COMP90051 AI-Generated Text Detection Report - Group 47

1. Introduction

As large language models (LLMs) evolve, machines are becoming increasingly adept at producing text that resembles human writing. This advancement in AI, if misused, can lead to serious consequences such as the dissemination of fake news, misinformation, intellectual property infringements, and the creation of deceptive product reviews, leading to detrimental effects (Chakraborty et. al, 2023). Therefore, there is a pressing need to devise algorithms that can discern between machine-generated and human-written texts.

This study addresses the challenge of detecting AI-generated text by crafting a binary machine learning classifier. We have explored and compared the efficacy of several machine learning models: Random Forest (RF), XGBoost, Logistic Regression (LR), Support Vector Machines (SVM), Multilayer Perceptron (MLP), and Recurrent Neural Network (RNN).

2. Final Approach

In our analysis, we employed Bag of Words (BoW) vectorisation with an MLP neural network. Main reasons for choosing this model include its suitability for classification tasks, ability to handle large input dimensions, and flexibility in accommodating complex architectures tailored to specific data characteristics. Given that the BoW approach can sometimes omit nuances, the MLP serves as a valuable tool in extracting deeper connections within the data.

2.1 Preprocessing and Vectorisation

We identified and removed 703 duplicates to ensure no duplicates between training and testing sets. We evaluated various vectorisation methods: BoW, binary representation, frequency count, and TF-IDF. Due to the stochastic nature of MLP, we conducted five trials for each approach to gauge consistency.

In our analysis, the baseline BoW approach surprisingly yielded the best results as demonstrated in Figure 1. This may be due to BoW's inherent simplicity have deterred overfitting, retaining a broader range of information. Given that our dataset was pre-tokenised which presented as number sequences rather than words, we lacked the granularity to filter out stop words. However, while TF-IDF effectively discounts frequent terms across documents, it might unintentionally diminish the importance of some vital tokens. Consequently, BoW's straightforward representation was deemed the most fitting for our final model.

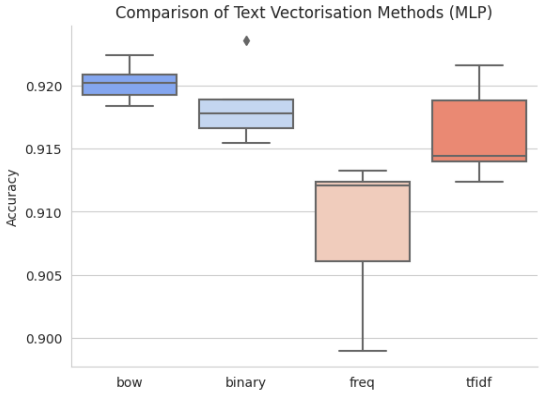

Figure 1. Vectorisation Comparison

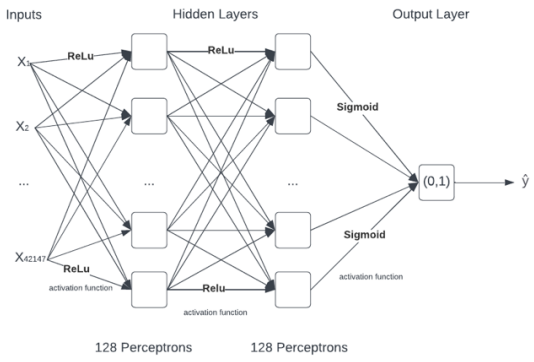

Figure 2. MLP Overview

2.2 Reasoning for Model

The architecture was designed based on the nature of our dataset. The initial layer was designated with 5,000 nodes, mirroring the 5,000 unique lexical entities present in our dataset. Subsequently, we incorporated two intermediate hidden layers, each equipped with 128 nodes, and utilised the Rectified Linear Unit (ReLU) activation function. This choice was motivated by ReLU's efficacy and computational efficiency in such contexts. To prevent overfitting, we interspersed dropout layers post each hidden layer to randomly deactivate certain nodes, thereby promoting model robustness. During the training phase, the stratify parameter was set to ensure uniform distribution of target classes across training batches. The terminal layer employs a sigmoid activation function, apt for our binary classification task. For the optimisation process, the Adam optimiser was chosen, primarily for its adaptability and superior convergence properties in gradient descent methodologies.

2.3 Evaluation

Our model garnered an accuracy rate of 85% on Kaggle. This performance can be attributed to our strategic handling of imbalanced data, detailed in Section 4, as well as the fine-tuning of our output's sigmoid function. During our training process, we observed false negative instances as our model mistakenly classifies human-generated text as machine-generated. Therefore, we adjusted our sigmoid function parameters to balance this bias.

In addition to accuracy-focused measures, we assumed that both accuracy and the Area Under the Receiver Operating Characteristic (AUROC) were pivotal metrics post-data rebalancing. The ROC curve, crucial for imbalanced data, highlights true and false positive rates, helping us understand potential biases in our model towards human or AI-generated data. Ultimately, our optimal model boasted a test accuracy of around 91% with an AUROC of 0.97.

3. Other Model

To solve the classification problem, we initially considered SVM, XGBoost, RNN and LR as our baseline approach with best performance comparison demonstrated in Figure 3. After visualising the data using a 3D PCA plot during our EDA, we observed potential for classification based on dimensions, prompting us to incorporate SVM into our approach. However, SVM is too computationally expensive considering the size of the training data, it takes much longer to train compared to other models.

The reasons we implement those models are that XGBoost is robust to bias and generally performs well on classification and RNN is designed for NLP and sophisticated approaches. Given that most of our models lean towards predicting machine-generated content, we suspect a significant bias in the data, leading to skewed predictions. To counteract this, we implemented techniques like over-sampling, under-sampling, SMOTE, and ADASYN. Additionally, we fine-tuned parameters like tree depth and prediction thresholds to mitigate data bias.

