Introduction

As LLMs improve, the act of recognizing AI generated texts becomes more and more important. Whether it’s in influencing the masses via social media, creativity in students’ assignments, or even the creation of datasets for reseach, knowing whether text is actually human created will be a key focus in the upcoming years. That is the topic I have decided to consider.

Project Objective

The end goal I had in mind was simply to see if I can fine-tune a pretrained model to perform AI detection in different domains. To that end, I wanted to leverage actual gramatical structure through handcrafted features in the form of parse trees. My hypothesis was that simply providing a parse tree can have a big impact in informing a model whether a piece of text was AI generated or not.

Methodology

Starting with a pretrained Roberta model (~125M parameters) that was trained on the answers from the HC3 dataset, a AI/human QA dataset, I would finetune it to two separate domains (Twitter and academic abstracts) via two additional datasets (Tweepfake and CHEAT) with and without parse tree information. These two finetuned domains are different from the original pretrained model, which had mostly financial and medical QAs.

To generate parse trees, I followed the process described by Kauchak, Leroy, and Hogue in “Measuring Text Difficulty Using Parse-Tree Frequency” (2017) with some modifications. All human created sentences in the respective training sets of each finetune dataset was run through the Berkley neural parser to generate a parse tree. Only human sentences were included in this parse distribution because I believed that marking any AI generated training samples that did not follow seen human grammatical structures would make it easier for the model to distinguish between human and AI.

Following the paper, all trees were pruned at a depth of three, below which parse trees become ungeneralisable – we do not want the parse structures to be too sample specific. And all parse structures were binned into 11 categories, with bin 1 being the top 1% of parses (in terms of occurrence in training samples), bin 2 being the top 10%, and so on. Each sentence in the training, validation, and testing datasets was parsed and assigned a category. If a parse did not exist in the frequency dictionary made by the training data, they were assigned to a zero bin (i.e. does not match any observed human parse structures).

The parses and the categorization were concatenated (with the ‘<s>’ separator) to the original text and became the new input for the tokenizer.

After preprocessing all the data for a dataset, a transformers Trainer was created and used to train and evaluate performance.

Hypothesis

Due to the difference between the Twitter and academia datasets, with Tweets often only being phrases and not full sentences – my hypothesis was that the performance of this methodology would be far less effective for Tweets.

Datasets

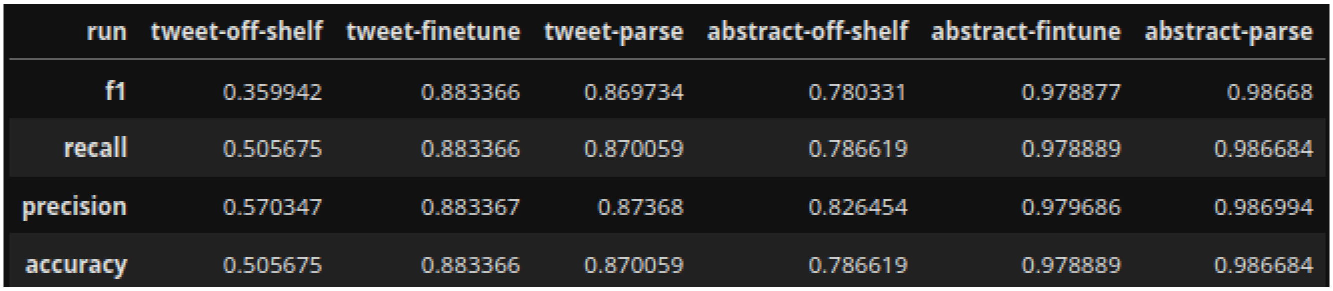
Both datasets were well balanced between human and AI samples – so no data manipulation was required for training.

TweepFake (https://www.kaggle.com/datasets/mtesconi/twitter-deep-fake-text)
(note: AI is split between RNN, hidden Markov, and GPT-2)
training set = 20712 samples (10358 human, 10354 AI)
valid set = 2302 samples (1150 human, 1152 AI)
test set = 2558 samples (1278 human, 1280 AI)

The CHEAT (https://github.com/botianzhe/CHEAT) dataset did not come with predetermined splits – I divided it with splits 80-10-10 to match similar ratios as the Tweepfake dataset.

