

Configuration Manual

MSc Research Project
Data Analytics

Yoshitha Ganji
Student ID: 23102268

School of Computing
National College of Ireland

Supervisor: Abubakr Siddig

National College of Ireland
MSc Project Submission Sheet
School of Computing



Student Name:Yoshitha Ganji.....
Student ID:23102268.....
Programme: Ms Data Analytics **Year:** ...2023-2024
Module: Research paper
Lecturer:Abubhakar Siddig
Submission Due Date:12-08-2024.....
Project Title: A Hybrid Approach to NLP-Based Depression Detection: Integrating BERT and Word2Vec.....
Word Count:1934..... **Page Count:**08.....

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:Yoshitha Ganji.....
Date: ...10-08-2024.....

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	<input type="checkbox"/>
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Yoshitha Ganji
Student ID:23102268

1.Introduction

This guide offers a structured approach for researchers aiming to analyze social media text for depression detection through Natural Language Processing (NLP). Central to this endeavor is the utilization of BERT for its deep semantic understanding and Word2Vec for its adeptness at capturing context within text(Yadav and Vishwakarma, 2020) . By following this manual, researchers will gain insights into setting up their project environment and leveraging these advanced NLP techniques to effectively process and analyze text data, unlocking valuable insights into mental health trends on social media platforms.(Asghar et al., 2018)

2. System Specification

To ensure the project runs smoothly, proper hardware and software environments are essential.

2.1 Hardware Requirements

- **OS:** Windows 11 Pro 64-bit
- **Processor:** 13th Gen Intel(R) Core(TM) i5-1335U 1.30 GHz. A modern multi-core processor is crucial for computational tasks involving large datasets and complex algorithms.
- **RAM:** 16GB
- **Storage:** 1 TB SSD
- **Graphics:** Intel(R)Iris(R)Xe Graphics

For network connectivity, an Ethernet connection facilitates robust and fast data transfer, supporting remote communications effectively.

2.2 Software and Tools

For the software setup of the project, the chosen operating system is Windows 11, which provides a stable and contemporary platform compatible with all necessary development tools and libraries. Python, the programming language of choice, I used a version 3.12.4 to ensure access to the latest features and library support. Some of the key Python libraries used in this project are Pandas and NumPy for data manipulation, Matplotlib and Seaborn for data visualization, TensorFlow and Keras for building and training machine learning models, and NLTK, Gensim, and Transformers for natural language processing tasks. Jupyter Notebook and Visual Studio Code provide flexible coding experiences from exploratory data analysis inside notebooks to script development and debugging inside an Integrated Development Environment. Visual Studio code is used for teamwork and coordinating the computation via

the cloud. Kaggle serves as a primary source for obtaining relevant datasets, featuring a vast collection of resources for various data science and machine learning projects.

3. Software Installation Instructions

3.1 Development Environment

Visual Studio Code:

Visual Studio Code is a versatile IDE that not only supports Python development but much more. Its rich ecosystem of extensions makes it a favourite for Python projects.

- **Download and Installation:** Visit the [official Visual Studio Code website](#) download the OS installer, and install according to the OS installer. Open Visual Studio Code upon the completion of the installation.
- **Setting up Python Support:** Open VS Code and go to the Extensions view on the left sidebar represented as a square icon Search for "Python" extension by Microsoft and click "Install." It includes advanced Python language support, IntelliSense, linting, debugging, and testing. Now install the extension "Jupyter" by Microsoft. This will allow support for Jupyter Notebooks in VS Code.
- **Usage:** I can create or open Python scripts and Jupyter notebooks right within VS Code with the Python and Jupyter extensions.directly in VS Code. The IDE provides tools for code writing, editing, running, and debugging, along with convenient features for version control and extension management.

