CSCI 49362: Natural Language Processing

Determining Machine-Generated Text from Human-Written Text

Daniel Perez Alfaro

Abstract

This study examines Machine-Generated and Human Written Text to highlight possible distinctions and eventually create accurate classifications. We explore a supervised labeling approach where a generative language model such as ChatGPT, produces responses based on short input from Human Wikipedia entries, creating binary representations. Features are essentially extracted from each respective class and later used to train a Gaussian Naive Bayes Classifier to distinguish between Human and Generated Text. So far, the model's best performance yields an accuracy score of 0.644, while on average yields an accuracy score of roughly 0.55942, significantly worse, suggesting much-needed improvement.

1 Introduction

Recently, language models have increasingly gained notoriety as they have become more accessible to the public, prompting an urge to meet demand through rapid business integration. However, an ethical dilemma begins to take form as content moderation becomes significantly more challenging when finely-tuned articles can almost achieve the sophistication of the human-written text. In a recent study titled "Machine Generated Text: A Comprehensive Survey of Threat Models and Detection Methods"(Crothers et. al, 2022), the researchers identified four major categories within their Threat Models where attacks were likely to cause significant social disruption.

These areas of concern were Facilitating Malware and Social Engineering, Spam and Harassment, Exploitative Authorship, and Online Campaign Influence. Each category was then divided into subcategories, where the consensus remained the same; to cause and facilitate malicious intent. The disruptive nature of Natural Language Generative systems becomes apparent, warranting concern for creating countermeasures for detecting and preventing malicious use and effects of these language models. The researchers suggested several Feature-Based, as well as Neural Language model, approaches to consider, some of which laid the foundation for this study.

2 Previous Work

A common approach in determining the authorship of a text is to use Stylometric techniques, or the statistical analysis for measuring the stylistic features of a text(Gomez Adorno et. al, 2018). Stylometry could examine a variety of features such as word length, function word counts, punctuation, as well as N-grams(Wermer-Colan, 2018).

However, when examining the work of research within the field of language models, previous techniques to identify authorship have proven to be rather ineffective. For instance, the study "The Limitations of Stylometry for Detecting Machine-Generated Fake News" explores how traditional Stylometry methodologies are limited when attempting to differentiate between genuine applications of language models and those that produce false information.

To assess the effectiveness of a classification model, both human and language model-generated fake news articles were subjected to stylometric techniques. The findings indicated that Stylometry was less successful in identifying machine-generated fake news compared to human-authored content, emphasizing the success of a language model's ability to closely mimic human writing styles. The study recognizes that stylometric techniques alone, may not be effective, however, adopting alternatives such as machine learning algorithms and advanced benchmarks for NLP techniques, could create significantly better outcomes for determining maliciously machine-generated content.

A recent 2020 study titled "Automatic Detection of Machine Generated Text: A Critical Survey"

Generate Intro	Wiki Intro
"In mathematics, spec..."	"In mathematics..."
"In Finnish Folk..."	"In Finnish Folk..."
"Robert Milner..."	"Robert Milner..."

Table 1: Example of dataset used for feature extraction from https://huggingface.com

addresses how Text Generative Models can create misleading responses and provide insight into their potential misuse. The study examines more nuanced features common throughout human writing that serve as viable indicators for detecting the use of language models. The researchers suggest that a LM can often leave detectable signatures in which a classifier can successfully be trained to identify. In previous classification models, the Bag of Words approach was shown to perform on par with more complex encoders. However, the study analyzes how features such as fluency, shortness, validity, and grammatical correctness of a text can provide significant discernible qualities when identifying authorship.

3 Data Collection

A large collection of 100,000 ChatGPT queries alongside human written Wikipedia entries was created by calling the OpenAI API to generate a short introduction given the first few lines of a Wikipedia article. The dataset also contains information such as the Wikipedia URL for querying, the generated and human written text, as well as the length for each text, measured in words as a form of token.

