SCIPEDIA Journal of Dawning Research

Utilizing BERT for the Detection of Depression in Social Media Messages

Michael Petrizzo

Abstract

Depression is a global mental health issue generalized by a decrease in mood and satisfaction. Treatments for individuals afflicted with depressive symptoms include prescribed medications that require diagnosis to acquire. The purpose of this investigation was to accurately assist psychiatrists in diagnosis procedures to prevent both false positive and false negative conclusions by utilizing machine learning on social media messages. This was done by training a machine learning algorithm which accurately predicted and detected depressive behaviors and communications. As social media messages encompass individual's general communications among long periods of time with high consistency and frequency, I hypothesized that social media messages could be used as a method to both train an accurate and consistent machine learning model for the detection of depression. Social media dataset messages rely on self-reported diagnoses. Based on F1 accuracy normalization across machine learning HyperTuning, average accuracy indicated 97% [+/-0.5%] among a ~7600 sample dataset. Utilizing generalized sentimental analysis has shown less accurate results (~80%) but needs further research.

OPEN ACCESS

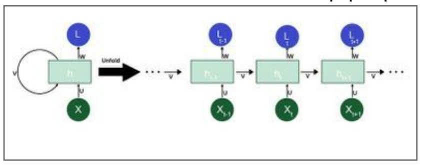
Published: 19/06/2024

Keywords:
Binary Classification
Natural Language Processing
Machine Learning
Computer Science
Sentimental Analysis

INTRODUCTION

Depression is a collective issue found globally with a rising diagnosis and frequency that negatively impacts the quality of life for those afflicted, especially in elderly populations and often requires medications (1). Vitally, diagnosis allows individuals to acquire medications, therapies, and awareness to a possibly life changing announcement from a professional. Unfortunately, diagnosis is often after extensive development of mental disease and hence detectable by professionals with a measly 50% accuracy (2).

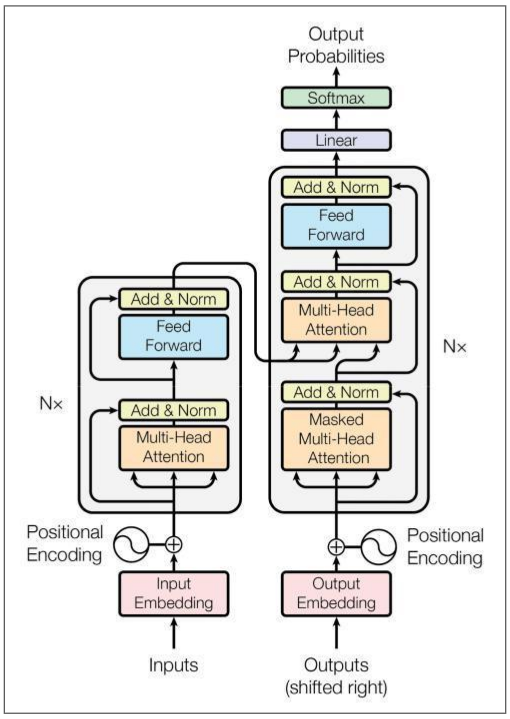
Machine learning is an oncoming development for perceived artificial intelligence which attempts to copy the brain capabilities of a human. Notably, machine learning has proven excellence in detecting themes that are typically difficult for a human to recognize such as underlying consistency between prominent labels. Prominent evidence regarding consistent accuracy in detecting specific mental illness is uncertain, as such more research is needed. Utilizing machine learning with social media text could detect depression early to allow individuals higher risk assessment and hence more time to develop proper response to this issue.



(**Figure 1**) - **RNN Model Visualization** (Adapted from Gupta). Visual representation of the recursive nature of an RNN, both in a condensed and uncondensed form.

Natural Language Processing (NLP) is the process of turning natural language into a form which is readable to a computer typically for the purpose of machine learning. This is a broad term to encompass anything from speech recognition to machine translation. NLP has preprocessing techniques for data such as padding, tokenization, lemmatization, and truncation. Padding is used to normalize data to have all values equal length done by adding filler tokens. Tokenization is the act of turning sentences into individual tokens representing each word numerically. Lemmatization is the act of turning words into their base form regarding context to both reduce complexity and introduce normalization of words, an example would be computing turned into compute (3).

