Comparative Analysis of Machine Learning and Transformer-Based Models for Multi-Class Medical Abstract Classification

Swati Rajwal
Emory University
201 Dowman Drive, Atlanta, GA 30322
Swati.Rajwal@emory.edu

Abstract

Automated text classification has been a center of attention due to its applicability in finding patterns within large textual datasets. This study explores supervised multi-class text classification of medical abstracts categorized into five distinct medical conditions. We trained a range of traditional machine learning algorithms alongside neural network and transformer-based architectures. Through experimental results, we found that RoBERTa achieved the highest accuracy of 0.620 and an AUC of 0.886. Also, linear SVM showed a comparable performance with an accuracy of 0.614. This work aims to contribute insights into the effectiveness of various models for automated text classification and to understand their potential and limitations within the realm of medical abstract categorization.

1 Introduction

Text classification is an important task in the field of natural language processing (NLP). It involves categorizing text into a predefined set of classes and has seen significant advancements since machine learning and deep learning techniques became popular. In particular, for supervised multi-class text classifications, a model is trained with labeled data to classify text into one of multiple categories. This technique has been applied in various domains, including sentiment analysis, topic labeling, medical diagnosis categorization, and many others. Therefore, the supervised learning approach relies on the availability of a very well-labeled dataset, where each text sample is tagged with one of the class labels.

Traditional machine-learning models such as Naive Bayes, Support Vector Machines (SVM), and Random Forests have been popularly used for a very long time for text classification (Ikonomakis et al., 2005). These standard models laid the groundwork for further advancements in NLP and that further set a perfect foundational mathematical-based stage for more complex approaches. Later on, with the evolution of deep learning, neural network-based models have become popular for this task. For instance, the introduction of word embeddings (Mikolov et al., 2013) helped in understanding the way text data is represented which further allowed models to capture more semantic meanings of words. Another such example is the adaptation of LSTMs (Hochreiter and Schmidhuber, 1997) for text classification tasks which gained momentum as part of the broader exploration of deep learning techniques in NLP. LSTMs can remember long-term dependencies which makes them particularly suitable for understanding the context and semantics in sequences of text (Dernoncourt et al., 2017). More recently, the emergence of Transformer-based models such as Bidirectional Encoder Representations from Transformers (or, BERT) by (Devlin et al., 2018) and Robustly optimized BERT approach (or, RoBERTa) by (Liu et al., 2019) has certainly elevated the state-of-the-art performance in text classification tasks.

In this project, the goal is to experiment with the standard machine learning models and much more advanced transformer-based networks' for a multi-class text classification problem. This work aims to contribute to the growing body of knowledge in automated text classification using various models and address both the potential and limitations. The dataset (Schopf et al., 2023) being used in this re-

Class label	Train	Test
Neoplasms	2530	633
Digestive system diseases	1195	299
Nervous system diseases	1540	385
Cardiovascular diseases	2441	610
General pathological conditions	3844	961
Total	11550	2888

Table 1: Class distributions within the dataset.

