Text analysis with deep learning techniques

Rosario Urso

December 2023

Contents

1 Introduction

The work conducted concerns the application of Deep Learning techniques, particularly **BERT**, which stands for *Bidirectional Encoder Representations from Transformers*, for Sentiment Analysis about the classification of tweets.
Due to the high computational power required, the work was carried out on a small part of the provided corpus, and in order to make comparisons between the implemented methods, an analysis was performed on a dataset downloaded from the Kaggle website.
The results detect good performance of the neural network model, considering both loss and accuracy of the model.
This was done by running several tests and going to set different hyperparameters and different training, testing and validation rates for each model each time.

2 Text analysis approach

2.1 Natural Language Processing

Natural Language Processing (**NLP**) is a branch of Artificial Intelligence by which machines are given the ability to read, understand and extract meanings from human language. [7]
In a world in which all of us are constantly inundated with textual data, NLP is particularly useful in that it is able to combine human language with the field of computer science in order to formulate models that can analyze texts and capture meaningful details from them.
Making very brief references to history, the first application of natural language processing occurred from the 1940s, with the goal of building a machine capable of translating text from one language to another in a fully automatic way. [8]
Closely applied to natural language processing can be found Sentiment Analysis, which is a subset of NLP that focuses on identifying, analyzing and understanding emotions, attitudes or feelings expressed in natural language through written texts, speeches or other forms of human communication. [8] In the presented case, with 2 datasets already labeled and an unlabeled corpus, the main purpose was to identify the nature of sentiment of Italian-language tweets.

2.2 Neural Networks

Sentiment Analysis applied to NLP employs various techniques and approaches, among which neural networks are included. The use of neural networks, such as Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), but especially with **transformers** (such as BERT, GPT, etc.) are able to capture and understand complex relationships within texts for sentiment analysis.
The work was conducted after training results on networks such as LSTM, Bi-LSTM, GRU (applied to two different Word Embedding models such as Word2Vec and FastText) and on BERT models.

3 A variety of Neural Networks

3.1 LSTM

An **LSTM** (*Long Short-Term Memory*) neural network is a type of recurrent neural network (**RNN**) designed to model long-term dependencies.
They are used as an alternative to Recurrent Neural Networks since the latter show more difficulty when the information within the text begins to increase: this is surely related to the context of the reference, and since the LSTM network can store more memory, it is able to achieve better performance. [9]
As an example, consider trying to predict the last word in the following sentence "*I grew up in France... I speak fluent French*".
The available information suggests that the word may be a language, but only because of the context and thus the fact that the information was captured previously can the word **French** be answered correctly. [9]
If we had used a simple RNN in this case, it probably would not have provided the correct result.
An LSTM neural network works in the following way:

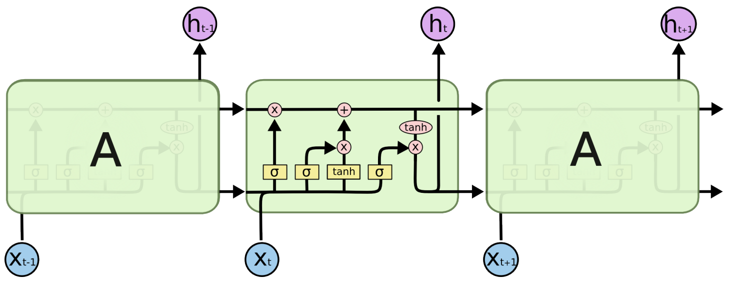


Figure 1: LSTM Neural Network

where each arrow carries a vector, the pink circles represent pointwise operations and the yellow symbols the layers learned by the neural network.
It should be kept in mind that the LSTM neural network has 3 types of layers:

- **Forget Gate**, which decides what information should be thrown out of the cell;
- **Input Gate**, which reports what information to save in the cell and is composed of two types of layers: a *sigmoid* layer (called the "**input gate layer**") and a *tanh* layer so defined in 1 and 2 equations:

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{1}$$

$$\tilde{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \tag{2}$$

These two layers are combined to obtain:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{3}$$

- **Output Gate**, where a sigmoid layer will be executed to select which parts of the cell state to produce:

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right) \tag{4}$$

Next, the cell state is passed through *tanh* and multiplied by the sigmoid gate output, so that only the previously selected parts are emitted:

$$h_t = o_t \cdot \tanh\left(C_t\right) \tag{5}$$

3.2 GRU

Cho et al. (2014) introduced a variant of the LSTM network called *Gated Recurrent Unit* (**GRU**).
The difference from the classical LSTM neural network is that the **update gate**, obtained by combining forget and input gate, is introduced in the GRU. [2] In addition, the hidden state and cell state are also joined.
Such a neural network is represented in figure 2:

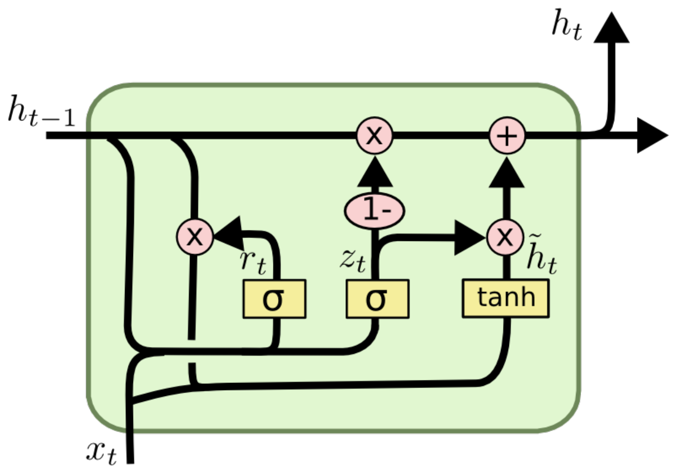


Figure 2: GRU Neural Network

where:

$$z_t = \sigma\left(W_z \cdot \left[h_{t-1}, x_t\right]\right) \tag{6}$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right) \tag{7}$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right) \tag{8}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{9}$$

It should be pointed out that there are several variants of GRUs and that some are better performing than LSTM networks.