Recurrent Neural Networks (RNNs) are a form of machine learning innovative for their recursive nature and self-attention (Figure 1). These models are used in NLP for speech recognition and text generation as self-attention encompasses a contextual aspect from textual data. RNN's are fundamental to many text-based processes which had the flaw of gradient decline resulting in loss of attention across large amounts of data (4). This was improved upon following Long Short-Term Memory (LSTM) models which introduces crucial information to all recursive nodes in the network, this line of information gathers importance and alters accordingly (5). An RNN is a fundamental aspect of the BERT model and shares similar core components

(4). 

(**Figure 2**) - **Transformer Model Visualization** (https://www.baeldung.com/cs/rnns-transformers-nlp) Figure representing the architecture associated with Transformers, the fundamental aspect of BERT.

Transformers added upon RNN models by utilizing the self-attention mechanism but differed by removing the recursive element (Figure 2). Furthermore, instead of individual processing of each input, the model worked utilizing parallelism of multiple tokens at once. All forms of transformers have an Encoder layer, with some having a Decoding layer. The feed-forward design introduces self-attention similarly to LSTM models, while also having a multi-head attention layer which is the core of the Transformer's parallelism (6).

BERT stands for the Bidirectional Encoder Representations (from) Transformers. BERT was created by Google's AI Language department to create a competitive RNN alternative specifically for word processing (7). BERT uses weighting to value all words equally with no representation to word index and will adjust these 'weights' to value phrases or specific keywords with higher correlation than others. This creates a model which searches for both specific words and general abstraction. BERT has been used robustly from sentiment analysis to language translation (8). This robustness is increased by the development of BERT expansions, which train the model with specific large data characteristics, such as medical or robust data (9).

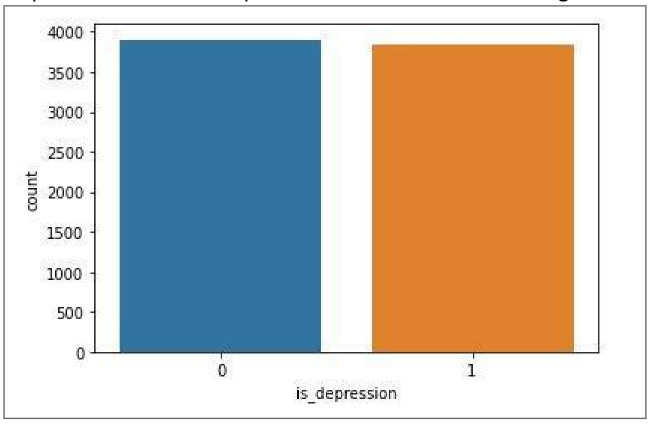
Social media posts and text are an excellent database for machine learning and more specifically for detecting mental illness with high accuracy (10). While identifying information regarding supposed diagnosis legitimacy cannot be achieved, all data being self-reported sustains authenticity to a degree that falsity will be trained out given large amounts of data (11).

Social media text processing has been utilized for mental evaluation prior, especially regarding multiclass identification (12). The purpose of this project was to create a machine learning model which could accurately detect depression in social media messages with the goal of assisting psychiatrists. This was done as former studies concluded fluctuating results among attempted machine learning models on similar marks of identifying depression from social media messages (13). Furthermore, a rising need for social workers, psychiatrists, psychologists (14) infer that those available would benefit from the use of an application in their work field. Machine learning is a rapidly growing field with constant developments warranting previous research as a basis, not an expectation. The use of BERT was dictated by availability of model, learnability, and previous use. Likewise, BERT being open source allowed for flexibility of internal hidden layers. This led to the hypothesis of utilizing BERT on social media messages to detect the presence of depression.

