Depression and Suicide Detection in Reddits

Vasilescu Andreea[a], Diaconescu Vlad-Eduard[b]

[a] *University of Bucharest, Academiei Street, no. 14, Bucharest, 077131, Romania*
[b] *University of Bucharest, Academiei Street, no. 14, Bucharest, 077131, Romania*

Abstract

This paper is about Detecting Depression and Suicide in Reddit.

We chose this topic because in recent years depression is impacting more and more people, resulting in a decreased quality of daily life.

To conduct this study we used selected posts on Kaggle from the subreddits r/SuicideWatch, r/depression and applied different models in order to conclude which one is the most effective for suicide and depression detection.

BIO (2023)

Keywords: Bio-Medical NLP, NER, AI, binary classification
https://colab.research.google.com/drive/123XhbUhOHXhao4r30S5arsn9$_zXTUh - I?usp = sharing$

1. Introduction

Biomedical natural language processing (NLP) is a specialized field that applies computational techniques to extract valuable insights from biomedical texts such as scientific articles and clinical records. It plays a crucial role in advancing biomedical research, aiding in tasks like named entity recognition, relation extraction, and text classification. Biomedical NLP has broad applications in healthcare, including evidence-based medicine, pharmacovigilance, clinical decision support, and drug discovery. It helps unlock knowledge, accelerate research, and improve patient care through the analysis of biomedical language and domain-specific terminology.

1.1. Related Studies

One study related to this topic that we found interesting was conducted in 2022. The authors identified that suicide has become a serious problem, and preventing suicide has become a very important research topic. Social media provides an ideal platform for monitoring suicidal ideation. Thir paper presents an integrated model for multidimensional information fusion. By integrating the best classification models determined by single and multiple features, different feature information is combined to better identify suicidal posts in online social media. They had several data-sets that analyzed several social media platforms, such as Reddit and Twitter. X

2. State of the ART

The following section will provide an overview of the current advencements and achievements in the Bio-Medical NLP field, as well as state of the art models, methodologies, limitations and potential applications. The purpose of this section aims to set a foundation for the conducted study that will be presented.

2.1. Supervised Learning

Supervised Learning is a Machine Learning technique that requires data to learn before it can make predictions, as well as labels for each data. This technique is used to solve classification and regression problems, as it help to predict unforeseen data.

2.2. Text Classification

Text classification represents a process of making the classification of documents based on their content, into predefined categories. Kamruzzaman (2006). This subject has been studied in a variety of fields such as machine learning, data mining, and databases and it has applications in numerous fields, including the following: medical diagnosis, organization of documents, filtering group news, target management, and so on. Aggarwal and Zhai (2012)

2.3. NLTK

NLTK (Natural Language Toolkit) is a Python library for natural language processing (NLP) tasks. It provides tools, data sets, and resources for various NLP applications, including tokenization, tagging, parsing, and machine learning. NLTK offers pre-trained models and corpora for research and experimentation. It supports tasks like part-of-speech tagging, sentiment analysis, and syntax parsing. NLTK integrates with other NLP libraries and frameworks. It is widely used in academia and industry for NLP development and analysis. NLTK's documentation and community support make it accessible to beginners and advanced practitioners.

2.4. Tokenization

Tokenization is a crucial step in NLP, breaking down text into smaller units called tokens. These tokens can be words, subwords, or characters, depending on the task. State-of-the-art techniques, like BPE or SentencePiece, handle OOV words effectively. Tokenization is especially important for large pre-trained models like BERT, where special tokens are added. Domain-specific tokenization addresses specialized vocabulary. Overall, tokenization plays a vital role in NLP, facilitating efficient text processing and enhancing language understanding for diverse applications, such as Sentiment Analysis, Machine Translation, NER or text classification.