3.3 Bi-LSTM

The set-up of the neureal networks **Bi-LSTM** (an acronym that stands for *Bidirectional Long-Short Term Memory*) bases on using a simple LSTM.
They are used to solve the problem of "*short memory*": in this case, two hidden layers are used to process data forward and backward (that's why <u>bidirectional</u>) based on LSTM networks, connecting the two hidden layers to the same output layer and storing both previous and subsequent information. For this reason, performance should be better than unidirectional LSTMs.
The Bi-LSTM neural network, shown in figure 3, works as follows:

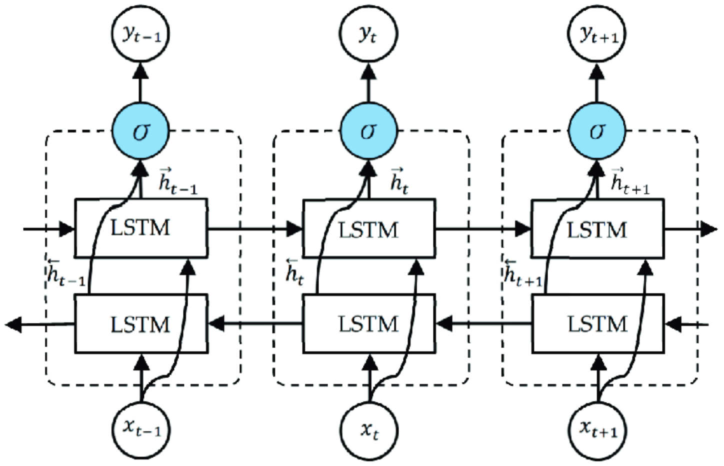


Figure 3: Bi-LSTM Neural Network

where σ is the activation function, H_t is the hidden layer input, and the output is generated by the forward $\overrightarrow{h_t}$ and backward $\overleftarrow{h_t}$ structure explicated as follows:

$$\overrightarrow{h_t} = \sigma\left(W_{x\vec{h}} x_t + W_{\vec{h}\vec{h}} \overrightarrow{h_{t-1}} + b_{\overleftrightarrow{h}}\right) \tag{10}$$

$$\overleftarrow{h_t} = \sigma\left(W_{x\overleftarrow{h}} x_t + W_{\overleftarrow{hh}} \overleftarrow{h_{t-1}} + b_{\overleftarrow{h}}\right) \tag{11}$$

$$H_t = W_{x\vec{h}} \vec{h} + W_{\overleftarrow{h}y} \vec{h} + b_y \tag{12}$$

3.4 BERT

BERT were introduced in 2018 by *Jacob Devlin et al.* of the Google AI Language group. This model, unlike recent language representation models, is designed to train deep representations from unlabeled text. [4]
There are two strategies for applying linguistic representations:

- *feature-based*, which is the pre-training step, in which the model is trained on unlabeled data.
- *fine-tuning*, phase in which the parameters are *tuned*.

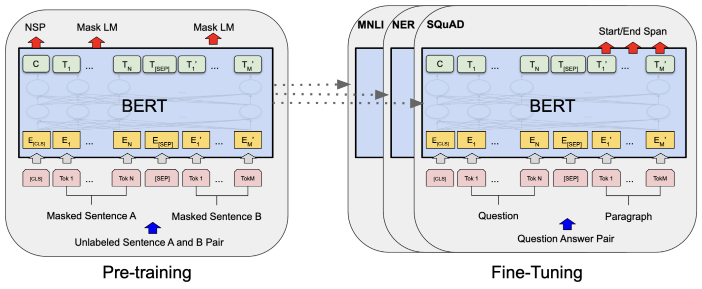


Figure 4: BERT Neural Network

In the case of fine-tuning approach (such as the **Generative Pre-trained Transformer - OpenAI GPT**, proposed by Radford et al. in 2018), the architecture is unidirectional (left-to-right) in which each token can only tend to the previous tokens. [11]
BERT overcomes this constraint by using a *masked language model* (**MLM**), in which, in a random sequence, some tokens are masked with the goal of reproducing them as accurately as possible.

In the figure proposed in 4, pairs of sentences are combined into a single sequence, separated by a special token [**SEP**], where we will differentiate the tokens related to sentence A and sentence B.
In the masking procedure, by which a percentage of the input tokens are randomly masked with the purpose of predicting them, the problem is not always solved by placing the token [**MASK**].
After that, the step of *Next Sentence Prediction* (**NSP**) is based on understanding the relationship between the two sentences so as to predict the sentence next to the one being considered. Finally, the fine-tuning phase is computationally less expensive than the pre-training phase.

