Kathlalu at SemEval-2024 Task 8: A Comparative Analysis of Binary Classification Methods for Distinguishing Between Human and Machine-generated Text

Lujia Cao and **Ece Lara Kılıç** and **Katharina Will**
University of Tübingen
{lujia.cao, ece-lara.kilic, katharina.will}@student.uni-tuebingen.de

Abstract

This paper investigates two methods for constructing a binary classifier to distinguish between human-generated and machine-generated text. The main emphasis is on a straightforward approach based on Zipf's law, which, despite its simplicity, achieves a moderate level of performance. Additionally, the paper briefly discusses experimentation with the utilization of unigram word counts.

1 Introduction

This paper addresses the task of classifying textual data as human or machine-generated, focusing on Subtask A Wang et al. (2024) with monolingual English data. The rise of technologies like ChatGPT has led to a surge in the use of machine-generated content in academia and workplaces. The task is crucial for ensuring the authenticity of texts, especially as individuals may potentially claim authorship of machine-generated content as their own work, raising concerns about academic integrity. By focusing on Subtask A and utilizing English-language data, this research addresses the challenges associated with the increasing prevalence of machine-generated text in academic and professional contexts, offering effective classification methods. In this paper, we use simple methods based on linguistic intuition to distinguish between human and machine-generated text. Our primary approaches involve leveraging Zipf's Law as one method, and employing word unigram counts as another.

We explored multiple approaches, ultimately narrowing our focus to two strategies. Although we submitted only one approach for leaderboard consideration, we believe the other one offers valuable insights as well. Surprisingly, we found that our simple methods based on linguistic intuition can rival the performance of large language models in the same task. While this approach could have been applied to the multilingual track, regrettably, time constraints prevented us from pursuing this direction. We ranked in the middle compared to other teams participating in this task.

2 Background

In configuring our task, we utilized the subtask A monolingual training data to train all three approaches, encompassing texts from both humans and models such as ChatGPT OpenAI (2022), Cohere Cohere (n.d.), Davinci OpenAI (n.d.), and Dolly Hugging Face (n.d.). Each data entry included the text, its source, the model used, the assigned label (either 0 or 1), and a unique identifier. For development purposes, we employed the subtask A monolingual development data, which featured texts generated by humans and the Bloomz BigScience (n.d.) model, along with their corresponding source, model, label, and ID. The final test data exclusively included texts and their respective IDs, with all other information omitted. The output data comprised jsonl files containing only the text IDs and their predicted labels (either 0 or 1). These files were generated once using the development data to refine our approach and again for the final test data, aligning with the task objective of predicting the label for a given text using our approach.

Our research delved into the practical implementation of Zipf's Law for binary classification. While consulting Linders and Louwerse's paper Linders and Louwerse (2020), as well as Nguyen-Son et al. Nguyen-Son et al. (2017), we found theoretical mentions of its potential application. However, none of these sources provided an actual approach. In contrast, our approach involves the concrete implementation of Zipf's Law, resulting in a functioning system.

Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024), pages 399–402

3 System Overview

3.1 Zipf's Law

Following an extensive review of the literature concerning methodologies aimed at discriminating between human and machine-generated textual content, our inquiry identified Zipf's Law as a potentially promising avenue of investigation. Despite the limited prevalence of existing methodologies leveraging this distribution, we deemed it worthy of investigation. Our rationale for pursuing this direction stems from the observed advantages in terms of computational efficiency and simplicity compared to Large Language Model (LLM) based approaches, which typically incur higher computational demands.

Zipf's Law is characterized by the following equation.

$$f = \frac{C}{r^s}$$

- $f(r)$ represents the frequency of the rank rth term.
- C is a constant.
- s is the Zipf exponent, typically close to 1.

This formula illustrates the inverse relationship between the frequency of a term and its rank in a given dataset, with the Zipf exponent governing the rate of decline in frequency as rank increases.

Our code initiates by tokenizing the text into individual words, followed by the computation of each word's frequency within the text. This preliminary step is pivotal for acquiring the empirical frequency distribution of words. After computing word frequencies and their corresponding ranks, we fit a curve to the Zipfian distribution. This step takes into account the Zipfian distribution function, word ranks, and frequencies as input parameters. By optimizing the scaling parameter s of the Zipfian distribution, fitting the observed data to a curve reveals the text's adherence to Zipf's law. This process aims to determine the optimal parameters (such as the scaling parameter s and constant C) for the Zipfian distribution function.