2.5. Stemming

Stemming is a widely used technique in natural language processing (NLP) that aims to reduce words to their root or base form, called the stem. It involves removing prefixes, suffixes, and other word affixes to normalize the text. Stemming helps in reducing word variations, simplifying vocabulary, and improving computational efficiency in NLP tasks such as information retrieval, text classification, and clustering. While stemming can be effective in certain scenarios, it may result in over-stemming or generating incorrect stems due to its rule-based or heuristic nature. Integrating stemming into NLP research and applications enhances text processing and facilitates improved information extraction, retrieval, and analysis of textual data.

2.6. Lemmatization

Lemmatization is a crucial linguistic technique in natural language processing (NLP) that aims to reduce words to their base or dictionary form, called the lemma. Unlike stemming, which truncates words to their root form, lemmatization considers the word's morphological properties and applies grammatical analysis to generate accurate lemmas. By reducing words to their canonical forms, lemmatization enhances text normalization, improves information retrieval, and supports various NLP tasks such as machine translation, information extraction, and sentiment analysis. State-of-the-art lemmatization approaches often leverage linguistic resources, part-of-speech tagging, and rule-based or statistical methods to handle different word forms and languages effectively. Robust and accurate lemmatization is essential for ensuring precise analysis and understanding of text data, making it a vital component in modern NLP research and applications.

2.7. NER

Named Entity Recognition is a subtask of information extraction that aims to identify and classify named entities within text into predefined categories such as person names, location or organization.

2.8. SVM

Support Vector Machine(SVM) is a **supervised learning algorithm** that solves a Convex Optimization Problem for the classification of objects in two categories, separated by a hyper-plane that maximizes the distance from it to the closest point of each category. This model can be used for text recognition, face detection, or spam filtering.

Each object we want to classify is represented as a point in n-dimensional spaces. Each of the points proposed for classification has its unique coordinates called features.

The classification consists of drawing a **hyper-plane**, also called **decision boundary**, represented by a line in 2D or a plane in 3D, splitting the categories on one side or another.

The purpose of SVM is to find a hyper-plane that separates the categories in the best way, by maximizing the distance from the hyperplane to the closest point from each category (**margin**). **Support vectors** are the closest points to hyper-plan.

SVM is a **supervised learning algorithm** because it requires a training set represented by a set of points already labeled with the correct category, needed for finding the hyper-plane. Points in each category must be on the proper side of the decision boundary.

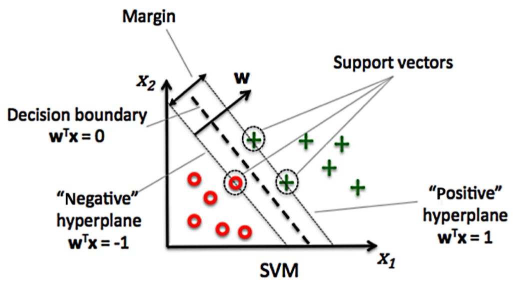


Figure 1: SVM representationAhmad (2020)

In the context of the SVM discussion, it is important to mention the Kernel Trick, a method that allows data to be mapped from 2-dimensional space (as in Support Vector Machine) to 3-dimensional space. The figure shows a graphical representation of how SVM, while the figure explains the Kernel Trick works.

2.9. XGBoost

Extreme Gradient Boosting (XGBoost) is an ensembling method, as it combines multiple methods to provide improved results. This model performs classification by taking outputs from individual trees and then combining them. It is designed to be highly flexible, efficient, and portable, implementing an algorithm of machine learning by the Gradient Boosting framework.

The xgboost algorithm is designed especially for huge and complex datasets and this model can be used for solving both **regression** and **classification** problems.

2.10. SVC

The purpose of clustering is to classify data based on a pre-defined criterion to group data. **Support Vector Clustering** is based on a Support Vector Machine. This can be achieved using methods such as:

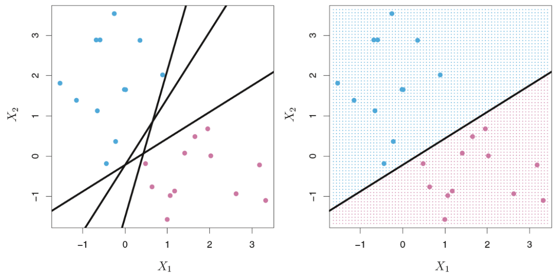


Figure 2: SVMGrace Zhang (2018)