3.1 Preprocessing

In the process of analyzing the data, the initial focus was on eliminating irrelevant details that did not contribute significantly to the overall classification. These included removing Wikipedia URLs, prompts, and irrelevant measurements within the columns. Additionally, any null rows and unwanted columns were excluded from the data frame after importing the CSV file.

Regarding the textual data, which comprised both machine-generated and human-written text, it was necessary to perform preprocessing techniques such as tokenization for more complex feature engineering. However, keeping the text in its original form was done to accurately determine the count of grammatical errors. Tokenization and further normalization could have compromised the accuracy of this assessment.

Wiki Entries	Gen. Entries	Total
10,000	10,000	3,252,291
150,000	150,000	48,821,139

Table 2: Number of entries and associated word totals.

Total Number of Entries	Model's Score
20,000	0.6105
300,000	0.640333

Table 3: Number of entries and associated model's score.

4 Baseline

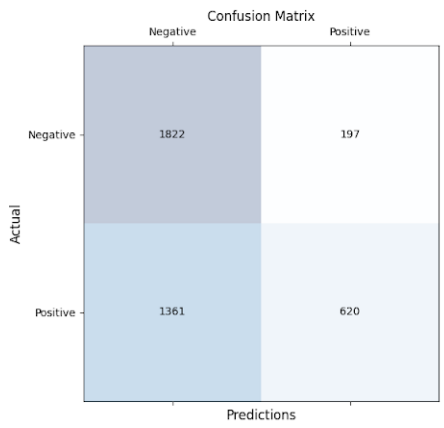
Table 2 represents a select number of entries along with their total number of words within that dataset. Selecting 10,000 entries from both Generated and Written columns, creates a collection of texts twice as large. Furthermore, extracting the token counts for each, highlights the number of words present within these datasets which could be used as a benchmark for understanding the data. As a baseline, a count of grammatical errors for each entry was extracted.

Listing 1: Grammatical Counts

```
# single grammar checking feature
def get_grammar(text):
    errors = grammar_check(text)
    return len(errors)
```

The idea being that because machine-generated-text, on average, creates fewer grammatical errors, a simple model to classify written and generated texts could rely solely on the count of the errors present within a text. When extracting the counts of grammatical errors using python's language tool, each entry consisted of a single feature with a target classification of true or false for Wikipedia and Machine Generated text, respectively. A Gaussian Naive Bayes model was used as a Baseline for identifying classifications with the resulting scores in Table 3.

The processed data frame was first used by adding new columns for the count of errors in Generated and Written texts, which were later extracted and reshaped as a 2-dimensional array to use as the set of features. The labeled data is created as a simple classification of either true or false values.



When evaluating the performance of the model, a confusion matrix was used to measure how well the model was able to accurately identity classifications for all possible classes. For an analysis of the first 10,000 rows of the data set, or 20,000 total entries, the model produced a score of 0.6105, while the total entries produced a slightly improved score of 0.640333. A limitation, however, is the single feature that creates a rather unrealistic expectation to the complexity of modern language models.

5 Methodologies

Having established a vastly oversimplified baseline model, multiple features were to be extracted in an attempt to account for a modern Natural Language Model's complexity.

5.1 Cosine Similarity

When examining the similarity between two words, a word and it's most frequent synset could be used to draw comparisons based on the words and their relative context. However, doing so proves to be time consuming and resource intensive. This study chooses to, instead, represent sentences as vectors once tokenized and transformed.

Building from the idea that a paragraph with a larger dot product or smaller theta, would have sentences more closely related. This becomes useful when attempting to determine whether human written or generated text, typically maintain concise and on topic sentence structure.

Cosine Similarity is essentially a normalized dot product of two vectors. For this approach, each paragraph or text, was first tokenized, computing the cosine similarity of each succeeding sentence, to finally compute an average similarity score.

$$Similarity = \frac{1}{N}\Sigma cos\theta = \frac{p_i \cdot p_{i+1}}{||p_i||||p_{i+1}||}\frac{1}{N} \quad (1)$$

Listing 2: Similarity Implementation

```
# sentence tokenization
tex = sent_tokenize(words)
# First Sentence
for i in range(len(tex)):
    # Second Sentence
    for j in range(i+1,len(tex)):
        # appending scores
        score = similarity(t[i],t[j])
        score_list.append(score))
    return np.mean(score_list)
```

5.2 Polarity

Sentiment Analysis often plays a role in identifying the meaning behind a given text. Likewise, Polarity scores can be used to denote a text's relative sentiment by returning a score between [-1,1] where negative and positive expressions lie on opposite extremes of one another. For the purpose of this study, a polarity score was extracted from both Generated and Written entries with the idea that human entries would on average include more biased sentiment, as opposed to a language model whose text would remain more neutral.

