Grad research paper

Artificial Intelligence has changed substantially over the past year. The Large Language Model (LLM) has taken over the public consciousness, with public facing chatbot programs finally arriving. It feels like the entire world is racing to figure out what a model like this cannot be used for. One particular use case that has become common, is students using the chatbot function for help on homework and tests at all grade levels, with improved models even being able to pass standardized tests like the Bar exam all on their own. As academic institutions come to terms with the new landscape of tools that students can use, some are adapting quickly with methods ranging from allowing the use of chatbots to getting rid of homework entirely. Others are relying on an unproven technology. The technology in question is itself a statistical model, which in theory can detect the difference between content generated by a chatbot, and content created by a human. However, these tools have been shown to work poorly or not all, leaving teachers without reliable methods to know if students are benefitting from ChatGPT or similar services.

Just after setting out to research this topic, OpenAI, the company behind ChatGPT published their own AI detection tool which they titled ‘AI Text Classifier’ in a lengthy blog post with a link to then beta version of the tool. Text can be pasted into a box, the classification run, and the model will return a classification, from ‘very unlikely’ to ‘likely’ to be AI-generated. In the blog post detailing the tool, OpenAI states that their tool “…correctly identifies 26% of AI-written text as ‘likely AI-written,’”. So what does this all mean? Essentially, detecting AI generated content is a task difficult, even the makers of ChatGPT have yet to solve it, and thus it is unlikely that we will solve this task either.

This paper is not an attempt at a solution to AI detection, consistently spotting AI generated text is a complex task that the creators of ChatGPT themselves have yet to complete. This project merely serves as an exploration into whether content that resembles human writing can be generated systematically and different methods of doing so, the characteristics of AI generated content, and creating models to compare current results from other research projects on this topic. After finding an appropriate dataset of articles and blog posts, we used those articles to generate text from ChatGPT, giving the bot the role of students of different grade levels. These texts are then run through different classification models, looking at human against AI writing, but also the different grade levels of AI writing compared against each other. Finally, we will examine the results and discuss possible ways to expand upon this work in the future. Existing tools for AI detection may not be perfect, but perhaps we can contribute something to the conversation.

**LLMs**

What exactly is an LLM? Simply put, it is a statistical model that takes a string of text as input, and outputs text that it predicts is desired based upon the input. The LLM is a tool used in the AI subfield of Natural Language Processing (NLP), which deals with analyzing and predicting text data from all different types of sources. The “Language” part of the LLM is straightforward enough, but the “Large” part is where these models start to shine. GPT 3, which was the most well-known of these models we when started this project, used an unfathomable amount of data as its training data. Approximately everything that had been put on the internet up until June of 2021 was downloaded, then arranged in such a way that all that data could be used for model creation, resulting in over 175 billion parameters. We can think of parameters as columns on a spreadsheet, with each row being an observation or training example.

LLMs are not exactly new though, in 1954 the Georgetown-IBM experiment successfully translated 60 Russian sentences into English in the first well-known public demonstration of this type of technology. Work on this topic stalled for some time, as computing power took time to catch up to the theoretical work of mathematicians, until 2001 when Bengio et al published their paper “A Neural Probabilistic Language Model” which outlined using neural networks to process large amounts of text much faster than had previously been accomplished, but they were still hampered by technological restrictions.

That all changed in 2017 when Google researchers published their paper titled “Attention is All You Need”, which introduced the Transformer model. Prior to Google’s paper, neural networks like the recurrent neural network, the convolutional neural network, and the long short-term memory network, had vastly improved over the 1990’s and 2000’s, but they were still expensive to train in terms of computing time and power required. The transformer model was a new type of neural network architecture that allowed for a model to be trained in parallel using many computers which could be equipped with millions of dollars of special processing units designed specifically for AI tasks. For reference, the GPT in ChatGPT stands for “Generative Pretrained Transformer”. Researchers were, practically overnight, now able to train a model with billions of parameters in a fraction of the time it would take to train a “traditional” neural network, and that is where we find ourselves now.

