National and Kapodistrian University of Athens

Department of Informatics and Telecommunications

Deep Learning for Natural Language Processing

Project IV - Transformers using BERT

Chalkias Spyridon

January 2022

Contents

1 BERT Sentiment Classifier

1.1 Task

The task is to perform sentiment analysis on a dataset consisting of tweets and be able to infer whether a new tweet is:

1. Neutral
2. Anti-vax
3. Pro-vax

by fine-tuning the pretrained **BERT**-base model available on **Hugging Face**.

1.2 Data Preprocessing

Real-world data tend to be incomplete, noisy, and inconsistent. This can lead to a poor quality of collected data and further to a low quality of models built on such data. In order to address these issues, Data Preprocessing provides operations which can organise the data into a proper form for better understanding in data mining process.
Some of the methods applied in order to prepare the data for analysis are:

- Data Cleaning
- Data Reduction

1.2.1 Data Cleaning

The cleaning process involved two (2) operations:

- Checking whether there's need to drop data rows that contain *null* values. Luckily, the data had no such rows.
- Cleaning tweets from
  - URLs.
  - Noise contained in '< >'parenthesis.
  - The phrase 'RT' (meaning retweet).
  - '@' and the following twitter account name.
  - Any punctuation.
  - '\r\n' which is present in some strings.
  - Numbers, capitalization and white space.

  which are totally useless and add noise to the data.

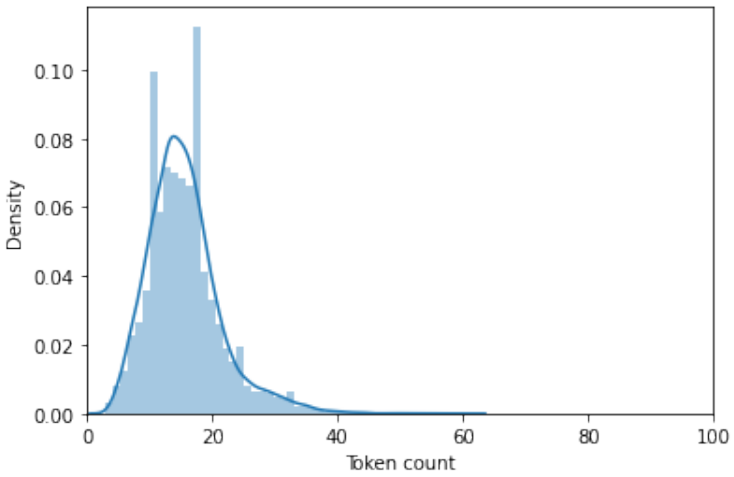
1.2.2 Data Reduction

Data had a significant reduction process through *Tokenization and Lemmatization*, in order to group together the inflected forms of the words so they can be analysed in less items which are more independent and have greater semantic content. Also, all the stopwords were removed, so that data delivers more semantic content with as few words as possible.

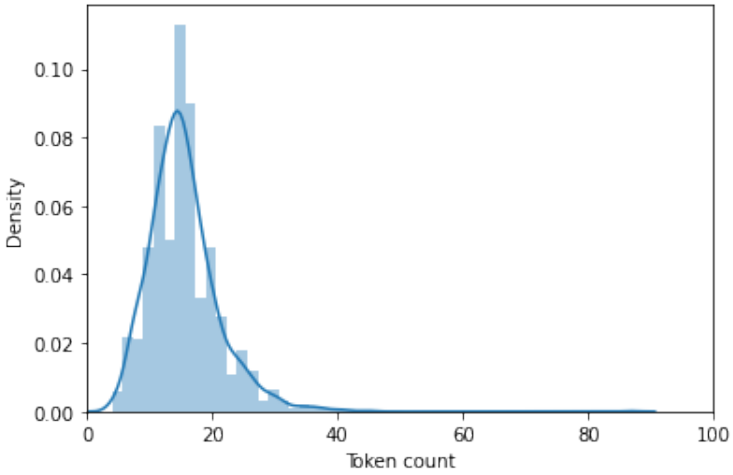
- The stopwords container used was provided by *NLTK* library.
- The lemmatization process was supported by *spaCy* library.
- The tokenization process was handled by Hugging Face's BERT Tokenizer.