5.3 Bigram Model

As a continuous sequence of N words, N-Gram models are useful in the understanding and analysis of word patterns and their associations. With numerous applications across NLP, the Bigram model was chosen for this study's feature extraction to potentially identify common word pairs and expressions within both classes. Once each text class was tokenized and removed of any stopwords and punctuation, a standard count of the ten most common Bigrams was returned.

5.4 Procedure

The process of feature extraction begins by first removing any rows where the text from each class can not be tokenized into sentences such that the initial and following tokenized text are not the same. This accounts for when the structure of a text is in any other form that can lose valuable information. Once, each row can be tokenized, and labeled as 'correct', all remaining rows are removed. In doing

so, the dataset decreased by a few thousand entries, leaving the appropriately tokenized sentences.

A new data frame is created to concatenate both Written and Generated text while classifying each as true or false, respectively. A Cosine Similarity Score is computed for each pair of sentences in succession, within a text.

The Cosine Similarity function was designed so that if a a sentence within the dataset was not properly removed, and a computation would result in an error, the following exception would return nan. Before proceeding, all rows containing nan scores were dropped. Once again, removing data whose similarity score produced an error.

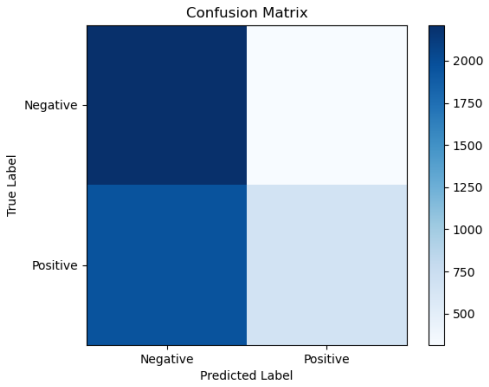
Both the Bigram Counts and Polarity scores are extracted by applying respective functions to the text columns and appending to the data frame.

Listing 3: Feature Extraction

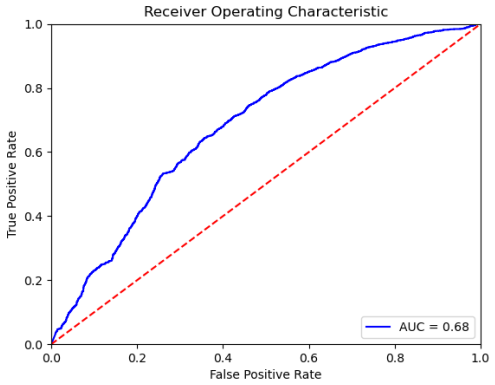
```
# removing rows w/o valid sent.
df = remove_false_tokens(df)

# computing cosine scores
df_new = similarScores(df)

# dropping NaN values
df_new = df_new.dropna()
```

6 Modeling and Evaluation

Finally, with the resulting data frame, both X and Y variables can be properly extracted and reshaped to perform a train-test-split. A Gaussian Naive Bayes Classifier is then used to fit the training data. When examining its performance, the model produces an accuracy score of 0.55942.

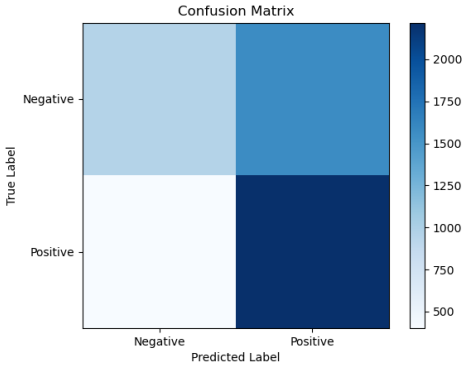


Using Accuracy as a stand alone measure, the model plots a greater true negative classification when compared to all possible instances. However, examining the model's ability to distinguish between classes reveals significantly worse performance.