3.4.1 Attention Mechanisms

BERTs, more generally the *Transformer Models*, is based on **attention mechanisms**, which avoids recurrence and allows parallelization within the training examples. [1]
The attention mechanism was introduced to improve the performance of the *encoder-decoder* model for machine translation. The idea behind the attention mechanism is to allow the decoder to use the most relevant parts of the input sequence in a flexible way, through a weighted combination of all the encoded input vectors, with the most relevant vectors being assigned the highest weight. [11]
The attention mechanism was introduced by Bahdanau et al. (2014), which is divided into graded calculations of alignment scores, weights and context vector: [3]

- The **alignment model** takes the encoded hidden states h_i and the previous output of the decoder s_{t-1} to compute the score e_{t-1}, which indicates how well the input sequence elements align with the current output at position t.
  The alignment model is represented by a function, $a(.)$, which can be implemented by a feed-forward neural network.

$$e_{t,i} = a(s_{t-1}, h_i) \tag{13}$$

- The **weights**, $\alpha_{t,i}$, were calculated using an operator *softmax* to the previously calculated alignment scores:

$$\alpha_{t,i} = softmax(e_{t,i}) \tag{14}$$

- The **context vector**, c_t is fed into the decoder at each step and is calculated as the weighted sum of all the hidden states of the encoder T:

$$c_t = \sum_{i=1}^{T} \alpha_{t,i} h_i \tag{15}$$

4 1st Empirical Analysis

4.1 Pre-processing and EDA step

The pre-processing step was performed on the datasets **Happy Parents** and **Sentipolc**. Regarding Happy Parents: this dataset (merged between training and test files) has **1507** tweets with different levels of sentiment, as illustrated in the figure below:

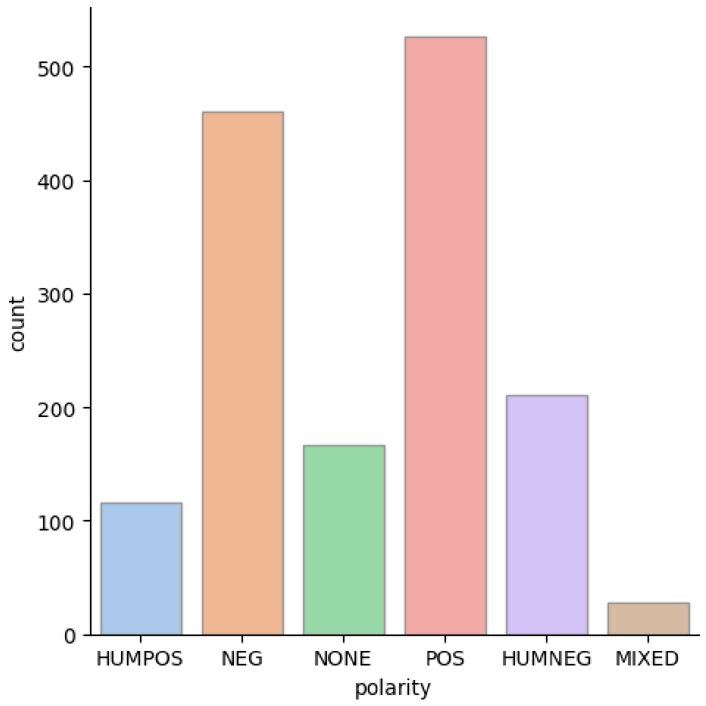


Figure 5: Happy Parents Distribution **before** pre-processing

For this reason, a merge was performed between categories (HUMPOS and POS and HUMNEG and NEG, while the other categories were removed since the classification to be performed is a binary type classification).6. After the first merge, the dataset is presented in figure 6:

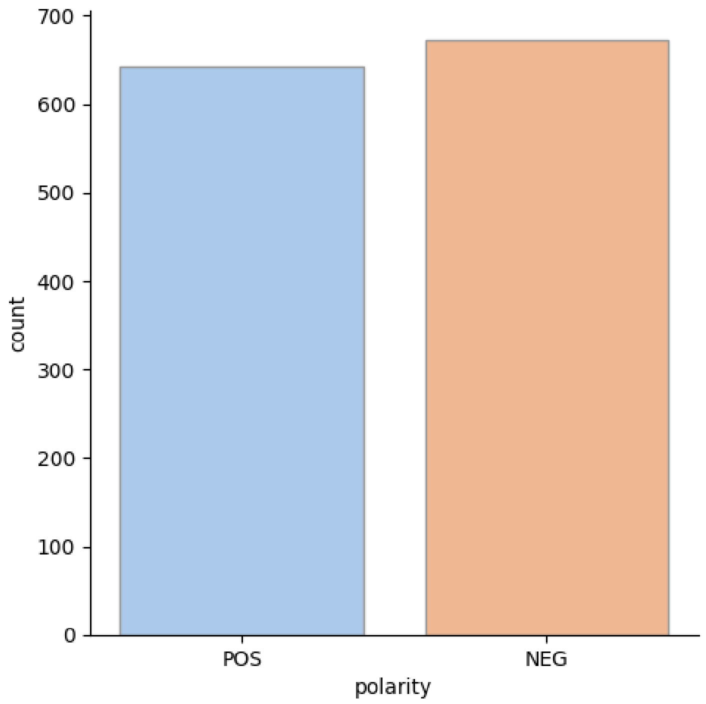


Figure 6: Happy Parents Distribution **after** pre-processing

where there is a balance between positive and negative sentiment.

Same pre-processing was done for **Sentipolc** (after training and test set were merged, the dataset contained **4310** observations). Initially, the classes were apparently very unbalanced, so again, the same coding was made as in *Happy Parents*. Positive sentiments (with *opos*) and negative sentiments (with *oneg*) were taken, with the differences being that the classes are coded as follows:

- $opos = 1$ e $oneg = 0$ as **positive**;
- $opos = 0$ e $oneg = 1$ as **negative**;
- $opos = 0$ e $oneg = 0$ as **neutral**;
- $opos = 1$ e $oneg = 1$ as **mixed**.

