APS360 PROJECT PROGRESS REPORT: AI GENERATED TEXT VS HUMAN WRITTEN TEXT

Vivian Huynh
Student# 1006978521
vivian.huynh@mail.utoronto.ca

Amelie Smithson
Student# 1006684651
amelie.smithson@mail.utoronto.ca

Ji sung Han
Student# 1006581815
jishan.han@mail.utoronto.ca

Hana Truchla
Student# 1006988422
hana.truchla@mail.utoronto.ca

ABSTRACT

This project progress report builds on our project proposal on creating a model that can decipher between AI generated text and text written by people. This report includes a brief description of the project, the individual contributions and responsibilities of group members, and a description of the work done thus far for data processing, the baseline model, and the primary model.

—-Total Pages: 9

1 BRIEF PROJECT DESCRIPTION

1.1 PROJECT PURPOSE AND IMPORTANCE

Our project aims to develop a deep learning model that can accurately distinguish between AI-generated text and human-written text. The model is trained on a variety of datasets and targets an accuracy of at least 70%.The rapid growth of AI-generated text has complicated maintaining academic integrity and verifying the reliability of online information (Kannan (2024)). Our model provides universities and other institutions with a reliable tool to ensure the authenticity of written content. Achieving over 70% accuracy ensures practicality, reduces errors, and increases reliability. Furthermore, as AI text generators improve, identifying AI-written text is becoming harder (Woodall (2024)). Many people lack the expertise to detect AI in papers. Even experts can benefit from using this model for confirmation, as it can classify text faster and more accurately.

Overall, the project adapts to the evolving landscape of AI-generated content, providing proactive solutions for content verification.

1.2 WHY DEEP LEARNING

Simple code to compare predefined features is often redundant and time-consuming, and such features may be biased or inaccurate. Deep learning, however, can handle complex language patterns and classifications efficiently due to its layered structure (Mahapatra (2018)). Its speed, accuracy, and ability to learn from raw text make it suitable for this task. In natural language processing (NLP), deep learning is advantageous as it automatically learns useful features, handles sequential data well with RNNs and Transformers, and supports end-to-end training. Pre-trained models can be fine-tuned for specific tasks, to achieve better performance. For detecting AI-generated text, deep learning excels at recognizing subtle machine-generated patterns, and can adapt as AI evolves.

The following Figure 1 demonstrates the objective of this project and outlines the data processing and architecture routes the program will follow.