ChatGPT is the fastest growing web app in human history, reaching 100 million users just 2 months after its public launch. Microsoft, their primary investor, had put hundreds of millions into the infrastructure to train the model behind the chatbot, GPT 3, but after the meteoric rise of GPT 3, they raised their investment by ten billion in January of 2023. Whether we like it, or are ready for it, this technology will shortly be inescapable. OpenAI has already started licensing their latest model, GPT 4, to companies all over the world who want to integrate AI services into their own products and has begun to offer a subscription service that grants users premium access to the chatbot features. ChatGPT was even used to write some portions of code for this project, in addition to our usage of the API to generate AI written text for our own model building.

In addition to OpenAI’s own classification tool, there are many prominent products that claim to be able to spot AI generated content. ZeroGPT and GPTZero (we assure you these are two different services) are two of the most notable, and though these tools appear to have improved substantially since the beginning of this project (a recent a viral reddit post showed ZeroGPT predicting that the US Constitution was written by an AI), they remain incomplete and suffer from the same problems as OpenAI’s tool. A single typo anywhere in a text, translating to a foreign language, having ChatGPT write in a foreign language and then using an online translator to go back to English, and even telling ChatGPT to write like a human, can cause the detector to classify the document as likely human written. One possible solution that OpenAI has publicly stated they are working on is a digital watermark, which forces their model to output text with specific characteristics and patterns that make it easily detectable by tools which are trained to find that watermark.

**Experiments**

The genesis of this project was before many of the AI detection tools had been published, so we had started with the goal of creating a model that could accurately detect the difference between AI and human generated content. Between finding a dataset of 190,000 human written articles and blog post from the website Medium.com and beginning to use the OpenAI API for their GPT models to generate AI content, we believed we could at least make a competent model. However, shortly after embarking on this journey, OpenAI published their detection tool, and the results were striking. As previously mentioned, their tool correctly classifies AI text 26% of the time, but the success rate for human written text is 91% according to their post about the model. Seeing that AI written text is difficult to classify but human text is relatively easy, and operating under the assumption that any results we could achieve would be below what OpenAI did, we decided to craft some additional experiments.

Seeing posts on social media where AI detection models were fooled when ChatGPT was instructed to “write like human”, this is where we had the idea to generate AI content at different grade levels. We decided to craft a prompt where different grade levels could be easily and systematically inserted, along with other necessary information, and this prompt could be fed to ChatGPT through the python API. These AI written texts could then be used as data for model creation. That said, we still wanted to compare AI vs. human text to see what sort of results we could achieve.

**Data**

In order to generate content though, ChatGPT needs a specific prompt. We had the framework but needed to engineer a specific method. This is where the dataset of Medium.com articles became useful. Through some exploratory analysis we found that there was a large distribution of word counts in the articles. ChatGPT whether through the chatbot interface or through the python API has a limit to how many tokens, or prominent word segments, can be read into the program via a prompt or output in a response, so we had to filter the dataset by word count. Ultimately, we chose to select only articles with between 500 and 1050 words.

Chart, histogram

Description automatically generated

This gave us just over 65,000 articles. The word count was also important as we knew that for creating our model, we wanted to do paired samples. While the article title would be the basis for the content, we decided to give the word count in the prompt, as having articles of similar lengths would eliminate a potential flag. For example, if ChatGPT thinks that 6th graders write shorter texts than college students do, than our model might easily pick up on that, and be able to classify too easily. Additionally, if we wanted to pair our AI samples with our human generated text, word counts should be similar in that regard so that when the human vs. AI model was created, obvious flags like word count would still be accounted for.

Next, we decided to find out the topics of each article. Fortunately, the dataset included tags for each article, as writers on Medium.com use these in order to help people find their articles. We decided to generate content under the role of different age students, so we decided to select topics that were “appropriate” or common for US schools. While this is subjective, the variety of the Medium.com dataset was far too wide for an average student to write about. The top 20 tags shown in the below graph demonstrate this. Sixth graders are not typically writing about data science, the blockchain, or startups.