CHEAT:
training set = 24632 samples (12304 human, 12058 AI)
valid set = 3079 samples (1565 human, 1514 AI)
test set = 3079 samples (1526 human, 1553 AI)

Results and Discussion



run	tweet-off-shelf	tweet-finetune	tweet-parse	abstract-off-shelf	abstract-fintune	abstract-parse
f1	0.359942	0.883366	0.869734	0.780331	0.978877	0.98668
recall	0.505675	0.883366	0.870059	0.786619	0.978889	0.986684
precision	0.570347	0.883367	0.87368	0.826454	0.979686	0.986994
accuracy	0.505675	0.883366	0.870059	0.786619	0.978889	0.986684

Finetuning did most of the work to increase accuracy by 38 points into the high 80s for the Twitter dataset and 29 points up to the high 90s for the abstract. This is unsurprising, echoing the often touted machine learning mantra “finetuning is all you need”.

I believe the abstract does so much better than the Twitter model because the pretrained model was trained on QA answers, so it is more used to longer texts. However, I’ve shown here that it can still be converted into identifying short LLM generated texts with some success.

As expected, adding parse tree information did nothing for the TweepFake dataset. In fact, the accuracy actually decreased. I expect this is due to the fact that Tweets are so short without many consecutive sentences, meaning the model has to decide from just a single parse. Also, the fact that people do often (compared with proper writings or even ELI5 Reddit posts) write random gibberish in them most likely means that parse trees can be incredibly broken.

However, adding parse trees just appended to the end of the text improved the accuracy of the detection of abstract generation by one point to almost 99%, which is an incredible increase at that level. Because how many sentences are in each abstract, I believe the addition of so many different parses meant the model had so much more information to work with. It may be that with so many parses, if any single order of parses (represented by the tokenizer as input_ids) did not seem to match something a human would write, it would have a high probabily of being AI generated.

Further Work

There were a lot of ideas during development that were not explored due to time, lack of knowledge, and resource constraints.

Late into development, I discovered the *multimodal-transformers* library (https://multimodal-toolkit.readthedocs.io/en/latest/notes/introduction.html), which allows you to pass tabular inputs, specifically multiple types of and multiple columns of data to a model's training. Instead of just relying on input_ids and an attention_mask of a parse tree representation and its category in the frequency count, this might help great improve the reliability of the data and therefore the training results. This does lead to a discussion of my odd choice of concatenating all the data (original text, parse tree, and parse distribution category) and using them as input data. Because transformers just use tokenized text to create predictions, I believed this was the only option open to me and hoped the concatenation of these three were enough to lead to interesting results. If I were to start again, I would like train using this multimodal transformers – we can use parse tree strings, category numbers, and perplexity scores, among other multimodal data as input without having to convert them to strings.

Speaking of perplexity, further grammatical relationships like perplexity scores may help improve the models, calculated against the word frequencies of each respective corpus domain. Given that one of the most successful methods of AI detection has been watermarking, where models always choose specific words in situations, adding a perplexity score to the input may greatly help identification of patterns of strange (in terms of a general English corpora) word patterns.

Expanding on the idea of perplexity – I also wondered if it was possible or if there was any interest in calculating the perplexity not of each word given the previous context, but the parse structure of a sentence given previous parses. Especially for longer texts like the scientific abstract dataset, can we discover different patterns in how humans follow up previous sentences with certain parse structures. AI may tend to repeat a sequence of parse structure more often than humans, who are often more creative with their parse structures. However, I was not sure of the validity of or how best to implement this idea.

After discovering (Son, et. al, 2012)'s "An application for plagiarized source code detection based on a parse tree kernel", I also wondered about the possibility of creating a kernal to compare each parse tree against all the potential trees in the human parse structure distribution. However, I was unsure of how to proceed in this direction.

Lastly, I do not address any of the potential adversarial attacks that can be applied to try and defeat AI detectors. Examples include paraphrasing or prompting. The CHEAT dataset provides a separate human/AI hybrid dataset for a much more difficult classification decision. I also had the idea of generating a small set of responses (prompted or paraphrased) from chatGPT-3.5 and Claude as an additional finetune layer to improve model's performance against text generated by models other than roberta. However, I did not have time to achieve these ideas.