AML Challenge III: Sentiment Analysis

EURECOM

Group 7

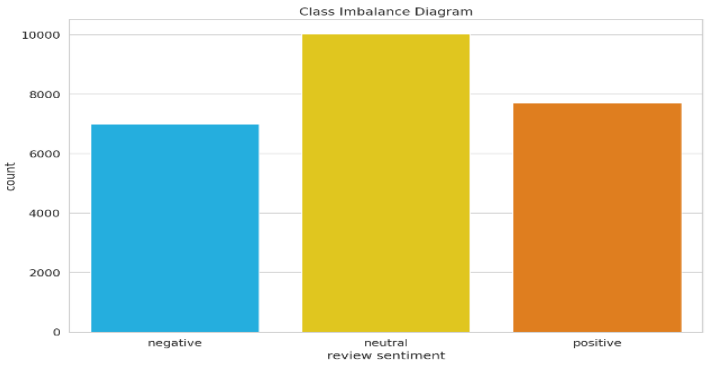
Emerson CARDOSO, Sameer HANS, Deepika SIVASANKARAN, Gaganjot SHAN

1. Introduction

The main aim of this challenge is to determine whether a tweet is positive, negative or neutral. We used a variety of models to achieve this. We experimented with a variety of simple and complex models. We compared their results after they were implemented and chose the one that performed the best.

2. Data exploration

The dataset contained 24732 samples and 2748 test samples. The training set is relatively balanced among the positive and negative. The dataset contains no null or missing values. The distribution is skewed between neutral and positive/negative tweets.



Plot 1: Sentiments Distribution

3. Data preparation

Some pre-processing was already done in the dataset where we see the "**selected_text"** column in place of "**text"** column. But, even the selected_text could be cleaned more to obtain only the relevant information. Therefore, we performed some basic data cleaning steps for NLP on the selected_text.

We did data cleaning starting from removing garbage noise such as punctuations, symbols, RT, mentions and emails. Then removed the stopwords using nltk library. We performed tokenization (Tokenization is a method of breaking down a piece of text (the tweet) into smaller pieces (words). It converts a string of text into a list of strings, each string representing one of the tweet's words.) Then we did some chat words replacement (after obtaining the tokens); such as "**ppl"** to "**people".**

Finally, we did Lemmatization. Lemmatization removes the grammar tense and restores the original form of each word. Lemmatization is the process of converting a word into its lemma form. For example, if we lemmatize the word "**books**", we get the term "**book**".

Original text	Cleaned	Stop words	Tokens	Lemmatization
!!!!!!!!!!!! @Roberts: Good Luck with your auction "	Roberts Good luck with your auction	Roberts good luck auction	[Roberts, good, luck, auction]	Good Luck auction

Example table of Data Cleaning steps

After data cleaning, we tried to do some data visualizations on the clean data such as finding the most common unigrams and bigrams.

4. Features Extraction

We must turn pre-processed data into features in order to analyse it. Text features can be built in a variety of ways. We used a variety of strategies to extract the desired attributes in this challenge: BERT, and Bag of Words.

a) BERT:

BERT stands for Bidirectional Encoder Representations from Transformers. BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. BERT's goal is to generate a language model. Its primary function is to extract high-quality language features from text data. Bert is trained with 2 training tasks:

i. Classification Task: determining which category the input sentence belongs in.
ii. Next Sentence Prediction Task: determining if the second phrase follows the first sentence.

The Bert Tokenizer was chosen as it was trained on a huge corpus of unlabelled text, including the entire Wikipedia, allowing us to obtain acceptable pre-trained weights. It's easy to fine-tune the model with these pre-trained weights. We experimented with many pre-trained models of BERT (cased and uncased). The uncased model worked better than the cased models.

b) Bag of Words:

A bag of words model is a method of extracting features from text for use in modelling techniques. The method is straightforward and adaptable, and it may be used to extract information from documents in a variety of ways. A bag of words is a text display that describes occurrence of words in a document. It involves 2 things:

i. A lexicon of well-known terms.
ii. A metric for determining the presence of known words.

It's the easiest way to vectorize a record, but the biggest drawback is that all of the words are weighted equally, which isn't true in all circumstances because word relevance varies depending on context.

c)Word2vec

The Word2vec is a group of related models that are used to produce so-called word embeddings. These models are shallow, two-layer neural networks, that are trained to reconstruct linguistic contexts of words. After training, word2vec models can be used to map each word to a vector of typically several hundred elements, which represent that word's relation to other words. This vector is the neural network's hidden layer.

5. Models

a) BERT

A transformer is a deep learning model that uses the attention mechanism to weigh the impact of various elements of the incoming data. It offers general-purpose architectures for sentiment analysis, such as BERT, GPT-2, XLM, and XLNet, with over 32+ pretrained models in 100+ languages. We used the Bert Classifier model with a pretrained Bert Tokenizer in our experiment. Bert models were chosen to pre-train both the model and the classifier because they are simple and empirically powerful. It is "deep bidirectional," which

means that during the training phase, it learns information from both the left and right sides of a token's context, which is more efficient.

b) LSTM

The LSTM is a deep learning recurrent neural network. The ability of LSTM to process sequences of data such as speech or video is due to its feedback connections. This is why we used it because we have word sequences. We utilized the Keras library to implement it. An embedding layer, an LSTM layer with dropout, a dense layer, and a final layer to output 3 dimensional vectors using softmax make up our neural network. As a result, the categorical entropy loss function was applied. Finally, we tuned all of the hyperparameters using cross-validation and the auc metric.