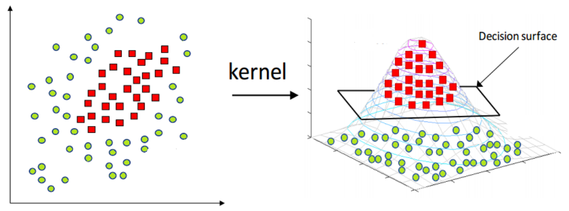


Figure 3: Kernel TrickGrace Zhang (2018)

- parametric model (only parameters are needed to predict unknown values; e.g. K-means Model, a basic model in clustering, which allows grouping data based on similar characteristics)
- grouping points based on distance, like hierarchical clustering algorithms Ben-Hur et al. (2001)

SVC transforms the points that lie in a given space into another space characteristic for them, but with a larger dimension, and then identifies a sphere in the characteristic space; In the initial space (the one before the transformation), the previously identified sphere will be transformed into a set of disjoint regions.

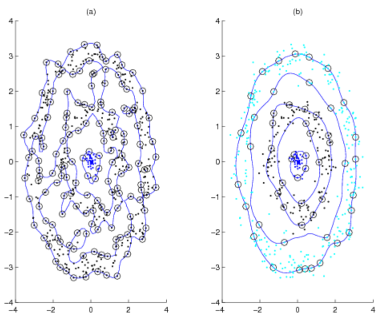


Figure 4: SVCBen-Hur et al. (2001)

2.11. *Gaussian Naive Bayes*

Gaussian Naive Bayes it is a probabilistic machine learning algorithm based on the research of Thomas Bayes, an English statistician whose name it bears. The algorithm has its practical application in solving classification problems such as spam detection, sarcasm detection, and so on.

Being a classic algorithm in machine learning, Naive Bayes has its origins in statistics and it represents a classifier family and is based on a conditional analysis that is further applied.

The **advantage** of the Naive Bayes algorithm lies in the balance between simplicity and power. Hence with the right hyperparameters, it can offer responses to the requests almost instantly.

The Naive Bayes algorithm is represented by a family of classifiers, based on conditional probability analysis. To perform the calculus of this probability, it is necessary to calculate the probability for individual event performed that composes the event.

2.12. *BERT*

BERT (Bidirectional Encoder Representations from Transformers) is a transformer-based model developed by researchers at Google. It was introduced in 2018 and has had a significant impact on natural language processing (NLP) tasks.

The key innovation of BERT is its ability to pretrain a language model using a large corpus of unlabeled text and then fine-tune it on specific NLP tasks. This pretraining process allows BERT to learn rich contextual representations of words by considering both the left and right context surrounding each word.

Unlike previous language models that only considered the left context (such as ELMo) or used autoregressive approaches (such as GPT), BERT utilizes a bidirectional approach. It employs a transformer architecture that consists of multiple layers of self-attention mechanisms and feed-forward neural networks.

2.13. *Neural Networks*

Neural Network is a method of Artificial Intelligence, which aims to process data similarly to the human brain. Artificial neurons and nodes, inspired by biological neurons, are used for this purpose.

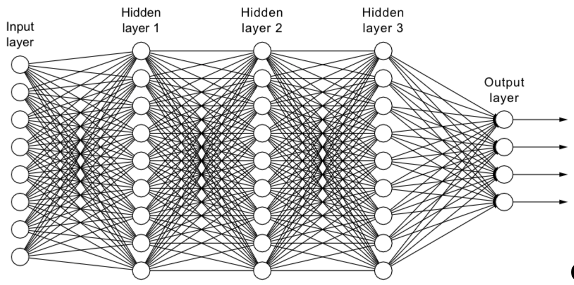
Neural Networks can be seen as oriented graphs, where each node represents a neuron. Neurons are distributed in layers. A neuron in one layer is connected to a neuron in the next layer by a weighted edge. The weight of an edge can be 0 or any positive value, but no less than 0.

