Peyton Skwarczynski

CSI 4180 – Natural Language Processing

Dr. Steven Wilson

# NLP Homework 4: NLP App

My NLP APP: AI Generated Text Detection

## 1. Introduction

My NLP App, AI Generated Text Detection, provides a way for users to compare the results of using multiple different models, both finetuned and base models, for AI generated text detection. The main purpose of the application itself is to provide a simple and easy-to-use graphical user interface for the user to make input decisions, select examples, and display the classification results. Outside of what is used for the GUI, there is very little code within the Hugging Face Space used to classify the input text. The majority of this classification functionality comes from the models I finetuned on various datasets containing AI generated text. I used multiple well-known base models, finetuned them on the datasets using Google Colab, pushed them to a model repository hosted on Hugging Face, and finally called the finetuned models within my application to classify an input text. AI Generated Text Detection contains a total 10 different finetuned and base models that can be used for classification. This allows users to see the differences in accuracy across the various models and datasets used for finetuning.

## 2. Usage

The AI Generated Text Detection app consists of an input section on the left side of the page, an output section on the right side of the page, and examples on the bottom of the page. In order to use the application, the user is required to provide 3 different fields of input: the model, the dataset the model is finetuned on, and the input text the finetuned model will be classifying. These inputs will then be used to print the output on the right side of the page.

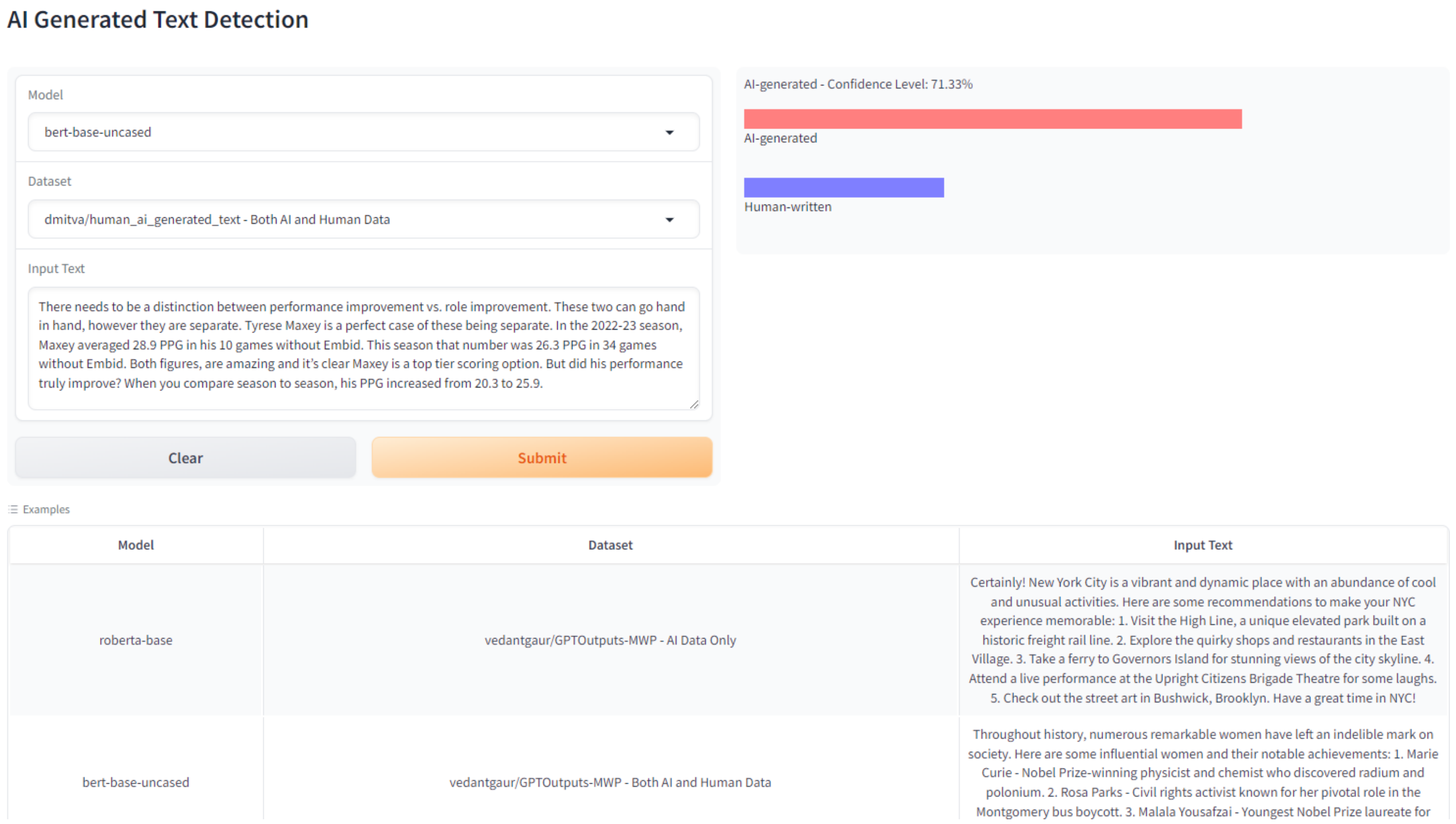
The model and dataset fields within the input section are drop boxes that allow users to select whichever model and dataset they would like to use. My application uses 2 different models, bert-base-uncased and roberta-base, each of which were finetuned on 5 different datasets: no dataset finetuning, vedantgaur/GPTOutputs-MWP AI data only, vedantgaur/GPTOutputs-MWP human data only, vedantgaur/GPTOutputs-MWP both AI and human data, and dmitva/human_ai_generated_text both AI and human data. By having a collection of 2 models and 5 different finetuning datasets, my application provides a total 10 different combinations, or 10 different models to classify the input text. It is important to note that 2 of these examples, using either model with no dataset finetuning, are not models that I created. These input selections simply call the base model on Hugging Face without any finetuning at all.