3.2 Libraries Installed

- **pandas:** Enables operation like `pd.read_csv()` for dataset loading, DataFrame operations for data exploration and manipulation, and data structuring for machine learning workflows.
- **numpy:** Facilitates array-based computation
- **tensorflow:** Implements algorithms and models, from constructing neural network layers (`tf.keras.layers`) to compiling (`model.compile`) and training models (`model.fit`).
- **keras:** Used for concise model definition and training, simplifying complex neural network creation with functions like `Sequential()` for model stacking and `Dense()` for fully connected layers.
- **scikit-learn:** Facilitates model selection (`train_test_split`), preprocessing (`StandardScaler`), model instantiation and training (`RandomForestClassifier().fit()`), and evaluation (`accuracy_score()`).
- **nlTK** (Natural Language Toolkit): It's essential for text preprocessing tasks such as tokenization, stemming, and stopwords removal. Tools for text analysis, including text tokenization (`word_tokenize`) and stopwords filtering.
- **transformers:** Facilitates the implementation of pre-trained models (`BertModel`) to encode text data into dense vectors encapsulating linguistic patterns.
- **matplotlib and seaborn:** Enable plotting of data distributions, trends, and model evaluation metrics through various visual formats like histograms, scatter plots, and heatmaps.

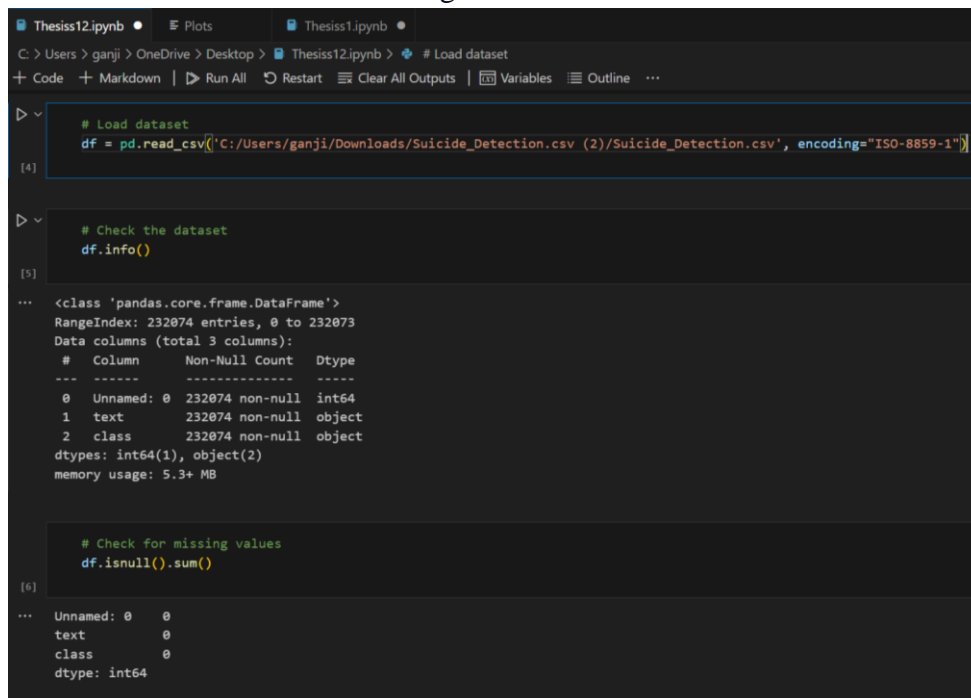
- **spaCy:** while NLTK and Gensim are very effective, spaCy provides additional functionalities for named entity recognition, dependency parsing, and part-of-speech tagging.

4.Data Source:

Primary Data Acquisition: Data is primarily obtained from Kaggle, chosen for its extensive repository of datasets pertinent to depression detection on social media. It's imperative to consider the relevancy, currency, and quality of dataset annotations during the selection process. Firstly, the search of the databases is done through the Kaggle website using relevant keywords to the project requirements. The data is then transferred into the working environment of the project which could be Visual Studio code and initially analysed and cleansed using the Pandas.