A neural network has several layers:

- **input layer** is the first layer

- **output layer** represents the last layer. In the case of a binary neural network, the output layer is represented by two neurons for the probability of each class belonging.

- **hidden layers** are layering between the above-mentioned ones, and the value of neurons in both hidden and output layers are the sum of weighted edge values.



Oliver Dürr and Murina (2020)

Figure 5: Neural Network Layers

	Predicted Negative	Predicted Positive
Actual Negative	TN	FP
Actual Positive	FN	TP

- **True Positive (TP):** The number of samples that were correctly classified as positive (correctly classified instances of the positive class).
- **True Negative (TN):** The number of samples that were correctly classified as negative (correctly classified instances of the negative class).
- **False Positive (FP):** The number of samples that were incorrectly classified as positive (instances of the negative class wrongly classified as positive).
- **False Negative (FN):** The number of samples that were incorrectly classified as negative (instances of the positive class wrongly classified as negative).

2.14. Interpret a Confusion Matrix

The Accuracy gives us an overall measure of how well our model performed by calculating the percentage of correctly classified instances (TP and TN) out of total number of instances.

Precision measures how many of the predicted positive instances are actually positive. It is calculated that $\frac{TP}{TP+FP}$. A high recall indicates a low false negative rate, meaning that model has a tendancy to classify negative instances as positive.

Recall, measures how many of the actual positive instances are correctly classified as positive. It is calculated as $\frac{TP}{TP+FN}$ A high recall indicates a low false negative rate, meaning that the model has a low tendency to classify positive instances as negative.

F1 Score is the harmonic mean of precision and recall and provides a single metric that balances both measures. It is calculated as $2 * \frac{(Precision*Recall)}{(Precision+Recall)}$. The F1 score considers both false positives and false negatives and is useful when you want to find a balance between precision and recall.

3. Experiment

In the following are going to explain our work done on the research of Depression and Suicide detection in Reddits, as well as present how the experiment was conducted. This chapter contains information regarding the implementation of the coding part that stands behind this paper, what I have tried and how we get to the final version. At the end of this chapter, we are going to provide the pros and cons of each model we've used on our data-set, providing a brief explanation.

3.1. Motivation for the theme

We have chosen this subject for the projects, as depression and suicidal thoughts are impacting people all around the world. Some scientists consider depression to be the illness of the century, as nowadays even kids starting age of 5 are experiencing depression symptoms. Many people are seeking for help online, so being able to detect this illness and offer help to the ones in need will be a great achievement.

For this experiment, , we used **Python** programming language.
We have used one of the main libraries in python such as:

- numpy for numerical operation, as it is considered to be one efficent on array operations, as well as mathematical functions.
- pandas for reading the data.
- NLTK, a library specifically designed for human language data manipulation itemspacy, popular library used for NLP processing, as it provides several models trained on large corpus texts.

Google Collaborator was our choice because we were already familiar with it from previous university projects we worked on. This tool offered us the possibility to test our models easier, collaborate when writing code, review each other modifications, and work together even though we weren't on the same room. Also, one big advantage for us was the fact that we could easily store the data in Cloud, and then be able to access our work from any device.

3.2. Data

For this thesis, we used **Python** programming language.
We have used one of the main libraries in python such as:

- **numpy** - for numerical operation, as it is considered to be one efficent on array operations, as well as mathematical functions.
- **pandas** - for reading the data.
- **NLTK**, - a library specifically designed for human language data manipulation item**spacy**, - popular library used for NLP processing, as it provides several models trained on large corpus texts.

3.3. *Reading the data*

As the data set was received in a CSV format, we have chosen pandas library, as it is easy to work with, it is good for tabular data structures, and well-suited for this type of work, allowing us to manipulate and analyze text data efficiently. Another key feature that made us to choose pandas is the integration with other popular libraries used in NLP, such as NLTK and Scikit-learn.

3.4. *Data -set*

For each Reddit proposed for classification the following information was provided:

index	Text	Class

The Text contains the content of user posts, while the class is the label for each text as follows: ”non-suicide” or ”suicide”.

