Classifying Human and Machine Generated Text

Norrec Nieh, Frank Rossi, & Jason Zhang
Khoury College of Computer Sciences
Northeastern University
Boston, MA 02115
nieh.c@northeastern.edu, rossi.f@northeastern.edu, zhang.haozhe1@northeastern.edu

Abstract

We investigated the ability of perplexity to classify texts as human or machine generated using two approaches, a single perplexity score and a sequence of word probabilities corresponding to the input text. The former was classified according to a single threshold and the latter was fed into a neural network. These perplexities and probabilities were generated using N-grams. Our best result for the single score, threshold classifier was 77% and our best result for the probability sequence, ANN classifier was 80%. Our work demonstrates that perplexity can be used as a feature to distinguish human and machine texts even with basic classifiers.

1 Introduction

GPT (Generative Pre-Trained Transformers) are neural data models developed by OpenAI that generate human-like text by way of pre-training on extremely large datasets, with GPT-3 trained on 175 billion parameters and around 400 billion tokens of text, over a hundred times larger than its predecessor GPT-1 (Brown et al.). The rapid development of these models in recent years and the increasing popularity of related tools such as ChatGPT have given rise to a host of ethical concerns regarding their usage, specifically in relation to the ease of "AI plagiarism" and forgery. The scope of this issue is evident when considering OpenAI's own performance evaluation of GPT-4, which can ostensibly achieve around 90-percentile scores on SAT and AP tests, among others. This study is an examination of methods used to distinguish GPT-generated texts from human-written texts, after similar studies such as Mitchell et al.'s "DetectGPT," which utilizes small "perturbations" in the human-written texts to compare probabilistic differences, and Fannie Lin's open-source version of GPTZero, which determines probabilistic classification through an ensemble method, considering both the *perplexity* and *burstiness* of a text.[1] Drawing the general intuition from both studies that a higher variance in probability values (or a higher perplexity)

[1] Perplexity measures how well a language model predicts new text data ("Perplexity of fixed-length models," 2023), while burstiness measures the tendency of certain words to occur in short, concentrated bursts within a given text corpus (Kleinberg, 2002), and is represented in Fannie Lin's open-sourced version of GPTZero as the maximum perplexity value pertaining to any sentence within the text (Lin, 2023)

within a text suggests that it is human-written, while a lower variance (or lower perplexity) suggests an AI-generated origin, we sought to investigate and present ways in which comparatively simplistic models can be used to classify human- and machine-generated texts through determining a text's overall perplexity.

2 Method

2.1 Dataset Generation

Our initial proposal involved using the PubMedQA dataset, a biomedical research dataset, whose entries comprise a long form answer written by a human expert in response to a technical question (Jin et al., 2019). We planned to use the technical questions as queries associated with two input texts – the original human-written text, which we label as class 0, and the machine-generated text, which we would produce by way of the GPT-2 transformer. However, due to the highly technical nature of the dataset, as well as the shorter lengths of the questions, the structure of the outputs were highly unpredictable and often fell in the category of dialogue scripts or bibliography citations. Consequently, we moved to the 20NewsGroups dataset, which consists of 20,000 news group forum posts falling under 20 different general news topics. Since many of the posts were short, we used a curated selection of longer posts as our human (class 0) texts, and generated texts of a similar length via GPT-2 for our machine (class 1) texts, removing truncated sentences at the end of the outputs (due to the predetermined character limit) to preserve the structural integrity of the sentences, and also the initial query at the beginning of the outputs. We proceeded with a dataset of 6000 human and 6000 machine texts, with each machine text generated from a topic statement pertaining to an extant human text.

```
subject_lines = fetch_and_extract_subject_lines()
subject_lines
```

```
18846
['Pens fans reactions',
 'Which high-performance VLB video card?',
 'ARMENIA SAYS IT COULD SHOOT DOWN TURKISH PLANES (Henrik)',
 'IDE vs SCSI, DMA and detach',
 'driver ??',
 'subliminal message flashing on TV',
 'Number for Applied Engineering',
 'Atlanta Hockey Hell!!',
 'Goalie masks',
 'Christians above the Law? was Clarification of pe',
 "14 Apr 93   God's Promise in 1 John 1: 7",
 'Fighting the Clipper Initiative',
 'OTO, the Ancient Order of Oriental Templars',
 'Krillean Photography',
 'Islam And Scientific Predictions (was Genocide is Caused by Atheism)',
 '"Stretching from the Adriatic Sea to the Great Wall of China"',
 '600RPM Floopy drives - UPDATE!',
```

Figure 1. Subject lines retrieved from 20NewsGroups dataset.