The input text field within the input section is a box in which the user can write or copy-and-paste any form of text. Whatever text is put within this box will be passed through the finetuned model to undergo classification. The selected finetuned model will take this input parameter and classify the text as either AI generated or human written. It should also be noted that extremely small forms of input text do not serve as the best inputs for sequence classification, as it can be undecipherable by the model whether or not the text is AI generated.

After all input selections are filled out, the user can press the “Submit” button at the bottom of the input section. This will take each of the user inputs, call the selected finetuned model, and pass the input text to the finetuned model for sequence classification. The result of this classification is then displayed in the outputs section. On the right side of the page, not only will there be a written explanation of the sequence classification, but there will also be a visual representation. For this, the results will form two bars that are colored to the confidence percentage of the classification results, where each of the two bars represent a different output classification. Additionally, if the user would like to clear both the input and output results, they can select the “Clear” button in the input section of the page.

The examples section at the bottom of the page provide 4 different examples of potential user input to demonstrate the application and sequence classification. Within the Hugging Face Space repository, I hard-coded 4 different examples of AI generated and human written text that can be used within the classification process. Each of these texts are then paired up, at runtime, with a random choice of model and dataset to fill out the other two input fields. If the user clicks on one of these examples, the model, dataset, and text will all be automatically copied over to each of the input fields.

Example Input/Output Using Self-Input:



AI Generated Text Detection

Model: bert-base-uncased

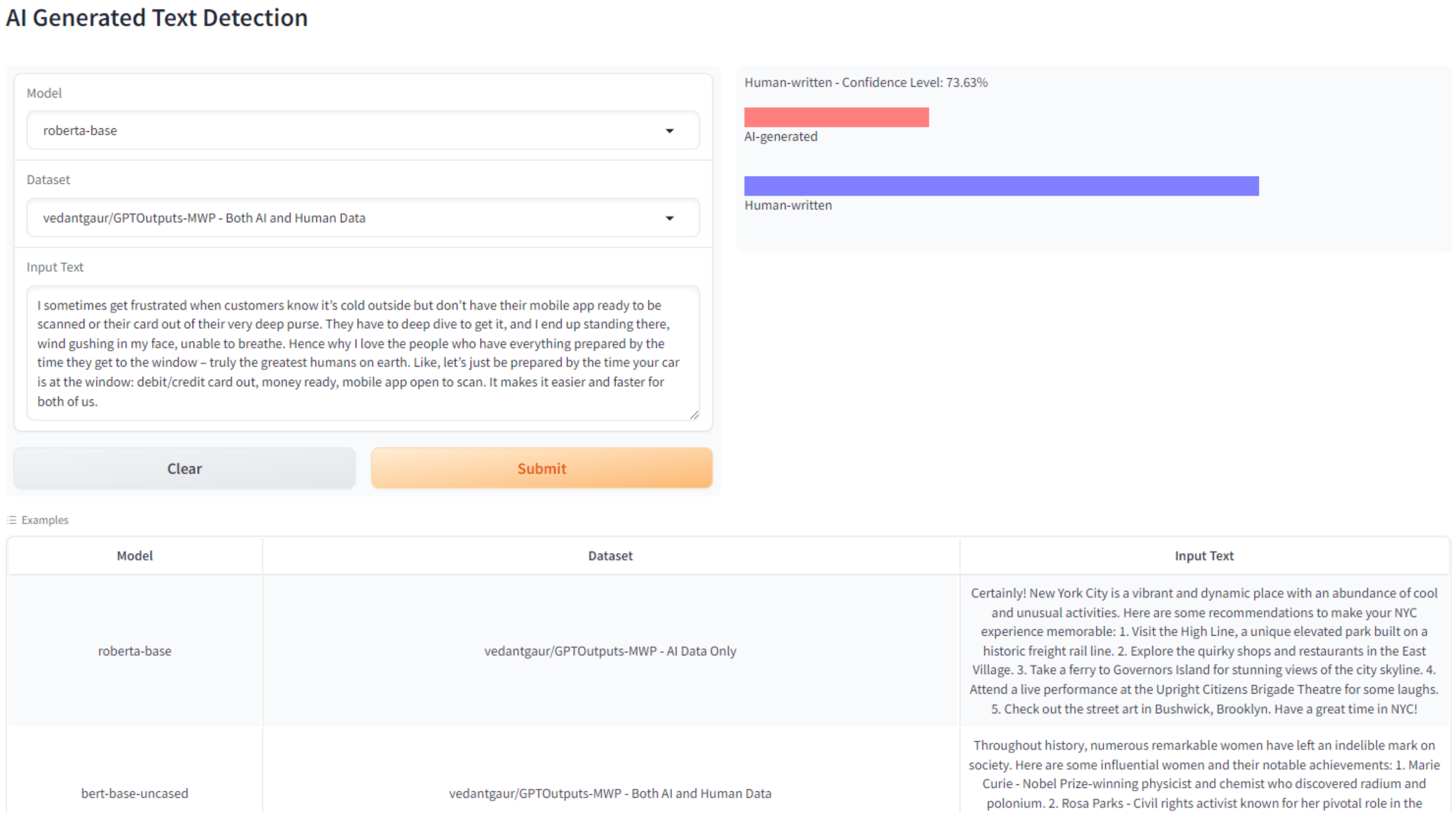
Dataset: dmitva/human_ai_generated_text - Both AI and Human Data

Input Text: There needs to be a distinction between performance improvement vs. role improvement. These two can go hand in hand, however they are separate. Tyrese Maxey is a perfect case of these being separate. In the 2022-23 season, Maxey averaged 28.9 PPG in his 10 games without Embid. This season that number was 26.3 PPG in 34 games without Embid. Both figures, are amazing and it's clear Maxey is a top tier scoring option. But did his performance truly improve? When you compare season to season, his PPG increased from 20.3 to 25.9.

| Model | Dataset | Input Text |
|---|---|---|
| roberta-base | vedantgaur/GPTOutputs-MWP - AI Data Only | Certainly! New York City is a vibrant and dynamic place with an abundance of cool and unusual activities. Here are some recommendations to make your NYC experience memorable: 1. Visit the High Line, a unique elevated park built on a historic freight rail line. 2. Explore the quirky shops and restaurants in the East Village. 3. Take a ferry to Governors Island for stunning views of the city skyline. 4. Attend a live performance at the Upright Citizens Brigade Theatre for some laughs. 5. Check out the street art in Bushwick, Brooklyn. Have a great time in NYC! |
| bert-base-uncased | vedantgaur/GPTOutputs-MWP - Both AI and Human Data | Throughout history, numerous remarkable women have left an indelible mark on society. Here are some influential women and their notable achievements: 1. Marie Curie - Nobel Prize-winning physicist and chemist who discovered radium and polonium. 2. Rosa Parks - Civil rights activist known for her pivotal role in the Montgomery bus boycott. 3. Malala Yousafzai - Youngest Nobel Prize laureate for |

In this example, the selected model classifies the input text as AI generated, but only with 71.33% confidence level. I found it to be important to include the confidence level, especially with visual representation, as it gives a better explanation of the thought process behind the classification. Specifically, this example shows that the model only has a somewhat strong belief that the AI generated text classification is correct.