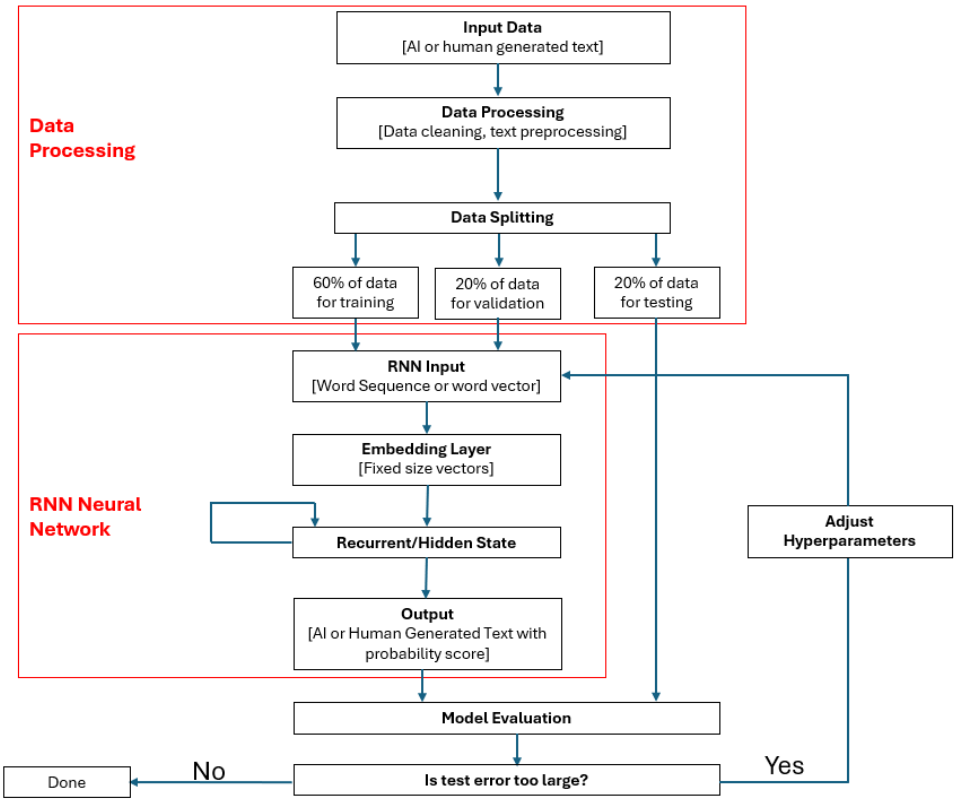

Figure 1: The flow chart for the RNN model and training process.

2 Individual Contributions and Responsibilities

2.1 How the Team Works Together

Our team uses Google Docs for project management, Discord for day-to-day communication, and GitHub and Google Colab for code sharing and versioning. These settings allow you to make real-time adjustments and track your progress.

2.2 Responsibilities and Work Produced for the Progress Report

Ji

Draft the initial project description, organize the team's responsibilities, provide resources for data processing, add and edit brief descriptions, write paragraphs about responsibilities, and create tables detailing contributions.

Hana

Responsible for collecting and organizing data and ensuring data quality for model training. She conducted the data preprocessing step to clean the data by extracting important information and removing duplicates. Additionally, the dataset was split into 60% training data, 20% validation data, and 20% test data. To enter the data in a consistent format, each essay was split into a list of words and organized into a list. Punctuation and capitalization were considered important information and were not removed. Hana made an important contribution in ensuring that models can be trained and evaluated effectively.

Viv

Responsible for implementing and testing the baseline model and documenting initial results. Specifically, Viv first converted text to numbers using the Bag of Words model. Then, implemented

the Random Forest model and performed hyperparameter tuning to achieve optimal performance. She also evaluated model performance, documented results, and created confusion matrices using validation and test data. Through this work, Viv verified the accuracy of the baseline model and provided a basis for comparison for the future model.

Amelie

Responsible for key model development details, architecture design, and hyperparameter tuning. Concluded that RNNs will be used to process text data and identify contextual connections through embedding layers and bidirectional LSTM layers. Furthermore, the model's output probability will be calculated by adding two fully connected layers. Finally, binary cross-entropy will be used as the loss function and updated the weights with the Adam optimizer. A portion of training data will be used to ensure the model is efficient and effective, and adjusted as needed. Key challenges predicted include model training time and Google Colab's RAM limitations.

Below is a summarized table of the internal deadlines.

TASK	PERSON	DEADLINE
Create Group	All	May 22
Project Proposal: Intro	Amelie	June 5
Project Proposal: Illustration	Hana	June 5
Project Proposal: Background	Hana	June 5
Project Proposal: Data Processing	Ji	June 5
Project Proposal: Architecture	Ji	June 5
Project Proposal: Baseline Model	Viv	June 5
Project Proposal: Ethical Considerations	Amelie	June 5
Project Proposal: Project Plan	Amelie	June 5
Project Proposal: Risk Register	Viv	June 5
Project Proposal Review and Submit	All	June 6
Project Progress: Intro	Ji	July 4
Project Progress: Responsibilities	Ji	July 4
Project Progress: Data Processing	Hana	July 4
Project Progress: Baseline Model	Viv	July 4
Project Progress: Primary Model	Amelie	July 4
Project Code	All	Ongoing
Project Progress Report	All	June 10
Project Presentation	All	August 10
Final Project Report	All	August 10

Table 1: Internal Deadlines of Project Proposal and Final Project

3 Data Processing

The data processing and splitting portion consisted of 4 sections: Receiving the input data, text preprocessing, data splitting, and data formatting.

