# In the name of God

Department of Computer Engineering

# Natural Language Processing

## Final Phase Report [*]

**Yeganeh Morshedzadeh**
Student Number: 96521488

Spring 2021

[*] All codes are available on my GitHub!

# Contents

# List of Figures

# List of Tables

## Abstract

In this project, we tried to use Natural Language Processing to better understand Depression and Anxiety posts.[1] The dataset is gathered from Reddit communities r/depression and r/Anxiety.

For this project, at first, we wrote a project proposal (Google Docs), and afterward, in the first phase (Google Docs), we gathered data and made some exploratory data analysis.

In the final phase, we went deeper and tried various NLP tasks, such as, computing Word2Vec, Tokenization, Parsing, and creating a language model based on our dataset.

[1] I am deeply sorry that some of the things discussed in this report may be unpleasant, disturbing, uncomfortable, and sometimes sad. In my opinion, we should pay good attention to mental health of people around us. This project is my effort to dig a little deeper into this important matter.

# Part I
# Word2Vec

**Filename:** 3_word2vec.ipynb

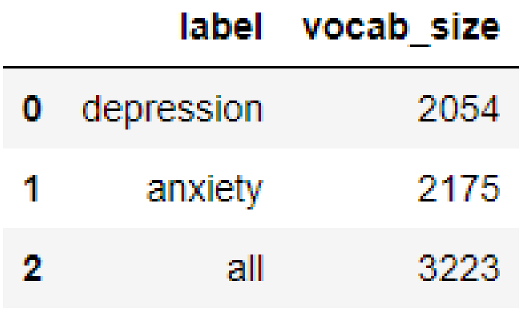
## 1 Overview

### 1.1 Code

We used Gensim implementation of word2vec to better draw the plots and show the relations between words.

For this part, we have three Word2Vec models, named dep_w2v_model, anx_w2v_model, and all_w2v_model. Moreover, with boolean parameters, load and save, the model will be saved and/or loaded in the my_word2vec function.

Table 1: Word2Vec vocabulary size



| | label | vocab_size |
|---|---|---|
| 0 | depression | 2054 |
| 1 | anxiety | 2175 |
| 2 | all | 3223 |

### 1.2 Results and Examples

#### 1.2.1 Analyze Common Words

In this part, we used t-SNE visualization. t-SNE is a non-linear dimensionality reduction algorithm that attempts to represent high-dimensional data and the underlying relationships between vectors in a lower-dimensional space.

To make the visualizations more relevant, we will look at the relationships between a query word (in **red**), its most similar words in the model (in **blue**).

In Figures 1, 2, and 3 we can see the words that are close to the word *life*.

- We can see that in depression model plot (Figure 1), the words are mostly about negative feelings, for example *useless*, *miserable*, *horrible*, and *meaningless*.

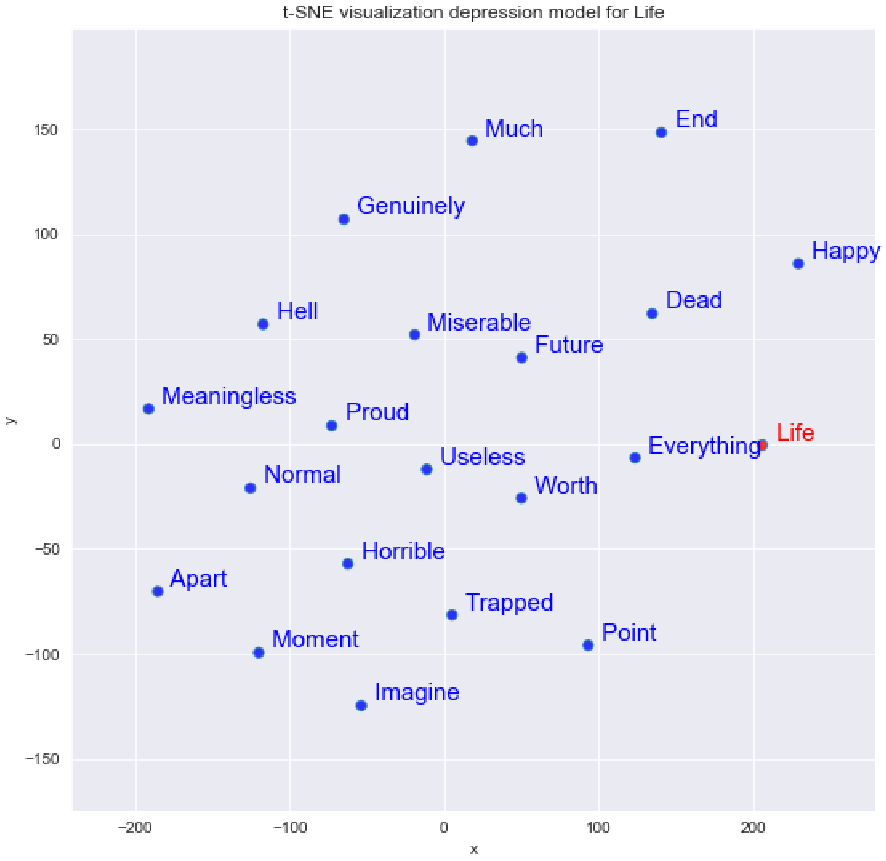


Figure 1: t-SNE visualization for life based on dep_w2v_model

- In Figure 2, the close words are mostly not negative and more about the life itself, such as *relationship* and *world*. More interestingly, we can see a couple of good vibe words, like *enjoy*, *incredibly*, *happy*, and *good*.