3.5. *Pre-processing*

We have started the pre-processing by spliiting the data into *reddit_paths*, *reddits*, and *reddit_classes*.

To remove the stop words from reddit texts, we used NLTK library and we imported stop words for English vocabulary. Then we have created a corpus with the reddits we applied filters to in order to prepare the data for analysis. We implemented a function called *cleanText*, so we can use it in the future in case we want to add more data on future work. We considered that our data was fully preoprcessed after we lowercased all the reddits, removed the white space, numbers, special charachter, email addresses, stop words (using NLTK) , links and tokenized the reddits.

3.6. *Named Entity Recognition*

Next, we used NER to identify the most used adjectives in our data set. More than that, we splitted the data into suicidal and non suicidal, as we wanted to see which are the most used adjectives for each category.

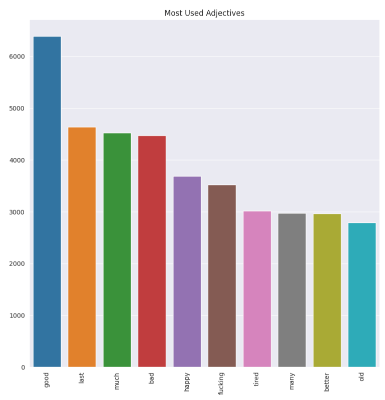


Figure 6: Most Used Adjective in Reddits

To achieve this, we have used the python code snipped that loads a specific language model from spacy.load(”en_core_web_sm”). SpaCy is a popular open source library used in Natural language processing, as it contains several language models for different languages, allowing to perform various NLP tasks such as tokenization and NER.

In order to see most common Adjectives in suicidal/ depressive reddits, we splitted the data into suicidal and non suicidal reddits based on their labels, and we used zip(cleanReddits, reddit_paths) because we didn't want to mess up the labels, so we took them in pairs.

I have plotted graphics for the most used words in all Reddits, most used in Reddits labeled as suicide, and the ones labeled as non suicide.

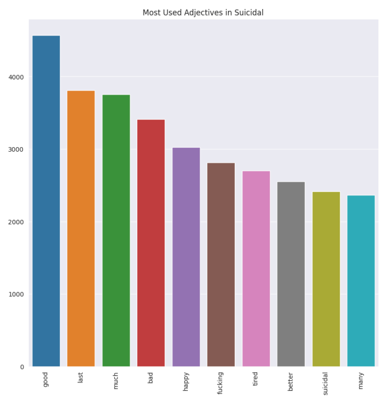


Figure 7: Most Used Adjective in Reddits labeled as Suicidal

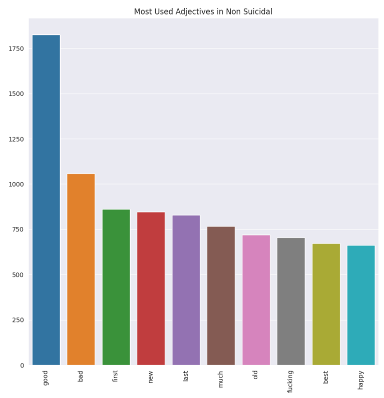


Figure 8: Most Used Adjective in Reddits labeled as non-suicidal

3.7. *Bag Of Words*

As a prerequisit to bag of words we convert all cleaned reddits to str type, and just to be sure, we checked all of them to see if after the cleaning there is any reddit that ended up as NaN, but it wasn't the case.

To implement BoW model, we used Count Vectorize, as this method make the text data to be directly used in ML and Deep Learning methods, such as text classification, which is exactly our case.

There are a total of 33153 reddit posts proposed for analysis.

The vector vocabulary as 48921 unique words.

The Matrix created has 33153 lines, and 48921 columns.

To implement this, 3 steps were followed:

1. Used the pre-processing class from Scikit-learn
2. Used the fit method, and called it on the vector in order to build an internal vocabullary. Basically, this is a dictionary with all the unique words from the text corpus (cleaned reddits).
3. Transformed the vector into an array, as now our sparse matrix will have a number of lines equal with the number of reddits, and a number of columns equal to the number of unique words.