We ended up choosing six topics: History, Statistics, American History, Geometry, African History, and Literature, and after selecting topics we were left with 1188 articles. From there we randomly selected 50 articles, because we knew we faced limits on total task size within our computational budget. We then constructed prompts based on article title and student grade level, such as:

“Write an approximately 700 word essay like a 10th grade student based on the following title: Capital Cities of Asian Countries“

Then we inserted the API prompt template into a loop and ran the loop for each article and each student level, where we chose 6th grade, 10th grade, and college. The OpenAI API is very straightforward and gave very few problems while working with it. In terms of text generation tasks, it seems to be an invaluable tool, allowing complex systems to be set up quickly and massive amounts of text to be generated with ease. The main section of the code was very short.

message = {"role": "user", "content": f"Write an approximately {word\_count} word essay like a {student} student based on the following title: {title}"}

completion = openai.ChatCompletion.create(

model="gpt-3.5-turbo",

messages=[message],

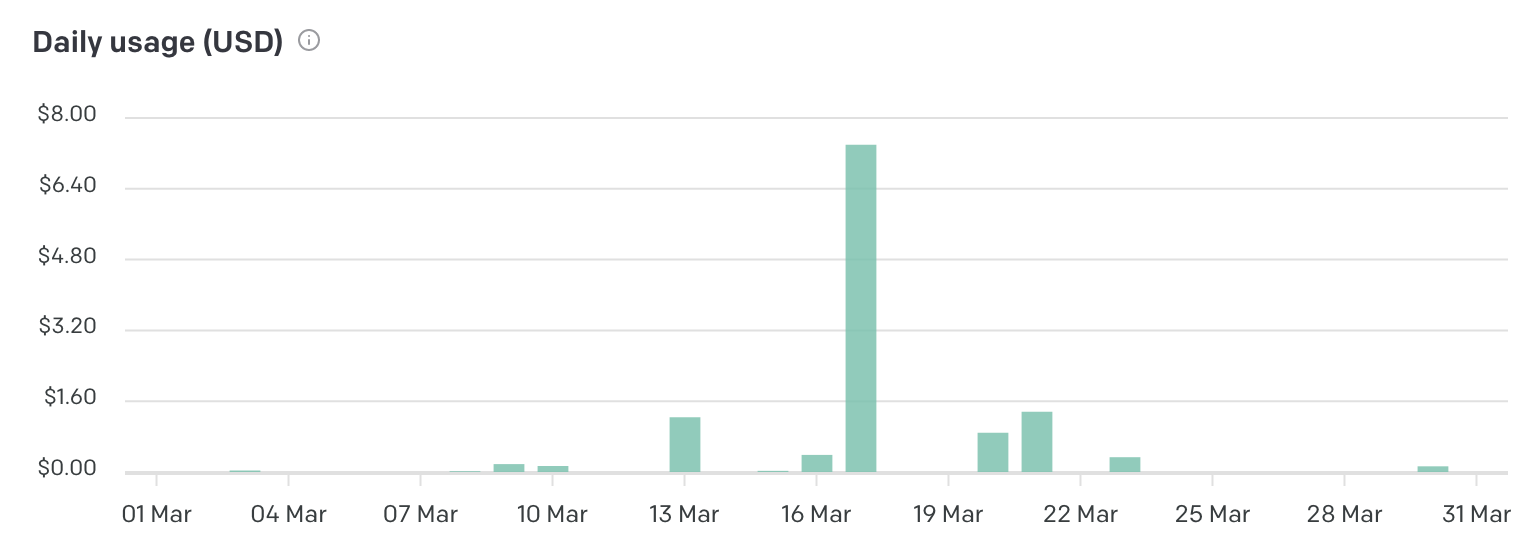
temperature=mytemperature

)

Above is the core of the code for querying the OpenAI model via python. The message variable contained the prompt, structured in the format that GPT needs, containing a role and content. There are other roles available for different purposes, but “user” was appropriate here. The word count, student, and title variables all came from our dataset and was filled in during each iteration of the loop. The model selected for our purposes was the “gpt-3.5-turbo” model. This was the latest model released at the time of this project; GPT 4 was released after we had run the bulk of our experiments.

