MinHash to Detect LLM Generated Content

Distinguishing text produced using Large Language Models (LLMs) as opposed to human-produced texts has been challenging for researchers due to models' abilities to generalize information and present it in new contexts. Even if model training sets are provided, predicting whether an answer depends upon the training set is challenging. In this paper, we propose that by focusing on a particular subfield it is possible to identify baseline similarities for LLM generated texts and exploit these similarities to classify human, and LLM generated texts.

Following our review of existing research, we found no previous work that utilized MinHash directly to compute similarity. We would like to investigate the practicality of using MinHash techniques, such as n-gram MinHash, k-shingling MinHash, SimHash, and similarity estimations, such as Jaccard similarity and cosine similarity, to distinguish human generated text from LLM generated text. We hypothesize that LLM generated content contains repeatable patterns, and by observing the average similarity between a set of LLM generated prompts, we can uncover a baseline similarity score. Using this baseline, we will calculate a threshold that we will use to classify future inputs (LLM or otherwise) by comparing them to our training set comprising of LLM responses.

Literature review

Kirchenbaur et al. [1] proposed a solution involving watermarking language model outputs which would allow for distinction between human written content. They describe getting the previous K tokens for the current word prediction. Then, using the tokens to compute a hash, which is used as a seed to a random number generator. Using the seed, the current word list can be partitioned into two sets, called the red list and the green list. The algorithm continually samples from the green list, never using the red list. To detect the watermark, they traverse the text and record the number of “violations” of the red list/green list rule [1]. Their intuition is that a human writer is expected to violate the rule continually since they are randomly choosing from one of the lists. An important caveat to the model is filtering out low-entropy sentences. These sentences assign high probabilities to a few predictions that make much more sense (e.g., if we consider the phrase George Bush, there is a very high likelihood Bush should be predicted). However, with the described methodology, we would miss these near-deterministic predictions. To overcome this shortcoming, the authors propose to discard these low entropy sentences by computing the change in quality of sentence given word changes. They further discuss the possibility of attacking the watermark and mitigating strategies.

Other approaches have involved more traditional classification algorithms to differentiate between human and LLM generated content. These algorithms include Naive Bayes, SVM, and. For example, Fröhling and Zubiaga [2] propose a classifier based on feature extraction of the text. They ascertain several facts about text generated by language models to identify key features they should focus on. These include a lack of syntactic diversity, wherein similar expressions, and overuse of adjectives while underusing nouns and pronouns. In addition, models suffer from a lack of coherence over larger paragraphs of generated text, termed “topic drift”. They generate datasets for humanly generated content by scraping web content and employing a filtering algorithm to improve the quality of the data. For machine LLM content, they use GPT2 and GPT3 outputs to test against. To understand the most important features, they evaluate their classifier on “individual subsets of features”. Then, to test their multi-feature models, they create a new dataset from generations composed of many models. They achieve especially stellar results on GPT2 XL, with over 95% accuracy.

Most existing approaches use the HC3, Human ChatGPT Comparison Corpus, which is comprised mostly of ELI5 (explain like I'm 5) question-answers which are inherently "low entropy" since most of the questions involved are not original work but rather a historical/factual recall. In this paper, we will use a dataset wherein we compare LLM generated and human generated abstracts.

Thesis:

We propose that by leveraging a variety of MinHash-based techniques to obtain similarity scores on academic abstracts, we can distinguish LLM generated abstracts from human generated abstracts.

To perform our survey on MinHash methods, we shall use the dataset listed in https://github.com/ChrisMJAndre/Detecting-AI-Authorship-Analyzing-Descriptive-Features-for-AI-Detection/blob/main/Datasets. The dataset includes a title and an abstract for research papers chosen from Arxiv. Each title appears twice, one representing the real abstract and one representing the output of a GPT3 query with the prompt to create an abstract for a given research paper title. We shall separate this data into two sets. We will first separate all LLM generated abstracts, from the human generated abstracts. We will then further split the LLM generated abstracts into two sets, 80% for training, and 20% for testing. We will keep all human generated abstracts for testing. Thus, our testing set will be comprised of both LLM and human generated abstracts. The training set will be X and the testing set will be A. It is important to note that all our results (i.e., accuracies) have been normalized to simulate a 50/50 chance of an abstract being LLM or human generated in our testing set.