The vocabulary size is 48921.

3.8. *Training Part*

At this point, the variables that matter are as follows:

- reddits $\rightarrow$ reddits before preprocessing $\rightarrow$ df['text']
- reddit_class$\rightarrow$ labels (bool) $\rightarrow$ df['class']
- cleanReddits $\rightarrow$ cleanText $\rightarrow$ df['cleanText']
- cleanLabels $\rightarrow$ labels after exclusion of nan reddits
- suicidal $\rightarrow$ suicidal reddits
- non_suicidal $\rightarrow$ non suicidal reddits

The first step into the training part was to split the data into test and train.

We decided to go with 25% of the data for testing, and 75% for training.

To further conduct the study, we used TensorFlow. The version 2.x is he one included in Colab.

After that, we decided that it will be relevant to shuffle the data, so we used the function train_test_split from sklearn, with the following parameters: test_size = 0.25, random_state = 2, and shuffle = True.

The shape of the data for training and testing was:

- x_train: (24864, 48921)
- x_train: (8289, 48921)

3.9. *SVM*

The first Natural Language Processing model we implemented was SVM:

We tested several coefficients to see which one is a better fit on our SVM for our data.

The conclusion is that the smaller the coefficient, the gratest the accuracy.

One issue we encountered here was ConvergenceWarning.

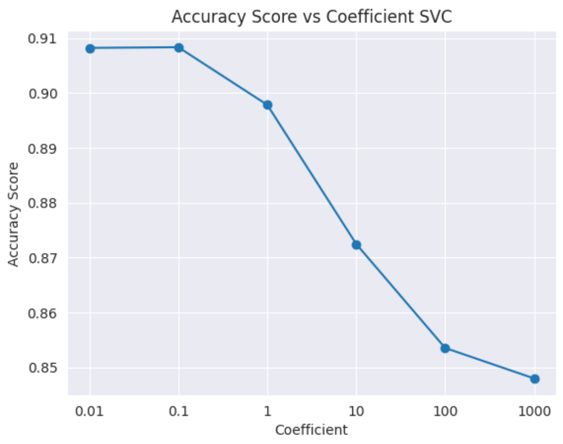


Figure 9: Accuracy SVC

This suggests that model did not converge with the default number of interations, meaning that for an optimal solution we will need to increase the number of iterations.

After we increased the maximum number of iterations by adding the parameter *max_iter* on our svm.LinearSVC function to 10000 we observed that convergence warning issue was fixed, but only for the coefficients 0.01, 0.1 and 1.

Another thing to mention is that accuracy has not changed.

Using matplotlib.pyplot and sklearn.matrix we were able to print the confusion matrix for the best accuracy when we used the regularization coefficient of 0.1.

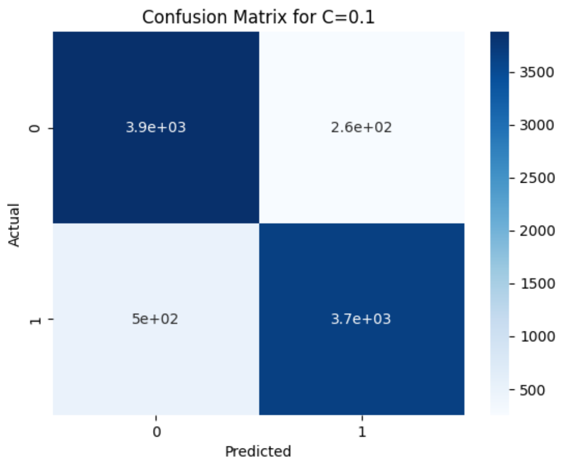


Figure 10: Accuracy SVC

3.10. *Neural Networks*

To implement Neural Networks for this paper we have used Tensorflow and the functions it offers.