MATERIALS AND METHODS

General Procedure

Programming followed a linear creation originating with simple experimentation of BERT encoding and single node dense layers to grid search of multi-layer dense and dropout layers. Versions visibly progressed in both complexities, testing availability, output, and model architecture. Architecture derived from the need of complexity, meaning if the model functioned with high accuracy (95%+) with minimal hidden layers, there was no need to alter this. The data's split between Depression and Non-Depression was also confirmed (Figure 3).



(**Figure 3**) – **Dataset Depression Non-Depression Split** (Graphic via Student, TensorFlow). Figure representing the split across Depressed and Non-Depressed data samples Final Procedure

Final Procedure

Modules such as Tensorflow-Text, Keras-Tuner, TensorFlow, and Sklearn were imported to be utilized in the creation of the model. TensorFlow as the main module for the BERT model, Keras-Tuner for grid search, and Sklearn for test-train-split. Data was imported from kaggle/depression-reddit-cleaned as a TensorFlow dataframe, which had two columns; clean_text which contains the textual information of the message and is_depression which contains a binary value representing if the individual has depression. Each textual portion had 2-7 sentences of information which was gathered through multiple Subreddits without documentation from specific locations. Data was already cleaned and ready to be used by the dataset owner. The Sklearn; Test-Train-Validation-Split, of data with 60% used for training, 20% used for testing, and 20% used for validation. Ensuring the same random_state for comparison. Clean_text is normalized through the BERT encoder inside the model architecture, ensuring that each message has 128 tokens and identical tokenization style. The BERT tokenizer and encoder are from the smallest BERT Model; bert_en_uncased_L-2_H-128_A-2 and bert_uncased_preprocess_3. These preprocessors tokenize inputs to 128 tokens, perform lemmatization, and apply padding. The model was created with a dropout layer, hidden layer, dense layer, two activation functions of RELU and Sigmoid, and an output layer. It utilized an AdamW optimizer, which was chosen by a variety of training models, comparing results, and discovery of general machine learning models utilizing AdamW in textual applications. Keras-Tuner grid search was utilized to test random values for dropout percentage, hidden layer node amount, and HyperTuning of AdamW optimizer: such as decay_steps and decay_rate. The HyperTuning values were chosen on variable dependent specifications, such as the necessity of a Dropout, this also determined total training time across all models (Table 1). Most scaling are logarithmic based on the large difference between the minimum and maximum values. Keras-Tuner gathered best validation accuracy from various model performance, and the highest accuracy model is utilized.

(**Table 1**) – **Hyper Tuning Scale Figure**. Table representing the hyperparameter values scaled and ranges.

	Minimum	Maximum	Scale
Dropouts	0.2	0.5	0.05
Units	16	64	2
Init_learning_Rate	1e-5	5e-5	1e-6
Decay_steps	10000	20000	5000
Decay_rate	1e-5	1e-3	Logarithmic
Weight_Decay	0.001	0.1	Logarithmic
Beta_1	0.9	0.99	Logarithmic
Beta_2	0.999	0.99999	Logarithmic
Epsilon	1e-8	1e-6	Logarithmic

Utilization Procedure

The appendix contains direct code relating to the entirety of the Final Procedure. Inside the code contains comments on both running and testing the model. The entire document must be downloaded into a python file and was intended to be run inside of Google Collab as a .ipynb file. The topmost section contains a link to the original file in Google Collab, and is sectioned into the "Complete Model Training" which imports the dataset, and performs the entire Final Procedure, this section is not necessary to test the model and instead will create a model with identical hyperparameters to the #1 ranked model. The second section is a loading section which will download the trained model as described in this paper, hence skipping the training. The last section allows testing with individual inputs and the entirety of the testing set.