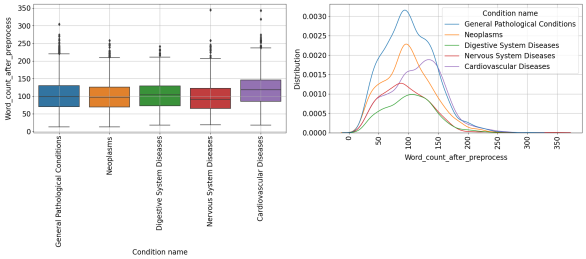

Figure 1: Word count after preprocssing

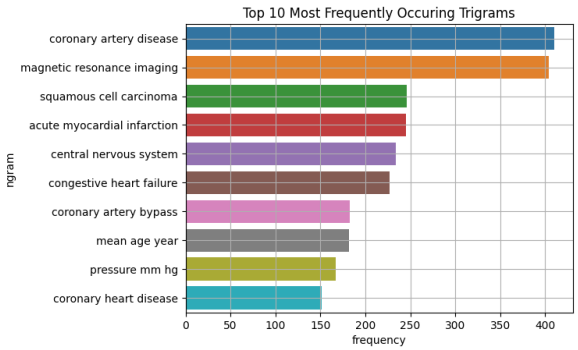

Figure 2: Frequent Trigrams

search project is based on medical abstracts which are labeled with one of the 5 different conditions namely: digestive system diseases, cardiovascular diseases, neoplasms, nervous system diseases, and general pathological conditions. The class label distribution within the train and test dataset are shown in Table 1. This dataset corpus is available under the Creative Commons CC BY-SA 3.0 license at GitHub page[1].

2 Methodolody

2.1 Preprocessing

The most important aspect of the dataset provided is the medical abstracts in textual format. Therefore, before proceeding to do anything, a standard preprocessing function was written that performed a number of operations on the texts. Specifically, the function first converted all texts to lowercase using the lower() method. This is beneficial for text analysis tasks because it helps to ensure that words are treated consistently regardless of their original case. Next, the nltk.word_tokenize function (NLTK) is used to split the text into individual words or tokens. The function then removes any stopwords (common words that often do not carry significant meaning) and non-alphabetic characters.

And finally, each word is lemmatized using the WordNetLemmatizer from the NLTK library. Lemmatization reduces words to their base or dictionary form, which helps in grouping together different forms of the same word. Figure 1 shows the word count after preprocessing. The preprocessed train and test dataset are stored as comma-separated values (CSVs) to be used for further analysis (namely, 'medical_tc_train_preprocessed.csv', and 'medical_tc_test_preprocessed.csv'). From this point forward, any reference to the train or test set should be understood as pertaining to the preprocessed versions of these datasets.

2.2 Feature Extraction

This section covers the data preparation process for our multi-class text classification task on medical abstracts. For the training set, the column 'medical_abstract' containing preprocessed text data served as the feature set, while 'condition_label' represented the target variable. A similar approach was applied to the test set. To convert the text data into a numerical format suitable for machine learning algorithms, the Term Frequency-Inverse Document Frequency (TF-IDF) vectorization method by scikit-learn in python (Tfi) was used. This technique transforms the text into a sparse matrix of TFIDF features. Through Exploratory data analysis (EDA), it was also found that the bi- and trigrams can be very useful as features (see Figure 2). Therefore, the vectorizer was configured to consider unigrams, bigrams, and trigrams by setting the value of ngram_range=(1, 3) and limiting the feature space to the top 5000 terms by frequency across the corpus (max_features=5000).

The 'fit_transform' method of the vectorizer was used on the input texts to learn the vocabulary from the training set and to transform the training set into a TFIDF-weighted document-term matrix. For the test set, the text was transformed into the document-term matrix using the 'transform' method to ensure the model is tested on the same feature space established by the training set. This rigorous process ensures our models are trained and evaluated on vectors that accurately reflect the significance of terms within the medical abstracts while mitigating the risk of overfitting by constraining the number of features.

[1] https://github.com/sebischair/Medical-Abstracts-TC-Corpus

2.3 Machine Learning Models

To classify the medical abstracts, 7 traditional machine-learning classification models have been trained independently. The models are Naïve Bayes, logistic Regression, K-Nearest Neighbors (KNN), Linear Support Vector Machine (SVM), Decision Tree, Random Forest, and Gradient Boosting. Each of these machine learning algorithms offers a unique set of advantages for multi-class text classification. Naïve Bayes is an excellent baseline model due to its simplicity and computation speed. Logistic Regression provides interpretable results and is computationally efficient for linearly separable data. KNN is a versatile, non-parametric method that adapts easily to new data. Linear SVM are effective in high-dimensional spaces and incorporate regularization to avoid overfitting. Decision Trees are easy to interpret and can handle complex, non-linear relationships without the need for feature scaling. Random Forests (built on Decision Trees' strengths) improve robustness and accuracy through ensemble learning. Finally, Gradient Boosting is known for its high accuracy and flexibility, capable of modeling complex structures in data through iterative improvement. Leveraging this diverse set of models allows for a comprehensive exploration of various decision boundaries and data characteristics which ultimately leads to the creation of a more robust and accurate model.