Inspect Data:

Loaded my dataset from a specified path, adjusting the path to where your dataset is stored. Employed methods like `.info()` and `.isnull().sum()` to inspect the dataset for understanding structure and identifying missing values. Below Fig 1: Shows how the data set is loaded in a dataframe and checked for missing values.



```

# Load dataset
df = pd.read_csv('C:/Users/ganji/Downloads/Suicide_Detection.csv (2)/Suicide_Detection.csv', encoding="ISO-8859-1")

# Check the dataset
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 232074 entries, 0 to 232073
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  --
 0   Unnamed: 0    232074 non-null  int64
 1   text          232074 non-null  object
 2   class         232074 non-null  object
dtypes: int64(1), object(2)
memory usage: 5.3+ MB

# Check for missing values
df.isnull().sum()

Unnamed: 0    0
text          0
class         0
dtype: int64

```

Fig 1 Code snippet of loading dataset and checking the missing values.

5. Preprocessing

5.1 Text Cleaning and Normalization

- **Lowercasing:** Converted text data to lowercase for uniformity by using `text_hammer`
- **Removing Noise:** Strip out URLs, hashtags, mentions, and special characters.
- **Tokenization:** Split texts into individual words or tokens by using `nlkt`.
- **Stop Words Removal:** Filter out common words that offer little value to the analysis.

- Stemming/Lemmatization (optional): Reduce words to their base or root form.(Shrivastava, Soni and Rasool, 2020).

5.2 Combine Original and Preprocessed Texts:

Used pandas DataFrame to combine the original texts ('original_text'), their preprocessed counterparts ('preprocessed_text'), and the text classifications or labels ('class').Fig 2. Displays the combined original and pre-processed text for further modelling.

```
> # Display the resulting DataFrame
print(df_preprocess[['original_text', 'preprocessed_text', 'class']].head())
```

	original_text	preprocessed_text	class
0	ex wife threatening suiciderecently i left my ...	wife threatening suiciderecently left wife goo...	suicide
1	am i weird i dont get affected by compliments ...	weird dont affected compliment coming someone ...	non-suicide
2	finally 2020 is almost over so i can never hea...	finally 2020 almost never hear 2020 year ever ...	non-suicide
3	i need helpjust help me im crying so hard	need helpjust help cry hard	suicide
4	iam so losthello my name is adam 16 and iave b...	losthello name adam iave struggling year afrai...	suicide

Fig 2. Displays the combined original and pre-processed text.

5.3 Sequence Padding and Visualisation of Text Data:

Sequences are standardized to a fixed length (150 tokens) using pad_sequences, ensuring uniform input sizes for model training.

Two primary visualization strategies: plotting the distribution of token lengths to understand text variability and generating word frequency graphs (before and after removing stopwords) to identify predominant words.Below Fig 3 and 4 is a bar graph of showing word frequency Before and After Removing Stopwords.

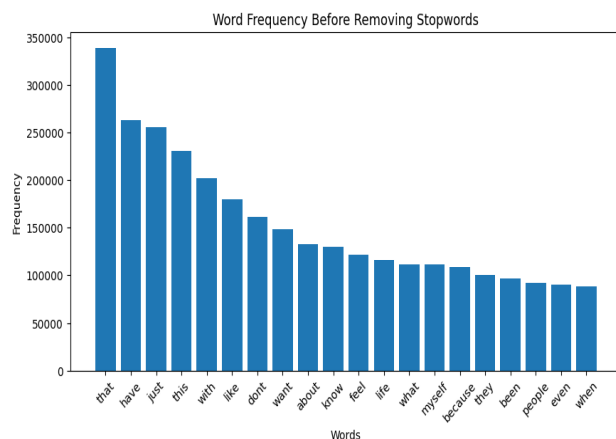


Fig 3 word frequency Before Removing Stopwords.