We look to establish a baseline similarity for LLM generated abstracts, by computing the average pairwise similarity of all LLM generated abstracts in our training set. This would result in $\frac{N\times(N-1)}{2}$ comparisons. We hope that by comparing the abstracts, we can understand similarities due to the content's machine generation. We will create a baseline similarity score $b(X)$ from this operation.

After we have established a baseline similarity for LLM content, which represents the average similarity of all LLM generated abstracts, we plan on computing a threshold as a function of our baseline similarity. Let us define $t(X) = y \times b(X)$ where $t(X)$ represents the threshold for a given training set X, y represents a constant where $0 < y < 1$, and $b(X)$ represents the baseline similarity. We will then compare $t(X)$ with abstracts from our testing set (both LLM and human generated). For each abstract in our testing set $a \in A$, we plan on computing the average similarity score between itself and each abstract in our training set. That is $s(a) = \frac{\sum_{x\in X}\sum similarity(a,x)}{|X|}$. We hope to see human abstracts having less similarity than our threshold, $t(x) > s(a)$ where a is human generated. And for LLM abstracts to have more similarity than our threshold, $t(x) < s(a)$ where a is LLM generated. We plan on varying y to maximize our accuracy.

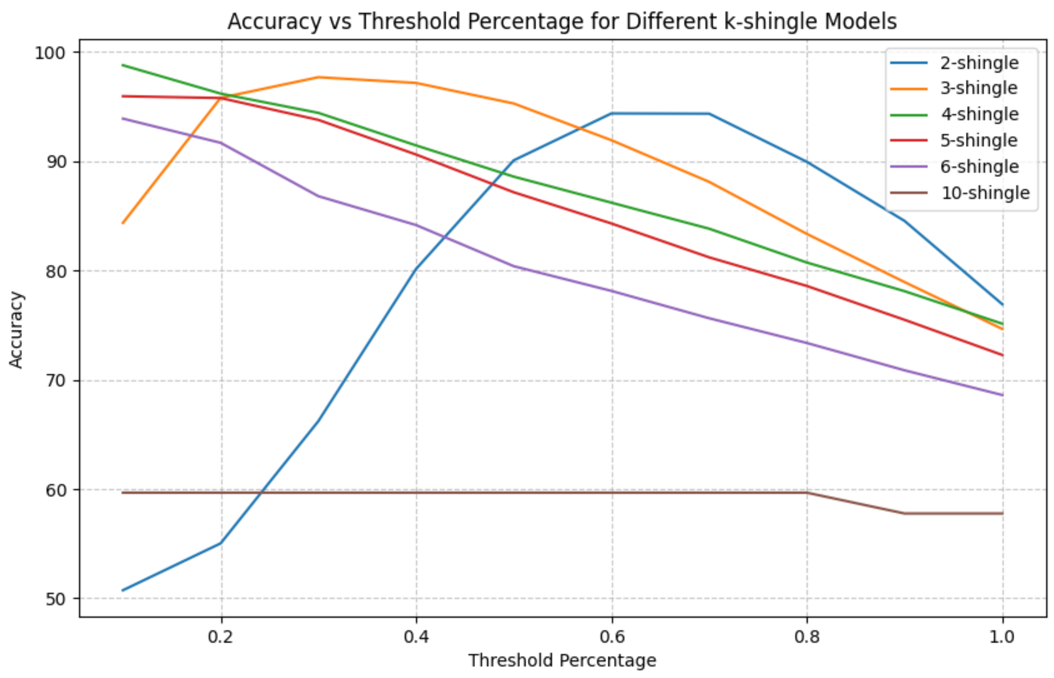
When calculating $t(X)$, the goal is to accurately classify the LLM cases, while minimizing the impact on human cases. As we lower y, our accuracy on classifying LLM cases improves, but at a cost of increasing our misclassification of human cases. The challenge lies in finding the optimal y value that strikes a balance between accurate identification of LLM generated content and minimizing misclassifications of human generated content.