Example Input/ Output Using Self-Input:

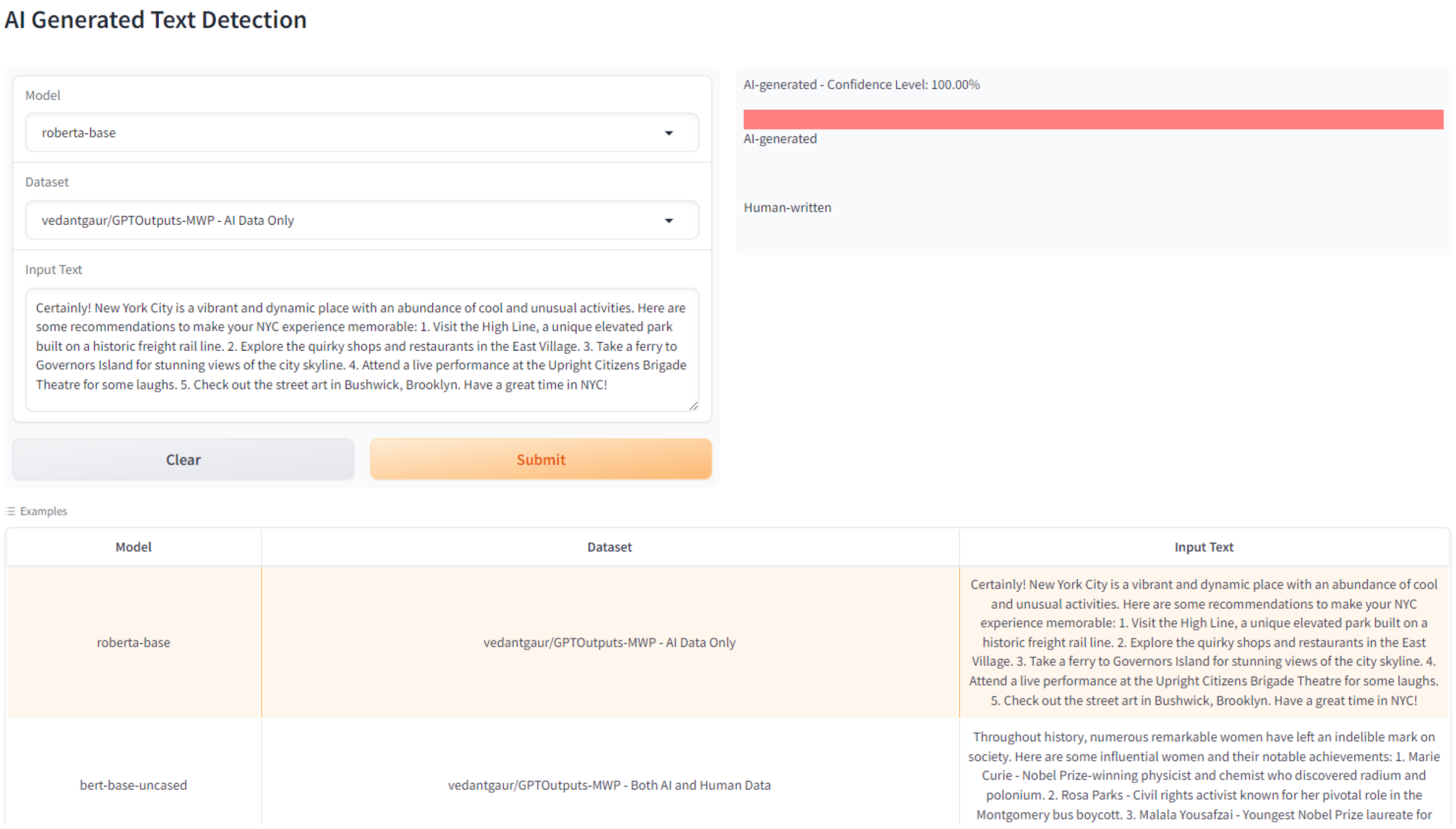

AI Generated Text Detection
Model
roberta-base
Dataset
vedantgaur/GPTOutputs-MWP - Both AI and Human Data
Input Text
Clear
Submit
Human-written - Confidence Level: 73.63%
AI-generated
Human-written
Examples