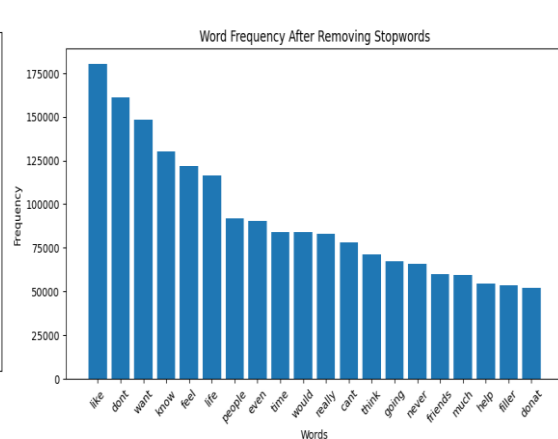


Fig4.wordfrequency After Removing Stopwords.

5.4 BERT and Word2Vec Embeddings:

Embedding Extraction:

The BERT model generates embeddings for each padded token sequence, capturing contextual information. Extract the [CLS] token's embedding from each sequence as it represents the aggregated information for classification purposes. Fig 5 gives the shape of BERT embeddings.

```
[59] print("Shape of embeddings:", embeddings.shape)
... Shape of embeddings: (232074, 768)
```

Fig 5 shape of BERT embeddings.

Word2Vec Embeddings:

Using SpaCy's 'en_core_web_lg' model, extract Word2Vec embeddings by averaging the vectors of all tokens in a text. This process supplements your feature set with embeddings derived from a different methodology, offering a broader semantic understanding. Fig 6 gives us shape of Word2Vec embeddings.

```
[97] print("Word2Vec embeddings shape:", word2vec_embeddings.shape)
... Word2Vec embeddings shape: (232074, 300)
```

Fig 6 shape of Word2Vec embeddings.

Combined BERT and Word2Vec embeddings are used for the creating the Hybrid Embeddings which can be used into training and Testing sets.

6. Model Implementation

BERT: Utilize TensorFlow's Sequential model to stack layers. Begin with a dense layer having 512 units and 'relu' activation, appropriate for the size of BERT embeddings (768 dimensions) (Devlin et al., 2019). Incorporate dropout layers with a rate of 0.5 to prevent overfitting. Compile the model with 'adam' optimizer and 'binary_crossentropy' loss, indicating a binary classification task. Monitor 'accuracy' as the performance metric. Fit the model on the BERT embeddings (X_train) and the corresponding labels (y_train_encoded), processed through a LabelEncoder and converted to float32. Set aside 20% of the data as a validation set during training.

Word2vec: Similar to the BERT model, but the input shape adjusts to the dimensionality of the Word2Vec embeddings. (Mikolov et al., 2013) Train this model on the Word2Vec-derived features (word2vec_X_train) with appropriately encoded labels.

Hybrid Model: Concatenate BERT and Word2Vec embeddings to form a comprehensive feature set, merging the strengths of both embeddings. Model Framework: Assemble a neural network with a configuration analogous to the individual BERT and Word2Vec models,

adjusting the input dimension to accommodate the combined feature set. give all this simple bullet point.

Additional Models:

Convolutional Neural Network (CNN): Adopts Conv1D layers suited for temporal data like text, with additional Dropout and dense layers for classification purposes.

Long Short-Term Memory (LSTM) and Bidirectional LSTM (Bi-LSTM) Models: Incorporate LSTM layers for capturing long-term dependencies in text data, with Bi-LSTM offering enhanced context awareness by processing data in both directions.

Hyper Parameter Tuning : The code employs GridSearchCV for an exhaustive evaluation of all specified hyperparameter combinations, including batch_size, learning_rate, and epochs, to meticulously identify the optimal settings that enhance the neural network model's performance.

Efficient Optimization with RandomizedSearchCV: By using RandomizedSearchCV, the code introduces a probabilistic approach, sampling a subset of the hyperparameter space across specified distributions. This method significantly reduces computational demand while effectively exploring potential hyperparameter configurations.