2.4 Neural Network

In addition to traditional machine learning models, a neural network was built using TensorFlow and TensorFlow Hub. This text classification neural network uses the Universal Sentence Encoder[2] for embeddings (converts text into high dimensional vectors) and a custom self-attention mechanism for capturing contextual relations within medical abstract texts.

Next, a self-attention mechanism is implemented using a custom TensorFlow layer. This block uses Long Short-Term Memory (LSTM) layers to create query and value sequences. It then applies additive attention to these sequences and processes them with global average pooling and batch normalization. The block finally generates a residual output combining the processed query and attention components.

The input layer of the neural network takes preprocessed text data and the Universal Sentence Encoder generates embeddings from the input text. A custom self-attention block (as explained earlier) is applied to the embeddings. The output from the self-attention block is combined with the original embeddings through element-wise multiplication and dense layers with an activation function are used for further processing. The final output layer uses a softmax activation function for classification which is perfectly suitable for multi-class in our dataset.

2.5 Transformer-based Models

The BERT ('bert-base-uncased') model (Devlin et al., 2018) was configured for multi-class text classification with our datasets' five output labels. The architecture of BERT allows it to process words in relation to all the other words in a sentence, unlike traditional models that look at words in isolation. Therefore, it captures the context more effectively. The model specifics include a maximum sequence length of 512 tokens (since the medical abstracts are very large) and a batch size of 16 for training. This configuration provided the best tradeoff in terms of model performance and available computational resources. The model was fine-tuned on our dataset. Also, CUDA was used to expedite the training process on GPU hardware. Another similar transformer-based model, RoBERTa (Liu et al., 2019), was utilized for the classification task. As an optimized version of BERT, RoBERTa enhanced performance. For the current project, the model used the same sequence length and batch size as used in BERT to allow for direct comparison. The training utilized CUDA for efficient computation on GPUs. Both models' performances were further analyzed using confusion matrices and detailed classification reports, providing insights into each model's precision, recall, and F1 score per class. The results are listed in further sections.

SimpleTransformers[3] is an easy-to-use library based on the Transformers library by Huggingface (Wolf et al., 2020). Specifically in the current project, the 'simpletransformers.classification.ClassificationModel' class within the library is utilized for training and evaluating transformer-based models (BERT and RoBERTa).

[2] https://tfhub.dev/google/universal-sentence-encoder/4

[3] https://github.com/ThilinaRajapakse/simpletransformers/

Metric	Description
$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$	Good starting point for evaluation but can be misleading in imbalanced datasets.
$\text{Balanced Acc.} = \frac{1}{N}\sum_{i=1}^{N}\frac{\text{TP}_i}{\text{TP}_i+\text{FN}_i}$	Crucial for datasets where some classes are underrepresented as observed during EDA (Table 1).
$\text{Precision}_i = \frac{\text{TP}_i}{\text{TP}_i+\text{FP}_i}$	Measures the accuracy of positive predictions for each class. High precision indicates a low rate of false positives.
$\text{Recall}_i = \frac{\text{TP}_i}{\text{TP}_i+\text{FN}_i}$	Shows the proportion of actual positives that were correctly identified. High recall means most of the positive cases are captured by the model.
$\text{F1 Score} = 2 \times \frac{\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$	Useful when you want to weigh each instance or prediction equally. It calculates metrics for each label and finds their unweighted mean. This does not take label imbalance into account.
$\text{F2 Score} = \frac{5\times\text{Precision}\times\text{Recall}}{4\times\text{Precision}+\text{Recall}}$	Gives more weight to recall than precision. It is useful when missing actual positives (low recall) is costlier than false positives (low precision).

