AUTHORSHIP ATTRIBUTION FOR NEURAL TEXT GENERATION (SYNTHETICALLY GENERATED TEXT DETECTION)

Charan Sai Sadla
charansa@buffalo.edu

Jasleen Kaur
jasleen2@buffalo.edu

Rachna Bhatia
rachnatr@buffalo.edu

Deepanshu
deepans2@buffalo.edu

1 Introduction

Recent advancements in language modeling using neural networks have brought us to a point where it's difficult to differentiate between human-written texts and those generated by machines. While these neural techniques have several advantages and utilities, there are contexts such as forensic linguistics, plagiarism detection, or tracing the origin of threats or anonymous messages, where identifying the author of a text is crucial. Therefore, it has become imperative to distinguish between machine-generated texts and those written by humans. One way to solve the problem of Authorship Attribution is to determine which neural language generation method produced the text in question. So in our implementation, we have 3 problems:

1. **Same method or not:** Given two texts T1 and T2, determine if both T1 and T2 are generated by the same NLG method (or human) or not.
2. **Human vs. Machine:** Given a text T1, determine if T1 is written by human or generated by any k NLG methods.
3. **Authorship Attribution:** Given a text T1, single out one NLG method (among k alternatives) that generated T1.

2 Baseline Architecture

Our baseline architecture (Figure 1.) for solving the NLG method identification problems consists of the following components:

2.1 Data Extraction

To support our modeling efforts, we gathered data from ten natural language generators and a human. we used the same prompts to extract data from OpenAI for both GPT3 and InstructGPT via OpenAI's API. In our Reddit case study, we obtained data from three subreddits - r/changemyview, r/Advice, and r/relationships. With the extracted data prompts for the Reddit posts, we extracted data from OpenAI API's for both GPT3 and InstructGPT.

2.2 Data Cleaning and Preprocessing

To prepare our data for analysis, we merged the Titles and Generation columns to include both text and headings. Next, we utilized the Neattext library to convert the data to lowercase, remove special characters, punctuations, and emojis, and eliminate HTML tags. We decided not to eliminate stopwords as they could provide valuable insights for various natural language generation (NLG) techniques. Lastly, we combined all 11 files into a single document that included the generated text and corresponding labels indicating which NLG (or human) produced it.

2.3 Problem P1: Identifying the NLG Method

In order to begin the research, the data from the CSV file was loaded into a Pandas data frame. The necessary libraries were imported, and then the data was divided into training and testing groups using an 80/20 split ratio. With this split, the model was able to be trained on a sizable quantity of data while still having access to sufficient data to evaluate its performance. The pre processing of the text data involved creating a function that tokenized the text, assigned a Part of Speech (POS) tag to each word, and turned the POS tags into a string of text. The features of the text data that could be used for classification were extracted as a result of this preprocessing phase. The TFIDF-Vectorizer, which assisted