For the above mentioned considerations, the neutral and mixed categories were dropped, as shown in figure 7:

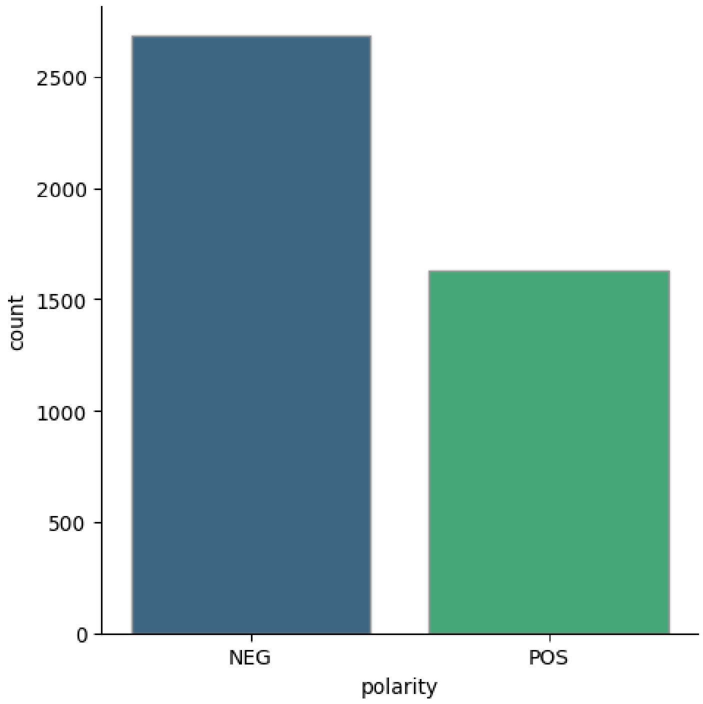


Figure 7: Sentipolc Distribution **after** pre-processing

In order to create a single labelled dataset, the two datasets were merged, forming one large dataset:

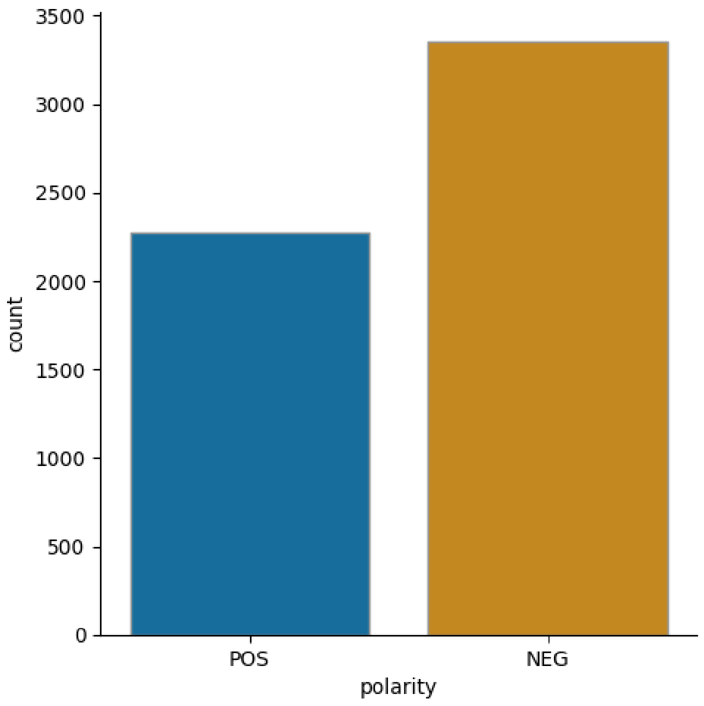


Figure 8: Sentipolc and Happy Parents joined Distribution

A cleaning task was performed on this entire dataset through which the words associated with the tweets in the datasets were made lowercase.
In addition, in order to proceed with the training of the **W2V** model and the creation of the **WordCloud**, punctuation and various symbols were eliminated, and finally stopwords were removed through the *nltk* library.
With the resulting tweets, a WordCloud was created, illustrated as follows:

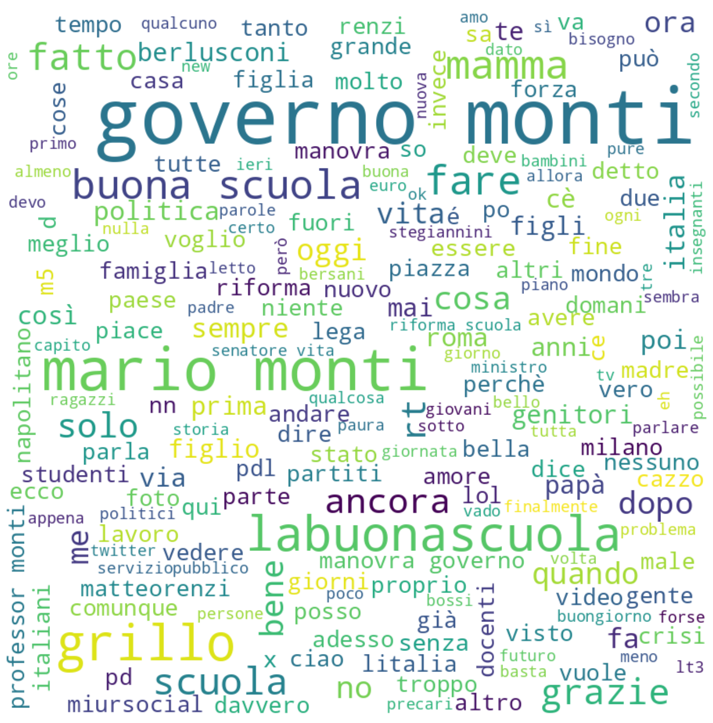


Figure 9: WordCloud Rappresentation

where the most frequent word appears to be related to the field of politics and institutions, namely "*government monti*", "*monti*" and "*school*". In order to proceed a more accurate textual analysis, the following is the **bigram** that presents a sequence of two consecutive elements within the tweets in the dataset:

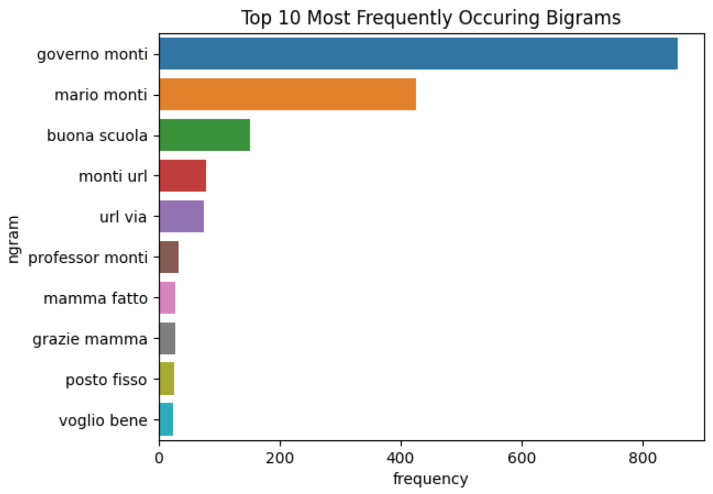


Figure 10: Bigram Rappresentation

In that graph it can be seen, confirming the WordCloud in the figure 9, that the most recurring words turn out to be "*government monti*" with a frequency greater than 800, "*mario monti*" and finally "*good school.*"

4.2 Model Training

After pre-processing the dataset through the above activities, the W2C model was trained on the unsupervised tweet corpus containing **15115421** tweets, dividing that corpus into training, test and validation set. **Word2VeC** is a word embedding algorithm that converts words into numerical vectors in a vector space of reduced dimension, and in the case under consideration in input was given:

- the **corpus of tweets**, which results in a list of lists, needed in order to proceed with training the model;
- the **size of vectors** containing the words, set to 300;
- the **window size** that the W2V model considers for training, set to 1;
- the **minimum word count** to be included in the vocabulary, set equal to 1, i.e., all words in the vocabulary are included;
- the **approach used** for training, in this case Skip-gram.
  This iterative approach, given a specific input word, attempts to predict the *approximate* words in the sentence given a specific reference context. It then aims to maximize the probability of generating the adjacent words (*skip words*) in a specified window. [6]

- **epoche**, which is how many times the model will see the entire dataset during training, in this case only once.

After calculating the matrix of embeddings for the words in the vocabulary, by means of the coefficients of the word considered (as an example, the word used here is "*lazio*"), a 3-dimensional graph was constructed through t-sne dimensionality reduction technique.

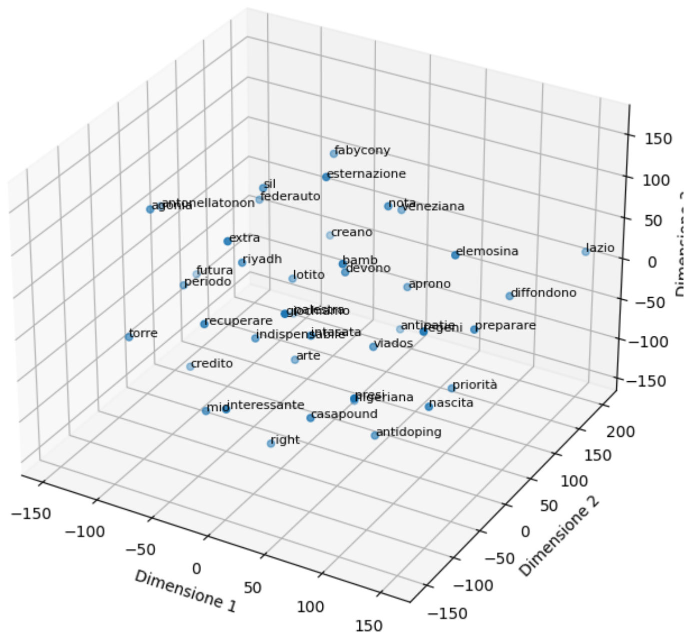


Figure 11: Embedding Space Word2Vec

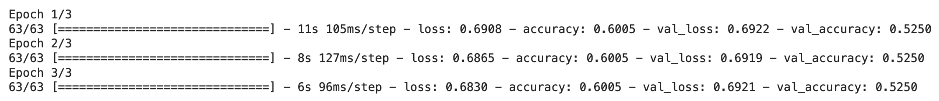
From the graph in figure 11, it can be seen that there are words related to the one considered, such as "*lotito*" and "*casapound*".
Medesima activity was accomplished using the model *FastText*, however, not present in this work.