Temperature is a variable used to determine what the style the model will write with is, lower values are more focused and coherent, where higher value are more diverse and creative but potentially less coherent. After testing different values, we ended up choosing a high value for temperature as this more closely resembled human writing. The 3.5 turbo model was able to produce surprisingly realistic content with ease, based solely an article title, grade level, and word count, with a temperature value of 0.9.

It should be noted that while using the chatbot interface, OpenAI does not charge money for its basic service. The company does offer a paid subscription which prioritizes the user’s activity with the program. However, using the API does cost money. When starting a new account, the company provided an $18 starting grant, but when generating 150 articles between 500-1000 words, we found the cost to be around $0.50. We were able to generate all data for this project within the free trial granted by OpenAI. Model creation is also a process that OpenAI charges for, but again, we were able to upload our custom dataset and train a model within our preview window. Also , a factor in our decision to focus on comparing different grade levels of AI generated text was influenced by not only the cost of generating AI content, but also by the availability of public datasets of AI content. We could not locate a dataset of all AI content, so we chose to refocus the project.



Now that we have a dataset, albeit a small one, of AI generated content we need to explore it and see what differences we can find between the grade levels but also against human text. For these purposes we decided to calculate some simple metrics to describe text data, and we chose average sentence length, average grammatical complexity, and sentiment score. For sentence length, we split each text at each period, then count the number of white spaces between each period and average them per text. To calculate grammatical complexity we can use the part-of-speech function from the python library Natural Language Toolkit (NLTK). First we tokenize the data, which is a method of standardizing, then we use the part-of-speech function to assign tags to complex words, count the number of complex words, and compute the average per text. Finally, to calculate sentiment we use the TextBlob package, which takes a text and computes the sentiment automatically, then assigns a score between -1.0 and 1.0.

We can visualize the data with some box and whisker charts:

Chart, box and whisker chart

Description automatically generated Chart, box and whisker chart

Description automatically generated

Chart, box and whisker chart

Description automatically generated

From the charts, we can see that there is noticeable difference, sometimes substantial, between our 3 categories of AI generated content. ChatGPT is a remarkable tool, and being able to replicate the more subtle differences of different author types is impressive. A Kruskal-Wallis test for each metric with n=100 in each group does show that sentence length and complexity have a significant difference, while sentiment does not. The p-values calculated are: 4 e-15, 1e-15, and 0.056, respectively.

Below are the graphs for human vs AI content in our dataset, where see even more interesting differences.

Chart, box and whisker chart

Description automatically generated Chart, box and whisker chart

Description automatically generatedChart, box and whisker chart

Description automatically generated

Interestingly, human writing has much more variation in sentence length and grammatical complexity, but less variation in sentiment. Overall, though, we see differences we would expect to see. For human vs. AI, after a Wilcoxon Rank-Sum test, all 3 variables show significant differences with p-values of 0.003, 5e-05, and 1e-14.

**Modelling**

For attempting to predict the category of given text, we have chosen two possible models. The first is Fine-Tuning from OpenAI. Through their API, people can upload their own custom natural language dataset, and using the weights and biases calculated by OpenAI based on the custom dataset, create a classification model. The second model we have chosen is an LSTM, created with the python library Tensorflow. Tensorflow is an open-source python package published by Google that allows for anyone to train neural networks on their own local computers.

The first models that we created were using the OpenAI Fine-Tuning. Their API provides a convenient framework to upload data, do the pre-processing automatically, and queue your model up for training in just a few lines of code that can all be run in the console. In theory, you could even create a dataset in Microsoft excel and run the training entirely from a command prompt without python, but there are ways to execute command prompts in python, making the whole process very painless.