Leveraging the parameter s, the Zipfian distribution, we computed mean values for texts of label 0 and label 1. Subsequently, we determined their midpoint (-0.125) to serve as the threshold for classifying a text as either label 0 (human-generated) or label 1 (machine-generated).

In the system overview, following label prediction for the development data in subtask A monolingual track, we conduct a thorough evaluation by manually computing a preliminary F1 score using the actual labels as references. Our prediction process involves assigning labels based on Zipfian values, where values below the predefined threshold receive label 0 and those exceeding it are labeled as 1. We systematically apply the Zipfian distribution method to the texts extracted from the development data, facilitating precise label determination during subsequent analysis.

Our system obtained an F1 score of 0.72 on the development set. We used the same threshold/model to predict the labels on the test set.

The F1 score obtained on the official leaderboard for this approach yielded a value of 0.729. This metric provides a robust assessment of our approach's performance in the context of the shared task.

	label 0	label 1
min	-0.539	-1.778
max	2.212e-09	-1.550e-10
mean	-0.111	-0.139

Table 1: Zipfian distribution of labels 0 and 1

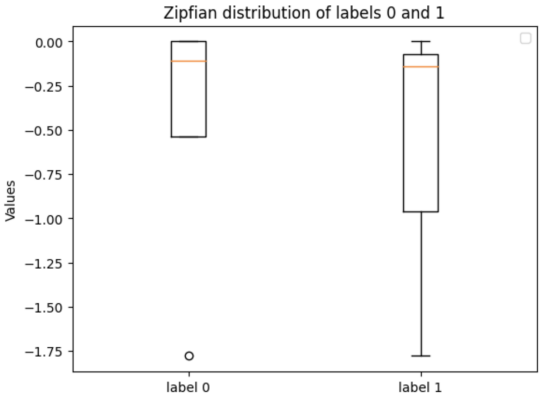

Figure 1: Zipfian distribution of labels 0 and 1

3.2 Unigram

Although we did not submit the predictions of the unigram approach, we will clarify its setup here. This method mirrors the structure of the Zipf's Law approach, yet diverges in its focus on calculating the number of words per text.

The mentioned values led us to establish a threshold of 450.303. With this threshold in place, the prediction process commenced: texts with word counts surpassing it were predicted as 0, indicating

	label 0	label 1
min	2	6
max	33220	2665
mean	583.755	316.850

Table 2: Word counts of labels 0 and 1

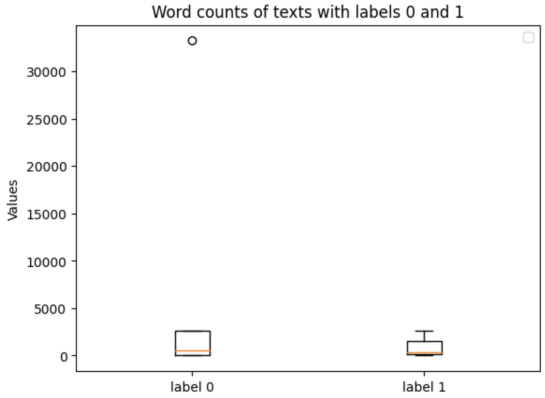

Figure 2: Word counts of labels 0 and 1

a human author, while those falling below were labeled as 1, indicating machine-generated text. Subsequently, we proceeded with text extraction for the test data, calculating the word counts for each text. We made predictions based on this.
Our preliminary F1 score for this approach stood at 0.59 on the development set, which, while respectable, fell short of the performance achieved by the Zipf's Law approach. This discrepancy led us to opt against pursuing further development of the unigram approach.

3.3 Comparing Zipf's Law and Unigram

The unigram method focuses on capturing information related to word frequencies, providing insights into the overall lexical diversity and richness of the text. In contrast, the Zipf method leverages the distributional characteristics of word frequencies, emphasizing patterns of occurrence and rank-order relationships. Together, these approaches offer complementary perspectives on textual content, enabling a more comprehensive analysis of linguistic features.
The unigram method may excel in scenarios where the distribution of word frequencies significantly impacts classification outcomes, such as detecting texts with distinct lexical signatures or vocabulary usage patterns. On the other hand, the Zipf method may prove more effective in identifying structural patterns and deviations from expected frequency distributions, particularly in texts generated by language models with predictable language patterns. While the unigram and Zipf methods differ in their primary focus and underlying principles, there is some overlap in the information they capture. The unigram method operates at the level of individual word occurrences, providing insights into the frequency and distribution of specific terms within the text, while the Zipf method considers the broader distributional patterns of word frequencies, focusing on rank-order relationships and overall distribution shapes.