After performing the procedure of the *One-Hot Encoding*, through the embeddings matrix, the Bi-LSTM neural network was trained: two bidirectional LSTM (*textbfLong short-term memory*) layers were used.
The first LSTM layer with 128 units and a dropout of 20% during training (*dropout* = 0.2), while the second LSTM layer has 64 units and a dropout of 0.2 again.
Finally, a single dense layer with 64 units and activation function **ReLU** was used. This activation function (*rectified linear ReLU unit*), which appears to be

the most widely used for neural networks, takes the following form: [5]

$$g(z) = (z)_+ = \begin{cases} 0 & \text{if } z < 0 \\ z & \text{otherwise} \end{cases} \tag{16}$$

This is followed by a final dense layer with 2 units for binary classification.
The results are shown below:



```
Epoch 1/3
63/63 [==============================] - 11s 105ms/step - loss: 0.6908 - accuracy: 0.6005 - val_loss: 0.6922 - val_accuracy: 0.5250
Epoch 2/3
63/63 [==============================] - 8s 127ms/step - loss: 0.6865 - accuracy: 0.6005 - val_loss: 0.6919 - val_accuracy: 0.5250
Epoch 3/3
63/63 [==============================] - 6s 96ms/step - loss: 0.6830 - accuracy: 0.6005 - val_loss: 0.6921 - val_accuracy: 0.5250
```

Figure 12: Bi-LSTM Results

The model was trained with a batch size of 64, 3 epochs, and a validation rate of 0.1. However, the results obtained do not appear satisfying since a strong limit was placed on the initial corpu
Therefore, in order to place a comparison between the different neural networks, it was necessary to perform the analysis on an additional dataset downloaded from the *Kaggle* platform.

5 2nd Empirical Analysis

For the purpose of elaborating comparisons on neural networks, a dataset was used from the Kaggle[1] website concerning mental illness.
This dataset is a collection of texts about people with anxiety, depression and other mental health problems, where the corpus consists of two columns: one containing comments and the other containing labels where **1** indicates the presence of any mental health problem, while **0** identifies its absence.

5.1 Pre-processing and EDA step

The pre-processing procedure follows the same approach done for the previous dataset. After loading the dataset, which includes **27977** observations, any duplicate tweets present in the dataset were dropped and at the end the classes related to mental illnesses are presented as follows:

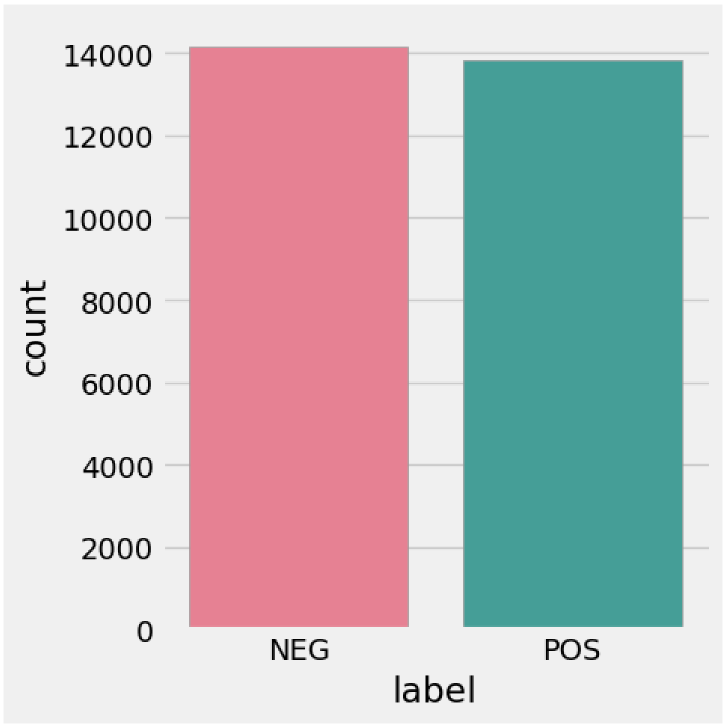


Figure 13: Dataset Structure **after** pre-processing

The classes thus appear to be balanced between negative and positive (**14139** observations for negative and **13838** for positive).

[1]The dataset can be downloaded from the following link: https://www.kaggle.com/datasets/reihanenamdari/mental-health-corpus

Differences between tweets defined as positive and negative emerge in terms of length, where negative tweets appear longer than positive tweets, which appear shorter and more concise; all except for the average word length where longer words are present on average in positive tweets. This just described can be deduced from the combined graph in figure 14:

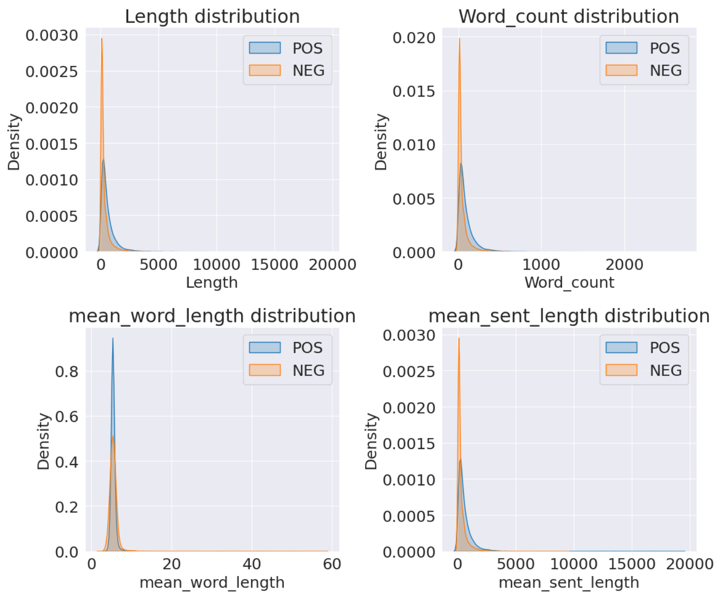


Figure 14: Text Analysis using KDE

In the following, a sequence of two consecutive elements within the tweets in the dataset can be identified, just as performed above, using the bigram exposed below.

