick and intuitive way to identify the most common or significant words in a dataset, offering a visual snapshot of the prominent themes or topics.

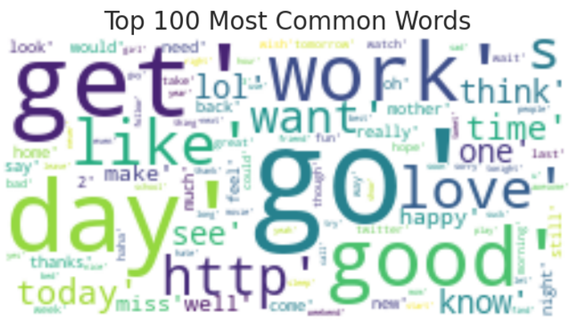

Figure 2: A Word Cloud of our dataset.

For our Naive Bayes and VADER models we are also going to tokenize the text using the *CountVectorizer* class. This class breaks down the input text into individual words or tokens. It removes any punctuation or special characters and converts all text to lowercase by default.

The pre-processing steps we are going to apply to our data in order to prepare it for the RoBERTa/SBERT model are similiar to the ones already mentioned, however they are automatized using the already implemented RoBERTa python libraries. Specifically, we are going to apply Tokenization using the *RobertaTokenizer* class, which will convert our data to PyTorch tensors.

4 Model selection

In the baseline, we are given two simple classifiers - a Naive Bayes classifier on the Bag-of-Words features of the training data, and VADER. We're also going to apply RoBERTa, a popular state-of-the-art method. We'll have two versions of this model: the first version utilizes RoBERTa as a feature extraction module, encapsulated within the `RobertaForSequenceClassification` model. This version serves as a classifier for our sentiment analysis task, which we aim to improve by fine-tuning the RoBERTa model. The other one uses a pre-trained and already fine-tuned RoBERTa model from Hugging Face which is ready-to-go for sentiment classification. In this case, we do not need to train the model ourselves. Instead, we can directly input our data into the pre-trained model and receive the corresponding output. Finally, we will utilize SBERT (from the same source as RoBERTa), a pre-trained sentence embedding model, in combination with a fully connected neural network. In this approach, we do not fine-tune the SBERT model but use it as a fixed feature extractor.

4.1 Naive Bayes

The Naive Bayes text classifier based on the principle of Bayes' theorem and assumes independence between features. Despite its simplicity, Naive Bayes performs well in many cases and is particularly effective in situations with high-dimensional data and limited training examples. It is widely used for tasks such as sentiment analysis, spam detection, and document categorization. We are going to compare this simple classifier with a more cutting-edge model.

4.2 VADER

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a rule-based text classifier specifically designed for sentiment analysis. Contrary to the other models, it doesn't use machine learning. This rule-based approach uses a pre-defined set of sentiment scores for words and phrases to determine the sentiment polarity of a given text. In python, this model will output a list of sentiments and the probabilites of each sentiment being true. We will use the given prediction function from the baseline which simply chooses the sentiment with the highest probability. It is important to note that VADER's performance can be limited by the coverage and Accuracy of its sentiment lexicon, and it may not capture more complex or nuanced expressions as well as more advanced machine learning models.

4.3 RoBERTa

We are also going to apply the model RoBERTa (Robustly Optimized BERT approach) - a state-of-the-art language representation model based on the popular BERT (*Bidirectional Encoder Representations from Transformers*) architecture. It was introduced as an improvement over BERT by Facebook AI Research. RoBERTa is trained on a large amount of text data using the Masked Language Modeling (MLM) objective, similar to BERT, however it incorporates several modifications in its training methodology to achieve better performance. It uses dynamic masking, which means that the masking pattern changes from one training iteration to another, allowing the model to see each word in the data at multiple positions. Additionally, RoBERTa implements extensive data augmentation techniques, including shuffling the order of sentences during training. These modifications enable RoBERTa to learn better contextual representations and capture deeper semantic relationships in the text. RoBERTa is a state-of-the-art model that has demonstrated excellent performance on a range of natural language processing tasks, such as text classification, sentiment analysis, and question answering. To enhance the sentiment analysis capabilities of RoBERTa, we will perform fine-tuning on the model. Fine-tuning involves training the pre-trained RoBERTa model on our specific sentiment analysis task, allowing it to adapt and improve its performance on sentiment-related predictions.

To assess the impact of fine-tuning RoBERTa on our sentiment analysis task, we will compare its performance with that of another pre-trained model specifically fine-tuned for sentiment analysis classification. This alternative model has already undergone the fine-tuning process on a dataset consisting of 5,304 manually annotated social media posts. Its hold-out Accuracy, which is the Accuracy achieved on a separate validation set, is reported as 86.1%.