In order to choose the average sequence length of data, the corresponding plots were designed. As you can see, most of the tweets contained less than 100 tokens:

Training set

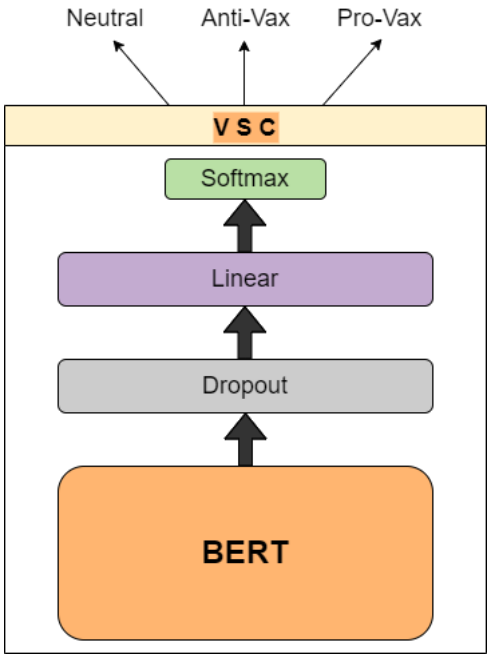


Validation set



1.3 Defining the Model

The main model is being represented by class **VSC** (*meaning Vaccine Sentiment Classifier*). As also shown in the diagram below, the class contains a pretrained BERT model, a dropout layer, a linear layer and finally a Softmax layer:



A **Pytorch Dataset** class has been implemented for all the tweets, named **TweetDataset** that contains all the necessary functionality in order to be fed into the aforementioned VSC model.

1.4 Hyperparameter Tuning

A custom **Trainer** class has been implemented, in order to fine-tune the BERT model. The class accepts a **configuration dictionary** and commences the training according to it. The parameters that can be tweaked are the following.

1.4.1 Number of training epochs

The ideal number of training epochs in which the model produces the best results and does **not** overfit, is **10 epochs**.

1.4.2 Batch size

Batch size was tested using values between 8 and 128 with the best results produced using **64 samples** per batch.

1.4.3 Dropout ratio for VSC class

The dropout layer in VSC class was tested using values between 0,1 and 0,5 with the best results produced using **0,3 dropout ratio**.

1.4.4 Maximum sequence length for data

Sequence length of data played one of the most important roles in fine-tuning. The length range was tested between values 30 and 170, with **60** being the **optimal sequence length**.

1.4.5 Optimizer

The project's classifier was tested by utilizing 4 optimizers:

- AdamW
- Adam
- SGD
- Adafactor

AdamW proved to yield the best training and validation losses and generalize the best amongst all its other competitors.

1.4.6 Learning rate

Because we are using AdamW as an optimizer, the best threshold for learning rate proved to be **1e-5**. The tested learning rates were $1e-5$, $3e-5$ and $0,1$.

1.4.7 Warmup steps for scheduler

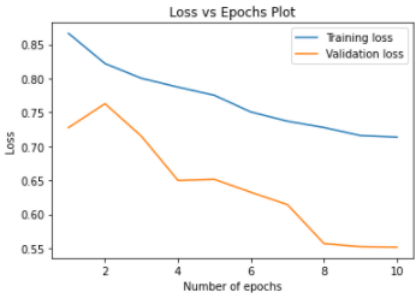
VSC class was trained using a scheduler whose warmup steps varied from 0 to 500. The optimal number of warmup steps for the scheduler proved to be **200 steps**.

1.5 Scores

Below is the demonstration of the experiments performed in order to fine-tune the BERT model. Initially, the model was trained with a random configuration and progressively the hyperparameters where tweaked in order to fine-tune the model.

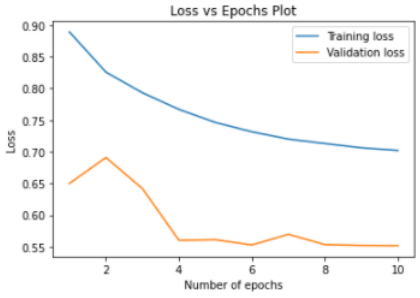
1.5.1 Scheduler's Warmup steps

```
Training with:
--------------
Batch size: 32
Number of epochs: 10
Max sequence length: 100
Dropout: 0.3
Optimizer: <class 'transformers.optimization.AdamW'>
Learning Rate: 1e-05
Warmup steps: 0
```