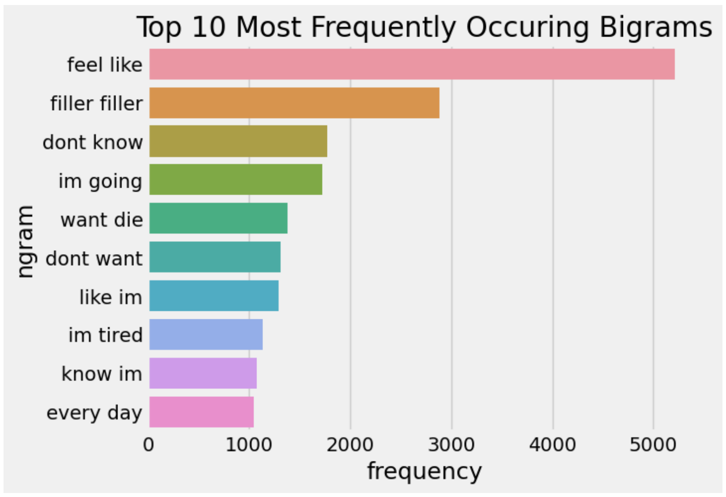


Figure 15: Bigram Rappresentation Mental Health

It can be seen from the bigram in figure **??** that the most frequent words in tweets are "*feel like*." It can be seen that words such as "*want die*" are present, identifying tweets presumably labeled as negative.
The trigram has also been reproduced, which does not differ significantly from the bigram and for that reason has not been reported here.
Instead, the wordcloud is present:

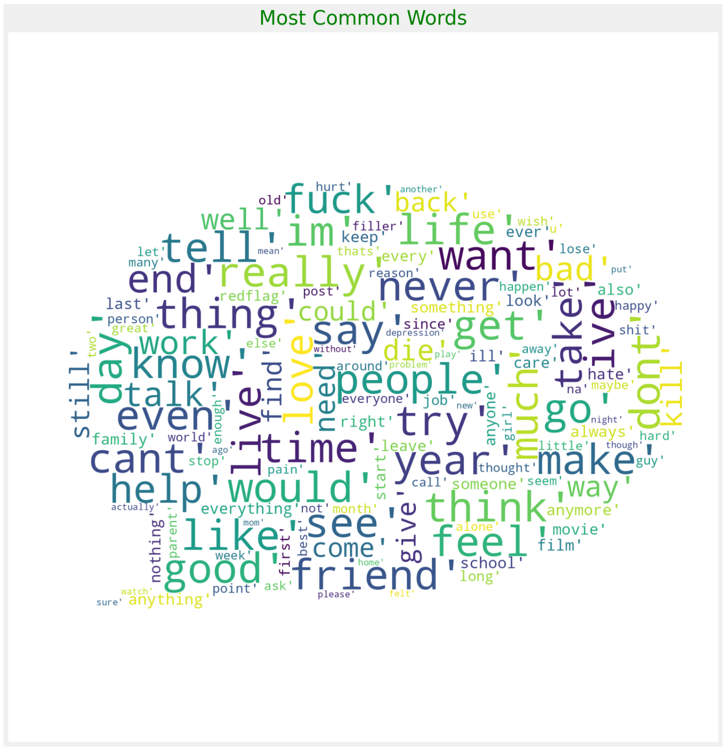


Figure 16: WordCloud Rappresentation Mental Health

Following this, the removal of stopwords was performed. It was followed by tokenization, again using the **nltk** library, through which any URLs, links and emoticons present in the various tweets within the dataset were also removed. In this case, a process of **lemmatization** was additionally applied by which the various forms of a word are mapped into the canonical form also known as lemma.
The dataset was divided as follows:

- **training set**, 80% of the observations i.e. 22381;
- **test set**, 20% of the observations i.e. 5596;
- **validation set**, 10% of the test observations.

5.2 Model Training

Performed the training of the W2V model (with the difference that in this case, due to the small size of the dataset, the number of epochs was set to 8 and not 1 like the previous dataset), the visualization of the embedding space taking the word ”*depression*” as reference is displayed below:

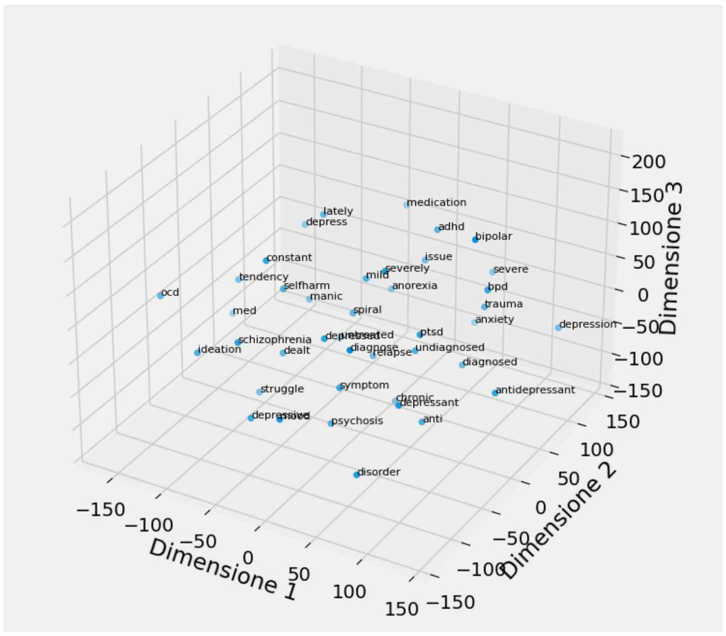


Figure 17: Embedding Space Word2Vec Mental Health

From the graph in figure 17 it can be seen that there is a strong semantic link between all the words in the 3-dimensional Word2Vec space.
Finally, following the training of the W2V model, several neural network models were estimated.