RESULTS

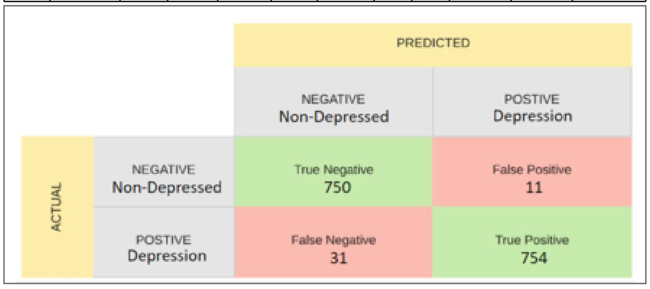
Hyper Tuning trials of models created various combinations which resulted in high accuracy (above 95%) and did not contain many similarities between models (Table 2). The final model was chosen based on a combination of high dropout and low epoch count to discourage overfitting to a maximum. Utilizing a low token count allowed for less data to still give accurate results as the model was trained on minimal tokens. Difficulty comparing outputs to another dataset's values as depressive data is often non-accessible without prior access, other forms of output do exist such as sentimental analysis but attempted output to a basis similar to sentimental (0-2) proved inconsistent results which fluctuated depending on the database. The use of a single database may introduce a selection bias as subjects are from solely Reddit, exact collection methods are not documented except binary labels which were self-reported, in aid of user anonymity. Comparing with a validation set ensures prevention of overfitting among model training and cross validation confirms accuracies with varying inputs, while using a stable randomization of data proved model consistency. The metric focused on was validation accuracy when HyperTuning, training accuracy when creating the individual model through AdamW and F1 Score created through testing results. The final model's accuracy in testing data supported the high training accuracy, which a confusion matrix can be utilized to visualize predictions and error variance between true positives, true negatives, false positives, and false negatives (Table 3). This table was generated by using direct prediction results on the testing portion of the dataset. Concluding, accurate results allow specific phrases or themes to be confirmed discriminants on detecting diagnosed individuals from non-diagnosed individuals, such "key words" could be used as a basis for further research.

(**Table 2**) - **HyperTuning of Models**. Accuracy Regarding Multi-Parameter HyperTuning with 128 Tokens, 33 individual pretrained models with labeled hyper parameters and accuracies.

Rank	Dropout	Units	init_lr	Decay Steps	Decay Rate	Weight Decay	Beta 1	Beta 2	Epsilon	Epochs	Accuracy
1	0.4	24	5.00E-05	10000	1.33E-05	2.86E-02	9.217E-01	9.99334E-01	7.77E-08	5	97.4%
2	0.2	36	4.50E-05	20000	7.05E-05	1.29E-02	9.525E-01	9.99010E-01	8.48E-08	5	97.3%
3	0.45	18	3.50E-05	15000	6.33E-05	2.07E-02	9.582E-01	9.99314E-01	4.19E-07	5	97.2%
4	0.45	58	5.00E-05	10000	6.80E-05	9.12E-02	9.727E-01	9.99960E-01	3.23E-08	3	97.0%
5	0.2	48	5.00E-05	10000	2.40E-04	1.60E-03	9.018E-01	9.99702E-01	2.32E-08	2	97.0%
6	0.45	58	5.00E-05	10000	6.80E-05	9.12E-02	9.727E-01	9.99960E-01	3.23E-08	5	97.0%
7	0.2	36	4.50E-05	20000	7.05E-05	1.29E-02	9.525E-01	9.99010E-01	8.48E-08	3	96.8%

