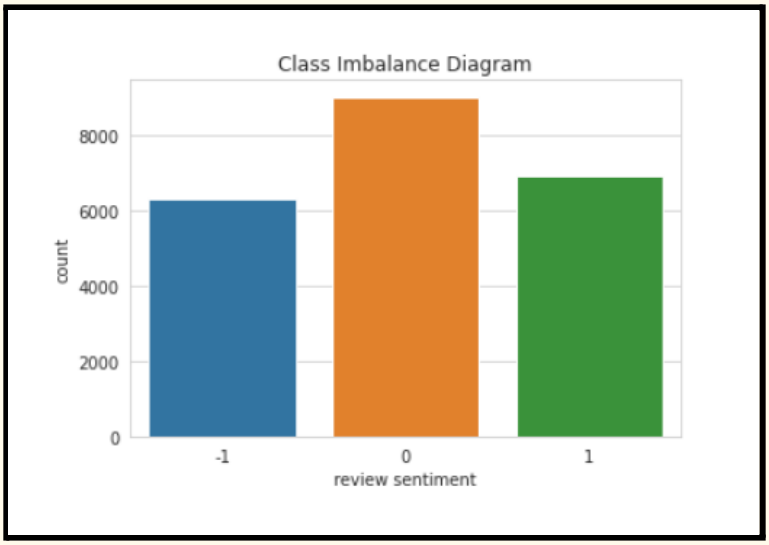
ABESSERA Zachary - CHAPPAT Quentin - COLIN Laure-Amélie - MAILLARD Lucien

AML Challenge 3

Table of content:

I. Exploring the dataset

The goal of this challenge is to classify tweets with positive, neutral or negative labels, reflecting the sentiment of the user writing the tweet. Our training dataset is made of around **24000 tweets**, which have **68 characters** in average, and the dataset is quite balanced, with a bit more neutral tweets (41%):



	positive tweets	negative tweets	neutral tweets
0	amazing	am	are
1	are	bad	back
2	awesome	be	be
3	be	bored	can
4	best	can	day
5	better	day	do
6	cool	don	get
7	day	feel	go
8	excited	get	going
9	fun	good	got
10	good	hate	have
11	great	have	is
12	happy	hurts	just
13	have	im	know
14	hope	is	lol
15	is	just	my
16	just	miss	not
17	love	my	now
18	mother	not	one
19	mothers	now	out
20	much	really	so

We also discovered what were the top words in positive, negative and neutral tweets. For instance, the words “love” and “amazing” are abundant in positive tweets whereas the word “bad” can be detected in negative tweets. Such words are essential in NLP.

In the baseline, the label “positive” is converted to 1, “neutral” to 0 and “negative” to -1. In order to not get a CUDA error in our final model, we converted “positive” to 2, “neutral” to 1 and “negative” to 0.

II. Cleaning of the data

The data are highly **heterogeneous** with emoticons, urls, whitespaces etc. That leads us to make a huge cleaning of the data for the future training of our model.

We applied some operations on our raw data as we cannot go straight from raw text to fitting a deep learning model. Here is a list of these operations:

- Erase all **whitespaces** in order to be sure that every word is separated by one space.
- Delete every **urls and usernames** in the dataset as they are not conveying any emotion.
- **Convert the emojis** into text:

Ex: :) = happy

- Use **spacy lemmatizer** to get words into basic form: lemmatization removes the 3 grammar types of tense and transforms each word into its original form.
- Remove **numerical numbers** : they are hard to analyze to detect the sentiment conveyed by a tweet.
- Remove **stopwords** : they are abundant in the tweets.. By removing them, we remove the low-level information from our text in order to give more focus on the emotion.
- Remove some **punctuation**

III. First models: Bayesian classifier and SVM

Our first approach was to try the **baseline code with Bag-of-Words** and perform our data processing on the data before applying the bayesian classifier. It slightly improves the results but we came soon to the conclusion that this model would not be good enough to produce a good sentiment analysis.

Then, we implemented a **SVM with 3 different kernel methods**: linear, polynomial and rbf kernel. But, again, it didn't help to improve the results a lot, even with a good data cleaning, which leads us to use a more advanced algorithm, after we read https://huggingface.co/docs/transformers/index .

IV. Bert Algorithm

BERT (Bidirectional Encoder Representations from Transformers) is a recent algorithm published by researchers at Google AI Language. It has been described as a **huge revolution in NLP** and it can be used for sentiment classification.

We first saw articles that show how they did sentiment prediction on the IMDB dataset using the BERT algorithm. Our code implementation of BERT is heavily inspired by an article found on Medium:
https://mostefasihamdi.medium.com/analyse-des-sentiments-avec-bert-a-laide-de-hugging-face-fde3c0993d26

To apply the pre-trained model BERT, we had to use the **tokenizer provided by the Hugging Face Transformers library**. Some basic operations can convert the text to tokens and tokens to unique integers (ids). BERT requires a specific, fixed vocabulary and we had to add special tokens to separate sentences and do classification.

Example:

```
 Sentence: When was I last outside? I am stuck at home for 2 weeks.
   Tokens: ['When', 'was', 'I', 'last', 'outside', '?', 'I', 'am', 'stuck', 'at', 'home', 'for', '2', 'weeks', '.']
Token IDs: [1332, 1108, 146, 1314, 1796, 136, 146, 1821, 5342, 1120, 1313, 1111, 123, 2277, 119]
```

We used the basic **BERT model** and built our classifier on top of it. We use a dropout layer for some regularization and a fully-connected layer for our output. To get the predicted probabilities from our trained model, we applied the softmax function to the outputs:

To **hypertune our Bert classifier**, we chose the following **parameters**, based on recommandations we found in the **litterature**:

Batch size	**32**
Learning rate	**3e-5**
Number of epochs	**3**

In the training loop, the BERT algorithm performs a forward pass to compute logits and loss. Then, it performs a backward pass to compute gradients:

```
loss.backward()
```

It clips the norms of the gradients to 1.0 to prevent extremely large updates in the neural network (exploding gradients):

```
nn.utils.clip_grad_norm_(model.parameters(), max_norm=1.0)
```

After that, it updates the model's parameters with:

```
optimizer.step()
```

And the learning rate with:

```
scheduler.step()
```

The evaluation of the algorithm is made by computing loss and accuracy rate over the validation set.

VI. Conclusion

This challenge offered us the opportunity to work with text data for the first time and to discover what their specificities were. The amount of time dedicated to the challenge was essentially divided in **2 big parts**:

1) **Manipulate text data and clean them**, even if the operations we applied on the tweets were sometimes less useful than we imagined.
2) Work/practice with **BERT tokenizer** and **BERT classifier**, which are advanced algorithms and which enable us to see what was the state-of-the-art in Natural Language Processing.

Finally, with a good data pre-processing and the BERT algorithm, we got our best result, which is a **fbeta score of 0.87**.