I sometimes get frustrated when customers know it's cold outside but don't have their mobile app ready to be scanned or their card out of their very deep purse. They have to deep dive to get it, and I end up standing there, wind gushing in my face, unable to breathe. Hence why I love the people who have everything prepared by the time they get to the window – truly the greatest humans on earth. Like, let's just be prepared by the time your car is at the window: debit/credit card out, money ready, mobile app open to scan. It makes it easier and faster for both of us.

| Model | Dataset | Input Text |
|---|---|---|
| roberta-base | vedantgaur/GPTOutputs-MWP - AI Data Only | Certainly! New York City is a vibrant and dynamic place with an abundance of cool and unusual activities. Here are some recommendations to make your NYC experience memorable: 1. Visit the High Line, a unique elevated park built on a historic freight rail line. 2. Explore the quirky shops and restaurants in the East Village. 3. Take a ferry to Governors Island for stunning views of the city skyline. 4. Attend a live performance at the Upright Citizens Brigade Theatre for some laughs. 5. Check out the street art in Bushwick, Brooklyn. Have a great time in NYC! |
| bert-base-uncased | vedantgaur/GPTOutputs-MWP - Both AI and Human Data | Throughout history, numerous remarkable women have left an indelible mark on society. Here are some influential women and their notable achievements: 1. Marie Curie - Nobel Prize-winning physicist and chemist who discovered radium and polonium. 2. Rosa Parks - Civil rights activist known for her pivotal role in the |

In this example, the chosen model is classifying the input text as human written. The model also contains 73.63% confidence that this classification is correct.

Example Input/Output Using Provided Examples:


AI Generated Text Detection

Model: roberta-base

Dataset: vedantgaur/GPTOutputs-MWP - AI Data Only

Input Text: Certainly! New York City is a vibrant and dynamic place with an abundance of cool and unusual activities. Here are some recommendations to make your NYC experience memorable: 1. Visit the High Line, a unique elevated park built on a historic freight rail line. 2. Explore the quirky shops and restaurants in the East Village. 3. Take a ferry to Governors Island for stunning views of the city skyline. 4. Attend a live performance at the Upright Citizens Brigade Theatre for some laughs. 5. Check out the street art in Bushwick, Brooklyn. Have a great time in NYC!

AI-generated - Confidence Level: 100.00%

| Model | Dataset | Input Text |
|---|---|---|
| roberta-base | vedantgaur/GPTOutputs-MWP - AI Data Only | Certainly! New York City is a vibrant and dynamic place with an abundance of cool and unusual activities. Here are some recommendations to make your NYC experience memorable: 1. Visit the High Line, a unique elevated park built on a historic freight rail line. 2. Explore the quirky shops and restaurants in the East Village. 3. Take a ferry to Governors Island for stunning views of the city skyline. 4. Attend a live performance at the Upright Citizens Brigade Theatre for some laughs. 5. Check out the street art in Bushwick, Brooklyn. Have a great time in NYC! |
| bert-base-uncased | vedantgaur/GPTOutputs-MWP - Both AI and Human Data | Throughout history, numerous remarkable women have left an indelible mark on society. Here are some influential women and their notable achievements: 1. Marie Curie - Nobel Prize-winning physicist and chemist who discovered radium and polonium. 2. Rosa Parks - Civil rights activist known for her pivotal role in the Montgomery bus boycott. 3. Malala Yousafzai - Youngest Nobel Prize laureate for |

In this example, the selected prompt shows a 100.00% confidence that the input text is AI-generated.

3. Documentation

The core components of my application are the finetuning of different models within numerous Google Colab notebooks, and then the classification procedure within the Hugging Face Space. Since I have previously completed the model finetuning on all the various combinations of models and datasets, there is no actual finetuning during the execution of the application. Instead, the flow from user input to output goes as follows:

1. The user inputs the model, dataset to finetune the model, and the input text
2. The chosen model and dataset are mapped to load in a specific Hugging Face model repository that is already storing the specific finetuned version of the model
3. Input text is then passed through the specific finetuned model for classification
4. Classification results undergo reformatting to best visualize data
5. Results are output to the user within the application