8	0.2	48	5.00E-05	10000	2.40E-04	1.60E-03	9.018E-01	9.99702E-01	2.32E-08	3	96.8%
9	0.3	64	4.50E-05	10000	1.84E-05	2.22E-02	9.775E-01	9.99611E-01	8.05E-08	3	96.8%
10	0.4	38	3.00E-05	20000	6.17E-05	8.18E-03	9.458E-01	9.99041E-01	6.61E-08	5	96.8%
11	0.45	18	3.50E-05	15000	6.33E-05	2.07E-02	9.582E-01	9.99314E-01	4.19E-07	3	96.8%
12	0.4	24	5.00E-05	10000	1.33E-05	2.86E-02	9.217E-01	9.99334E-01	7.77E-08	3	96.8%
13	0.2	26	3.00E-05	20000	1.37E-05	4.18E-03	9.340E-01	9.99282E-01	2.30E-07	5	96.7%
14	0.4	32	5.00E-05	5000	1.46E-04	5.23E-02	9.077E-01	9.99164E-01	2.26E-07	3	96.6%
15	0.2	36	4.50E-05	20000	7.05E-05	1.29E-02	9.525E-01	9.99010E-01	8.48E-08	2	96.6%
16	0.45	52	4.50E-05	10000	6.77E-05	7.16E-02	9.033E-01	9.99531E-01	1.11E-08	3	96.6%
17	0.4	16	5.00E-05	5000	2.31E-04	4.16E-03	9.589E-01	9.99819E-01	3.31E-07	3	96.4%
18	0.4	48	4.00E-05	5000	6.94E-04	7.99E-02	9.737E-01	9.99963E-01	1.52E-08	3	3
19	0.25	46	3.50E-05	20000	2.79E-04	1.56E-03	9.245E-01	9.99746E-01	2.05E-07	3	96.4%
20	0.4	18	1.50E-05	15000	7.52E-04	4.49E-03	9.206E-01	9.99363E-01	1.42E-07	5	96.4%
21	0.3	64	4.50E-05	10000	1.84E-05	2.22E-02	9.775E-01	9.99611E-01	8.05E-08	2	96.4%
22	0.45	58	5.00E-05	10000	6.80E-05	9.12E-02	9.727E-01	9.99960E-01	3.23E-08	2	96.3%
23	0.35	48	1.50E-05	15000	2.76E-04	7.67E-03	9.845E-01	9.99431E-01	3.75E-07	5	96.3%
24	0.3	64	4.00E-05	5000	2.87E-04	1.30E-03	9.321E-01	9.99554E-01	1.35E-08	3	96.2%
25	0.35	18	3.00E-05	20000	9.19E-05	3.99E-03	9.013E-01	9.99293E-01	5.00E-08	2	96.1%
26	0.4	48	4.00E-05	5000	1.50E-04	1.30E-02	9.035E-01	9.99543E-01	1.24E-07	2	96.1%
27	0.4	48	4.00E-05	5000	1.50E-04	1.30E-02	9.035E-01	9.99543E-01	1.24E-07	3	96.1%
28	0.4	48	4.00E-05	15000	7.74E-04	3.26E-02	9.888E-01	9.99676E-01	1.07E-07	2	95.7%
29	0.2	34	3.50E-05	10000	3.66E-05	5.84E-02	9.729E-01	9.99263E-01	1.78E-08	2	95.2%
30	0.2	58	1.50E-05	15000	1.75E-05	4.57E-02	9.397E-01	9.99657E-01	3.18E-08	2	91.5%
31	0.3	48	3.00E-05	5000	1.66E-05	2.51E-02	9.312E-01	9.99230E-01	1.55E-08	1	90.8%

32	0.2	64	1.00E-05	10000	2.86E-05	4.59E-03	9.032E-01	9.99776E-01	3.46E-07	3	90.1%
33	0.2	48	3.00E-05	5000	1.07E-04	1.10E-02	9.260E-01	9.99516E-01	1.42E-08	1	89.6%
Average	0.329	42.18	3.83E-05	11818	1.60E-04	2.73E-02	9.409E-01	9.99492E-01	1.21E-07	3.09	0.958
Top 5 Avg	0.340	36.80	4.60E-05	13000	9.10E-05	3.10E-02	9.414E-01	9.99464E-01	1.27E-07	4.00	0.972
Std. Dev	0.0992	15.38	1.16E-05	5,422	2.05E-04	2.87E-02	2.882E-02	2.88635E-04	1.26E-07	1.23	0.02075
Top 5 Std. Dev	0.1294	16.53	6.52E-06	4,472	8.64E-05	3.51E-02	2.891E-02	3.70193E-04	1.65E-07	1.41	0.00185



		PREDICTED	
		NEGATIVE Non-Depressed	POSTIVE Depression
ACTUAL	NEGATIVE Non-Depressed	True Negative 750	False Positive 11
	POSTIVE Depression	False Negative 31	True Positive 754

(**Table 3**) – **Confusion Matrix Visualized** (https://abeyon.com/ai-performance-measurement-f1score/). Adapted by Student) Figure representing the highest accuracy model testing results.