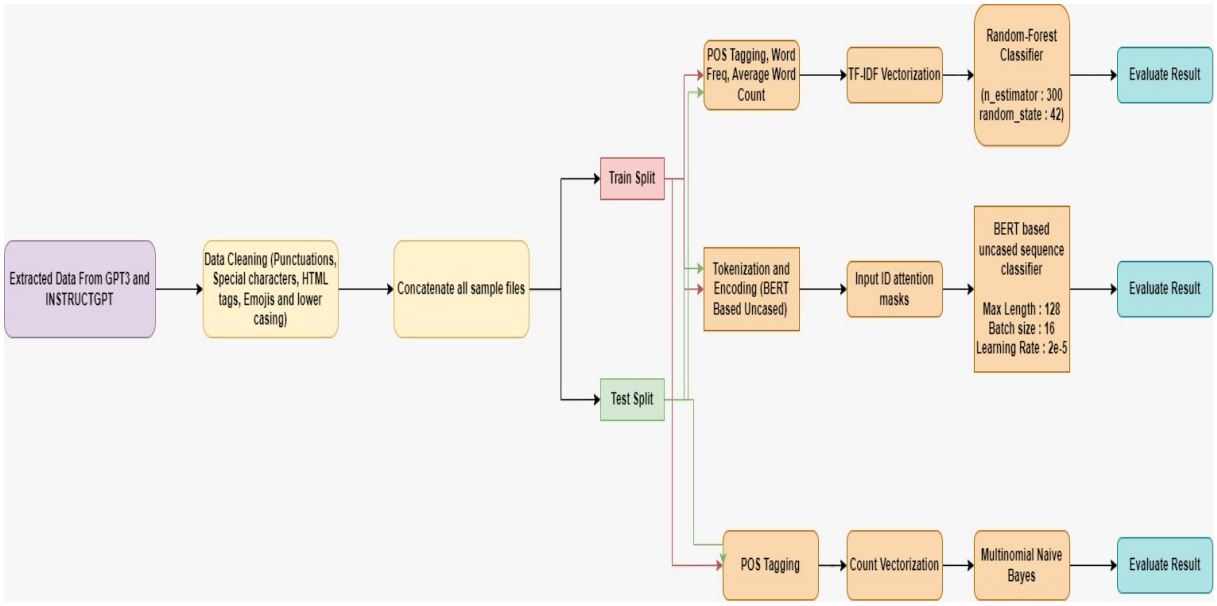

Figure 1: Baseline Architecture.

in converting the text data into numerical features, was used to retrieve the text features. For both the training and assessment sets of data, the extracted text features were combined with the POS tagging features. After that, the training set of data was fed into a Random Forest Classifier, which was initially started with 500 estimators and a random state of 42. The trial data's class was then predicted using the trained classifier. By comparing the predicted classes with the real classes of the testing data, the accuracy of the classifier was determined. The model was provided with 2 input prompts to predict their class and then compared whether both are generated by same NLG(or Human) or not. Finally, a classification report that provided a more in-depth analysis of the model's performance was produced, showing the precision, recall, f1-score, and support for each class.

2.4 Problem P2: Human vs Machine

The model is expected to predict whether the given input text T is written by a Human or a language generative model. To perform this task, we need a binary classifier to classify whether the given input is classified into one of two categories or classes. BERT (Bidirectional Encoder Representations from Transformers): BERT is a pre-trained transformer-based model that has achieved and proved the state of art results on many NLP tasks. Considering these factors, we used BERT ("Bert based uncased") version for the classification task (Figure 2.). BERT is completely an encoder-based network composed of several layers each having a self-attention mechanism which is a great addition to any model. We trained BERT based uncased model using the tokenized and encoded input data to make predictions about whether the given prompt is generated by a human or a machine. The model gave accuracy and precision of around 92 percent in the final epoch (Figure 3.).

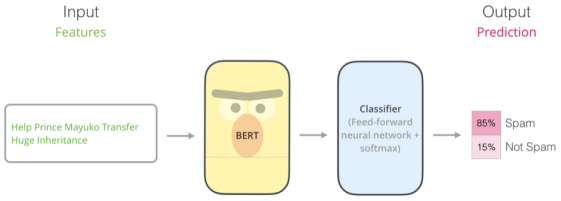

Figure 2: A diagram of the BERT model architecture.

2.5 Problem 3: Given a synthetic text T1, single out one NLG method (among k alternatives) that generated T1.

The objective of the task was to identify the NLG method responsible for generating a given text T from a set of k possible alternatives. To accomplish this, we utilized popular machine learning models, logistic re-

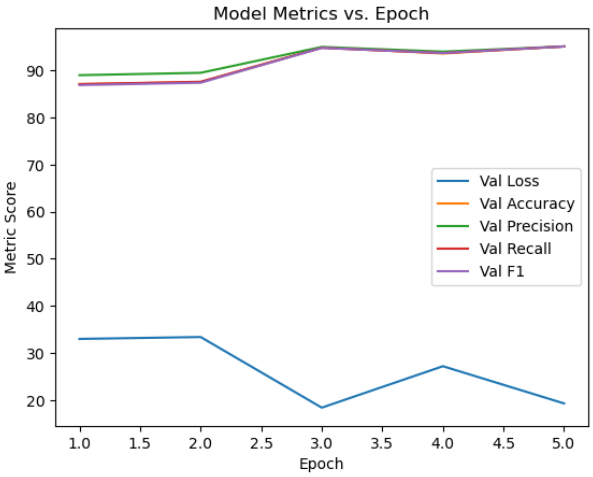

Figure 3: Model Metrics vs Epochs.