Creating a graphical representation of the model's performance, the ROC curve for positive classification produces a score of 0.68 which is slightly better than randomly guessing, as highlighted with the comparison against the diagonal. This shift in performance could be attributed to multiple issues. For instance, upon data preprocessing, it's likely that a class imbalance could have been occurred as rows with improper sentence structures were removed, creating a bad fit.

6.1 Variations



Augmentation of the N-Gram by varying the number of n-gram counts typically resulted in roughly the same performance, if not worse. A possible explanation could be that when appending more counts, the model is essentially dealing with a sparse matrix, simply adding more noise. Feature selection played a pivotal role in the success of the model's performance. For instance, decreasing the number of Bigrams present within each text class yielded an accuracy score of 0.6113, almost on par with the original baseline.

7 Conclusion

From the previous results, it's not entirely unlikely for the model, along with the features selected to

accurately distinguish between machine-generated and human-written text. As before, choosing which attributes to examine is perhaps critical for attempting to identify any discernible characteristics of an article.

The poor performance in the model's ability to discriminate between classes, reveals an area of concern. The AUC highlighted the model's inability to accurately identify machine-generated text. In this study, the complexity of a modern language model was not taken into account. Previous research had suggested applying different techniques for Neural Network based models, suggesting much greater room for improvement.

Nonetheless, with the rapid development of text generative models, new techniques must be examined to accurately identify and moderate potentially malicious and dishonest content.

7.1 Future Improvements

In hopes of improving performance, it could be worth examining a classification model that recognizes biases in certain tokens. Additionally, Language Models that are trained on solely domain-specific information, could help create models that recognize coherency and fact-based content. For instance, LMs may not be fully up to date with recent world events and so may produce predictable inaccurate information.

While not examined within this study, a change in the model could produce noteworthy results. A clear consideration is through the use of ensemble methods. These techniques use various iterations through different models to reduce the variance while not increasing bias.

References

Abdul-Mageed et. al. "Automatic Detection of Machine Generated Text: A Critical Survey." University of British Columbia, 2020

Bhat, Aaditya. "GPT-Wiki-Intro." Huggingface, 2023, huggingface.co/datasets/aadityaubhat/GPT-wiki-intro

Bonthu, Harika. "Rule-Based Sentiment Analysis in Python for Data Scientists." Analytics Vidhya, 2021 www.analyticsvidhya.com/blog/2021/06/rule-based-sentiment-analysis-in-python/

Crothers, Evan, et al. "Machine Generated Text: A Comprehensive Survey of Threat Models and Detection Methods." ArXiv, 2022, /abs/2210.07321.

Gomez Adorno, Helena Montserrat, et al. "Stylometry-Based Approach for Detecting Writing Style Changes in Literary Texts." Computación Y Sistemas, vol. 22, no. 1, 30 Mar. 2018,/ doi.org/10.13053/cys-22-1-2882

Ma, Yongqiang, et al. "AI Vs. Human – Differentiation Analysis of Scientific Content Generation." ArXiv, 2023, /abs/2301.10416

Reimers, Nils. "Semantic Textual Similarity — Sentence-Transformers Documentation." Www.sbert.net, Ubiquitous Knowledge Processing (UKP) Lab

Tal Schuster, Roei Schuster, Darsh J. Shah, Regina Barzilay; "The Limitations of Stylometry for Detecting Machine-Generated Fake News." *Computational Linguistics*, 2020

Trischler, Adam, et al. "NewsQA: A Machine Comprehension Dataset." ArXiv, 2016, /abs/1611.09830.

Wermer-Colan, Alex. "Research Guides: Stylometry Methods and Practices: Methods." Guides.temple.edu, 2018, guides.temple.edu/stylometryfordh/methods