Initially, we have set an embedding dimension of 16, and used tf.Keras.seqluential as we wanted to learn our neural network epoch by epoch. For the layers we had:

- tf.keras.layers.GlobalAveragePooling1D()
- tf.keras.layers.Dense(24, activation = 'relu')
- tf.keras.layers.Dense(1, activation = 'sigmoid')

As per the model.compile, we used for loss binary_crossentropy, and optimizer 'adam'.

The next step was to use to use Ragged Rensor for sentances with different lengths.

We created tokens for every word in the corpus, and then see the tokes in the word index.

After, we have stored our history of training our neural network for a model with a number of 10 epchs, and set as validation_data the testing data we split on the pre-processing part.

After 10 epochs, the accuracy was 0.86:

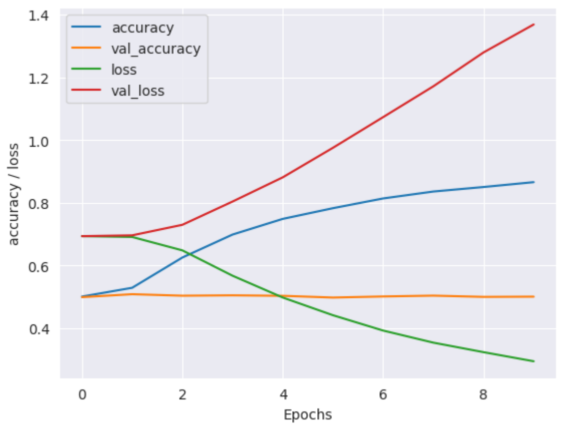


Figure 11: Training on 10 epochs

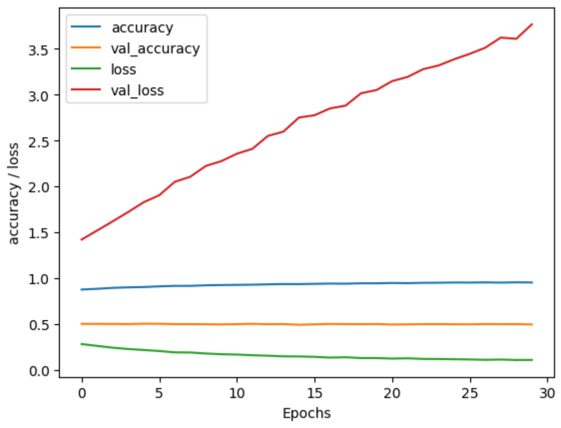


Figure 12: Training on 30 epochs

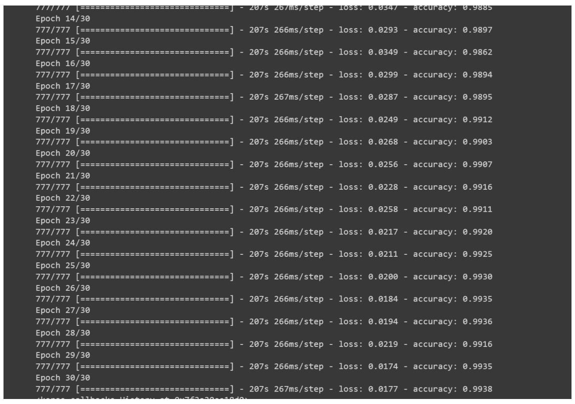
4. BERT

For the BERT model we installed transformers, and then loaded a pre-trained model and tokenizer.

The name of the pretrained model we used is BERAT-base-uncased.

For both training and testing we have used a maximum length of 128. Then, we have converted the inputs to Tensorflow data sets, by using slices with dictionaries.