2.2 Data Preprocessing

We prepared our dataset by casting it in the form of a pandas dataframe, initially with a text column and a label column (class 0 if human-written or class 1 if machine-generated). We then defined functions to perform cleaning, as well as the splitting of a text into sentences, in addition to the padding of a split text with start and end tags (pertaining to the length *n* of the ngrams) as a measure of preparation for n-gram training. Cleaning consisted of standard operations such as the

removal of URLs, email addresses, punctuation, and stopwords, as well as conversion to lower-case characters and lemmatization using the NLTK WordNetLemmatizer. We used these functions variably on each input text to create two distinct versions of preprocessed text: in one, the text is represented as a list of sentences, each having been cleaned and properly padded with start and end tags, and in the other, the text is cleaned as a whole. These new forms of the text are then respectively written into the original dataframe under the `text_split_cleaned` and `text_cleaned` columns. Finally, the updated frame is saved as a CSV file for future use in modeling and visualization.

	text	label
0	I have following softwares for sale:\n\nNEW IT...	0
1	Rewording the Second Amendment (ideas) of the ...	1
2	1)! What books and articles on X are good fo...	1
3	Moonbase race. Both were used to keep a low pr...	1
4	Several years back one of the radar detectors ...	0
...	...	...
11995	# |## |#2. Professors get summers off; industr...	0
11996	Ok, So I was a little hasty...\n\n...but a gir...	1
11997	GW2000 and SIMMS (for instance if you have a f...	1
11998	I recently backed out of purchasing an almost-...	0
11999	[purile babble deleted]\n\nWell, some form of ...	0

	text_split	label
0	['I have following softwares for sale: NEW IT...	0
1	['Rewording the Second Amendment (ideas) of th...	1
2	[' 1)! ', 'What books and articles on X are g...	1
3	['Moonbase race. ', 'Both were used to keep a ...	1
4	["Several years back one of the radar detector...	0
...	...	...
11995	['# |## |#2. ', "Professors get summers off; i...	0
11996	['Ok, So I was a little hasty... ', " ...but a...	1
11997	['GW2000 and SIMMS (for instance if you have a...	1
11998	['I recently backed out of purchasing an almos...	0
11999	["[purile babble deleted] Well, some form of ...	0

	text_cleaned	label
0	follow softwares sale new items never open 1 l...	0
1	reword second amendment ideas us constitution ...	1
2	1 book article x good beginners book good beg...	1
3	moonbase race use keep low profile battlefield...	1
4	several years back one radar detectors manufac...	0
...	...	...
11995	2 professors get summer industry employees do...	0
11996	ok little hasty girl come room look others tit...	1
11997	gw2000 simms instance simms might able use g2c...	1
11998	recently back purchase almostunused sony tcdd3...	0
11999	purile babble delete well form guarantee healt...	0

	text_split_cleaned	label
0	['<s> <s> follow softwares sale new items neve...	0
1	['<s> <s> reword second amendment ideas us con...	1
2	['<s> <s> 1 </s> </s>', '<s> <s> book article ...	1
3	['<s> <s> moonbase race </s> </s>', '<s> <s> u...	1
4	['<s> <s> several years back one radar detecto...	0
...	...	...
11995	['<s> <s> 2 </s> </s>', '<s> <s> professors ge...	0
11996	['<s> <s> ok little hasty </s> </s>', '<s> <s>...	1
11997	['<s> <s> gw2000 simms instance simms might ab...	1
11998	['<s> <s> recently back purchase almostunused ...	0
11999	['<s> <s> purile babble delete well form guara...	0

Figure 2. Updated 20NewsGroups dataset.

2.3 Probability and Perplexity

Our methodology relies on the mapping of words and sequences to probabilities and perplexities. We used N-gram probabilities in this assignment, where the probability of each word occurring is approximated by the previous n-1 words. For trigrams, this probability is generated with the following equation:

$$P(w_n|w_{n-2}w_{n-1}) = \frac{C(w_{n-2}w_{n-1}w_n)}{C(w_{n-2}w_{n-1})}$$

One variation on this methodology tries to account for unseen sequences due to the finiteness of the training set by "smoothing" the probabilities, adding one to each count. This is called Laplace smoothing and can be represented by this modified equation, where V is the size of the vocabulary:

$$P(w_n|w_{n-2}w_{n-1}) = \frac{C(w_{n-2}w_{n-1}w_n)+1}{C(w_{n-2}w_{n-1})+V}$$

2.4 Single-Score Classifier

While both of our classification models use n-grams as part of the training process, our first classifier uses n-grams also to directly evaluate the perplexity score of a text in its entirety against a predetermined threshold, which we adjusted as we refined and implemented