Our study aims to compare various MinHashing techniques to identify the most effective method of distinguishing between LLM generated and human generated abstracts. We will explore n-gram MinHashing, k-shingling MinHashing, and SimHashing. We plan to explore varying values of n and k. Additionally, we will explore using tokenizers for MinHash (instead of n-gram or k-shingling). For our similarity scores, we will explore using cosine and Jaccard similarities. For our experiments, we used 128 hash functions to make our MinHashes.

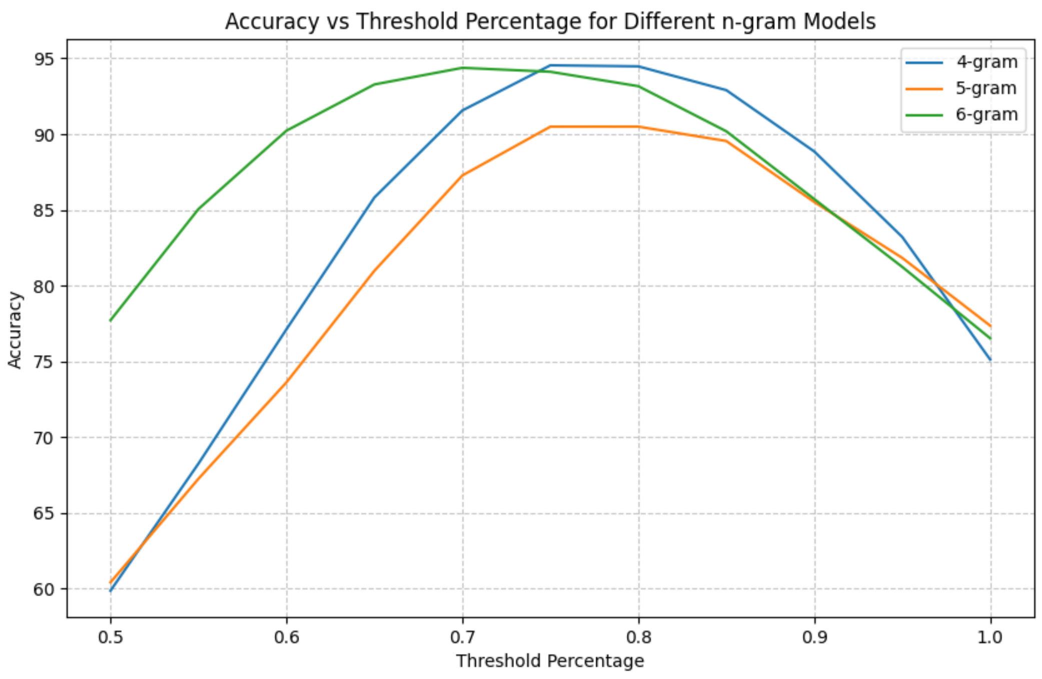
Plots and 1-2 paragraphs summarizing experimental result for each use case:

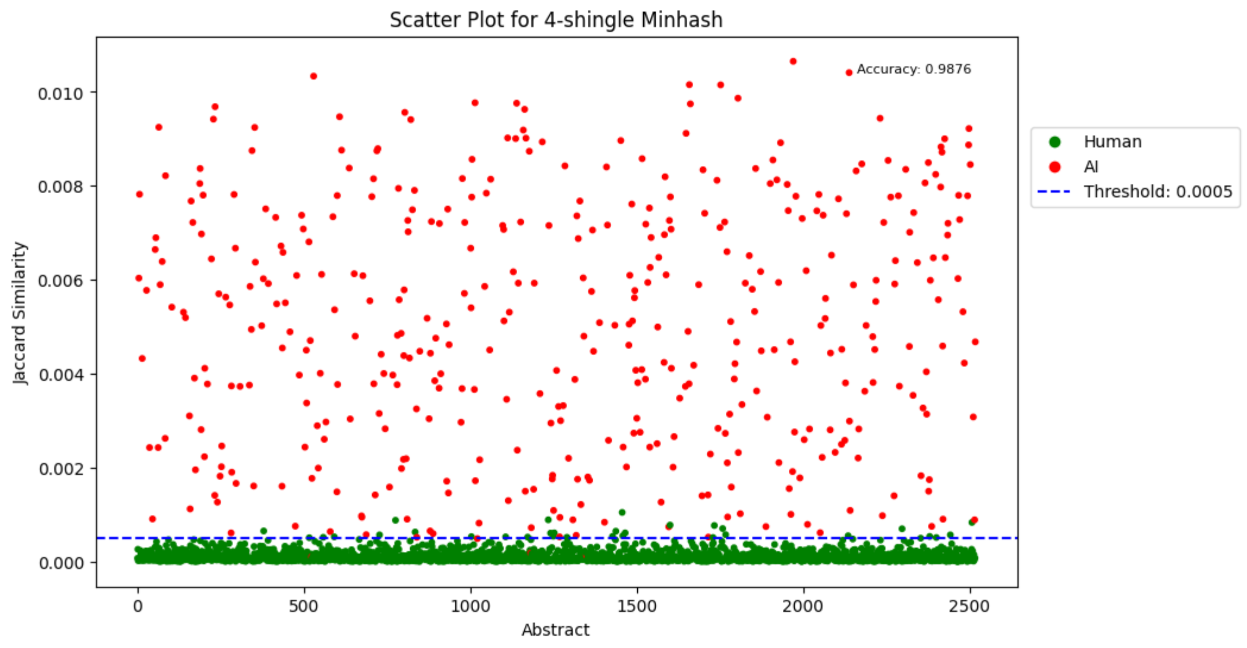
We were considering our use cases to implement different MinHash techniques for our training dataset comprised of LLM generated inputs. We made graphs to determine optimal parameters for using k-shingle and n-grams. Further, we graphed out the accuracy of our classification for each experiment as a function of adjusting the threshold to see which produced the best accuracy. Using 4-shingle MinHash using BertTokenizer we achieved the highest accuracy for the classification of our testing set.

We generated graphs for each parameter in every method. We found some unsatisfactory results; for example, when we applied SimHash we got extremely muddled accuracy scores, or when our threshold was too high (i.e., when y was larger).