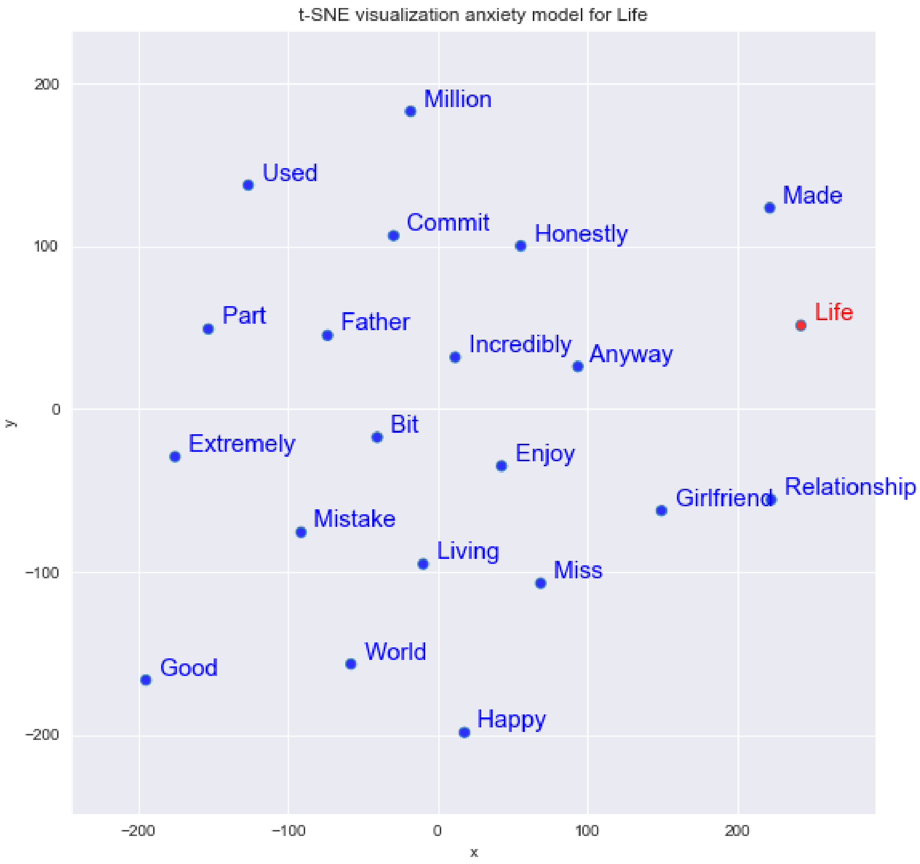


Figure 2: t-SNE visualization for life based on anx_w2v_model

- Lastly in Figure 3, the words are a mixture of the words in the previous model, but they are more general, for example *live*, *living*, *dead*, *future*, and *world*. These most close words to *life* does not contain negative meaning same as *life* itself.

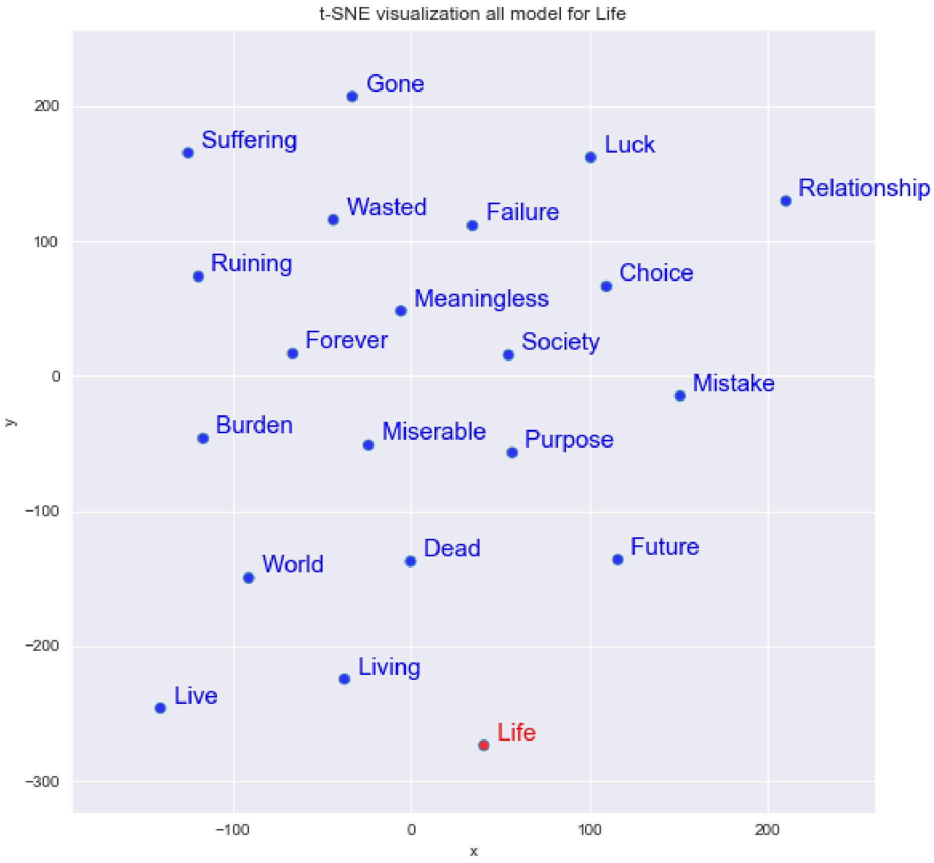
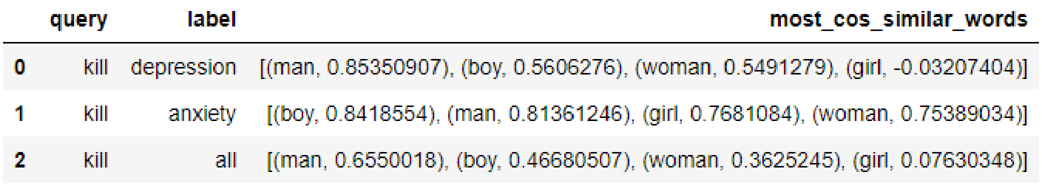


Figure 3: t-SNE visualization for life based on all_w2v_model

### 1.2.2 Find Biases

To find biases that often are based on gender, we have calculated the cosine distance of each query word from the words in target_list = [”woman”, ”man”, ”girl”, ”boy”].

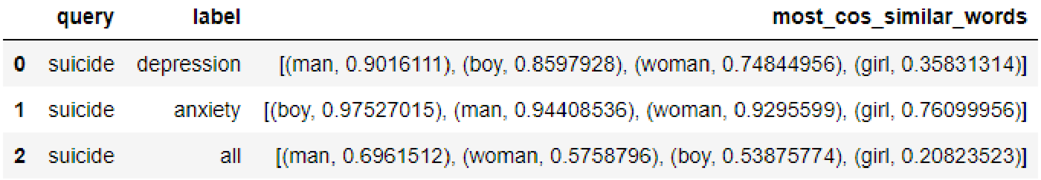
- In the Figure 4, we can see that males are much more closer to the word *kill* in all the models (ignoring the difference between boy and woman resulted from all_w2v_model).



| | query | label | most_cos_similar_words |
|---|---|---|---|
| **0** | kill | depression | [(man, 0.85350907), (boy, 0.5606276), (woman, 0.5491279), (girl, -0.03207404)] |
| **1** | kill | anxiety | [(boy, 0.8418554), (man, 0.81361246), (girl, 0.7681084), (woman, 0.75389034)] |
| **2** | kill | all | [(man, 0.6550018), (boy, 0.46680507), (woman, 0.3625245), (girl, 0.07630348)] |

Figure 4: Cosine distance between *kill* and target list words

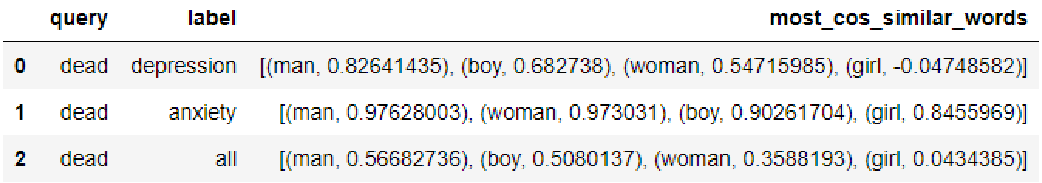
- Similarly, in the Figure 5, males are closer to the word *suicide* comparing to females in all the models.