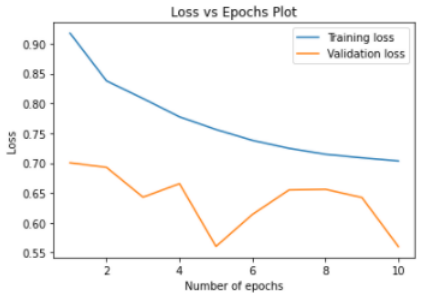
	precision	recall	f1-score	support
Neutral	0.80	0.80	0.80	1065
Anti-Vax	0.47	0.38	0.42	296
Pro-Vax	0.70	0.74	0.72	921
accuracy			0.72	2282
macro avg	0.66	0.64	0.65	2282
weighted avg	0.72	0.72	0.72	2282

```
Training with:
--------------
Batch size: 32
Number of epochs: 10
Max sequence length: 100
Dropout: 0.3
Optimizer: <class 'transformers.optimization.AdamW'>
Learning Rate: 1e-05
Warmup steps: 200
```



	precision	recall	f1-score	support
Neutral	0.82	0.79	0.80	1065
Anti-Vax	0.52	0.40	0.45	296
Pro-Vax	0.70	0.78	0.74	921
accuracy			0.73	2282
macro avg	0.68	0.66	0.66	2282
weighted avg	0.73	0.73	0.73	2282

```
Training with:
--------------
Batch size: 32
Number of epochs: 10
Max sequence length: 100
Dropout: 0.3
Optimizer: <class 'transformers.optimization.AdamW'>
Learning Rate: 1e-05
Warmup steps: 500
```

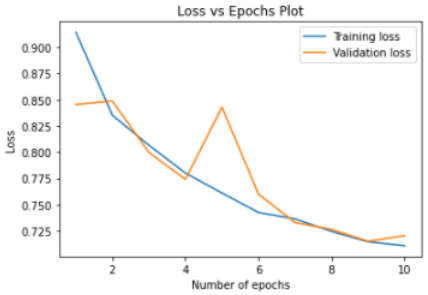


	precision	recall	f1-score	support
Neutral	0.80	0.80	0.80	1065
Anti-Vax	0.53	0.43	0.47	296
Pro-Vax	0.70	0.74	0.72	921
accuracy			0.73	2282
macro avg	0.68	0.66	0.66	2282
weighted avg	0.72	0.73	0.73	2282

In the above examples, **200 warmup steps** seem to perform the best, since weighted average precision, recall and f1-score are at **73%**, while the other models do not achieve such scores. However, the loss vs epoch diagrams seem to underfit a bit, so we'll try to fix that in the following experiments.

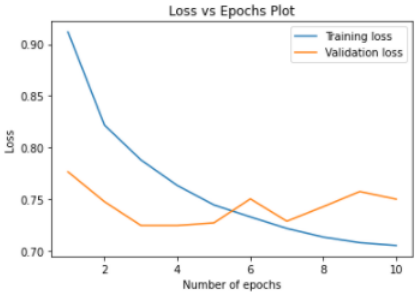
1.5.2 Sequence Length

```
Training with:
--------------
Batch size: 64
Number of epochs: 10
Max sequence length: 100
Dropout: 0.3
Optimizer: <class 'transformers.optimization.AdamW'>
Learning Rate: 1e-05
Warmup steps: 200
```



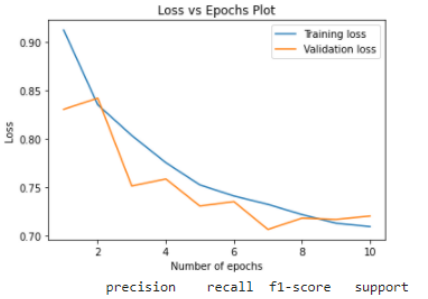
	precision	recall	f1-score	support
Neutral	0.82	0.77	0.80	1065
Anti-Vax	0.53	0.42	0.47	296
Pro-Vax	0.69	0.79	0.74	921
accuracy			0.73	2282
macro avg	0.68	0.66	0.67	2282
weighted avg	0.73	0.73	0.73	2282

```
Training with:
--------------
Batch size: 64
Number of epochs: 10
Max sequence length: 60
Dropout: 0.3
Optimizer: <class 'transformers.optimization.AdamW'>
Learning Rate: 1e-05
Warmup steps: 200
```