As with the text generation, training models using the OpenAI API is not free. For training models with around 400 examples at about 1000 tokens per example, and 10 iterations through the data, we ended up spending $0.75 per training. Again, this was covered by the introductory grant, but as the amount of data goes up, and number of iterations goes up, so does cost. With that in mind, the cost of building a computer that can compete with the GPT models could cost hundreds of millions of dollars, so for most use cases, the cost is an acceptable trade off.

For our first set of models, we chose to go with the Fine-Tuning. This is the latest and greatest tool for NLP classification, and we can upload our own custom dataset to OpenAI’s network and create a custom model using their pre-trained GPT 3. Before we look at the results, we should remember the performance of OpenAI’s classifier tool, which correctly identified AI written text 26% of the time, and human written text 91% of the time, which means their model thought that almost everything it saw was human. This is important for interpreting our results, as we need something to compare to, however we are using a vastly different dataset which is likely much smaller, and much more specific.

Here are graphs of our model’s accuracy and loss functions, computed at each epoch, for human written text against AI text. Accuracy is on the left, and loss on the right:

Chart

Description automatically generatedChart, histogram

Description automatically generated

As you can see, we were able to classify with 100% accuracy! Why are we seeing results like this, though, as we know that much larger models are not this accurate? We cannot be entirely certain as Fine-Tuning models do not provide much associated data, but we can speculate. First, it likely has to do with our data, using professional and semi-professional writers to compare to an AI that is playing the role of different age students, one can fathom that the texts would be quite different, which we confirmed earlier in our examination of the data. We cannot pass explanatory variables such as grammatical complexity into the Fine-Tuning model, but that does not mean that Fine-Tuning models are calculating similar variables and using them in their predictions. Another factor here is that Fine-Tuning models are pretrained, and we are taking advantage of the fact that GPT 3 has seen almost the entire internet, potentially including our training data. GPT does not behave like a search engine specifically though, it generally can’t recall the specific text of an article from several years ago, so we are not suggesting that the model recognizes the specific articles we are using for training data. We are suggesting that it does have a vague memory of articles like ours and can rely on that to make accurate predictions in our specific case. We would show a confusion matrix, but it is not interesting, as there are no false positives or false negatives.

We had some idea that creating a model based on a limited dataset might either yield results that were very accurate or very inaccurate, this is why we decided to run another model comparing different “levels” of AI generated text. Below are graphs describing the training process in terms of accuracy and loss:  
Chart, line chart

Description automatically generated Chart, histogram

Description automatically generated

For the 3-class model comparing different grade levels, our results landed at 61% accuracy. Loss was like the human vs AI classification, but our accuracy was much lower. A confusion matrix is quite helpful in visualizing the predictions here.

Chart, square

Description automatically generated

In the above chart, the darker the square, the higher the frequency of entries in that square. Correct predictions fall along the diagonal from top left to bottom right, so in the top left square, all entries located there were predicted to be 10th grade, and were indeed 10th grade, and the same goes for the middle square and the bottom right regarding 6th grade and college, respectively. So, what does this tell us? We see that 10th grade and college were often mistaken for one another, but 6th grade was very accurate. This does line up with the exploration we did earlier where we saw that for sentence length and grammatical complexity, 10th grade and college were very similar.

Since we do not know much about how exactly OpenAI’s Fine-Tuning models are created, we chose to create some custom models from scratch using the Tensorflow library. These models examine the same class predictions, the three levels of AI text, and human vs AI text, but instead of being only the text they include the metrics that we calculated earlier. One of the main benefits of using Tensorflow is its ability to be customized to any dataset, but there are tradeoffs. Fine-Tuning models can be incredible for NLP classification tasks as GPT may be the most sophisticated language model in existence, but what if we don’t want to perform classification, or we want a more transparent approach to modelling? That’s where libraries to create models on a local machine come into play. If one desires, they could even upload their code to a cloud service that offers access to special computers for training neural networks.

For our purposes, we have a small enough dataset that running it on our local computer is no problem. Getting the model up and running while mixing text and numerical data was tricky, but with some help from the ChatGPT chatbot, we were training models in a short amount of time. Starting with the human vs. AI model, our results are closer to what we expected from the Fine-Tuning.