| | query | label | most_cos_similar_words |
|---|---|---|---|
| **0** | suicide | depression | [(man, 0.9016111), (boy, 0.8597928), (woman, 0.74844956), (girl, 0.35831314)] |
| **1** | suicide | anxiety | [(boy, 0.97527015), (man, 0.94408536), (woman, 0.9295599), (girl, 0.76099956)] |
| **2** | suicide | all | [(man, 0.6961512), (woman, 0.5758796), (boy, 0.53875774), (girl, 0.20823523)] |

Figure 5: Cosine distance between *suicide* and target list words

- Like previous figures, in the Figure 6, males are closer to the word *dead* in most cases in all the models (without considering the difference between boy and woman resulted from anxiety_w2v_model).



| | query | label | most_cos_similar_words |
|---|---|---|---|
| **0** | dead | depression | [(man, 0.82641435), (boy, 0.682738), (woman, 0.54715985), (girl, -0.04748582)] |
| **1** | dead | anxiety | [(man, 0.97628003), (woman, 0.973031), (boy, 0.90261704), (girl, 0.8455969)] |
| **2** | dead | all | [(man, 0.56682736), (boy, 0.5080137), (woman, 0.3588193), (girl, 0.0434385)] |

Figure 6: Cosine distance between *dead* and target list words

## Part II
# Tokenization

**Filename:** 4_tokenization.ipynb

## 2 Overview

### 2.1 Code

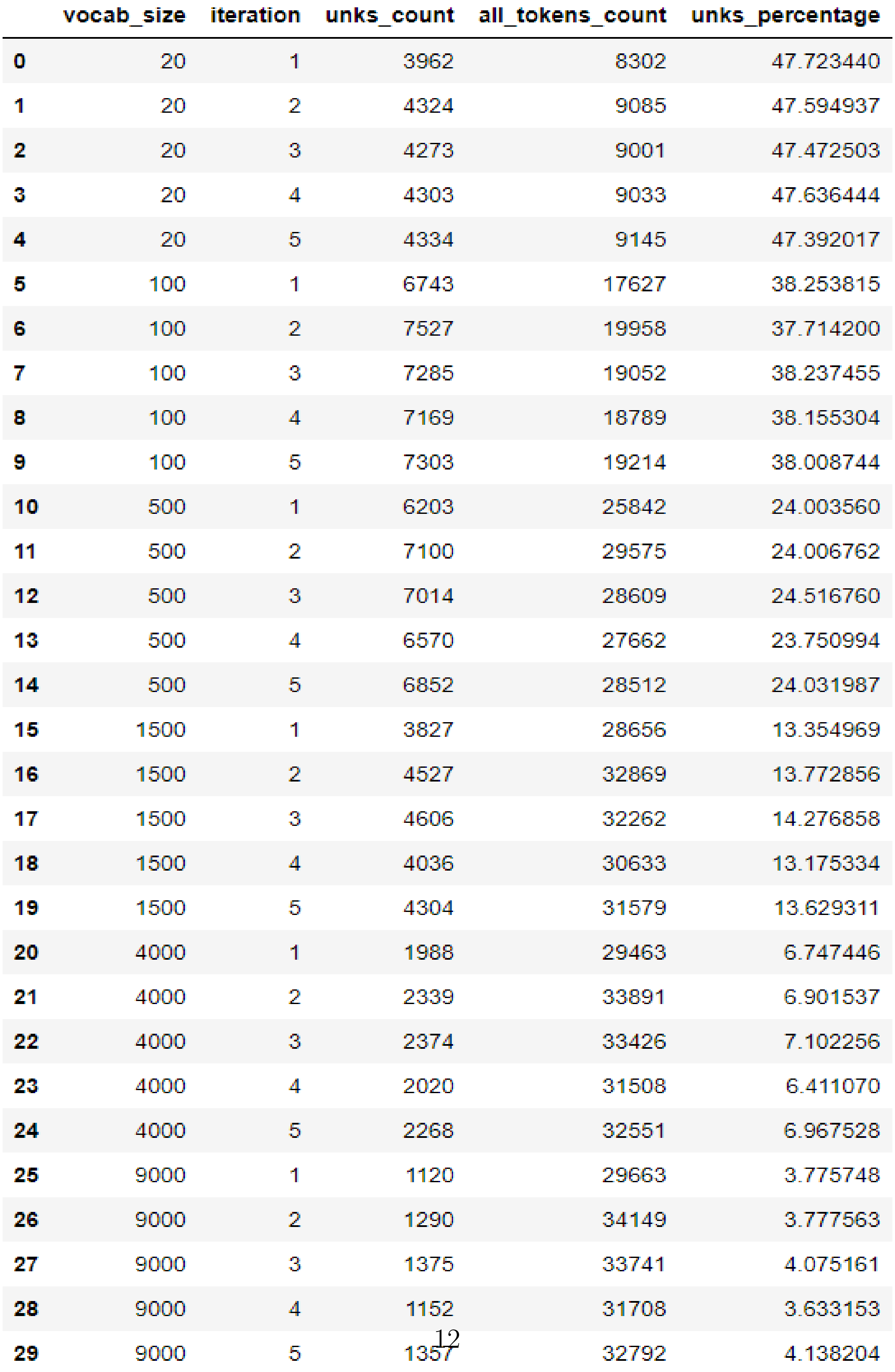
In this part, we have used KFold to split(=5) our data into train and test. Afterward, we train the SentencePiece model based on the data. Lastly, we evaluate the model by computing <UNK> on our test dataset and finally choosing as well as hard coding the best model for the tokenizer.

### 2.2 Results and Examples

Based on Table 2, in which the percentage of unk tokens to all tokens is calculated for each vocabulary size, we can conclude that by setting higher vocabulary size for the tokenizer, the number of <UNK>s decreases.

Therefore, the best tokenizer is the one with a larger vocabulary size (=9000).

Table 2: Tokenizer outcome based on different vocabulary sizes



| | vocab_size | iteration | unks_count | all_tokens_count | unks_percentage |
|---|---|---|---|---|---|
| **0** | 20 | 1 | 3962 | 8302 | 47.723440 |
| **1** | 20 | 2 | 4324 | 9085 | 47.594937 |
| **2** | 20 | 3 | 4273 | 9001 | 47.472503 |
| **3** | 20 | 4 | 4303 | 9033 | 47.636444 |
| **4** | 20 | 5 | 4334 | 9145 | 47.392017 |
| **5** | 100 | 1 | 6743 | 17627 | 38.253815 |
| **6** | 100 | 2 | 7527 | 19958 | 37.714200 |
| **7** | 100 | 3 | 7285 | 19052 | 38.237455 |
| **8** | 100 | 4 | 7169 | 18789 | 38.155304 |
| **9** | 100 | 5 | 7303 | 19214 | 38.008744 |
| **10** | 500 | 1 | 6203 | 25842 | 24.003560 |
| **11** | 500 | 2 | 7100 | 29575 | 24.006762 |
| **12** | 500 | 3 | 7014 | 28609 | 24.516760 |
| **13** | 500 | 4 | 6570 | 27662 | 23.750994 |
| **14** | 500 | 5 | 6852 | 28512 | 24.031987 |
| **15** | 1500 | 1 | 3827 | 28656 | 13.354969 |
| **16** | 1500 | 2 | 4527 | 32869 | 13.772856 |
| **17** | 1500 | 3 | 4606 | 32262 | 14.276858 |
| **18** | 1500 | 4 | 4036 | 30633 | 13.175334 |
| **19** | 1500 | 5 | 4304 | 31579 | 13.629311 |
| **20** | 4000 | 1 | 1988 | 29463 | 6.747446 |
| **21** | 4000 | 2 | 2339 | 33891 | 6.901537 |
| **22** | 4000 | 3 | 2374 | 33426 | 7.102256 |
| **23** | 4000 | 4 | 2020 | 31508 | 6.411070 |
| **24** | 4000 | 5 | 2268 | 32551 | 6.967528 |
| **25** | 9000 | 1 | 1120 | 29663 | 3.775748 |
| **26** | 9000 | 2 | 1290 | 34149 | 3.777563 |
| **27** | 9000 | 3 | 1375 | 33741 | 4.075161 |
| **28** | 9000 | 4 | 1152 | 31708 | 3.633153 |
| **29** | 9000 | 5 | 1357 | 32792 | 4.138204 |