Table 2: Classification results across various evaluation metrics

2.6 Evaluation Metrics

The models as stated earlier are benchmarked across multiple metrics to provide a comprehensive view of the model's performance, considering not just overall accuracy but also how well the model performs on each individual class, how it balances the trade-off between precision and recall, and its capability to handle imbalanced datasets. The metrics used in this project are described in Table 2. AUC (Area Under the ROC Curve) provides an aggregate measure of performance across all possible classification thresholds. It is useful for evaluating the probability outputs of a classifier, and it's robust against imbalanced data.

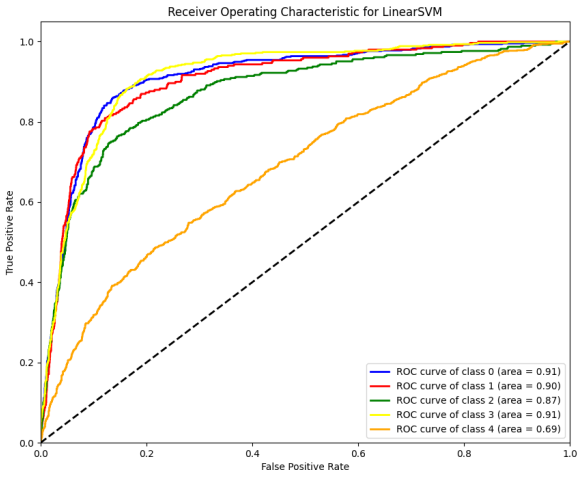

Figure 3: Linear SVM ROC curve

3 Experimental Results

3.1 Hyperparameter Selection

Hyperparameter tuning is a critical aspect in the construction of an optimized model. To ensure the best parameter configuration, a grid search cross-validation approach is employed by using the GridSearchCV library (Pedregosa et al., 2011) with the value of cross-validation (CV) set to 5. This method allowed us to systematically traverse the hyperparameter search space, evaluating each combination's performance through cross-validation. Subsequently, the best estimator was identified from the grid search results using the grid_search.best_estimator_ attribute. The final optimal model parameters were selected based on a subset of the training dataset. The selected hyperparameters have been documented in detail and are presented in Table 3 for reference.

3.2 Classification Model Results

Following the selection of hyperparameters, the ML models were trained on the dataset. Subsequently, the trained models were evaluated using the provided test dataset. Similarly, for the neural networks, BERT and RoBERTa model, the out-

Model	Hyperparameters
Naïve Bayes	alpha=0.01
Logistic Reg.	C=1, penalty='l2', solver='liblinear'
Decision Tree	max_depth=10
Random Forest	min_samples_split= 5, n_estimators=100
Linear SVM	loss='hinge',penalty='l2',alpha=0.001, max_iter=5, tol=None
KNN	n_neighbors=7
Gradient boost.	n_estimators=200, learning_rate=0.01

Table 3: Class distributions within the Medical Abstracts dataset.

Model	Accuracy	Balanced Acc	F1 Micro	F1 Macro	F2 Micro	F2 Macro	Precision	Recall	AUC
Naive Bayes	0.586	0.591	0.586	0.586	0.586	0.589	0.582	0.591	0.857
Logistic Reg.	0.591	0.565	0.591	0.577	0.591	0.569	0.598	0.565	0.874
Decision Tree	0.500	0.406	0.500	0.400	0.500	0.400	0.461	0.406	0.714
Random Forest	0.473	0.458	0.473	0.462	0.473	0.459	0.472	0.458	0.785
Linear SVM	0.614	0.607	0.614	0.595	0.614	0.599	**0.612**	0.607	0.862
KNN	0.542	0.545	0.542	0.532	0.542	0.538	0.531	0.545	0.810
GradientBoosting	0.537	0.461	0.537	0.479	0.537	0.463	0.592	0.461	0.833
Neural Network	0.594	0.587	0.594	0.582	0.594	0.584	0.587	0.587	0.852
BERT	0.591	0.603	0.591	0.590	0.591	0.597	0.581	0.603	0.873
RoBERTa	**0.620**	**0.632**	**0.620**	**0.618**	**0.620**	**0.626**	0.609	**0.633**	**0.886**

Table 4: Classification results across various evaluation metrics

comes, including classification accuracy, micro-averaged F1 score, macro-averaged F1 score, and various other evaluation metrics are listed in Table 4. Among the ML models, the Linear SVM has the best overall performance with the highest scores across many evaluation metrics (see the AUC Figure 3). But overall, RoBERTa performed better.