The Google Colab notebook of the data processing can be found here:

`https://colab.research.google.com/drive/1sULFK7nwkcg_HvZY831L0iaxVmfOHXPE?usp=sharing`

3.1 Receiving the Data

The AI vs Human generated essay data is in the form of a csv file obtained from the daigt-v3-train-dataset from Kaggle (Darek Kleczek). Each CSV file contains 5 columns which are outlined below:

1. Essay in the form of a paragraph
2. Category label (0 for human, 1 for AI)
3. Prompt name

4. Source
5. Column denoting if the data entered is in the RDizzle3-seven competition

3.2 TEXT PREPROCESSING

There are two main data preprocessing steps that are required; extracting the important information and removing duplicate entries from the CSV files. Both steps were addressed within the same function: looping through all the entries in the CSV file and deleting any duplicate entries. As each entry is sorted through, a new CSV file is written that contains single instances of the essays and the corresponding category. It was determined that the remaining 3 CSV columns were unnecessary. The prompt column was removed since the model should be able to determine if it is AI or human regardless of the topic being written about. Since we already know from the category column whether the essay is AI or human generated the source of the essay is unnecessary and thus, was removed. Lastly, the model does not require knowing if our data is present in a competition or not, thus, it was removed.

3.3 SPLITTING DATA

The data set will be split up into 60% training, 20% validation and 20% testing data. A function is created that goes through the dataset and splits the required amount of data into a separate CSV file, one for each training, validation, and testing. By ensuring that the testing data is stored in a completely separate CSV file it can be assured that the model will not see the test data prior to the testing phase. Furthermore, the dataset contains 27,371 human generated essays and 37,962 AI generated entries, thus careful steps need to be taken to ensure that there is a balanced dataset. Thus, the splitting of the data will only include 27,371 AI generated essays to ensure a 50/50 split of AI vs human.

Once the data splitting was run, it was found that there is a total number of 32,846 training samples, 10,498 validation samples and 10,948 testing samples.

To allow for a more consistent format of data entering the model, each essay was taken and split into a list of the corresponding words and then entered into a list of the form:

[[[word1], [word2], . . .], category]], [[word1], [word2],...], [category]],...]

Punctuation and capitalization contains important information about the essays and thus it was kept rather than being removed to give the model as much information as possible. However, if this becomes an issue during training it can be easily removed from the data. An example of a processed essay can be seen in Figure 2.

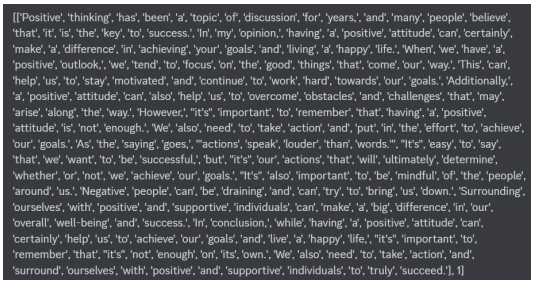

Figure 2: An example of how the text is split.

3.4 ADDITIONAL TESTING DATA

In addition to the testing data obtained from the original dataset, it was decided that more testing data should be obtained that was not used anywhere else to allow for a more accurate evaluation of the model. Getting a GPT model to produce these essays is one efficient way of getting AI generated samples. On the other hand, collecting data first-hand proved to be quite difficult since each essay is around 360 words total. The amount of time required to produce a substantial amount of human

generated essays to give an honest evaluation of the model proves to be impossible given the time constraints. Thus, a new dataset from Kaggle titled "AI Vs Human Text" can be used to provide further evaluation of the model (Gerami (2024)).

3.5 Challenges

One of the main challenges associated with the data processing section was determining which elements of the essays should be kept and which should be discarded. At first it was thought that punctuation should be removed as it does not provide much valuable information to the model. However, upon further considerations, this was not considered the case as punctuation serves to guide the reader and express important emotions within the essay.

4 Baseline Model

Google Colab Notebook Link: `https://colab.research.google.com/drive/1tgLEdtP1XONEFNAL67NICj9AJ6v7VU5y?usp=sharing`

For this project, a Random Forest Model will be used as a baseline model to compare to. Since this project evaluates pieces of text, using the Bag of Words model first will allow the model to work more effectively. The Bag of Words model enables the text to be interpreted numerically. For each data point, the model does a word frequency count, creating a list of features of limited length (max_features parameter). This data is then passed through the Random Forest Model. Within the model, it is passed through multiple trees, each one receiving a sample set of the data. This sample set is sampled using the bootstrapping method. This means it has the same number of data points as the original dataset with the possibility of repeats due to sampling with replacement. For validation and testing, each tree produces an output prediction, then the final prediction is obtained by finding the majority result (see Figure #). Both the Bag of Words and Random Forest models (Chidananda (2018)) were created using the Python module sklearn (see the full code in the Google Colab link at the beginning of this section).