# Part III
# Parsing

**Filename:** 8_parsing

## 3 Overview

In this part, we used Stanza, which is a Python NLP Package, and a collection of accurate and efficient tools for the linguistic analysis of many human languages. Starting from raw text to syntactic analysis and entity recognition, Stanza brings state-of-the-art NLP models to languages of your choosing.

More specifically, we used their Online Demo to create a manual CoNLL file based on our dataset (my_test.conll). Later, we can use Universal Dependencies CoNLL viewer to automatically generate a parse tree from the CoNLL file.

### 3.1 Results and Examples

We chose 10 sentences from our dataset as shown below.

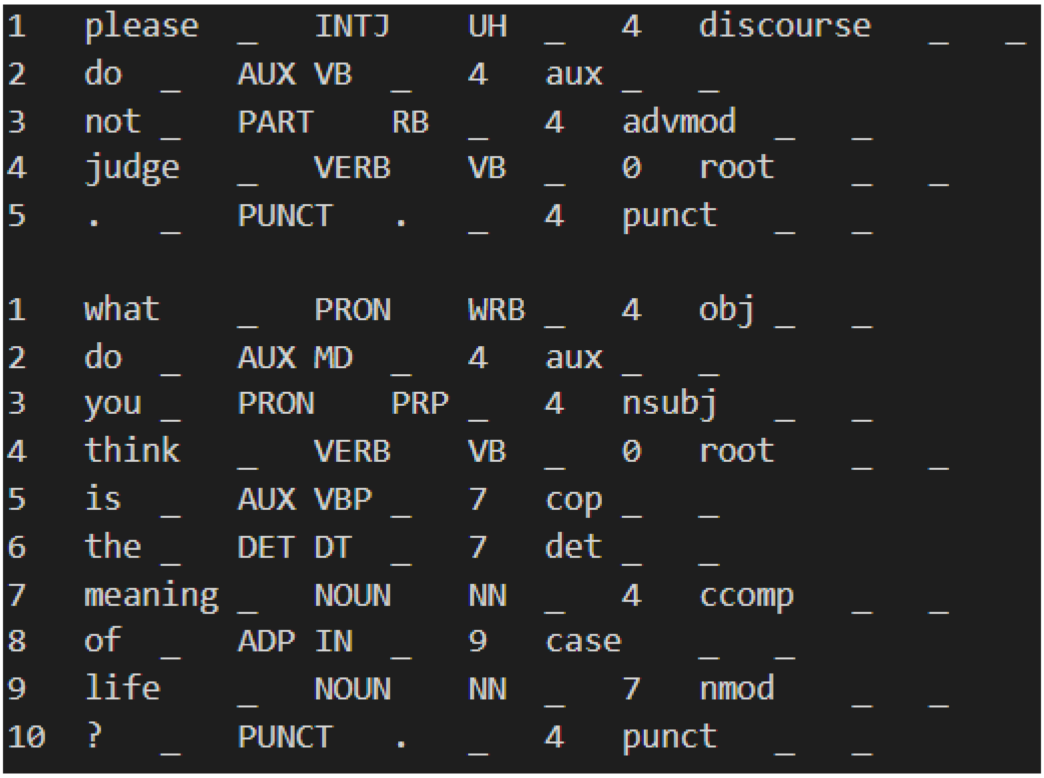
1. I never really showed any sadness when I am with someone.
2. he is a good person, and I know that.
3. how am I feeling?
4. last year I went through a huge amount of stress.
5. how can you tell the difference between an actual issue or anxiety?
6. please do not judge.
7. what do you think is the meaning of life?

8. today was a very strange day.

9. all these emotions are too much sometimes.

10. everybody is going to die.

Afterwards, we created a CoNLL file as shown in Figure 7.

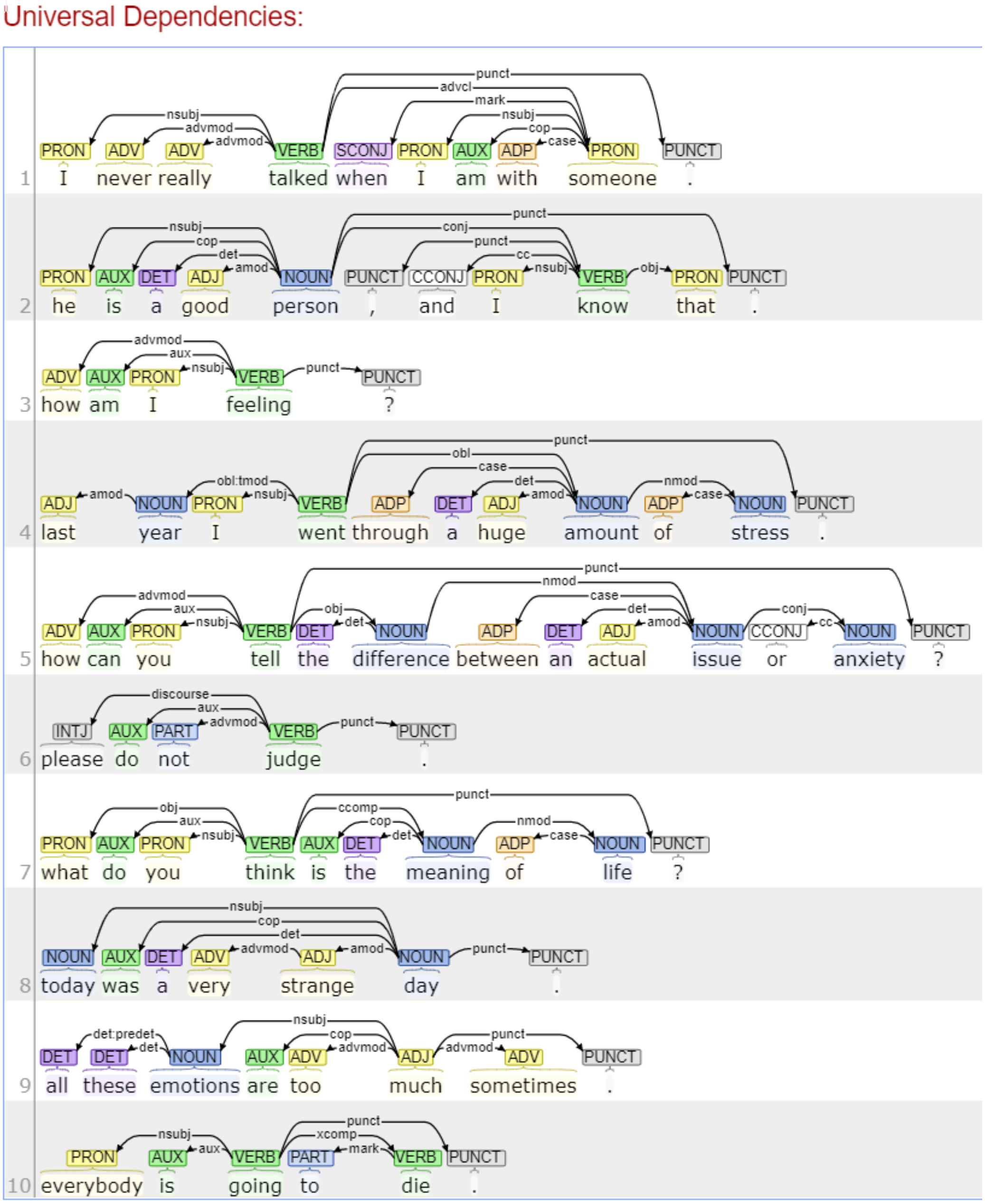


```
1   please  _   INTJ    UH  _   4   discourse   _   _
2   do  _   AUX VB  _   4   aux _   _
3   not _   PART    RB  _   4   advmod  _   _
4   judge   _   VERB    VB  _   0   root    _   _
5   .   _   PUNCT   .   _   4   punct   _   _

1   what    _   PRON    WRB _   4   obj _   _
2   do  _   AUX MD  _   4   aux _   _
3   you _   PRON    PRP _   4   nsubj   _   _
4   think   _   VERB    VB  _   0   root    _   _
5   is  _   AUX VBP _   7   cop _   _
6   the _   DET DT  _   7   det _   _
7   meaning _   NOUN    NN  _   4   ccomp   _   _
8   of  _   ADP IN  _   9   case    _   _
9   life    _   NOUN    NN  _   7   nmod    _   _
10  ?   _   PUNCT   .   _   4   punct   _   _
```