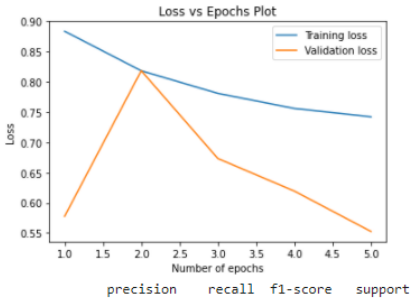
	precision	recall	f1-score	support
Neutral	0.81	0.79	0.80	1065
Anti-Vax	0.56	0.47	0.51	296
Pro-Vax	0.71	0.77	0.74	921
accuracy			0.74	2282
macro avg	0.69	0.68	0.68	2282
weighted avg	0.74	0.74	0.74	2282

```
Training with:
--------------
Batch size: 64
Number of epochs: 10
Max sequence length: 30
Dropout: 0.3
Optimizer: <class 'transformers.optimization.AdamW'>
Learning Rate: 1e-05
Warmup steps: 200
```



	precision	recall	f1-score	support
Neutral	0.82	0.78	0.80	1065
Anti-Vax	0.53	0.41	0.46	296
Pro-Vax	0.69	0.78	0.73	921
accuracy			0.73	2282
macro avg	0.68	0.66	0.66	2282
weighted avg	0.73	0.73	0.73	2282

```
Training with:
--------------
Batch size: 24
Number of epochs: 5
Max sequence length: 170
Dropout: 0.3
Optimizer: <class 'transformers.optimization.AdamW'>
Learning Rate: 1e-05
Warmup steps: 200
```



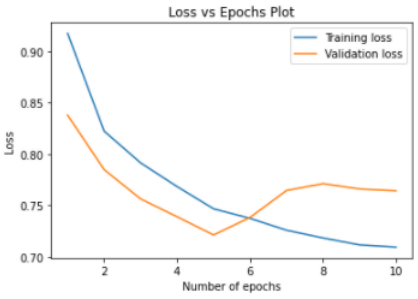
	precision	recall	f1-score	support
Neutral	0.79	0.79	0.79	1065
Anti-Vax	0.54	0.38	0.45	296
Pro-Vax	0.68	0.75	0.71	921
accuracy			0.72	2282
macro avg	0.67	0.64	0.65	2282
weighted avg	0.72	0.72	0.72	2282

As seen above, the ideal value for sequence length is **60**, since it produces

weighted average precision, recall and f1-score at **74%**, higher than any configuration so far. Also, the loss vs epochs diagram is pretty good, as the generalization gap is small, indicating that the model is neither overfitting or underfitting.

1.5.3 Dropout ratio

```
Training with:
--------------
Batch size: 64
Number of epochs: 10
Max sequence length: 60
Dropout: 0.5
Optimizer: <class 'transformers.optimization.AdamW'>
Learning Rate: 1e-05
Warmup steps: 200
```



	precision	recall	f1-score	support
Neutral	0.81	0.79	0.80	1065
Anti-Vax	0.57	0.44	0.50	296
Pro-Vax	0.69	0.76	0.72	921
accuracy			0.73	2282
macro avg	0.69	0.66	0.67	2282
weighted avg	0.73	0.73	0.73	2282

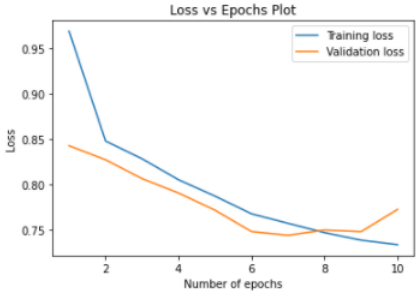
```
Training with:
--------------
Batch size: 64
Number of epochs: 10
Max sequence length: 60
Dropout: 0.1
Optimizer: <class 'transformers.optimization.AdamW'>
Learning Rate: 1e-05
Warmup steps: 200
```

	precision	recall	f1-score	support
Neutral	0.82	0.77	0.80	1065
Anti-Vax	0.56	0.42	0.48	296
Pro-Vax	0.68	0.78	0.73	921
accuracy			0.73	2282
macro avg	0.69	0.66	0.67	2282
weighted avg	0.73	0.73	0.73	2282

Changing the dropout ratio, does not seem to have any positive effect to the overall scores so far.