To train the BERT model we have used the exact same values as we used on NN, but the results was quite unimpressive. The training was made on 30 epoch, because trying more it will take us a lot of time.



```
777/777 [==============================] - 207s 267ms/step - loss: 0.0347 - accuracy: 0.9885
Epoch 14/30
777/777 [==============================] - 207s 266ms/step - loss: 0.0293 - accuracy: 0.9897
Epoch 15/30
777/777 [==============================] - 207s 266ms/step - loss: 0.0349 - accuracy: 0.9862
Epoch 16/30
777/777 [==============================] - 207s 266ms/step - loss: 0.0299 - accuracy: 0.9894
Epoch 17/30
777/777 [==============================] - 207s 267ms/step - loss: 0.0287 - accuracy: 0.9895
Epoch 18/30
777/777 [==============================] - 207s 266ms/step - loss: 0.0249 - accuracy: 0.9912
Epoch 19/30
777/777 [==============================] - 207s 266ms/step - loss: 0.0268 - accuracy: 0.9903
Epoch 20/30
777/777 [==============================] - 207s 266ms/step - loss: 0.0256 - accuracy: 0.9907
Epoch 21/30
777/777 [==============================] - 207s 266ms/step - loss: 0.0228 - accuracy: 0.9916
Epoch 22/30
777/777 [==============================] - 207s 266ms/step - loss: 0.0258 - accuracy: 0.9911
Epoch 23/30
777/777 [==============================] - 207s 266ms/step - loss: 0.0217 - accuracy: 0.9920
Epoch 24/30
777/777 [==============================] - 207s 266ms/step - loss: 0.0211 - accuracy: 0.9925
Epoch 25/30
777/777 [==============================] - 207s 266ms/step - loss: 0.0200 - accuracy: 0.9930
Epoch 26/30
777/777 [==============================] - 207s 266ms/step - loss: 0.0184 - accuracy: 0.9935
Epoch 27/30
777/777 [==============================] - 207s 266ms/step - loss: 0.0194 - accuracy: 0.9936
Epoch 28/30
777/777 [==============================] - 207s 266ms/step - loss: 0.0219 - accuracy: 0.9916
Epoch 29/30
777/777 [==============================] - 207s 266ms/step - loss: 0.0174 - accuracy: 0.9935
Epoch 30/30
777/777 [==============================] - 207s 267ms/step - loss: 0.0177 - accuracy: 0.9938
```

Figure 13: BERT

In the end, the total accuracy for the testing data was 0.50 .

5. Conclusions

In conclusion, the best accuracy obtained during our research was achieved using Neural Networks

Model_Method	Accuracy
SVM	0.90
SVC	0.84
Gaussian Naive Baye	0.88
Neural Network	0.92
Bert	0.50

Issues encountered

1. We asked for a data set, and completed some files, but the information was approved by Mental Health Center in America, and was confidential, so we didn't received it yet.

2. At the BoW part, the processing power was to low, so we needed to updgrade the collab to premium in order to have more GPU and to be able to further conduct the study.
3. On our first try we have received the following warning message: $usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py$: 1244 : *ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.warnings.warn.* This suggests that model did not converge with the default number ofinterations, meaning that for an optimal solution we will need to increas the number of iterations. This might help.
4. For the BERT model the accuracy was very low.

6. Future Work

We would like to apply the approved data set that we initially wanted to, but we didn't received it yet.

We would like to apply search for a data set in a different study, as most studies nowadays are in English, but people need help all around the world.

References

, 2023. Bio medical nlp group.

Aggarwal, C.C., Zhai, C., 2012. A survey of text classification algorithms , 163–222.

Ahmad, I., 2020. 40 Algorithms Evry Programmer Should Know: The SVM Algorithm .

Ben-Hur, A., Horn, D., Siegelmann, H.T., Vapnik, V., 2001. Support vector clustering. Journal of machine learning research 2, 125–137.

Grace Zhang , 2018. What is the kernel trick? URL: `https://medium.com/@zxr.nju/what-is-the-kernel-trick-why-is-it-important-98a98db0961d`.

Kamruzzaman, S.M., 2006. Text classification using artificial intelligence. Journal of Electrical Engineering, The Institution of Engineers, Banglades EE 33, No. I II, 315–344.

Oliver Dürr, B.S., Murina, E., 2020. Neural network architectures URL: `https://manningbooks.medium.com/neural-network-architectures-74527000a798`.

X, Zhou W, H.B.G.L., . Linguistic analysis for identifying depression and subsequent suicidal ideation on weibo: Machine learning approaches. .