Figure 7: conll format of 6th and 7th sentences

For creating the .conll we used Stanza visualization to help us understand Universal Dependencies. (Figure 8)


Universal Dependencies:

1 I never really talked when I am with someone .

2 he is a good person , and I know that .

3 how am I feeling ?

4 last year I went through a huge amount of stress .

5 how can you tell the difference between an actual issue or anxiety ?

6 please do not judge .

7 what do you think is the meaning of life ?

8 today was a very strange day .

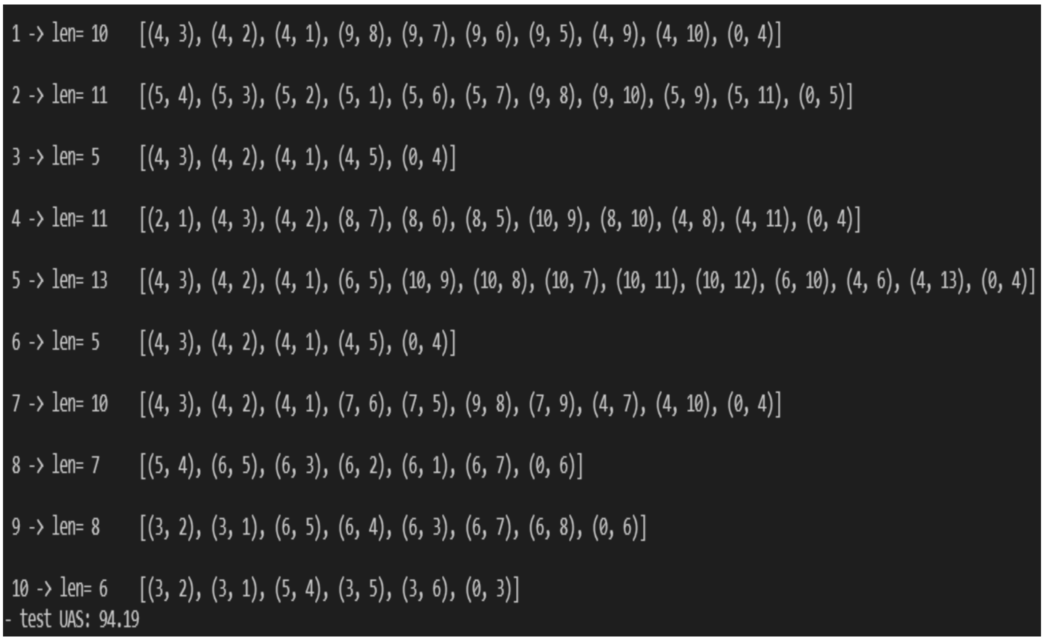
9 all these emotions are too much sometimes .

10 everybody is going to die .

PRON ADV ADV VERB SCONJ PRON AUX ADP PRON PUNCT
nsubj advmod advmod advcl mark nsubj cop case punct
PRON AUX DET ADJ NOUN PUNCT CCONJ PRON VERB PRON PUNCT
nsubj cop det amod conj punct cc nsubj obj punct
ADV AUX PRON VERB PUNCT
advmod aux nsubj punct
ADJ NOUN PRON VERB ADP DET ADJ NOUN ADP NOUN PUNCT
amod obl:tmod nsubj obl case det amod nmod case punct
ADV AUX PRON VERB DET NOUN ADP DET ADJ NOUN CCONJ NOUN PUNCT
advmod aux nsubj obj det nmod case det amod conj cc punct
INTJ AUX PART VERB PUNCT
discourse aux advmod punct
PRON AUX PRON VERB AUX DET NOUN ADP NOUN PUNCT
obj aux nsubj ccomp cop det nmod case punct
NOUN AUX DET ADV ADJ NOUN PUNCT
nsubj cop det advmod amod punct
DET DET NOUN AUX ADV ADJ ADV PUNCT
det:predet det nsubj cop advmod advmod punct
PRON AUX VERB PART VERB PUNCT
nsubj aux xcomp mark punct