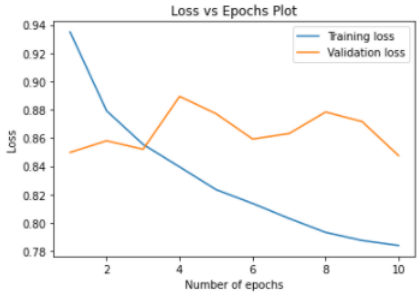
1.5.4 Optimizer

```
Training with:
--------------
Batch size: 64
Number of epochs: 10
Max sequence length: 60
Dropout: 0.3
Optimizer: <class 'torch.optim.adam.Adam'>
Learning Rate: 1e-05
Warmup steps: 200
```



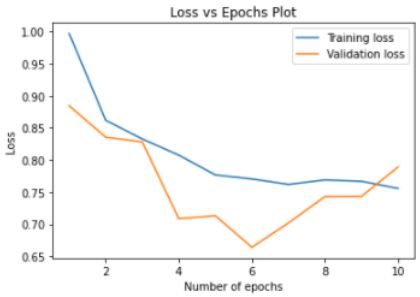
	precision	recall	f1-score	support
Neutral	0.81	0.79	0.80	1065
Anti-Vax	0.52	0.42	0.46	296
Pro-Vax	0.69	0.76	0.72	921
accuracy			0.73	2282
macro avg	0.67	0.65	0.66	2282
weighted avg	0.72	0.73	0.72	2282

```
Training with:
--------------
Batch size: 64
Number of epochs: 10
Max sequence length: 60
Dropout: 0.3
Optimizer: <class 'torch.optim.sgd.SGD'>
Learning Rate: 0.1
Warmup steps: 200
```



	precision	recall	f1-score	support
Neutral	0.78	0.81	0.79	1065
Anti-Vax	0.00	0.00	0.00	296
Pro-Vax	0.64	0.82	0.72	921
accuracy			0.71	2282
macro avg	0.47	0.54	0.50	2282
weighted avg	0.62	0.71	0.66	2282

```
Training with:
--------------
Batch size: 64
Number of epochs: 10
Max sequence length: 60
Dropout: 0.3
Optimizer: <class 'transformers.optimization.Adafactor'
Learning Rate: None
Warmup steps: 200
```

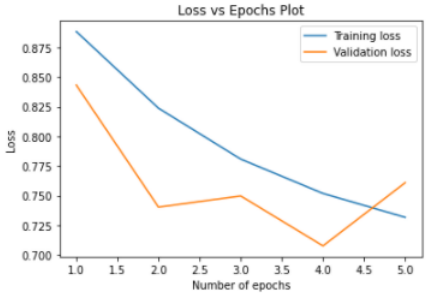


	precision	recall	f1-score	support
Neutral	0.83	0.76	0.79	1065
Anti-Vax	0.51	0.38	0.44	296
Pro-Vax	0.67	0.78	0.72	921
accuracy			0.72	2282
macro avg	0.67	0.64	0.65	2282
weighted avg	0.72	0.72	0.72	2282

Based on the classification report, Adam performed better than SGD and Adafactor, but not as good as AdamW.

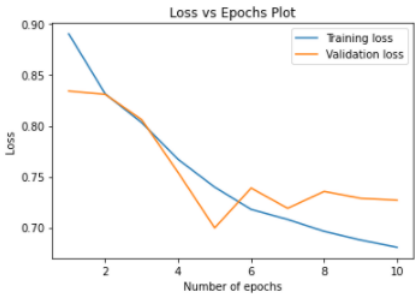
1.6 Learning Rate

```
Training with:
--------------
Batch size: 64
Number of epochs: 5
Max sequence length: 30
Dropout: 0.3
Optimizer: <class 'transformers.optimization.AdamW'>
Learning Rate: 3e-05
Warmup steps: 200
```



	precision	recall	f1-score	support
Neutral	0.80	0.80	0.80	1065
Anti-Vax	0.55	0.37	0.44	296
Pro-Vax	0.69	0.77	0.73	921
accuracy			0.73	2282
macro avg	0.68	0.65	0.66	2282
weighted avg	0.73	0.73	0.73	2282

```
Training with:
--------------
Batch size: 64
Number of epochs: 10
Max sequence length: 60
Dropout: 0.3
Optimizer: <class 'transformers.optimization.AdamW'>
Learning Rate: 3e-05
Warmup steps: 200
```