For XGBoost, it generally performs well on base params (tree depth = 3, #features=5000), and the cross-validation score is increasing until get (tree depth=9, #features=10000). The accuracy is remaining the same and the accuracy on the test dataset is slightly decreasing when the tree depth is increasing after 9 depths. The XGBoost performance graph shows that increasing the number of the features would not help to improve the accuracy. The peak of cross-validation accuracy is 96% but the accuracy on the split test set is 86% which is a big gap between them. We found that after performing over-sampling on the training dataset there is duplicated data that affect cross validation accuracy which might lead to overfitting. Therefore, we focus more on split test test set accuracy for comparing model performances.

For LR, despite the high dimensionality of data, it proved to be efficient and was more computationally efficient compared to XGBoost with the accuracy reaching 80% on the split test set. The average cross validation score of LR is 80% and it is the same accuracy on the split test set. However, the model exhibited a notable prediction bias, and due to its simplicity, further accuracy improvements were challenging. Hence, we moved to neural network approaches to identify and understand the deep connection between the elements of the data.

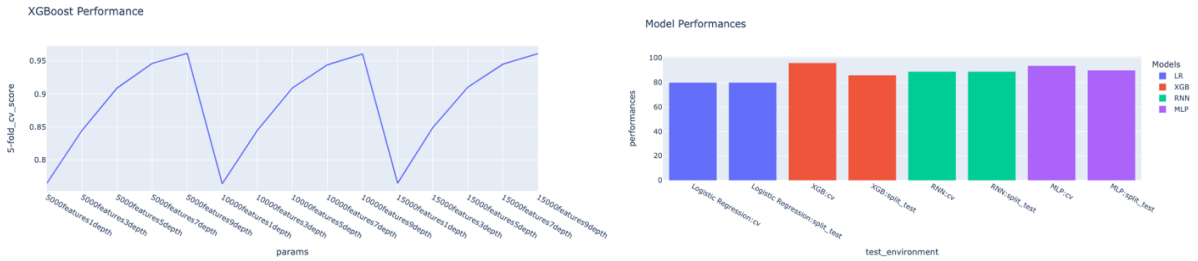

Figure 3. Other Model Performance

For the neural network approach, we use two models RNN apart from MLP. RNNs are specifically designed for sequential data like stock prices or language patterns. While we previously utilised n-grams to help models grasp data sequences, the inherent design of RNNs is to comprehend such sequences. Given that our tokens are already vectorised, our primary task was to pad the data before training. In our model, we incorporated a bidirectional layer to better capture the relationships between elements in a sentence. However, RNN did not generate desired results. Potential reasons may be that RNN has a limitation of input length which we set as 200-400 words and our need to select part of the features may result in loss of crucial information. Furthermore, the limitation for text length above 400 is computationally expensive and accuracy did not improve much.

4. Discussion on Domain Impact

Upon examining, neither domain 1 nor domain 2 individually covered all the features in the test data. Thus, we combined them as the training set to fully represent the test set features.

To address imbalanced dataset, we employed ADASYN (Adaptive Synthetic Sampling), which adaptively generates synthetic samples for the minority class based on their density distribution using K-Nearest Neighbours classifier. ADASYN demonstrates advantages over SMOTE as it uses unsupervised machine learning techniques to discover inherent distribution patterns within the dataset and is more adaptive in setting decision boundaries for overlapping classes. With application of ADASYN, our class distribution saw a notable shift, equalising the representation of human-generated and machine-generated text to a 1:1.00 ratio from the former imbalance of 1:1.83 as demonstrated in Figure 4.

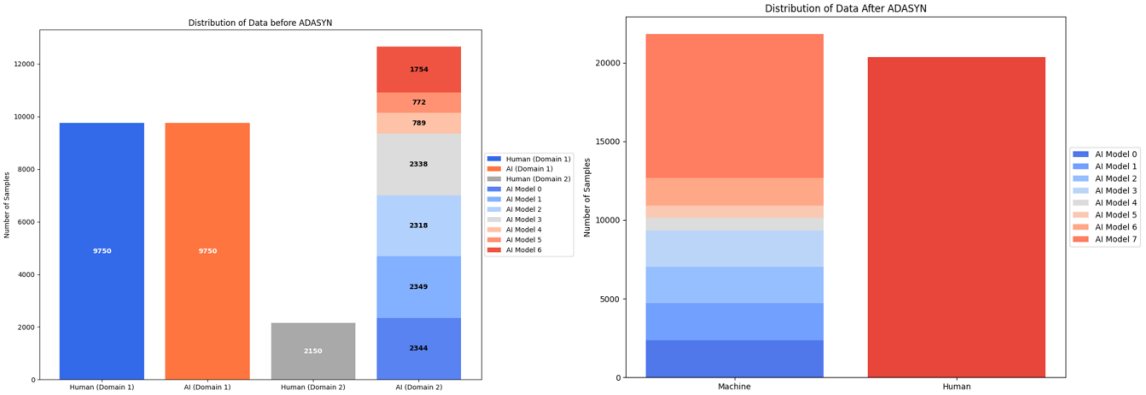

Figure 4. Data Distribution Before and After ADASYN (model 7 refers to AI model in domain 1)

5. Reference

Chakraborty, S., Bedi, A. S., Zhu, S., An, B., Manocha, D., & Huang, F. (2023). On the Possibilities of AI-Generated Text Detection. https://doi.org/10.48550/ARXIV.2304.04736