DISCUSSION

Testing between higher tokenization values such as 256 to incorporate more textual information was tested, which resulted in similar accuracies with far longer training cycles with a model taking multiple hours compared to several minutes. Higher tokens theoretically incorporated higher contextual information from messages but finalized validation accuracy with HyperTuning proved similar results when compared with 128 tokenization (+/- 0.5%). Processing time increased dramatically, as the double of tokens across thousands of inputs increased complexity.

In addition, the implementation of multiple NLP processing techniques such as the removal of keywords, pre-tokenizing, and padding were also tested. The removal of keywords removed contextual information from messages which fundamentally altered the text, the advantage of BERT's ability to understand context would be nullified by this. As such the removal of keywords showed a lower (85%) accuracy with no HyperTuning. Pre-tokenization conflicted with BERT's own tokenization, while processing ran with similar efficiency to finalized models, the model lacked readability of inputs with pre-tokenized inputs. Lastly, padding is introduced in BERT's encoder and tested individually which normalized data into a form which would prevent bias from input lengths, this was a vital part of ensuring validation accuracy.

Machine learning can be a highly successful application for the detection of depression in textual information. Improvement of previous studies was achieved by a vast majority, and the potential for application among the diagnosis of individuals shows to be increasingly promising. The model architecture is not defined solely for the purpose of depression, only the dataset, which allows adoption into other Binary Classifications. The availability of code for creation of models utilizing BERT is minimal, this implementation also may serve as assistance for the creation of BERT-Hybrid models.

ACKNOWLEDGMENTS

Thank you to Ms. Beatty for reviewing and assisting with this project!

REFERENCES

[1] Hofmann SG, Sawyer AT, Witt AA, Oh D. Effect of treatments for depression on quality of life: A meta-analysis. Cogn Behav Ther. 2017;46(4):265-286. doi:10.1080/16506073.2017.1304445.

[2] Reece AG, Danforth CM. Instagram photos reveal predictive markers of depression - EPJ Data Science. doi:10.1140/epjds/s13688-017-0110-z.

[3] "What Is NLP? - Natural Language Processing Explained - AWS." What Is Natural Language Processing (NLP)?, Amazon, 2024. Available from: https://aws.amazon.com/what-is/nlp/

[4] GfG. "Introduction to Recurrent Neural Network." GeeksforGeeks, 4 Dec. 2023. Available from: https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/

[5] "Recurrent Neural Networks." Understanding LSTM Networks -- Colah's Blog, 27 Aug. 2015. Available from: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[6] Baeldung. "From RNNs to Transformers." Baeldung for Computer Science, 23 Jan. 2024. Available from: https://www.baeldung.com/cs/rnns-transformers-nlp

[7] Devlin J, Chang MW, Lee K, Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805. 2018. Available from: https://arxiv.org/abs/1810.04805

[8] Turchin A, Kuznetsov M, Gromov A, et al. Comparison of Bert implementations for natural language processing of narrative medical documents. Informatics in Medicine Unlocked. 2023;36:101139. doi:10.1016/j.imu.2022.101139.

[9] Liu Y, Ott M, Goyal N, et al. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692. 2019. Available from: https://arxiv.org/abs/1907.11692)

[10] Kim NH, Kim J, Kim J, et al. Analysis of depression in social media texts through the patient health questionnaire-9 and natural language processing. DIGITAL HEALTH. 2022;8:205520762211142. doi:10.1177/20552076221114204.