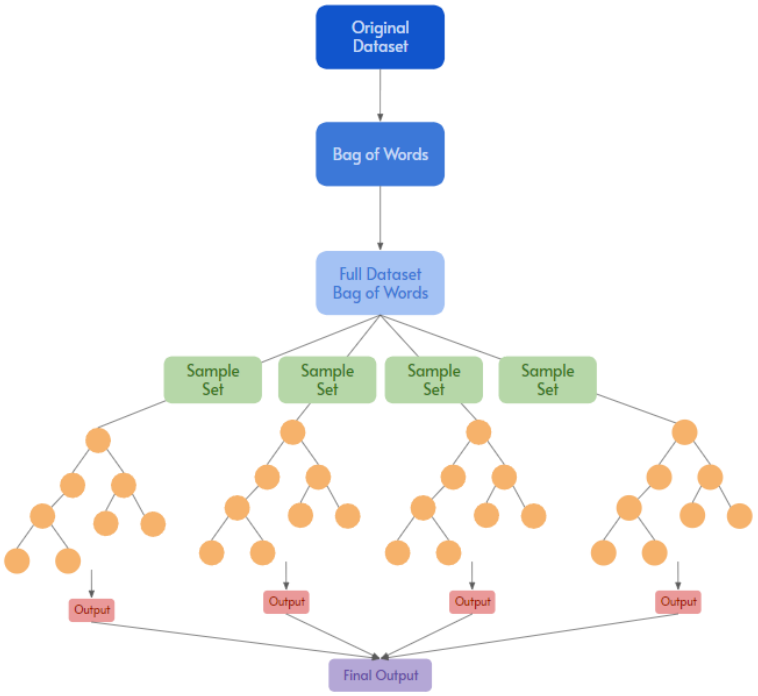

Figure 3: A simple diagram of the Random Forest Model with Bag of Words.

The first iteration of the baseline model used 5000 features, 500 trees and the rest of the parameters are set to the common values as determined through research (Probst et al. (2019)). For splitting, the Gini impurity rule was used. Additionally, since this is a classification problem, the minimum number of samples in a leaf node was set to 1. For the number of features at each split (max_features), it is determined to be most efficient at the square root of the total number of features. Finally bootstrapping is used for sampling, where N samples are drawn per tree where N is the same size as the original dataset.

This default Random Forest model resulted in an accuracy of 55% and an F1 score of 0.671(see Fig Figure 4 for the confusion matrix).

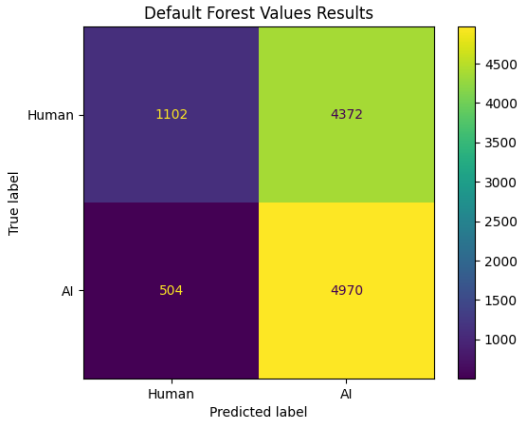

Figure 4: The confusion matrix after training the random forest model on the default values.