Even though the process of calling a model, passing input text, and reformatting the output for display within a GUI does not seem very difficult, the other major portion of the creation of this application was finetuning the models on the various datasets. One of the major problems with model finetuning was the available datasets on this topic. After doing some research on available Hugging Face data, there was not a single dataset that contained classification information for both AI generated and human written text mixed within a single column, as seen before in previous homework assignments. Instead, there were multiple datasets that were formatted in a way that contained fields for both AI generated and human written text in separate columns. Additionally, instead of having a separate column for the labels of the text, each column was either entirely AI generated or human written text. Compared to the datasets that we used in previous homework assignments, the ones I found contained a completely different organization from what I needed. I ended up doing a lot of troubleshooting within the training process to not only get the data configured in a way in which the finetuning process could occur, but also in a way that optimized performance, while not sacrificing accuracy of the training. Additionally, after successfully completing the first few model's finetuning, I realized a major issue regarding the accuracy of the updated model, requiring me to revamp my finetuning techniques to improve the model accuracy. I will go further into these troubleshooting techniques and the limitations I faced in the Limitations section of this user guide.

Within my application, I used a total of two different models, bert-base-cased and roberta-base, and two different datasets to finetune the models, vedantgaur/GPTOutputs-MWP and dmitva/human_ai_generated_text. Due to the conflicts mentioned above, I initially found it quite difficult to conceptualize how to undergo finetuning. Even though this ended up not being very accurate for sequence classification, I first began by up splitting these datasets up into separate portions of AI generated and human text and used these separate sections for individual finetuning of the models. By allowing the user this additional option of only training on a subset of the total data, I was able to get a larger number total finetuned models present within my application, even if some of these models are not very accurate.

It is very important to understand the characteristics of both the models and datasets you are working with. Descriptions of each of the models and datasets used within this application follow:

Models-

bert-base-uncased:

BERT, or Bidirectional Encoder Representation from Transformers, is a transformer -based machine learning model created and published by Google. This specific model, bert-base-uncased, is the base uncased variation of BERT, which is just one out of many different variations. The model is designed to pre-train bidirectionally from text by jointly conditioning from both sides of the text in all layers. This functionality allows BERT to be an extremely flexible model that can be easily finetuned by simply adding an additional output layer. Bert-base-uncased is a 12-layer, 768-hidden, 12-headed, 110 million parameter architecture. The model is trained on Wikipedia and BooksCorpus using masked language modeling and sentence prediction. Due to the size of the model, bert-base-uncased is a computationally expensive architecture. For reference, the original BERT model was trained on around 64 TPUs for 4 days straight. It was due to this computational complexity that I was forced to purchase Google Colab Pro to increase my compute units. When trying to finetune the large model on the base version of Colab, I kept getting kicked off of runtime due to my code requiring too many resources. Some of the known limitations of BERT include struggling with tasks that involve a very large sense of knowledge, including common human sense. Additionally, BERT models are also quite sensitive to input restructuring. This means that different outputs can occur even when only slightly altering the original input.

roberta-base:

RoBERTa, or Robustly Optimized BERT Pretraining Approach, is one of the many BERT variants. Created and published by Facebook, RoBERTa builds on the classic BERT model by adjusting parameters, removing the next-sentence prediction object previously held by BERT, and also training using large mini batch rates and various learning rates as well. As for the architecture of roberta-base, it is a 12-layer, 768-hidden, 12-headed, 125 million parameter model. Regarding the training procedure, RoBERTa used the same exact data as BERT, but with larger batch sizes and byte pair encoding. Facebook trained this model using 1024 V100 GPUs. Since RoBERTa is a larger model and also requires the use of larger batch sizes than BERT, RoBERTa is more computationally expensive than BERT. After researching this, my purchase of Google Colab Pro was even more justified. It would have been extremely time consuming to finetune each of these large models, if I could even be allocated runtime by Colab. Finally, RoBERTa contains very similar limitations to BERT. RoBERTa also can tend to struggle with common sense and large world knowledge and is also very sensitive to input restructuring.

Datasets-

vedantgaur/GPTOutputs-MWP:

The vedantgaur/GPTOutputs-MWP dataset was the very first dataset I ended up attempting to finetune the models on. Since the dataset contains no readme card, there is absolutely no information regarding where the data was gathered from. The only provided information is that the dataset contains both a train and validation split, each with a Prompts (human text), Outputs (AI generated text), prompt_length, and output_length field. Between the two splits, there is a total of around 3.75 thousand rows of data. When initially finetuning the bert-base-uncased model on this dataset, I thought to separate the Prompts and Outputs column and train the model individually. I thought that maybe the model could gather enough information to know exactly what either AI generated or human written text would look like, so that way the model would never need to be trained on both fields together. After running a couple of tests and re-finetuning the model on either portion of the dataset, it was clear that only training the model on a single classification causes two grave problems. First of all, the main problem with finetuning in this manner is that since the model will only ever contain knowledge about a particular classification, there is no possible way for the model to classify an input text as anything outside of what it was originally trained for. Secondly, this one-sided finetuning caused overfitting of the model. Since the model would only ever be trained on a particular classification, the model would start to memorize what inputs are classified as what, rather than gaining the ability to process input text that was not originally included in the finetuning data. Collectively, these two issues made me realize the need for finetuning on both classifications, and in this case, both columns of the model. After hours of troubleshooting, I figured out a way to concatenate the columns, shuffle them to provide an even mix of each classification, and then undergo finetuning of the dataset. This allowed for a finetuned model that was not overfit and produced significantly higher accuracy.

dmitva/human_ai_generated_text:

The dmitva/human_ai_generated_text dataset was the second dataset I ended up using within my application. This dataset contains a single train split of 1 million rows, with fields for human_text, ai_text, and a task given to both the AI and human responding to the prompts. Unfortunately, this dataset was structured in the same exact organization as the previous dataset, where instead of there being a single column with a mix of both classifications, there are two different categories, one for each classification. On the other hand, similar to the previous dataset, this data also contains absolutely no further information regarding where or how this information was gathered. After going through countless errors and troubleshooting regarding the previous model, I already had a solid idea of how to tackle this structuring concern. Unlike the finetuning with the other dataset, I decided not to include separate finetuned models for AI generated and human written text. Since those models contained accuracy and overfitting errors, there was no point in using that strategy with this dataset. If I were to do the same, these new finetuned models would also contain detrimental accuracy and overfitting errors. As a result, I decided to only finetune the models when utilizing both classifications found within this dataset. After testing these finetuned models, they displayed slightly different results compared to the previous finetuning on both classifications, but still showed very high accuracy.

## 4. Contributions

My personal contributions to this assignment were large. The basis of my application was to create a Hugging Face Space that allows users to actively compare the results of various AI generated text detection models. In order to carry out this goal, I not only needed to implement a navigable and visually appealing GUI with a way to pass the user inputs through to the model, but I also needed to create all of the finetuned models I was going to use within this application. The only part of my code that was somewhat carryover from previous assignments was the base template of how to finetune models on a dataset. This process, however, had to be significantly updated from my work on previous homework due to the datasets and models used within the finetuning process. As mentioned previously, the datasets I used caused plenty of issues regarding the general formatting of the data. This caused me to implement numerous methods of data concatenation and shuffling in order to gather the best possible collection of data to finetune the models on. Additionally, I had to continuously reconfigure the trainer objects and overall finetuning process to adjust for the computationally expensive models I was implementing within my application. In order to increase training speeds to the perfect amount, to maximize efficiency but minimize loss of accuracy, I had to research different ways of modifying the training process. Some of the features I looked into included data augmentation, different batch sizes, learning rates, different epoch sizes, and weight decay. For some of the model finetuning, these training enhancements were not necessary, but for others, these modifications were essential to increase accuracy and efficiency of the finetuning process.

## 5. Limitations

One of the major limitations within my application comes with the finetuned models that are only trained on one classification, either AI generated or human written text. As previously mentioned, since these models were only ever finetuned on data containing a single classification, they are unable to accurately classify an input text as one of the two labels. This lack of knowledge from the model causes the output classification to always be a high confidence for the label the model was trained on. Since the model has no knowledge of text outside the label it was trained on, it cannot predict an external classification. Even though this format of training seemed to be beneficial in optimizing the time needed for finetuning, it ended up being detrimental to the overall accuracy of the model. After coming across this error when testing my models finetuned on the first dataset, I shifted away from utilizing this form of training during model finetuning to the second dataset. Since training on only a single form of classification does not provide the model with enough information to accurately classify an input text as either AI generated or human written, there was no point in creating more of these finetuned models.

Another side limitation that should be noted, although not detrimental to the overall scope of the application, is that since these models are simply finetuned on a single dataset, they cannot be fully guaranteed to output the correct classification. Instead, the model is only able to output the classification it believes the input text to be, based off of the data the model has been trained on. It is essential to realize that this is the same as any other AI text detection technology. There is no fully guaranteed way to know whether or not an input text was generated by AI; this application only provides a highly accurate method of doing so.