[11] "The Importance of Having Large Sample Sizes for Your

Research." Charlesworth Author Services Authors. Available from: https://www.cwauthors.com/article/importance-of-having-large-sample-sizes-for-research

[12] Zanwar S, Deshmukh P, Zanwar S, et al. Exploring Hybrid and Ensemble Models for Multiclass Prediction of Mental Health Status on Social Media. arXiv preprint arXiv:2212.09839. 2022. Available from: https://arxiv.org/abs/2212.09839

[13] Seth P, Agarwal M. Uatta-EB: Uncertainty-Aware Test-Time Augmented Ensemble of BERTS for Classifying Common Mental Illnesses on Social Media Posts. arXiv preprint arXiv:2304.04539. 2023. Available from: https://arxiv.org/abs/2304.04539

[14] "Projecting Health Workforce Supply and Demand." HRSA Health Workforce. Available from: https://bhw.hrsa.gov/data-research/projecting-health-workforce-supply-demand

APPENDIX

```
1. -*- coding: utf-8 -*-

"""Depression_Classification8.ipynb

Automatically generated by Colab.

Original file is located at

https://colab.research.google.com/drive/1Fb67GkbTcbpU9aK4bTuWK6_68E0p0Z_o

"""
!pip install -U "tensorflow-text==2.13.*"

!pip install keras-tuner

import os

import numpy as np

import pandas as pd

import shutil

import tensorflow as tf

import tensorflow_hub as hub

import tensorflow_text as text

from tensorflow import keras

from sklearn.model_selection import train_test_split

import matplotlib.pyplot as plt

import keras_tuner as kt

tf.get_logger().setLevel('ERROR')

["""https://www.kaggle.com/datasets/infamouscoder/depression-reddit-cleaned """https://www.kaggle.com/datasets/infamouscoder/depression-reddit-cleaned]

>

Dataset here

"""
url =  ['https://drive.google.com/file/d/1ayX-QruP29SWjSZLqT0-JSVyL5D7NrSu/view?usp=sharing'#Depression 'https://drive.google.com/file/d/1ayX-QruP29SWjSZLqT0-JSVyL5D7NrSu/view?usp=sharing'#Depression] default

[url='https://drive.google.com/uc?id=' url='https://drive.google.com/uc?id='] + url.split('/')[-2]

df = pd.read_csv(url)

df

X_train, X_test, y_train, y_test = train_test_split(df['clean_text'],df['is_depression'], test_size = .2, random_state = 1)

X_train, X_val, y_train, y_val =  train_test_split(X_train,y_train, test_size = .25, random_state = 1)

"""# Complete Model Training"""

tfhub_handle_encoder = ['https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-2_H-128_A-2/1'  'https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-2_H-128_A-2/1']

tfhub_handle_preprocess = ['''https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3''' '''https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3''']

print(f'BERT model selected : {tfhub_handle_encoder}')

print(f'Preprocess model auto-selected: {tfhub_handle_preprocess}')

bert_preprocess_model = hub.KerasLayer(tfhub_handle_preprocess)

bert_model = hub.KerasLayer(tfhub_handle_encoder)

def build_hypermodel(hp):

text_input = tf.keras.layers.Input(shape=(), dtype=tf.string,name='sentences')

preprocessing_layer=hub.KerasLayer(tfhub_handle_preprocess, name="preprocessing")

encoder_inputs= preprocessing_layer(text_input,training=True)

encoder = hub.KerasLayer(tfhub_handle_encoder,trainable=True, name='BERT_encoder')

outputs = encoder(encoder_inputs)

neural_net = outputs['pooled_output']

1. Hyperparameters to tune

hp_dropout = hp.Float('dropout', min_value=0.2, max_value=0.5, step=.05)

hp_units = hp.Int('units', min_value=16, max_value=64, step=2)

neural_net = tf.keras.layers.Dropout(hp_dropout)(neural_net)

neural_net = tf.keras.layers.Dense(hp_units, activation='relu')(neural_net)

neural_net = tf.keras.layers.Dense(1, activation='sigmoid', name='classifier')(neural_net)