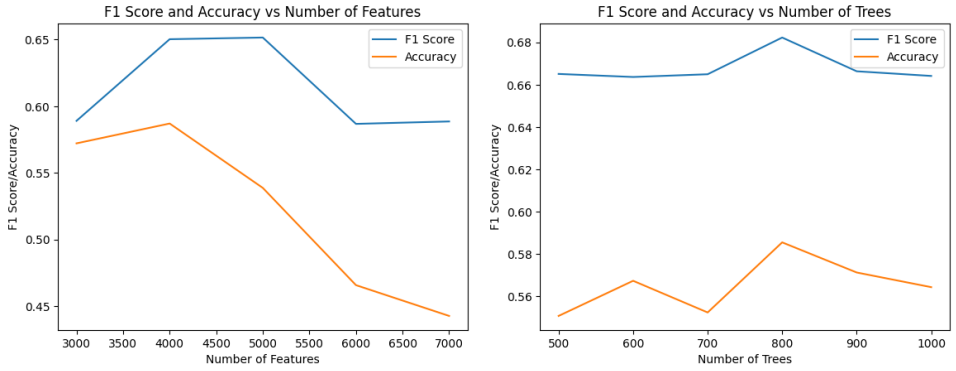

Figure 5: Tuning the hyperparameters, number of features, and number of trees. For the final model, it was decided that 4000 features and 800 trees would be used.

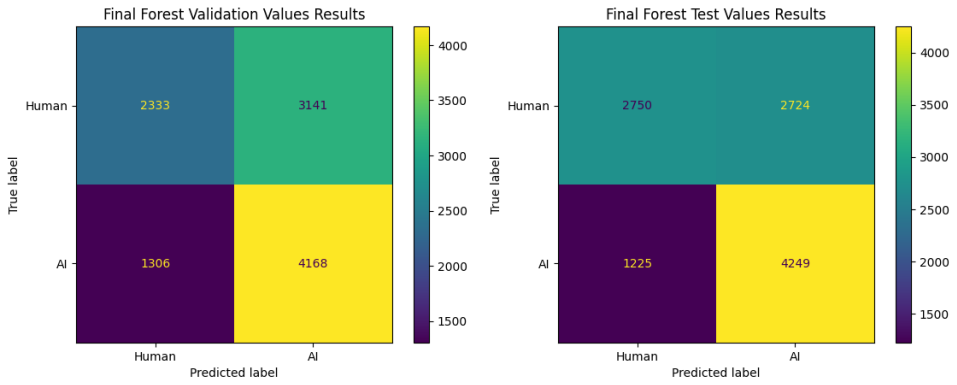

Figure 6: (Left) the confusion matrix for the validation data results on the final model. (Right) the confusion matrix for the testing data results on the final model.

In terms of challenges, the largest one was training time. Since the dataset is large, creating the Bag of Words followed by training took several minutes. As a result, the parameters had to be tuned slightly to allow the model to train within 10 minutes. Furthermore, the allocated resources of Google Collab also posed a challenge. At first, when the number of features was set to no limit, the RAM usage was exceeded quickly and the model could not complete training. After some trial and error, setting 5000 features was chosen since it did not max out the resources, allowing for proper hyperparameter tuning later.

The chosen hyperparameters to tune were the number of features and the number of trees. For the number of features, values in increments of 1000 were chosen from 3000 to 70000. This maximized the use of computational resources while keeping runtime for each iteration at a reasonable length ($<$10mins). As demonstrated in Figure 5, 4000 features maximized the accuracy and F1 score of the model. Similarly, with the number of trees, values in increments of 100 were chosen from 500 to 10000. This also maximizes the computational resources while exploring within a reasonable range as determined by research (Probst et al. (2019)). In Figure 5 it is found that 800 features maximizes the accuracy and F1 score of the model.

Finally, after tuning the hyperparameters to 800 trees and 4000 features, the final test accuracy was 64% and F1 score of 0.693. The validation and testing confusion matrices for the final model are presented in Figure 6.

5 Primary Model

The primary model that will be used in order to classify a body of text as written by AI or by a human will be a recurrent neural network, or RNN. As stated in our project proposal, RNNs are very effective with sequential data like text as they are good at identifying contextual links between words (geeksforgeeks).