configurations in our model. As detailed below in our results and ablation study, these configurations included using bigrams and 4-grams instead of trigrams, which resulted in under- and over-fitting, as well as evaluating the test dataset on n-grams derived from only the class 1 (machine-generated) texts, only the class 0 (human-written) texts, or both. We also determined the *burstiness* of each text, as proposed in Fannie Lin's summary of her reconstruction of GPTZero, by way of three different approaches, alternately defining it as 1) the maximum perplexity among all sentences within a given text, 2) the average perplexity among all sentences within a given text, or 3) the difference between the min and max perplexities within a given text. The burstiness values derived from these definitions were compared to and (in theory) used as part of an ensemble technique to augment our results. We detail in the following section the outcome of evaluating burstiness as a supplementary technique in our model. We used five splits in both classifiers using the sklearn module for the purpose of cross-validation.

2.5 Probability Sequence Classifier

Our second idea for feature extraction was to see if there was any additional variation in the dataset beyond a single perplexity score for each text. Perhaps there was not only a lower average perplexity in the machine generated texts, but a more "stable" perplexity as well, with individual word probabilities not varying as much compared to human texts. In an attempt to capture this additional dimensionality in the data, we created a separate method of feature extraction to convert texts to sequences of word probabilities. We used the same methods of preprocessing and n-gram generation to assign each word in the text an n-gram probability. To standardize sequence length, we could have chosen either padding or truncation. We chose truncation to ensure that differences in text lengths between human and machine texts would not bias the model with additional dimensions only filled by one class (machine text was generated to be 256 tokens, human text lengths were quite variable). These probability sequences were fed into a basic neural network with alternating dense and dropout layers. Neural Networks were chosen for this task because of their adeptness in finding complex decision boundaries in high dimensional spaces. Differently structured networks may perform better on this data, but limited variations produced no significant improvement and more complex designing of networks was beyond the scope of this project.

3 Results and Analysis

3.1 Single-Score Classifier

Our isolated attempts to classify a text based on burstiness provided a definitively worse result across the differing configurations, at worst producing no better than random outcomes at around 50% accuracy, and at best producing 6% less accurate results than a result determined from the perplexity of a text in its entirety. We can see in the graphs below a distribution of accuracy across possible burstiness thresholds for both the maximum-perplexity and average-perplexity definitions of burstiness, pictured beside the distribution graph of accuracy across possible perplexity thresholds. While the perplexity graph is of a standard shape, with a prominent peak

representing an optimal threshold of 24, the burstiness graphs present an irregular distribution and show a maximal accuracy of around 54% and 52%. As such, we were unable to apply burstiness as a part of an ensemble method. This would comprise an extended study given more time, but we hypothesize that the burstiness scores were negatively (and fatally) affected by our decision to train our model on the machine-generated dataset instead of using the pre-trained models associated with GPT-2, from which our machine data was generated.