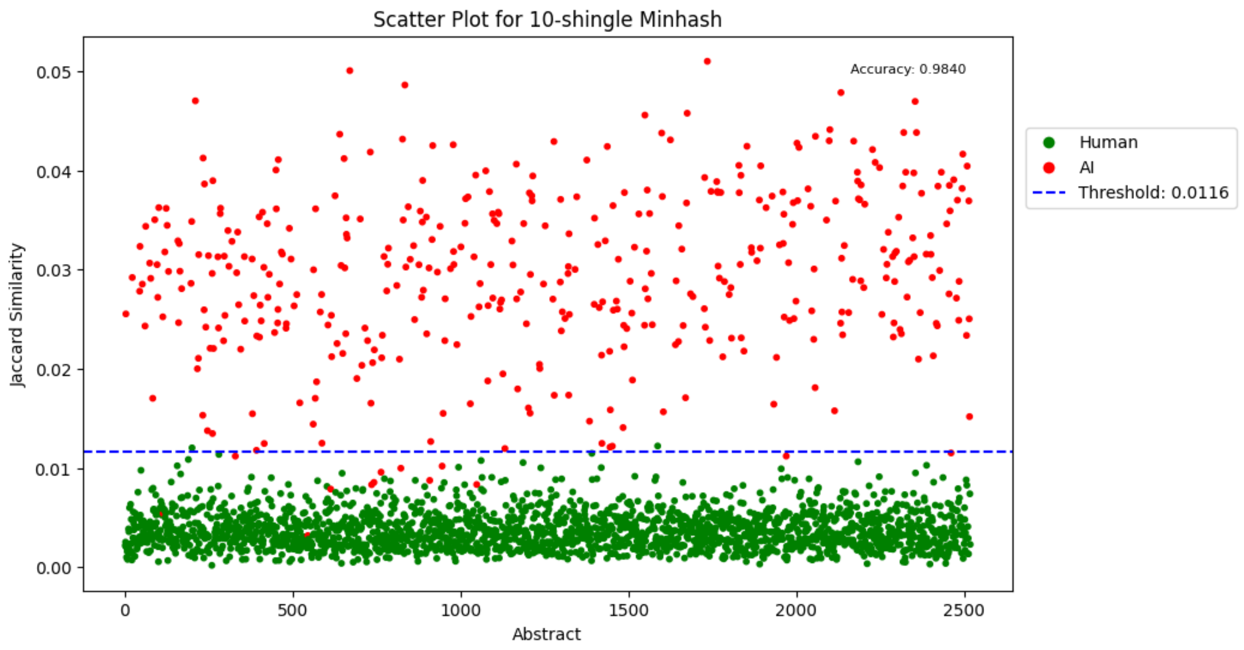


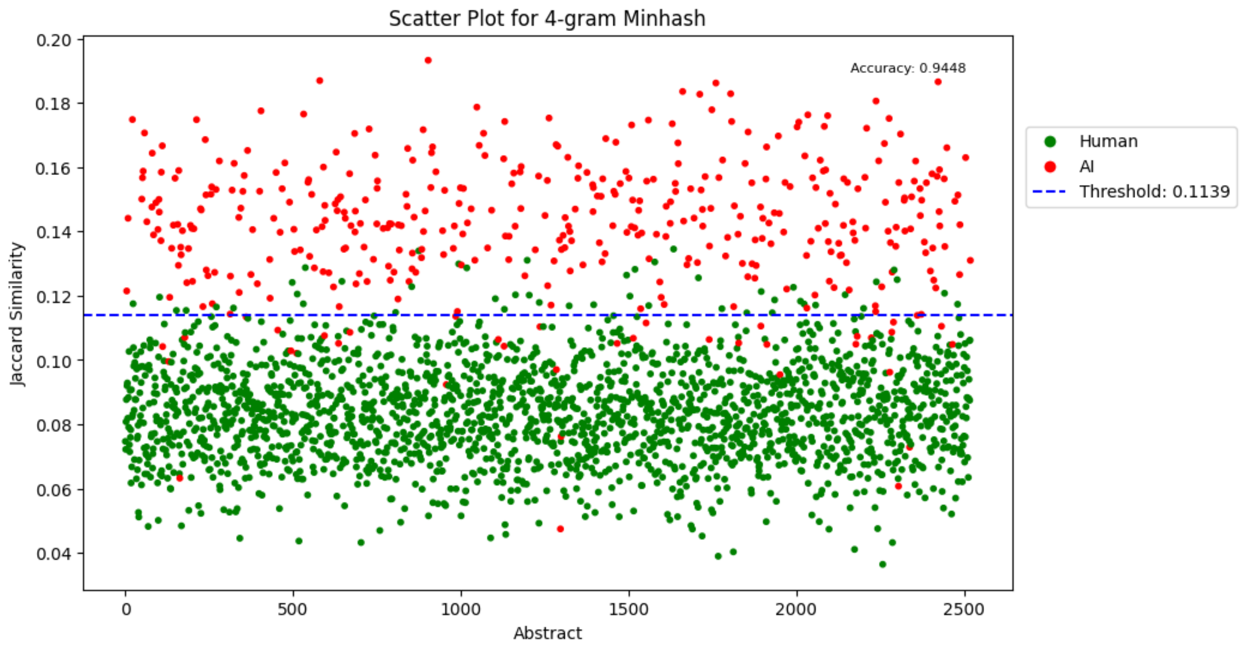
*This is for using k-shingle with BertTokenizers. Where threshold percentage represents values of y.