5.1 RNN Model and Layer Description

The first layer in an RNN model creates a numerical representation of the text data using a sequence of token indices. The encoding layer then converts this to a vector that the neural network can more easily process, with similar words having similar vectors. A sequential model is then used to create the neural network layer by layer, starting with the output of the encoding layer above. For our model, an embedding layer will then be used to reduce and standardize the size of the vectors (Saxena). This will be followed by a bidirectional long short term memory (Bi-LSTM) layer. LSTMs selectively retain and forget information over time in order to capture past context of the input sequence and resolve the vanishing gradient issue (Anishnama). Bi-LSTMs process neural networks in forward and backward directions and can capture past and future context of the input sequence. Following this there will be two dense fully connected layers. The first dense layer will

utilize a rectified linear unit (ReLU) activation function. The following dense layer will utilize a sigmoid activation function finally outputting the probability of the text data beingAI generated.

Binary cross entropy will be used in order to calculate error of the model since this is a binary classification model. In order to compile the data, the adaptive moment estimation (Adam) optimizer will be used to update each weight at its own rate.

Before training all of the text data, a small sample of the training data will be passed through the RNN model in order to ensure the model is efficient and effective. Additionally, this checks that the model can overfit the data and therefore is able to learn.

The steps outlined above to define the RNN model are summarized below using pseudocode. It is assumed that data processing has already been completed at this point.

5.2 PSEUDOCODE

This program will classify a body of text as written by AI or by a human.

Import tensorflow

Import numpy

TextVectorization

Sequential(

    TextVectorization,

    Embedding,

    Bidirectional(LSTM),

    Dense(ReLU),

    Dense(Sigmoid)

)

Loss = BinaryCrossEntropy

Optimizer = Adam

5.3 CHALLENGES

Major challenges that may come from our primary model is the time it takes to train the model as well as RAM limitations implemented by Google Colab. These will need to be taken into consideration in order to ensure we have the time to train our model and make any adjustments deemed necessary while still ensuring that our model is accurate and effective.

6 LINK TO GITHUB OR COLAB NOTEBOOK

Google Colab Notebook Links:

`https://colab.research.google.com/drive/1sULFK7nwkcg_HvZY831L0iaxVmfOHXPE?usp=sharing`

`https://colab.research.google.com/drive/1tgLEdtP1XONEFNAL67NICj9AJ6v7VU5y?usp=sharing`

The link to the Github: `https://github.com/ji24077/Cerberus-Detection`

REFERENCES

Anishnama. Understanding bidirectional lstm for sequential data processing — anishnama20. https://medium.com/@anishnama20/understanding-bidirectional-lstm-for-sequential-data-processing-b83d6283befc. [Accessed 05-07-2024].

Rajath Chidananda. Bag of words meets bags of popcorn. https://www.kaggle.com/datasets/rajathmc/bag-of-words-meets-bags-of-popcorn-, 2018. [Accessed 05-07-2024].

month=Dec Darek Kleczek, year=2023. Daigt-v3-train-dataset. URL https://www.kaggle.com/datasets/thedrcat/daigt-v3-train-dataset?resource=download.

geeksforgeeks. Rnn for text classifications in nlp - geeksforgeeks — geeksforgeeks.org. https://www.geeksforgeeks.org/rnn-for-text-classifications-in-nlp/. [Accessed 05-07-2024].

Shayan Gerami. Ai vs human text, Jan 2024. URL https://www.kaggle.com/datasets/shanegerami/ai-vs-human-text.

Prabha Kannan. How much research is being written by large language models? https://hai.stanford.edu/news/how-much-research-being-written-large-language-models, 2024. [Accessed 07-06-2024].

Sambit Mahapatra. Why deep learning over traditional machine learning? — towardsdatascience.com. https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063, 2018. [Accessed 05-07-2024].

Philipp Probst, Marvin N Wright, and Anne-Laure Boulesteix. Hyperparameters and tuning strategies for random forest. *Wiley Interdisciplinary Reviews: data mining and knowledge discovery*, 9 (3):e1301, 2019.

Sawan Saxena. Understanding embedding layer in keras — medium.com. https://medium.com/analytics-vidhya/understanding-embedding-layer-in-keras-bbe3ff1327ce. [Accessed 05-07-2024].

Nate Woodall. Why it's getting harder to tell ai-generated images from the real deal online. https://www.abc.net.au/news/2024-04-27/artificial-intelligence-ai-faces-fake-images-social-media/103627436, 2024. [Accessed 07-06-2024].