Figure 8: Universal Dependencies for 10 sentences

The reported Unlabeled Attachment Score (UAS) on our test file was 94.19, and the output of the code for dependencies are show in Figure 9.



```
1 -> len= 10   [(4, 3), (4, 2), (4, 1), (9, 8), (9, 7), (9, 6), (9, 5), (4, 9), (4, 10), (0, 4)]

2 -> len= 11   [(5, 4), (5, 3), (5, 2), (5, 1), (5, 6), (5, 7), (9, 8), (9, 10), (5, 9), (5, 11), (0, 5)]

3 -> len= 5    [(4, 3), (4, 2), (4, 1), (4, 5), (0, 4)]

4 -> len= 11   [(2, 1), (4, 3), (4, 2), (8, 7), (8, 6), (8, 5), (10, 9), (8, 10), (4, 8), (4, 11), (0, 4)]

5 -> len= 13   [(4, 3), (4, 2), (4, 1), (6, 5), (10, 9), (10, 8), (10, 7), (10, 11), (10, 12), (6, 10), (4, 6), (4, 13), (0, 4)]

6 -> len= 5    [(4, 3), (4, 2), (4, 1), (4, 5), (0, 4)]

7 -> len= 10   [(4, 3), (4, 2), (4, 1), (7, 6), (7, 5), (9, 8), (7, 9), (4, 7), (4, 10), (0, 4)]

8 -> len= 7    [(5, 4), (6, 5), (6, 3), (6, 2), (6, 1), (6, 7), (0, 6)]

9 -> len= 8    [(3, 2), (3, 1), (6, 5), (6, 4), (6, 3), (6, 7), (6, 8), (0, 6)]

10 -> len= 6   [(3, 2), (3, 1), (5, 4), (3, 5), (3, 6), (0, 3)]
- test UAS: 94.19
```

Figure 9: Outputs for the my_test.conll

When comparing Figure 8 and Figure 9, we can see couple of mistakes:

- **Sentence no.2**: *he is a good person, and I know that.*

  The parent of *and*, *person*, and *,* should be *know*, and that *know* is the root. However, our model thinks*person* is the root.

- **Sentence no.5**: *how can you tell the difference between an actual issue or anxiety?*

  The parent of *or* should be *anxiety*, whereas our model thinks it is *issue.*

Therefore, it looks like that the model in some cases fails to completely understand dependencies in complex sentences.

## Part IV
# Language Model

**Filename:** 5_language-model.ipynb

## 4 Overview

In this part, we have developed a model of the text that we can then use to generate new sequences of text. It is worth noting that we have used *selftext_clean* column of our data frame to train this model.

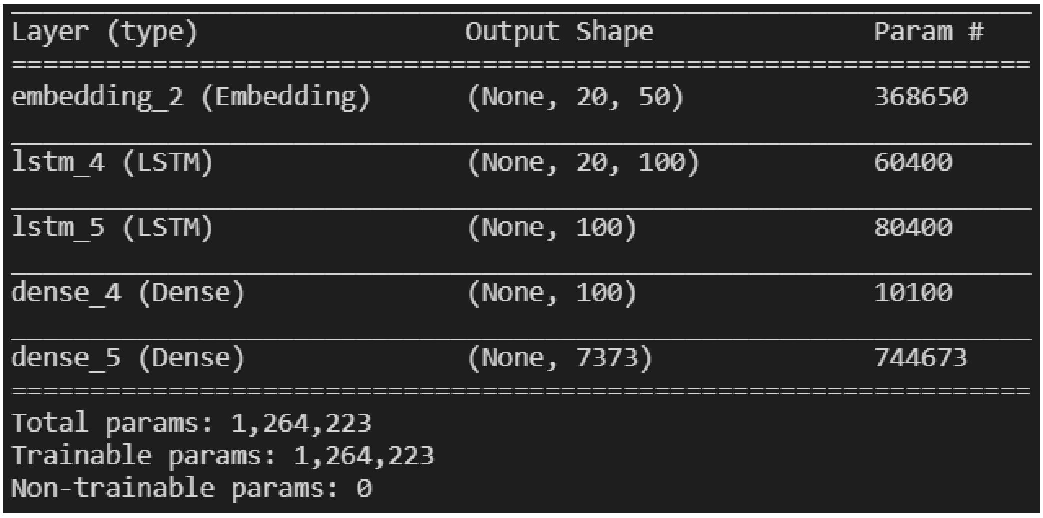
### 4.1 Code

We will start by preparing the data for modeling. We have picked a length of 50 words for the length of the input sequences, and we have attached 1 output word at the end of the sequences. Afterward, we save the long list of sequences.

For our language model, after loading our dataset, we used an Embedding Layer to learn the representation of words, and a Long Short-Term Memory (LSTM) recurrent neural network to learn to predict words based on their context. More precisely, we have used two LSTM hidden layers with 100 memory cells each. Following, we have a dense fully connected layer with 100 neurons connects to the LSTM hidden layers to interpret the features extracted from the sequence. The summary of the model is shown in Figure 10.

After the model is trained, we use generate_seq function that takes as input the model, the tokenizer, input sequence length, the seed text, and the number of words to generate.