	LSTM	**Bi-LSTM**	**GRU**
TrainAccuracy	0.6739	0.7688	0.7018
ValidAccuracy	0.7258	0.7950	0.6637

Table 1: LSTM, Bi-LSTM and GRU Accuracy

A graphic representation concerning accuracy and loss is provided in the figure below:

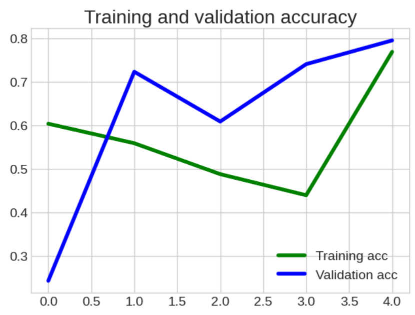


Figure 18: Training and Validation Set Accuracy

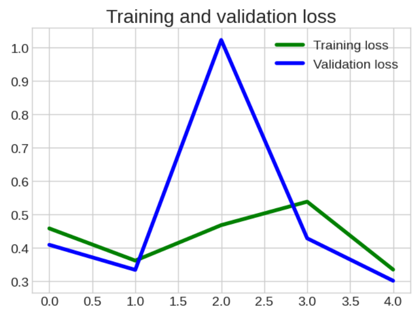


Figure 19: Training and Validation Set Loss

As a result of the different tests by changing values to the hyperparameters and the multiple settings computed to the relevant networks (epochs, number of layers, batch size), the network that best fit the data was the **Bidirectional LSTM**.
Once the best neural network was identified, the model was also trained on the test set with the following results:

	Accuracy	**Loss**
Test Set	0.80575	0.30904

Table 2: Test Set Accuracy and Loss

Finally, attention was focused on the BERT neural network model, which provided the following results:

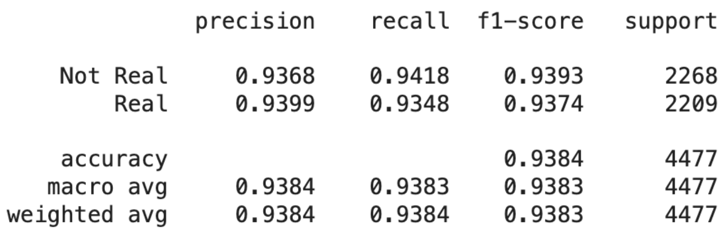
```
              precision    recall  f1-score   support

    Not Real     0.9342    0.9372    0.9357      2802
        Real     0.9368    0.9338    0.9353      2794

    accuracy                         0.9355      5596
   macro avg     0.9355    0.9355    0.9355      5596
weighted avg     0.9355    0.9355    0.9355      5596
```

Figure 20: Test Set Metrics

where for *Not Real* the category 0 is represented, and for *Real* the label 1 is reported. Regarding the metrics on the test set, they showed significantly better results than the previously tested networks with an accuracy of **0.9355**.
The same conclusions can be drawn when considering the Validation Set, which shows the following results:



```
              precision    recall  f1-score   support

    Not Real     0.9368    0.9418    0.9393      2268
        Real     0.9399    0.9348    0.9374      2209

    accuracy                         0.9384      4477
   macro avg     0.9384    0.9383    0.9383      4477
weighted avg     0.9384    0.9384    0.9383      4477
```

Figure 21: Validation Set Metrics

As shown in figure 21, the Validation Set returned even better accuracy than the Test Set, confirming that the BERT network in this particular case worked better than LSTM, Bi-LSTM and GRU.

References

[1] Francisca Adoma Acheampong, Henry Nunoo-Mensah, and Wenyu Chen. "Transformer models for text-based emotion detection: a review of BERT-based approaches". In: *Artificial Intelligence Review* (2021), pp. 1–41.

[2] Junyoung Chung et al. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. 2014. arXiv: `1412.3555 [cs.NE]`.

[3] Stefania Cristina. "The attention mechanism from scratch". In: *Machine Learning Mastery* 20 (2021).

[4] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[5] Gareth James et al. *An introduction to statistical learning*. Vol. 112. Springer, 2013.

[6] Vishal Kharde, Prof Sonawane, et al. "Sentiment analysis of twitter data: a survey of techniques". In: *arXiv preprint arXiv:1601.06971* (2016).

[7] Diksha Khurana et al. "Natural language processing: State of the art, current trends and challenges". In: *Multimedia tools and applications* 82.3 (2023), pp. 3713–3744.

[8] Prakash M Nadkarni, Lucila Ohno-Machado, and Wendy W Chapman. "Natural language processing: an introduction". In: *Journal of the American Medical Informatics Association* 18.5 (2011), pp. 544–551.

[9] Christopher Olah. "Understanding lstm networks". In: (2015).

[10] Lewis Tunstall, Leandro Von Werra, and Thomas Wolf. *Natural language processing with transformers*. " O'Reilly Media, Inc.", 2022.

[11] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).