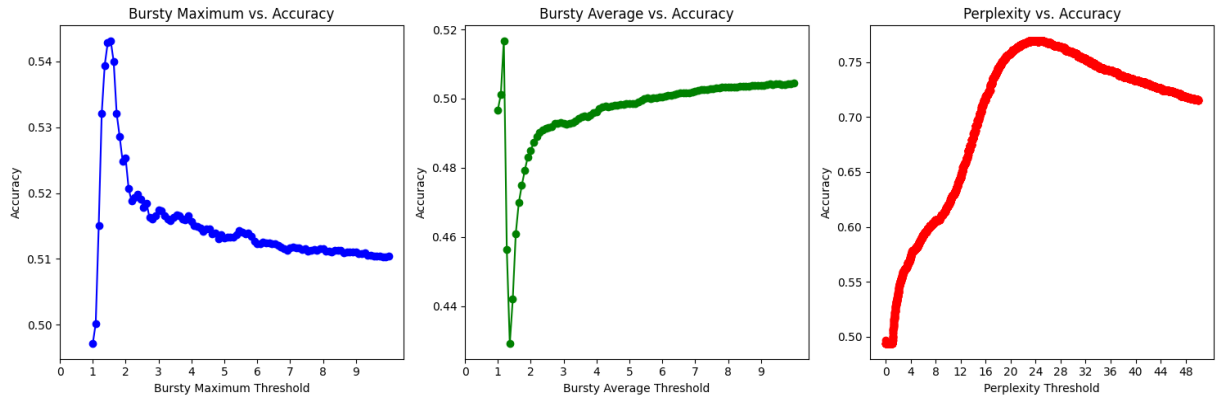

Figure 3. Distribution of accuracy across burstiness and perplexity thresholds

Nevertheless, we achieved a successful outcome of 77% probability based only on evaluating perplexity alone against an optimized threshold. These results establish evidence toward our hypothesis that comparatively simplistic algorithms such as a single-score classifier can "detect" or classify machine-generated texts using a perplexity score mechanism. In figure 4, we see the distribution of perplexities resulting from the first of five splits. While there is considerable crossover in perplexity values between the two classes in this model implementation, there is also a distinct threshold at which point all higher perplexities are mapped to the human class. This trend is verifiable in figure 5, which contains the distributions of all five folds, further corroborating our assertion.

An ablation study was carried out for our single-score classifier, where we configured n-gram lengths ranging from 2 to 4, alternatively created n-grams using human or machine data, and tested the impact of incorporating Laplace smoothing. The results were obtained through 5-fold cross validation using the sklearn module. Appendix A, figure 2 details the full results of our findings, including our highest accuracy, obtained from employing 4-grams based on machine data without Laplace smoothing, resulting in a perplexity accuracy of 77.3%. This indicates that using 4-grams presents a middle ground between overfitting and underfitting. Generally speaking, we also see that testing based on machine-generated n-grams provides higher accuracies in overall perplexities classification, and Laplace smoothing, while reducing noise in overall perplexity scores, it reduces the fluctuations in individual word probabilities that are essential for the classifier's performance, resulting in slightly lower accuracy rates.

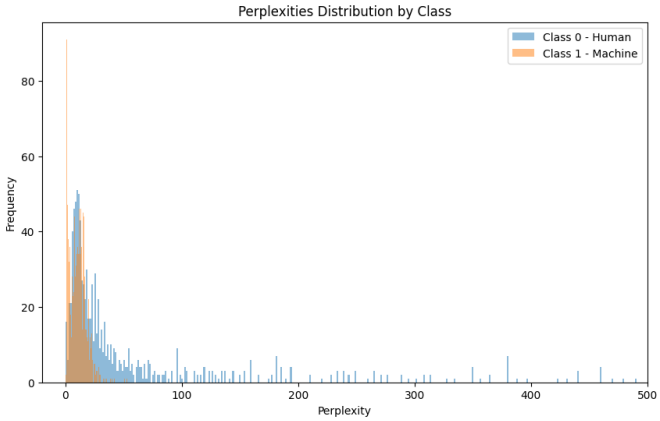

Figure 4. Perplexity distribution of texts by class in first split

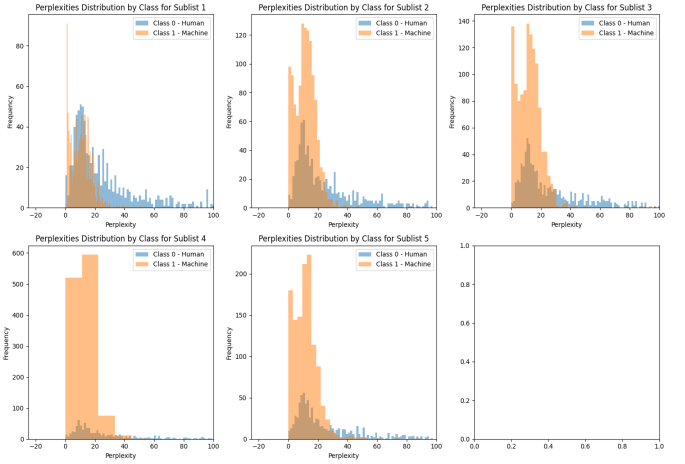

Figure 5. Perplexity distribution of texts by class in five splits

3.5 Probability Sequence Classifier

We conducted an ablation study to determine the best configuration of this classifier by varying N-gram lengths from 2-4, generating the N-grams from the human data, machine data, or both, and using or not using Laplace smoothing. All results were generated using 5-fold cross validation from the sklearn module. Our results are summarized in the following table, with our greatest accuracy coming from the use of trigrams generated from the machine texts without laplace smoothing, at an 80.8% test accuracy rate. These parameters can be explained by first recognizing that an n-gram length of 3 balances underfitting vs overfitting. Second, machine data is the most stable for n-gram generation. Finally, although laplace smoothing may reduce noise in the overall perplexity scores, it also reduces the variability in individual word probabilities that the classifier relies on. The full results are available in Appendix A, figure 2.