Moreover, with boolean parameters, load_bool and save, the specific model will be saved and/or loaded.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 20, 50)            368650
_________________________________________________________________
lstm_4 (LSTM)                (None, 20, 100)           60400
_________________________________________________________________
lstm_5 (LSTM)                (None, 100)               80400
_________________________________________________________________
dense_4 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_5 (Dense)              (None, 7373)              744673
=================================================================
Total params: 1,264,223
Trainable params: 1,264,223
Non-trainable params: 0
```

Figure 10: Model architecture for simple language model

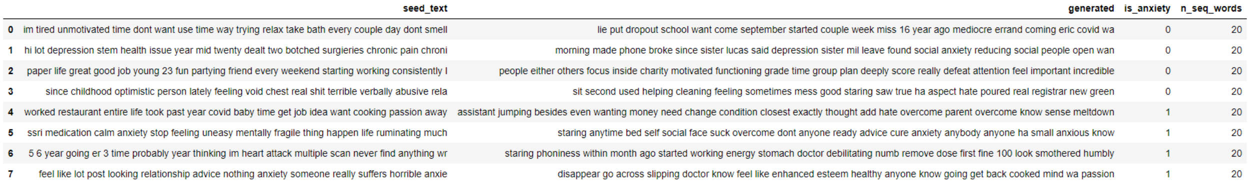
## 4.2 Results and Examples

Since we have used the *selftext_clean* column of our data and therefore the stopwords, as well as punctuation symbols, are removed, we see that our model can not follow grammatical rules and make complete sentences.
In addition, as we have not used pre-train models, the model requires more data to generate better sentences.

Overall, the model was somehow not bad at capturing the context of the dataset.

In Table 3, by feeding the seed text, obtained randomly from either depression or anxiety dataset, the model generates a sequence.

Table 3: Examples for simple language model



| | seed_text | generated | is_anxiety | n_seq_words |
|---|---|---|---|---|
| **0** | im tired unmotivated time dont want use time way trying relax take bath every couple day dont smell | lie put dropout school want come september started couple week miss 16 year ago mediocre errand coming eric covid wa | 0 | 20 |
| **1** | hi lot depression stem health issue year mid twenty dealt two botched surgieries chronic pain chroni | morning made phone broke since sister lucas said depression sister mil leave found social anxiety reducing social people open wan | 0 | 20 |
| **2** | paper life great good job young 23 fun partying friend every weekend starting working consistently I | people either others focus inside charity motivated functioning grade time group plan deeply score really defeat attention feel important incredible | 0 | 20 |
| **3** | since childhood optimistic person lately feeling void chest real shit terrible verbally abusive rela | sit second used helping cleaning feeling sometimes mess good staring saw true ha aspect hate poured real registrar new green | 0 | 20 |
| **4** | worked restaurant entire life took past year covid baby time get job idea want cooking passion away | assistant jumping besides even wanting money need change condition closest exactly thought add hate overcome parent overcome know sense meltdown | 1 | 20 |
| **5** | ssri medication calm anxiety stop feeling uneasy mentally fragile thing happen life ruminating much | staring anytime bed self social face suck overcome dont anyone ready advice cure anxiety anybody anyone ha small anxious know | 1 | 20 |
| **6** | 5 6 year going er 3 time probably year thinking im heart attack multiple scan never find anything wr | staring phoniness within month ago started working energy stomach doctor debilitating numb remove dose first fine 100 look smothered humbly | 1 | 20 |
| **7** | feel like lot post looking relationship advice nothing anxiety someone really suffers horrible anxie | disappear go across slipping doctor know feel like enhanced esteem healthy anyone know going get back cooked mind wa passion | 1 | 20 |

## Part V
# Fine-tuning

## 5 Classification

**Filename:** 6_finetune_classification.ipynb

Since some files exceeded GitHub's maximum file size (100Mb), we have uploaded them in Google drive.

- bert_classification_lm-pytorch_model.bin

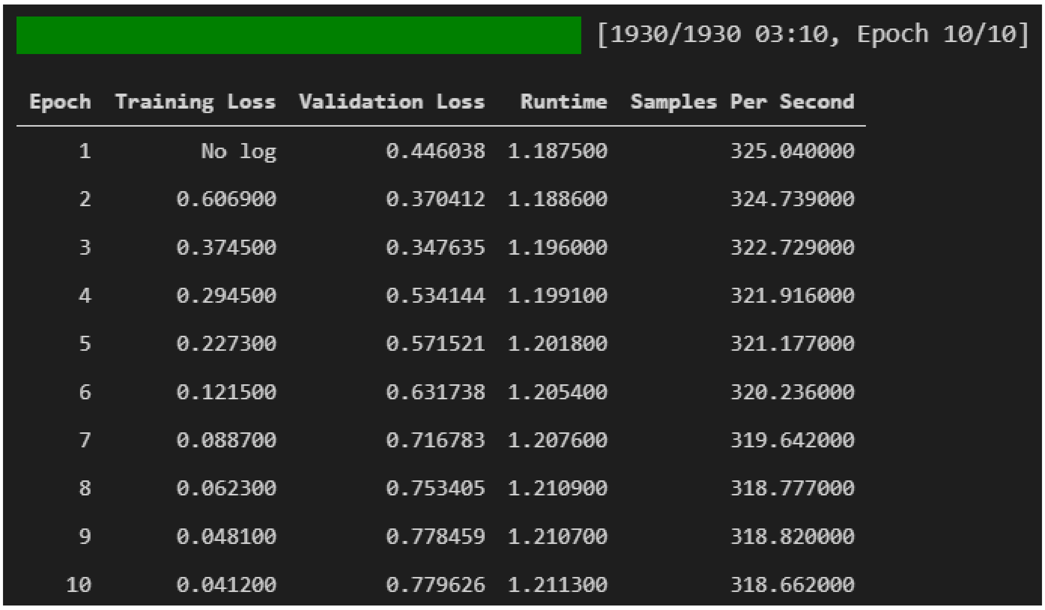
In this part, we have trained three distinct models, two of which are used to generate a model with respect to the label (depression or anxiety), and one model is used to classify the text into two categories (depression and anxiety).

### 5.1 Code

For classification, we used the pre-trained *google/bert_uncased_L-4_H-512_A-8* model. We then tokezie, truncate, and pad our dataset to a specific length (=max_length), and create a torch dataset out of them.

We used Trainer API to train our model. The Hyperparameters that are being used are set in the TrainingArguments.

Table 4: Training bert classification language model

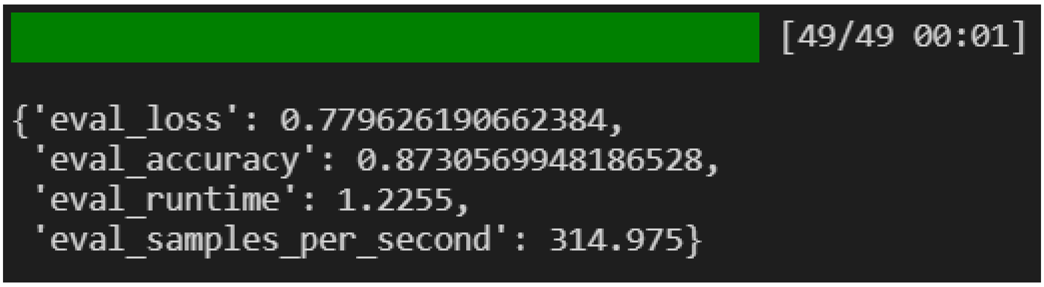


[1930/1930 03:10, Epoch 10/10]

| Epoch | Training Loss | Validation Loss | Runtime | Samples Per Second |
|---|---|---|---|---|
| 1 | No log | 0.446038 | 1.187500 | 325.040000 |
| 2 | 0.606900 | 0.370412 | 1.188600 | 324.739000 |
| 3 | 0.374500 | 0.347635 | 1.196000 | 322.729000 |
| 4 | 0.294500 | 0.534144 | 1.199100 | 321.916000 |
| 5 | 0.227300 | 0.571521 | 1.201800 | 321.177000 |
| 6 | 0.121500 | 0.631738 | 1.205400 | 320.236000 |
| 7 | 0.088700 | 0.716783 | 1.207600 | 319.642000 |
| 8 | 0.062300 | 0.753405 | 1.210900 | 318.777000 |
| 9 | 0.048100 | 0.778459 | 1.210700 | 318.820000 |
| 10 | 0.041200 | 0.779626 | 1.211300 | 318.662000 |

## 5.2 Results and Examples

As shown in Figure 11 , the accuracy of the model on the validation dataset is 87.30%.



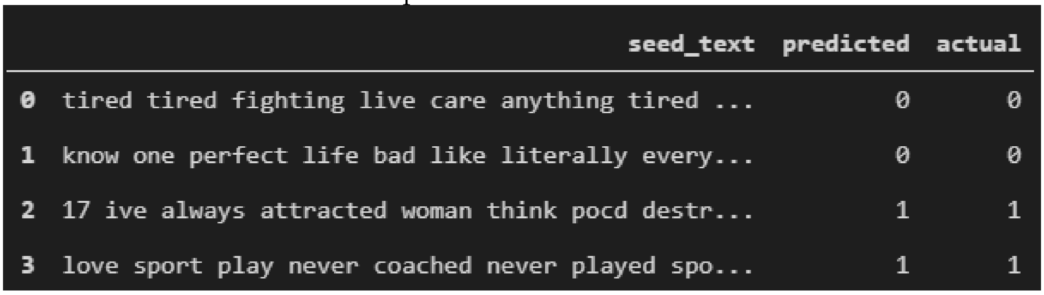
```
[49/49 00:01]