Chart, line chart

Description automatically generated Chart

Description automatically generated

Peaking at 74% accuracy around 450 training iterations, we see some excellent results there. In terms of loss, while actual numbers are relative, we would have like to see values closer to 0 instead of 1. This indicates that the model was not as effective as it could be. Looking at the confusion matrix this becomes more clear.

Chart, treemap chart

Description automatically generated

Oddly, we see that ai text was never predicted to be human writing, almost the exact opposite of the OpenAI classification tool. However, human text was almost entirely predicted to be AI generated. While we can’t be entirely certain why this is happening without more research, which is beyond the scope of this project, a lack of data is likely a big factor. The Fine-Tuning from OpenAI requires very small amounts of data, where Tensorflow (and PyTorch, the other common deep learning python library) can require large amounts of data to create sound models. That said, we wanted to create these models locally as a sort of sanity check, i.e., was what we saw with the Fine-Tuning model what we should expect generally, or was that on outlier in terms of performance? The same philosophy was applied to 3 class model.

Looking at the grade levels model created in Tensorflow, we see results more in line with what the Fine-Tuning model achieved. Accuracy and loss graphs show very similar results:

Chart, line chart, histogram

Description automatically generated Chart

Description automatically generated

Accuracy was at it’s peak around 375 epochs at approximately 56%, and loss values were low and convergent, but at a value of 1.25. While this isn’t great, achieving excellent results was not the goal of this project. The confusion matrix tells more of the story, though:

Chart

Description automatically generated

As before, correct predictions are along the diagonal, starting in the top left, and we can that 6th grade and college both were predicted relatively well, but 10th grade writing was often misclassified as the other categories. Overall, with the 3-class model in TensorFlow, it was reassuring to see very similar results to what we saw with the Fine-Tuning model.

As a final sanity check, we did take some of the content written by GPT 3 and the original article that we used a prompt and gave it to the chatbot interface and ask it whether it thought it was written by a human or not. Here are the results after asking about the AI written article:

Text

Description automatically generated

And here is the output from the original article:

Text

Description automatically generated

**Future Work**

While this work was interesting and we learned quite a bit, there are many ways this work could be improved upon. Here are just a few of the ideas we have had about modifying these experiments:

* Create a variable importance function to run along side TensorFlow models. Shapley values could be used, or permutation feature importance using the ELI5 library.
* Removing the text from the model and running only numeric data. We would not need a neural network for this, we could use a larger variety of models such as logistic regression or gradient boosted machines. Some of these models have built in variable importance functions as well. In addition to modelling only numeric data, we could model only text, like the Fine-Tuning, but with more control via TensorFlow.
* Investigating a method for passing numeric features to the Fine-Tuning method through prompt engineering. If we can pass this data into the model, we should. Be able to get somewhat better results.
* Experiment with different prompts for data generation. Ours is a straightforward prompt, but in asking GPT 3 to write like a student, and write an essay, we might have been “tipping our pitches”, and this might have allowed unusually high results in the human vs AI experiments.
* Generate a lot more data, in different roles, using different sources as topics. We were limited in the amount of data we could gather and the sources we could gather it from and believe this hampered our ability to craft a good model from TensorFlow.
* Engineer more features of our text data. Burstiness, or increases and decreases of events, is a popular feature that we did not calculate for this experiment. We could also look further into ways to quantify syntax and vocabulary to see if there are additional patterns in AI writing.

It is an exciting time for AI and natural language, as the world begins to see what the LLM is truly capable of. However, we do need to proceed with caution. If ChatGPT can pass the Bar exam and help students plagiarize their homework, it might also be capable of writing malicious social media posts or crafting spam emails. Over the course of this project, we learned quite a bit about using both the chat interface and the API to interact with the GPT model, what it is capable of and what it is not. Though our results were not definitive in detecting AI content, we did learn some of what goes into the task of creating such a model, and some ways a model could be improved.