4 Conclusion

We investigated the ability of ngram word probabilities and perplexity scores to classify text as either human or machine generated. We demonstrated that significant accuracy can be achieved with basic classifiers. More advanced methods with complicated techniques (such as DetectGPT) exist to achieve higher accuracies, but improvements may also be possible while sticking with similarly simple techniques to those described here. First, it may be useful to fetch ngram counts from an outside pretrained source for more accurate probabilities. Second, a more custom configuration of the neural network may provide better classification, as clearly there is some useful information in the sequences not captured in the overall perplexity score and a different network may better capture these high dimensional variations. There are also multiple other ways of extracting the sequences, either by creating sequences of perplexities over sliding windows of text or sequences of perplexities corresponding to each sentence.

5 References

[1] Brown, T.B., et al. (2020). Language Models are Few-Shot Learners. arXiv preprint arXiv:2005.14165. https://arxiv.org/abs/2005.14165

[2] OpenAI Research. (2023, March 14). GPT-4. OpenAI. https://openai.com/research/gpt-4

[3] Jin, Q., et al. (2019). PubMedQA: A Dataset for Biomedical Research Question Answering. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (pp. 2567-2577). https://aclanthology.org/D19-1259

[4] Lin, F. (2023, January 29). Open-Sourced GPTZero. Indiehackers. https://www.indiehackers.com/post/open-sourced-gptzero-788e8c9d1a

[5] Mitchell, M., et al.(2023). DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature. arXiv preprint arXiv:2301.11305. https://arxiv.org/pdf/2301.11305.pdf

[6] Jurafsky, D., & Martin, J.H. (n.d.). Speech and Language Processing (3rd ed.). https://web.stanford.edu/~jurafsky/slp3/3.pdf

[7] Rennie, J. (2008, January 14). 20 Newsgroups Dataset. Retrieved from http://qwone.com/~jason/20Newsgroups/

Appendix A: Results

	N-gram Length	Model	Laplace Smoothing	Feature Extraction	Classification Algorithm	Bursty Max PP Accuracy	Bursty Avg PP Accuracy	Overall PP Accuracy
0	2	human	True	Single Score	Threshold	0.514250	0.502500	0.548333
1	2	human	False	Single Score	Threshold	0.514250	0.502500	0.549083
2	2	machine	True	Single Score	Threshold	0.544583	0.518583	0.754333
3	2	machine	False	Single Score	Threshold	0.542083	0.516250	0.756750
4	3	human	True	Single Score	Threshold	0.515500	0.498083	0.562917
5	3	human	False	Single Score	Threshold	0.514917	0.497250	0.562333
6	3	machine	True	Single Score	Threshold	0.544583	0.518333	0.769500
7	3	machine	False	Single Score	Threshold	0.544167	0.518583	0.770000
8	4	human	True	Single Score	Threshold	0.519583	0.500833	0.566167
9	4	human	False	Single Score	Threshold	0.519417	0.500750	0.565917
10	4	machine	True	Single Score	Threshold	0.542000	0.507250	0.772750
11	4	machine	False	Single Score	Threshold	0.541917	0.507000	0.773250

Figure 1. Ablation study table for single score threshold classifier

	N-gram Length	N-gram Base	Laplace Smoothing	Feature Extraction	Classification Algorithm	Accuracy
0	2	human	True	Word Prob Sequence	ANN	0.637500
1	2	human	False	Word Prob Sequence	ANN	0.792500
2	2	machine	True	Word Prob Sequence	ANN	0.713917
3	2	machine	False	Word Prob Sequence	ANN	0.790083
4	2	both	True	Word Prob Sequence	ANN	0.681833
5	2	both	False	Word Prob Sequence	ANN	0.683833
6	3	human	True	Word Prob Sequence	ANN	0.803000
7	3	human	False	Word Prob Sequence	ANN	0.801750
8	3	machine	True	Word Prob Sequence	ANN	0.800500
9	3	machine	False	Word Prob Sequence	ANN	0.808083
10	3	both	True	Word Prob Sequence	ANN	0.501667
11	3	both	False	Word Prob Sequence	ANN	0.692500
12	4	human	True	Word Prob Sequence	ANN	0.800917
13	4	human	False	Word Prob Sequence	ANN	0.800167
14	4	machine	True	Word Prob Sequence	ANN	0.782667
15	4	machine	False	Word Prob Sequence	ANN	0.793583
16	4	both	True	Word Prob Sequence	ANN	0.501667
17	4	both	False	Word Prob Sequence	ANN	0.649500

Figure 2. Ablation study table for probability sequence classifier