4 Experimental Setup

For implementing the Zipf's Law and word unigram approach, we relied on Counter Python Software Foundation (2022), numpy NumPy (2022), and curve_fit from scipy.optimize SciPy (2022), without requiring any additional external tools or libraries. We utilized the provided data without creating additional splits. During testing on the development data, we employed the function f1_score from sklearn.metrics scikit-learn (2022).

5 Results

Our initial two approaches exhibit commendable performance in accurately predicting labels. The F1 score for the labels predicted by the Zipf's Law approach was 0.729 in the official ranking, with the task organizers' baseline set at 0.884. Our submission secured the 83rd position out of 137 in the official rankings. The preliminary F1 score for the unigram approach was 0.60, reflecting the test phase; however, this result was not included in the final submission.

6 Conclusion

In conclusion, we are satisfied with the performance of our Zipf's Law system in the shared task, particularly given its simplicity compared to other model-based approaches. The unigram system demonstrated commendable performance as well. We also explored training a linear Support Vector Classifier (SVC) scikit-learn (n.d.a) using character n-grams and employing a sublinear tf-idf scikit-learn (n.d.b) approach. We integrated several models, partitioning the training data into distinct files representing specific models used, such as ChatGPT, Cohere, Davinci, and Dolly, ensuring a balanced distribution of human-generated texts across all model categories. Despite our careful

preparations, all four models unexpectedly produced identical labels for all texts during prediction, rendering the system ineffective. This unexpected outcome highlights the necessity of rigorous testing and debugging to ensure the reliability of our methods. Identifying and resolving the underlying issues will be crucial for future improvements in model performance and credibility. Moving forward, we intend to enhance both the Zipf's Law and unigram systems through a comprehensive review of relevant literature. Additionally, we're dedicated to fixing the bug in our tf-idf vectorizer to maximize its potential in future iterations.

References

BigScience. n.d. Bloomz. Hugging Face. Accessed: February 13, 2024.

Cohere. n.d. Cohere. Cohere. Accessed: February 13, 2024.

Hugging Face. n.d. Databricks/dolly-v2-12b. Hugging Face. Accessed: February 13, 2024.

Guido M. Linders and Max. M. Louwerse. 2020. Zipf's law in human-machine dialog. In *Proceedings of the 20th ACM International Conference on Intelligent Virtual Agents*, IVA '20, New York, NY, USA. Association for Computing Machinery.

Hoang-Quoc Nguyen-Son, Ngoc-Dung T. Tieu, Huy H. Nguyen, Junichi Yamagishi, and Isao Echi Zen. 2017. Identifying computer-generated text using statistical analysis. In *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1504–1511.

NumPy. 2022. NumPy: Array processing for numbers, strings, records, and objects. Accessed: February 13, 2024.

OpenAI. 2022. ChatGPT. AI language model. Accessed: February 13, 2024.

OpenAI. n.d. DaVinci. DaVinci. Accessed: February 13, 2024.

Python Software Foundation. 2022. Python Collections Module. Accessed: February 13, 2024.

scikit-learn. 2022. scikit-learn: Machine Learning in Python. Version 1.0.2.

scikit-learn. n.d.a. Linear SVC. scikit. Accessed: February 18, 2024.

scikit-learn. n.d.b. tf-idf. scikit. Accessed: February 18, 2024.

SciPy. 2022. SciPy: Scientific Library for Python. Accessed: February 13, 2024.

Yuxia Wang, Jonibek Mansurov, Petar Ivanov, jinyan su, Artem Shelmanov, Akim Tsvigun, Osama Mohammed Afzal, Tarek Mahmoud, Giovanni Puccetti, Thomas Arnold, Chenxi Whitehouse, Alham Fikri Aji, Nizar Habash, Iryna Gurevych, and Preslav Nakov. 2024. Semeval-2024 task 8: Multidomain, multimodel and multilingual machine-generated text detection. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, pages 2041–2063, Mexico City, Mexico. Association for Computational Linguistics.

Acknowledgements

We extend our appreciation to Çağrı Çöltekin for his assistance during this shared task.