gression, svm and Multinomial Naive Bayes, which are commonly used in text classification. By employing various linguistic features such as word frequency, average word count, and POS tagging, we gained a better understanding of the input data. The input features were then transformed into a matrix of token counts using count vectorization. The training set was divided into a 70/30 ratio, and the models were trained on the vectorized data. Predictions were made on the test set, and a classification report was generated to evaluate the model's performance, which included accuracy, precision, recall, and F1 scores for our predicted data. The Multinomial Naive Bayes model gave better results among the other alternatives with an accuracy and precision rate of approximately 67 percent.

Evaluation Metrics:

Problem #	Parameters	Precision	Recall	F1-Score	Support
Problem 1	accuracy	-	-	0.70	2345
	macro avg	0.70	0.70	0.70	2345
	weighted avg	0.70	0.70	0.70	2345
Problem 2	Epoch 1.0	0.89	0.871	0.869	-
	Epoch 2.0	0.895	0.876	0.874	-
	Epoch 3.0	0.95	0.948	0.948	-
	Epoch 4.0	0.94	0.937	0.937	-
	Epoch 5.0	0.951	0.951	0.951	-
Problem 3	accuracy	-	-	0.66	3518
	macro avg	0.78	0.64	0.63	3518
	weighted avg	0.76	0.66	0.63	3518

Figure 4: Evaluation Results.

3 Analysis of Baseline Results

Figure 4. represents our evaluation results for all the subtasks.

In problem P1, we experimented with several models such as Naïve Bayes, Logistic Regression, and Random Forest to predict classes for two input prompts. Among these models, the Random Forest gave the best results with an accuracy, precision, and recall of around 0.70. While we also attempted to use word2vec embeddings to train our model, the results showed a decline in performance. Although Random Forest provided better results, we believe that using neural network models could further improve the accuracy of our predictions.

For problem P2, we trained an uncased BERT model to classify input prompts as either human-generated or machine-generated. The model achieved an accuracy, precision, and recall of 0.92 in the final epoch. BERT, being a transformer model with hidden layers, was able to learn the input more effectively than other machine learning models. We observed that the learning improved until epoch 3, after which it stabilized.

Regarding problem P3, we applied Multinomial Naïve Bayes and Logistic Regression models to predict the NLG method used to generate a given input prompt. The Multinomial Naive Bayes model gave the best results with an accuracy, precision, and recall of approximately 0.66. We observed that using word2vec embeddings led to a decline in performance, and countvectorizer gave better results.

4 Areas of Improvement

• Perform additional linguistic analysis: By conducting more in-depth linguistic analysis, such as sentiment analysis, LIWC Features, Rhetorical Style, or named entity recognition, we can gain a better understanding of the underlying themes and patterns in the text data. This can help to inform more targeted and accurate classifications.

• Explore different neural models: Neural models, such as recurrent neural networks (RNNs) or convolutional neural networks (CNNs), may be able to capture more complex relationships and patterns in the text

data than traditional models like the Random Forest Classifier used in this analysis. By exploring different neural models, we can find the best one for our specific task in the final implementation.

• Experiment with hyperparameters: Hyperparameters are parameters that are set prior to training a model, such as learning rate or number of hidden layers. By experimenting with different hyperparameters, we can optimize the performance of our models and potentially improve accuracy. We will conduct experiments with hyperparameters in our next phase.

• Use ensemble methods: Ensemble methods involve combining multiple models to improve performance. By using ensemble methods, we can reduce the risk of overfitting and increase the robustness of our predictions. If given time, we will explore ensemble methods as well.

5 References

- https://github.com/AdaUchendu/Authorship-Attribution-for-Neural-Text-Generation
- https://towardsdatascience.com/multi-class-text-classification-with-deep-learning-using-bert-b59ca2f5c613
- https://towardsdatascience.com/cleaning-preprocessing-text-data-by-building-nlp-pipeline-853148add68a
- https://www.geeksforgeeks.org/random-forest-classifier-using-scikit-learn/
- https://www.geeksforgeeks.org/applying-multinomial-naive-bayes-to-nlp-problems/