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(

initial_learning_rate= hp.Float('init_lr', min_value=1e-5, max_value=5e-5, step=1e-6),

decay_steps=hp.Int('decay_steps', min_value=10000, max_value=20000, step=5000),
decay_rate=hp.Float('decay_rate', min_value=1e-5, max_value=1e-3, sampling='log')

)

1. AdamW

optimizer = tf.keras.optimizers.AdamW(learning_rate=lr_schedule,

weight_decay = hp.Float('weight_decay', min_value=0.001, max_value=0.1, sampling='log'),

beta_1=hp.Float('beta_1',min_value=0.9,max_value=0.99,sampling='log'),

beta_2=hp.Float('beta_2',min_value=0.999,max_value=0.99999,sampling='log'),

epsilon=hp.Float('epsilon',min_value=1e-8,max_value=1e-6,sampling='log'),

name='Adamw')

model = tf.keras.Model(text_input, neural_net) #Passed into model

model.compile(

optimizer=optimizer,

loss='binary_crossentropy',

metrics=['accuracy']

)

return model

tuner = kt.Hyperband(

build_hypermodel,

objective='val_accuracy',

max_epochs=3,

factor=2,

directory=os.path.join('drive','MyDrive','GridSearch'), # Directory to store logs and checkpoints, note this will break unless redefined to personal directory

project_name='Trial 4'

)

1. Get the best hyperparameters

best_hps = tuner.get_best_hyperparameters(num_trials=1)[0]

1. Build and compile the final model with the best hyperparameters

final_model = tuner.hypermodel.build(best_hps)

1. Train the final model with the entire training set

final_model.fit(X_train, y_train, epochs=5, validation_data=(X_val, y_val))

tf.keras.utils.plot_model(final_model)

final_model.save("DepressionClassificationModelV8")

"""# Load the model

> Training skipped

"""

1. Commented out IPython magic to ensure Python compatibility.

1. %cd /content

!mkdir DepressionClassificationModelV8

1. %cd DepressionClassificationModelV8

!mkdir variables

1. %cd variables

!wget --no-check-certificate ['https://docs.google.com/uc?export=download&id=1-Y6J1XGkQhTXT1l1FkSpmQ2zY0H3jrrs' 'https://docs.google.com/uc?export=download&id=1-Y6J1XGkQhTXT1l1FkSpmQ2zY0H3jrrs'] -O variables.index

!wget --no-check-certificate ['https://docs.google.com/uc?export=download&id=1-PM-9FQyHjmCwv6hQG2KfkNxnOZUc_4U' 'https://docs.google.com/uc?export=download&id=1-PM-9FQyHjmCwv6hQG2KfkNxnOZUc_4U'] -O variables.data-00000-of-00001

1. %cd /content

1. %cd DepressionClassificationModelV8

!mkdir assets

1. %cd assets

!wget --no-check-certificate ['https://docs.google.com/uc?export=download&id=1-Vq8Br4gtbiQaOTNaO2G2LE8cyv1DM0g' 'https://docs.google.com/uc?export=download&id=1-Vq8Br4gtbiQaOTNaO2G2LE8cyv1DM0g'] -O vocab.txt

1. %cd /content

1. %cd DepressionClassificationModelV8

!wget --no-check-certificate ['https://docs.google.com/uc?export=download&id=1-P4gGGS-hzxZFHIm5LCcGBJj2aHsiGuX' 'https://docs.google.com/uc?export=download&id=1-P4gGGS-hzxZFHIm5LCcGBJj2aHsiGuX'] -O saved_model.pb

!wget --no-check-certificate ['https://docs.google.com/uc?export=download&id=1-Jlz2RaFcrSR-6gAjCRvqvqpyjTTneg3' 'https://docs.google.com/uc?export=download&id=1-Jlz2RaFcrSR-6gAjCRvqvqpyjTTneg3'] -O keras_metadata.pb

!wget --no-check-certificate ['https://docs.google.com/uc?export=download&id=1-DDZEh_iH3sXEFhD592d6oaZO98fPmCb' 'https://docs.google.com/uc?export=download&id=1-DDZEh_iH3sXEFhD592d6oaZO98fPmCb'] -O fingerprint.pb

fm = tf.keras.models.load_model("/content/DepressionClassificationModelV8")

fm.summary()

"""# Test Model"""

text = np.array([input("Enter an input")])

prediction = final_model.predict(np.array(text))

print(prediction)

loss, accuracy = final_model.evaluate(x=X_test, y=y_test)

print(f'Loss: {loss}')
print(f'Accuracy: {accuracy}')
```