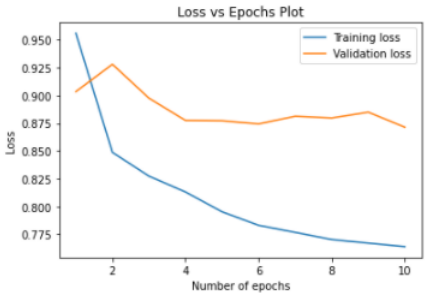


	precision	recall	f1-score	support
Neutral	0.81	0.79	0.80	1065
Anti-Vax	0.52	0.47	0.50	296
Pro-Vax	0.70	0.74	0.72	921
accuracy			0.73	2282
macro avg	0.68	0.67	0.67	2282
weighted avg	0.73	0.73	0.73	2282

$3e-05$ produced pretty high precision, recall and f1 scores (73%), but not as high as $1-e05$ did.

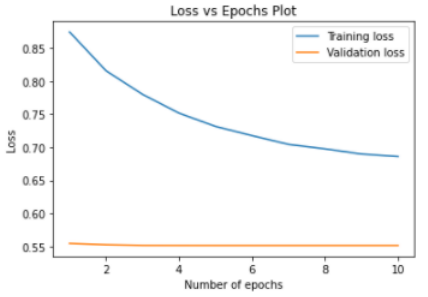
1.6.1 Batch size

```
Training with:
--------------
Batch size: 128
Number of epochs: 10
Max sequence length: 60
Dropout: 0.3
Optimizer: <class 'transformers.optimization.AdamW'>
Learning Rate: 1e-05
Warmup steps: 200
```



	precision	recall	f1-score	support
Neutral	0.77	0.80	0.79	1065
Anti-Vax	0.00	0.00	0.00	296
Pro-Vax	0.63	0.81	0.71	921
accuracy			0.70	2282
macro avg	0.47	0.54	0.50	2282
weighted avg	0.62	0.70	0.66	2282

```
Training with:
--------------
Batch size: 8
Number of epochs: 10
Max sequence length: 60
Dropout: 0.3
Optimizer: <class 'transformers.optimization.AdamW'>
Learning Rate: 1e-05
Warmup steps: 200
```



	precision	recall	f1-score	support
Neutral	0.81	0.79	0.80	1065
Anti-Vax	0.56	0.44	0.49	296
Pro-Vax	0.70	0.77	0.73	921
accuracy			0.74	2282
macro avg	0.69	0.67	0.68	2282
weighted avg	0.73	0.74	0.73	2282

By looking at all of the above, the best results were produced by using **64 samples** per batch. 128 samples per batch produced poor classification report, while 8 samples per batch underfitted the model.

1.7 Best Performing Model

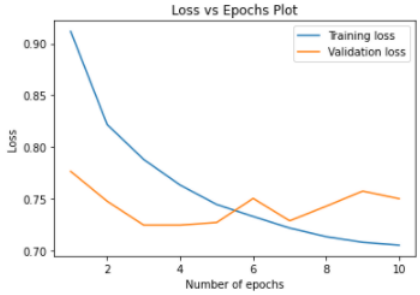
The overall best performing model is a pipeline that consists of:

- A pretrained BERT-base-uncased model.
- A dropout layer with $0,3$ dropout ratio.
- A linear layer.
- A softmax layer.
- Batch size: 64
- Number of training epochs: 10
- Max sequence length: 60
- Optimizer: AdamW

- Loss function: Cross Entropy Loss
- Learning rate: 1e-05
- Scheduler's warmup steps: 200

Let's plot the results again:

```
Training with:
--------------
Batch size: 64
Number of epochs: 10
Max sequence length: 60
Dropout: 0.3
Optimizer: <class 'transformers.optimization.AdamW'>
Learning Rate: 1e-05
Warmup steps: 200
```



	precision	recall	f1-score	support
Neutral	0.81	0.79	0.80	1065
Anti-Vax	0.56	0.47	0.51	296
Pro-Vax	0.71	0.77	0.74	921
accuracy			0.74	2282
macro avg	0.69	0.68	0.68	2282
weighted avg	0.74	0.74	0.74	2282

2 Sources

- *Hugging Face*
- *Kaggle*
- *CS224N Stanford class*