{'eval_loss': 0.779626190662384,
 'eval_accuracy': 0.8730569948186528,
 'eval_runtime': 1.2255,
 'eval_samples_per_second': 314.975}
```

Figure 11: Evaluating bert classification language model

We also selected some sentences from the dataset randomly and fed them to the model to see how it does. The results are shown in Table 5.

Table 5: Examples for bert classification model



| | seed_text | predicted | actual |
|---|---|---|---|
| 0 | tired tired fighting live care anything tired ... | 0 | 0 |
| 1 | know one perfect life bad like literally every... | 0 | 0 |
| 2 | 17 ive always attracted woman think pocd destr... | 1 | 1 |
| 3 | love sport play never coached never played spo... | 1 | 1 |

# 6 Language Model

**Filename:** 7_finetune_language-model.ipynb

Since some files exceeded GitHub's maximum file size (100Mb), we have uploaded them in Google drive.

- depression.bert_lm-pytorch_model.bin
- anxiety.bert_lm-pytorch_model.bin

In this part, we have used our Reddit dataset to fine-tune our distilgpt-2.

For this purpose, we first prepare the dataset and build a TextDataset, load the pre-trained GPT-2 model and tokenizer, initialize Trainer with TrainingArguments, and finally, train and save the model.

## 6.1 Code

The TextDataset is a custom implementation of the Pytroch Dataset class implemented by the transformers(4.2.2) library. Also, we have used the tokenizer from the distilgpt-2 model on huggingface. Afterwards, we initialize the Trainer class that provides an API for feature-complete training, and we set the Hyperparameters we are going to use in the TrainingArguments.

## 6.2 Results and Examples

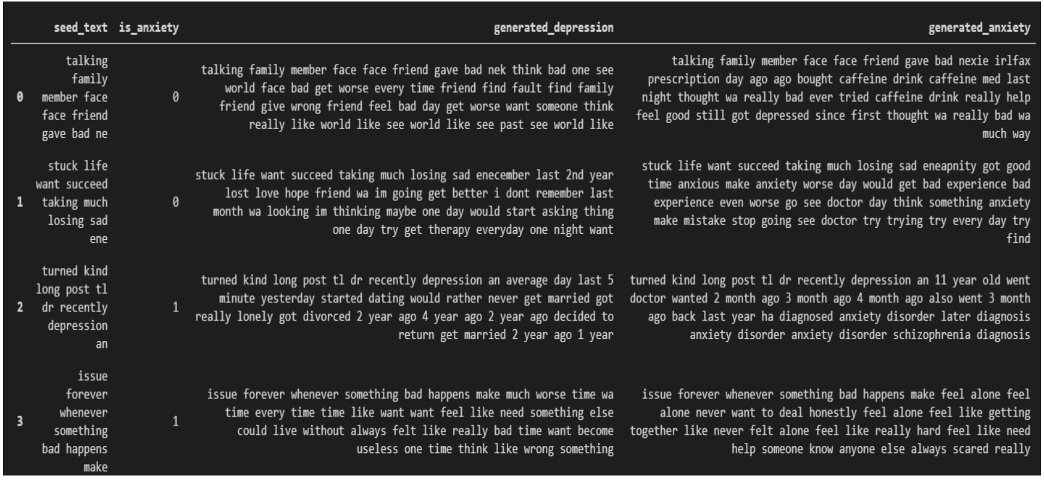
This section is quite the same as the section in the previous part (Language Model).

Similar to that part we have used the *selftext_clean* column of our data and therefore the stopwords, as well as punctuation symbols, are removed, we see that our model can not follow grammatical rules and use pronouns. The difference is that we have used a far better and more powerful model to do our language modeling task.

The model does pretty well at producing sentences similar to the context of the sentence considering the label (depression or anxiety). In other words, the quality of the generated text is high, and the sentence is much more meaningful.

Moreover, as shown in Table 6, the difference between the generated sentences for the same seed text is more distinguishable and really fits the context of each label.

Table 6: Examples for distilgpt2 language model



|   | seed_text | is_anxiety | generated_depression | generated_anxiety |
|---|---|---|---|---|
| 0 | talking family member face face friend gave bad ne | 0 | talking family member face face friend gave bad nek think bad one see world face bad get worse every time friend find fault find family friend give wrong friend feel bad day get worse want someone think really like world like see world like see past see world like | talking family member face face friend gave bad nexie irlfax prescription day ago ago bought caffeine drink caffeine med last night thought wa really bad ever tried caffeine drink really help feel good still got depressed since first thought wa really bad wa much way |
| 1 | stuck life want succeed taking much losing sad ene | 0 | stuck life want succeed taking much losing sad enecember last 2nd year lost love hope friend wa im going get better i dont remember last month wa looking im thinking maybe one day would start asking thing one day try get therapy everyday one night want | stuck life want succeed taking much losing sad eneapnity got good time anxious make anxiety worse day would get bad experience bad experience even worse go see doctor day think something anxiety make mistake stop going see doctor try trying try every day try find |
| 2 | turned kind long post tl dr recently depression an | 1 | turned kind long post tl dr recently depression an average day last 5 minute yesterday started dating would rather never get married got really lonely got divorced 2 year ago 4 year ago 2 year ago decided to return get married 2 year ago 1 year | turned kind long post tl dr recently depression an 11 year old went doctor wanted 2 month ago 3 month ago 4 month ago also went 3 month ago back last year ha diagnosed anxiety disorder later diagnosis anxiety disorder anxiety disorder schizophrenia diagnosis |
| 3 | issue forever whenever something bad happens make | 1 | issue forever whenever something bad happens make much worse time wa time every time time like want want feel like need something else could live without always felt like really bad time want become useless one time think like wrong something | issue forever whenever something bad happens make feel alone feel alone never want to deal honestly feel alone feel like getting together like never felt alone feel like really hard feel like need help someone know anyone else always scared really |

# References

[1] https://towardsdatascience.com/goodbye-world-4cc844197d51

[2] https://colab.research.google.com/github/google/sentencepiece/blob/master/python/sentencepiece_python_module_example.ipynb#scrollTo=ee9W6wGnVteW

[3] https://gmihaila.github.io/tutorial_notebooks/gpt2_finetune_classification/

[4] https://colab.research.google.com/github/philschmid/fine-tune-GPT-2/blob/master/Fine_tune_a_non_English_GPT_2_Model_with_Huggingface.ipynb

[5] https://github.com/huggingface/notebooks/blob/master/examples/language_modeling.ipynb

[6] https://www.kaggle.com/pierremegret/gensim-word2vec-tutorial

[7] https://machinelearningmastery.com/how-to-develop-a-word-level-neural-language-model-in-keras/

[8] https://huggingface.co/transformers/training.html

[9] https://www.kaggle.com/achintyatripathi/gensim-word2vec-usage-with-t-sne-plot

[10] https://colab.research.google.com/github/borisdayma/huggingtweets/blob/master/huggingtweets-demo.ipynb

[11] https://huggingface.co/transformers/custom_datasets.html

[12] https://colab.research.google.com/github/google/sentencepiece/blob/master/python/sentencepiece_python_module_example.ipynb#scrollTo=-k5KbVgiYae-

[13] https://colab.research.google.com/drive/1ZZ9vJ_nIQkgjW776lnPvwVfRHuOKvbz8