4 Discussion

There are a couple of conclusions that we can make based on the results shown in Figure 4 for a multi-class medical abstracts classification task. First and foremost, RoBERTa stands out as the top-performing model with an accuracy of 0.620, a balanced accuracy of 0.632, and an AUC of 0.886. This indicates its advanced capability for capturing the variance in language which we can attribute to the model's deeper bidirectional context understanding. Overall, the transformer-based models (BERT and RoBERTa) demonstrated strong performance in terms of AUC scores, with BERT at 0.873 and RoBERTa at 0.886.

Among the traditional ML models, the Linear SVM exhibits relatively high performance with an accuracy of 0.614 and an AUC of 0.862. In contrast, the decision tree showcased limitations in its ability to manage the classification task effectively with the lowest accuracy of 0.500 and a lower AUC of 0.714. The F1 Micro and Macro scores are highest for RoBERTa (0.620 for both) indicating a balanced performance across various classes. The Linear SVM follows right behind RoBERTa with an F1 Micro of 0.614 and an F1 Macro of 0.595.

Based on the experimental results, we can observe that the candidate models trained such as Gradient Boosting and KNN show that these models performed poorly which may suggest that they may struggle with the complexity of the text classification task compared to deep learning-based models. The LSTM Neural Network exhibits a moderate level of performance, with an accuracy of 0.594 and a balanced accuracy of 0.587. These metrics are notably higher than some traditional machine learning models such as the Decision Tree and Gradient Boosting but are outperformed by the Linear SVM and both transformer-based models, BERT and RoBERTa. The LSTM's F1 scores, both Micro and Macro, are 0.594 and 0.582 respectively, which are competitive with BERT's 0.591 and 0.590 but fall short of RoBERTa's

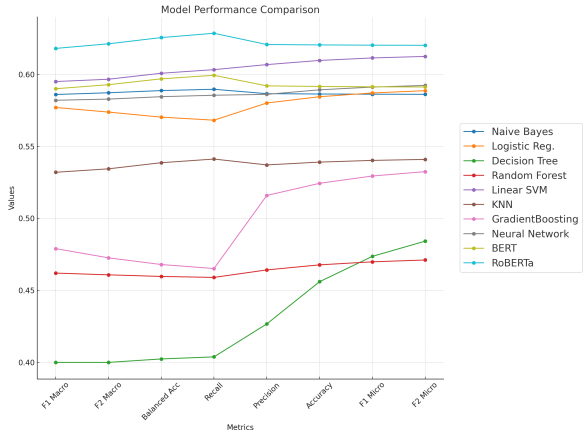

Figure 4: Comparative Performance Metrics of Classification Models

0.620 for both metrics. The neural net scored a fairly high AUC (0.852), however, not as high as BERT's or RoBERTa's. This suggests that while LSTM models are known for capturing long-term dependencies within text, the bidirectional context understanding of the transformer-based models is providing a more subtle understanding of text data. Another nice observation about this experimental dataset is that there is a consistent pattern where models with higher accuracy also tend to have higher scores across other metrics such as F1 scores and AUC, which suggests that accuracy is a good indicator of overall performance for this dataset. Although logistic regression has a higher precision score of 0.598 compared to BERT's 0.581, BERT outperforms it in recall, F1, and AUC scores. This could indicate that logistic regression is more conservative in its predictions, leading to higher precision but potentially at the cost of recall.

In summary, the experimental results suggest that RoBERTa and BERT are better-performing models than the traditional machine learning models for our dataset. However, traditional models like the linear SVM still hold competitive ground especially when considering the simplicity and computational efficiency of such models.