c) SVM and Decision Tree

We also test more traditional machine learning approach SVM and Decision Tree using TFIDF, to do that the following steps were followed:

i. Pre-processing and TFIDF corpus create:

The pre-processing phase consists in doing the cleaning of each input of the dataset, doing steaming and lemmatization of the words, removing numbers, stop words, punctuation, and tags. After cleaning the dataset, we proceed to TFIDF calculation for each input and create the final dataset. TF-IDF (term frequency-inverse document frequency) is a numerical statistic that intends to indicate how important a word or token is for a document in a corpus (Stein & Silva, 2016). Its value is determined by multiplying the value of TF (term-frequency), which represents the number of occurrences of a term t in a document d, by the IDF (inverse document frequency), which is determined by the logarithm of the inverse fraction of documents d that contains that term in corpus D, that is:

$$TFIDF = TF \times IDF$$

$$TF(t,d) = \frac{number\ of\ time\ t\ appears\ in\ d}{total\ number\ of\ t\ in\ d}$$

$$IDF(t) = \log \frac{total\ number\ of\ d}{number\ of\ d\ contains\ t}$$

At the end our dataset looks like the following table:

	t1	..	tn	output
d1	TFIDF	TFIDF	TFIDF	Y1
..	TFIDF	TFIDF	TFIDF	..
dn	TFIDF	TFIDF	TFIDF	Yn

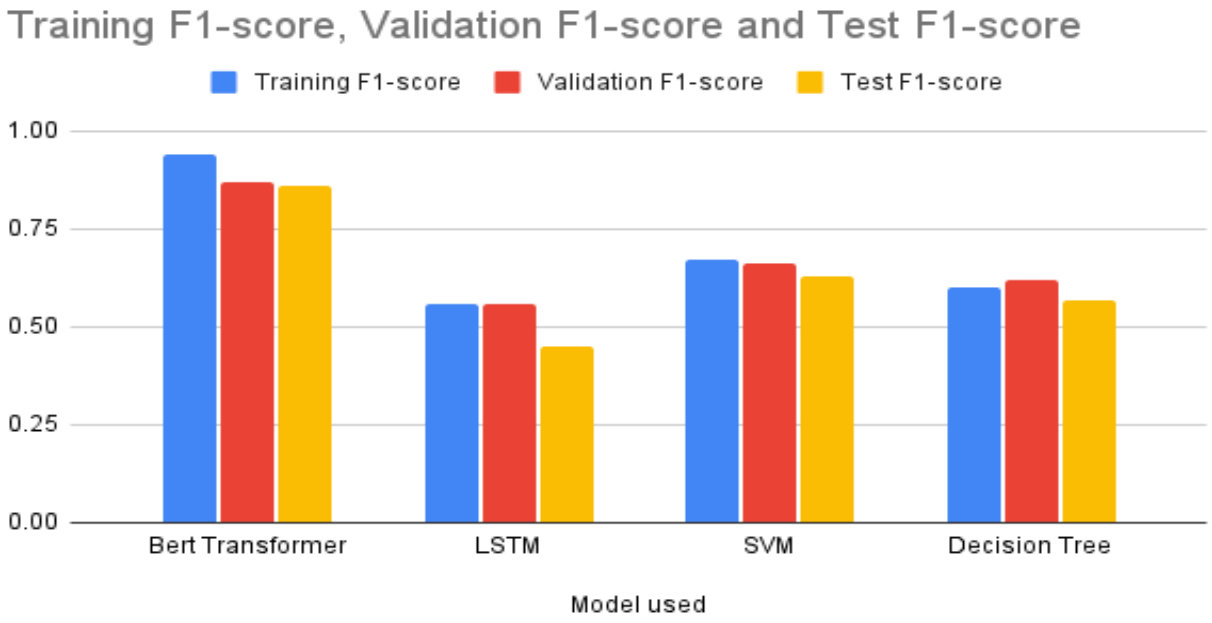
ii. Training and test

Using the previous table as our dataset we then pick SVM and Decision tree to test our hypothesis. This approach can be very time and computational consuming especially we you have very large dataset, which was the case of this approach, we couldn't try every scenario (parameters optimization due to the that both models take to training) but we are confident that this approach can lead to better results.

Models	Parameters
SVM	`kernel = 'linear', gamma='auto'`
Decision Tree	`max_depth=25`

6. Experiment and Results

Model used	Training F1-score	Validation F1-score	Test F1-score
Bert Transformer	0.94	0.87	0.86
LSTM	0.56	0.56	0.45
SVM	0.67	0.66	0.63
Decision Tree	0.60	0.62	0.57



Results

7. Conclusion

In this challenge, we performed several embedding techniques such as bag of words, and Bert. We learned how to perform basic pre-processing on data. We chose the Bert transformer after assessing all the models based on accuracy and auc. We use the outputs of the Bert transformer model for the submissions. We observed how tough it was to tune models and pre-process data in order to achieve good auc and accuracy. We could improve our model in the future by detecting the phrases that the model uses to classify the sentiment as positive, negative, or neutral.

8. Additional Resource:

Colab Notebook TFIDF