7.Evaluation

Model Training: Each model is trained on the corresponding embeddings (BERT, Word2Vec, or combined embeddings for the Hybrid model), utilizing encoded labels. Validation splits further support model tuning.

Performance Metrics: Post-training evaluation utilizes accuracy, precision, recall, and F1-score to quantify model effectiveness.

Accuracy Evaluation: Post-training, evaluate each model on the test set (X_test, y_test_encoded) to ascertain the accuracy and loss, offering insights into model performance. Fig 7 Give us the model accuracy for all the models performed.

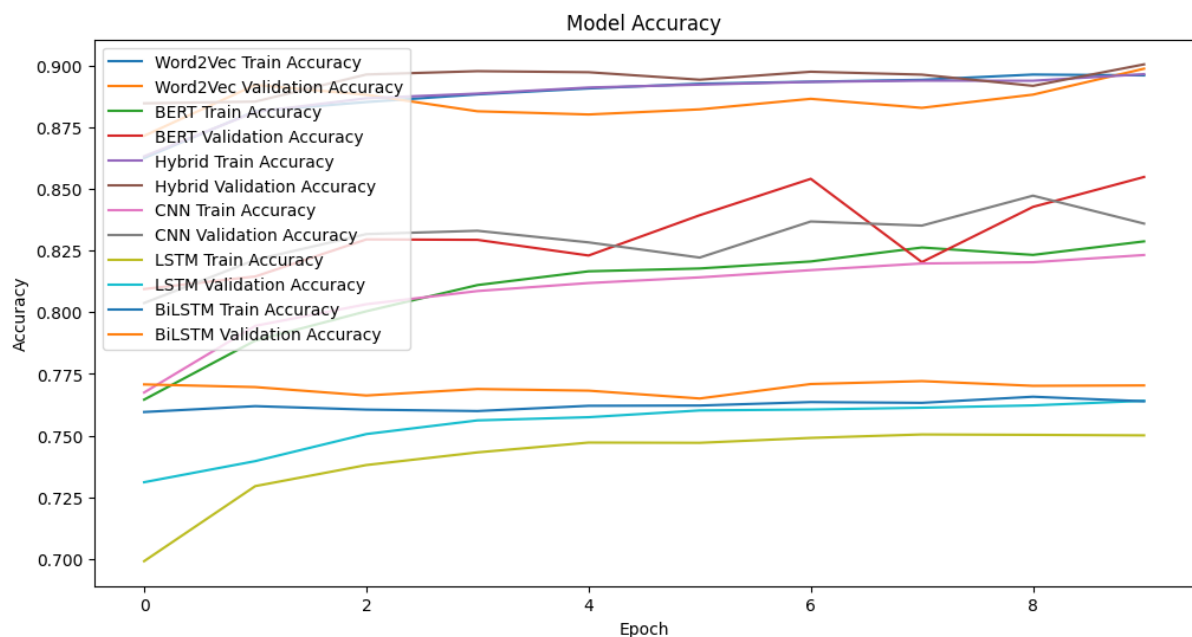


Fig 7: Model Accuracy for the models.

Confusion Matrix:

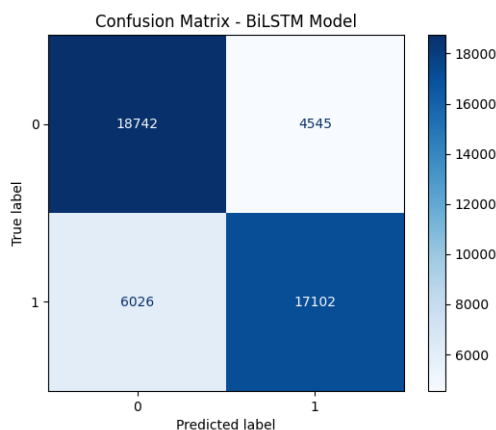


Figure 8: Confusion Matrix-BiLSTM model

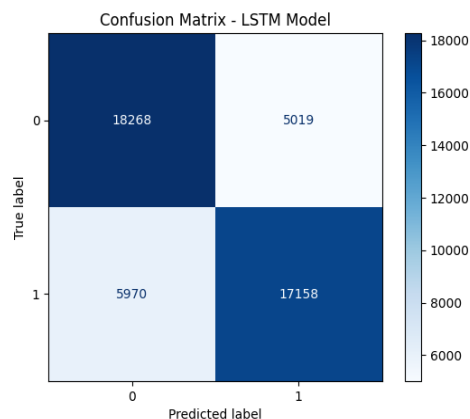


Figure 9: Confusion Matrix-LSTM model

The above Fig 8 and Fig 9 images show confusion matrices for LSTM and BiLSTM models in depression detection. Both models show strong performance with high true positive and true negative counts. Overall, the matrices suggest both models are effective, with slightly different trade-offs in error types.

Performance metrics: In a comparative analysis of neural network models applied to text classification, the Hybrid model, which integrates features from BERT and Word2Vec, outperforms others with the highest accuracy (90.06%) and an impressive balance between precision and recall, as evidenced by its F1-score (89.82%). While BERT and RoBERTa models show commendable performance, with RoBERTa achieving an 88.23% accuracy, CNN, LSTM and BiLSTM models lag in overall effectiveness, despite their advanced sequence processing capabilities. The Word2Vec model notably surpasses traditional deep learning models, securing an 89.59% accuracy. This analysis underscores the importance of model selection tailored to specific project requirements and highlights the advantages of feature fusion in achieving superior classification performance.

	Model	Accuracy	Precision	Recall	F1-score
0	BERT	0.850716	0.883336	0.806987	0.843437
1	Word2Vec	0.895874	0.933757	0.851435	0.890698
2	Hybrid	0.900598	0.921762	0.875800	0.898193
3	CNN	0.835269	0.807720	0.878545	0.841645
4	LSTM	0.763245	0.763654	0.763245	0.763129
5	BiLSTM	0.772250	0.773314	0.772250	0.771993
6	RoBERTa	0.882301	0.883617	0.882301	0.882186

Fig 10: Comparison of Performance metrics for all models

Conclusion:

Drawing on advanced NLP techniques, this guide outlines a clear path for detecting depression through social media text analysis, emphasizing the crucial role of a carefully configured system. It navigates from setup to execution, spotlighting the importance of selecting the right computational tools and models, such as BERT and Word2Vec, for insightful data interpretation. The manual encourages meticulous model tuning and evaluation, deploying practical metrics and visual tools for a nuanced understanding of model strengths and weaknesses. This tailored approach not only enhances the precision of depression detection but also enriches the broader field of NLP applications in mental health, embodying a bridge between technology and impactful social research.

References

- Asghar, M.Z. *et al.* (2018) ‘T-SAF: Twitter sentiment analysis framework using a hybrid classification scheme’, *Expert Systems*, 35(1), p. e12233. Available at: <https://doi.org/10.1111/EXSY.12233>.
- Devlin, J. *et al.* (2019) ‘BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding’, *Proceedings of the 2019 Conference of the North*, pp. 4171–4186. Available at: <https://doi.org/10.18653/V1/N19-1423>.
- Mikolov, T. *et al.* (2013) ‘Efficient Estimation of Word Representations in Vector Space’, *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings* [Preprint]. Available at: <https://arxiv.org/abs/1301.3781v3>
- Shrivastava, P., Soni, K.K. and Rasool, A. (2020) ‘Classical Equivalent Quantum Unsupervised Learning Algorithms’, *Procedia Computer Science*, 167, pp. 1849–1860. Available at: <https://doi.org/10.1016/J.PROCS.2020.03.204>.
- Yadav, A. and Vishwakarma, D.K. (2020) ‘Sentiment analysis using deep learning architectures: a review’, *Artificial Intelligence Review*, 53(6), pp. 4335–4385. Available at: <https://doi.org/10.1007/S10462-019-09794-5/METRICS>.