By considering this pre-trained model's high Accuracy on sentiment analysis, we can consider it as a baseline for evaluating the performance of our own fine-tuned RoBERTa model. A comparison between the two models will help us determine the extent of improvement achieved through our fine-tuning process. The model can be found on this link.

4.3.1 Hyperparameter tuning

To attain the best possible performance from our fine-tuned RoBERTa model, we conducted hyperparameter tuning. Using a grid search and considering the time constraints (around 7 minutes per epoch), we decided on optimize the following: number of epochs, learning rate λ and batch size n. Our grid consisted of the following values: $epochs \in \{1,2,3\}, \lambda \in \{10^{-5}, 3 \times 10^{-5}, 5 \times 10^{-5}\}, n \in \{16, 32, 64\}$. The scoring metric we used is the F1-score. The following combination of parameters has proven to perform the best on the validation set:

$$epoch = 2,\ \lambda = 3 \times 10^{-5},\ n = 64$$

4.4 SBERT

The last model that we used is SBERT (Sentence-BERT). This feature extraction method generates sentence embeddings. It encodes the phrases into a fixed-length vector, in order to capture their semantic meaning. It basically works on sentence pairs by learning with a so-called siamese architecture which means that two identical BERTs are trained in parallel that share the same network weights. The goal is to maximize the similarity for similar pairs and minimizing it for dissimilar pairs. This method seemed very useful to us, because it can handle sentences better than the other techniques used. But one also has to keep in mind that a tweet can consist of multiple sentences, so it's not assured that this method will give us very good results, but we've still decided that it's definitely worth a try.

After performing feature extraction with SBERT, we proceed to train a fully connected neural network. This network is a simple feed-forward architecture consisting of one hidden layer that utilizes the ReLU activation function.

4.4.1 Hyperparameter tuning

In order to have a fair comparison between the methods, we'll tune the hyperparameters of the fully connected neural network used for classification. Using the grid search, we will be optimizing the following parameters: learning rate λ, batch size n and number of neurons in the hidden layer h. Our grid consisted of the following values: $\lambda \in \{0.001, 0.01, 0.1, 1\}$, $n \in \{64, 128, 256, 512\}$, $h \in \{50, 100, 200, 400, 800\}$. We tested the model with the same scoring metrics as before, Accuracy and F1-score. The parameters we have found to be the best are as follows:

$$\lambda = 0.01,\ n = 128,\ h = 800$$

5 Model performance

Since this is a classification task, we will use Accuracy and F1-score to evaluate the model's performance. The Accuracy score is the ratio of correctly identified samples and the total sample size. The F1-score, on the other hand, is defined as follows:

$$F1 = \frac{2 * precision * recall}{precision + recall}$$

It's a harmonic mean of two scores: *precision*, which measures the Accuracy of positive predictions, and *recall*, which assesses the completeness of positive predictions. Both of these evaluation metrics are values belonging to the interval $[0, 1]$, where higher values are desirable.

The first model we can analyze is VADER, which does not have satisfactory results. In fact, on the validation test it gives us 48.67% and 0.3746 F1-score. Here we can see the effect of VADER's lexicon's limitations, which we assume is correlated to how tweets don't necessarily follow correct sentence structure and use plenty of slang.

Next, we can analyze is the Naive Bayes, which has a relatively short training time (just under 30 seconds, same as VADER) and yields in 65.60% Accuracy on the validation set. Additionally, the F1-score of this model is 0.6557. Overall, these results are better than VADER, but not entirely satisfactory and can certainly be improved.

If we take a look at the models we've implemented, there is a clear improvement in performance overall. Both the Accuracy and F1-score got better in each case. The worst from our proposed approaches is SBERT with an Accuracy of 69.03% and an F1-score of 0.6935. The RoBERTa models perform better than SBERT and the clear winner is our own fine-tuned model which yields an Accuracy of 79.94% and a F1-score of 0.7985.

Model	Accuracy	F1-score
VADER	48.67%	0.3746
Naive Bayes	65.60%	0.6557
RoBERTa - Hugging Face	71.14%	0.7150
RoBERTa	**79.94%**	**0.7985**
SBERT	69.03%	0.6935

6 Conclusion

We chose the RoBERTa model on which we performed the fine-tuning process ourselves, since it performed the best on our training data, with the validation Accuracy of 79.94%. The final score achieved by this model on Kaggle is 0.79435. This is a very satisfactory result considering that the provided baseline result is 0.63433 and goes to show the power of state-of-the-art models such as RoBERTa. In this specific task, fine-tuning RoBERTa demonstrated superior performance compared to using a pre-trained Sentence-BERT model. However, it is important to note that the effectiveness of fine-tuning Sentence-BERT or utilizing other BERT models could yield different results.

All of the written code can be found in a public notebook on Kaggle with all appropriate credits to various sources we used.