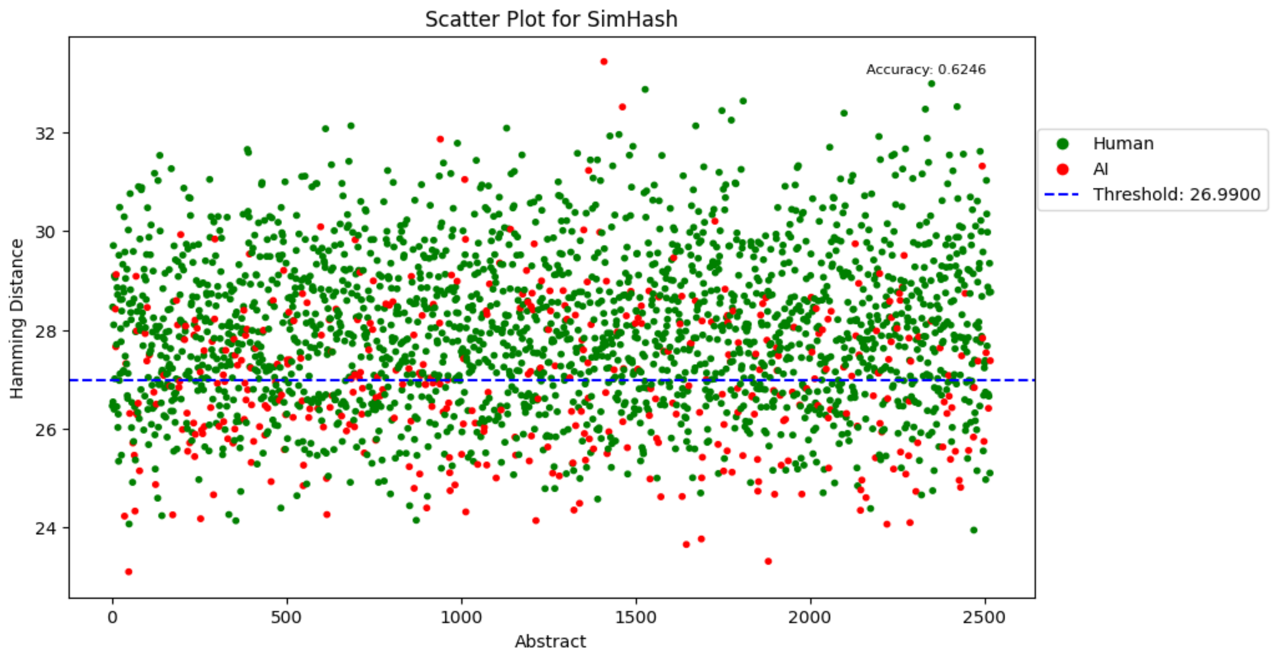




*This is for using k-shingle with BertTokenizers. We obtained the highest rates of accuracy using this method.


Scatter Plot for 10-shingle Minhash
Accuracy: 0.9840
Human
AI
Threshold: 0.0116
Jaccard Similarity
Abstract
0.05
0.04
0.03
0.02
0.01
0.00
0
500
1000
1500
2000
2500


Scatter Plot for 4-gram Minhash
Accuracy: 0.9448
Human
AI
Threshold: 0.1139
Jaccard Similarity
Abstract
0.20
0.18
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0
500
1000
1500
2000
2500



Case Study: We contend that the most impactful application of our findings lies in prompt-based essay classifications. In this context, we envision using essays generated by a transformer as the baseline for comparison. Unfortunately, we were not able to get data from courses at Rice University due to privacy concerns. However, to test out our hypothesis, we made a simple model with 3 ChatGPT generated essays for an architectural prompt-based essay, and used Yassin's submitted human generated essay; one of our models was able to correctly classify it, along with a different ChatGPT generated essay. It is noteworthy that our success was achieved through some prompt engineering, which was implemented to enhance the diversity of our training set.

As proof of concept, this serves as a possible commercial implementation of our methodology. We could develop a system with say, 50 ChatGPT calls (with prompt engineering) serving as the testing set and could classify essays from this. We are currently looking into the viability of providing such a system and extending out this project.

1 paragraph conclusion of why you think you have justified 1 and 3:

In our investigation of various similarity classifications, including MinHash, SimHash, cosine similarities, we have gained valuable insights into their applicability to our specific use case: the identification of LLM generated content in academic work, specifically research abstracts. Effectively identifying parameters that align with our requirements has enabled us to consistently and accurately detect LLM generated abstracts in a significant majority of cases. Through this process, we have also realized the limitations of specific parameters, such as the challenges posed by tokenizations that are either too small or too large. We feel this is strong proof of work for similarity classification through MinHash in an increasingly relevant problem space of detecting LLM generated content.

References

[1] J. Kirchenbauer, J. Geiping, Wen, J. Katz, I. Miers, and T. Goldstein, “A Watermark for Large Language Models,” arXiv: 2301.10226 [cs.LG] , June 2023

[2] L. Fröhling and A. Zubiaga, “Feature-based detection of automated language models: tackling GPT-2, GPT-3 and Grover,” *PeerJ. Computer science*, vol. 7, e443, Apr. 2021