5 Limitations and Future Directions

Through experimental analysis, it has been reported that linear SVM performed nearly as well as the best-performing model. It'd be interesting to see if the model's performance could still potentially be improved with hyperparameter tuning with larger search space, feature engineering, and other techniques.

Due to limited computational resources, there was a constraint that imposed limitations on the depth and breadth of model training. Due to this, the models were trained for a limited number of epochs, which might have affected their ability to fully learn from the training dataset. In future work, the plans include a comprehensive exploration of model parameters, longer training durations, and potentially developing lightweight versions without significant loss in performance.

While RoBERTa outperformed all the candidate models for this dataset, it is important to mention here that these complex models act as black boxes which makes it difficult to interpret their decision-making processes. Enhancing model interpretability to understand the reasons behind classifications will be a key area of my future research.

Conclusion

The research work discussed here presents an in-depth comparison of various machine learning models for the multi-class classification of medical abstracts. The findings indicate that while traditional machine learning models provide a strong baseline, the transformer-based model (i.e., RoBERTa) showcased higher performance across multiple evaluation metrics. The study reveals a consistent trend where models with higher accuracy also achieve higher scores across other metrics, suggesting accuracy is a robust indicator of performance for the dataset used. The paper also highlighted the limitations, such as computational constraints that restricted extensive hyperparameter tuning and model training. Future directions include exploring more comprehensive hyperparameter optimization, longer training durations, and enhancing the interpretability of complex models to better understand their decision-making processes.

Ethics Statement

This work has been conducted as part of the CS 571 coursework at Emory University. The dataset utilized for the project is publicly available[4] and has been accessed in accordance with its intended use and the terms of service associated with it. No private or sensitive personal data has been collected

[4] https://github.com/sebischair/Medical-Abstracts-TC-Corpus/blob/main/medical_tc_train.csv

or analyzed as part of this project. All analyses carried out on the dataset are purely for educational purposes, within the scope of the course, and the findings will not be used for any commercial benefit. This report and any associated code or output will be used solely for academic assessment within the NLP course and will not be disseminated outside of this context without proper review and approval by the course instructors. Should the project be considered for any form of publication or presentation beyond the classroom, further ethical review and approval will be sought as per university regulations and academic best practices.

Data and Code Availability

The dataset used in this work is publicly available in this Github repository. Python 3 was used as the programming language and code related to this project can be located in this GitHub Repository: https://github.com/cosinesimilarity1/Swati_rajwal_cs571_project.

Acknowledgements

Many thanks to Dr. Fei Liu (Emory University) for being an amazing educator, especially for coursework CS 571. This work has been inspired by the concepts learned during the lectures.

References

Sklearn tfidfvectorizer. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html. Accessed: December 2023.

Franck Dernoncourt, Ji Young Lee, and Peter Szolovits. 2017. Neural networks for joint sentence classification in medical paper abstracts. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 694–700, Valencia, Spain. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

M Ikonomakis, Sotiris Kotsiantis, Vasilis Tampakas, et al. 2005. Text classification using machine learning techniques. *WSEAS transactions on computers*, 4(8):966–974.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

NLTK. Nltk tokenizer package. https://www.nltk.org/api/nltk.tokenize.html. Accessed: October 2023.

F. Pedregosa, G. Varoquaux, and et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Tim Schopf, Daniel Braun, and Florian Matthes. 2023. Evaluating unsupervised text classification: Zero-shot and similarity-based approaches. In *Proceedings of the 2022 6th International Conference on Natural Language Processing and Information Retrieval*, NLPIR '22, page 6–15, New York, NY, USA. Association for Computing Machinery.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

A Appendix

A.1 Implementation Details

Python 3+ version was used for all the coding purposes. Major packages required to run the scripts are: pandas, matplotlib, seaborn, nltk, tqdm, collections, numpy, sklearn, joblib, tensorflow, tensorflow_hub, simpletransformers.