MODEL SUMMARY

Competition: LLM - Detect AI Generated Text
Team Name: Davide Cozzolino
Private Leaderboard Score: 0.969132
Private Leaderboard Place: 6
GitHub repository: https://github.com/davin11/entropy-based-text-detector

Team Member
Name: Davide Cozzolino
Location: Naples (ITALY)
Email: davide.cozzolino@unina.it

1. Background

I am an Assistant Professor at the University Federico II of Naples. I have a technical background in signal processing and programming (C/C++, Matlab, Python, Pytorch, and Keras). My research focuses on multimedia forensics. I decided to participate in this competition due to its relevance to my research area. My prior experiences in research significantly contributed to my success in this competition. I dedicated a focused effort, spending only four or five days during weekends and holidays.

2. Summary

The most important aspects of my approach are:

- I opted to employ a OneClass SVM trained exclusively on human-written essays. This choice is based on my prior experience in multimedia forensics, where I observed that training only on authentic data leads to better generalization, reducing the risk of overfitting [1].
- I used synthetic features based on entropy, drawing inspiration from literature on text classification [2].
- The entropy-based features leverage word probabilities computed through a pre-trained Large Language Model. I utilized a recent LLM, Phi-2 [3], which is licensed under the MIT license.

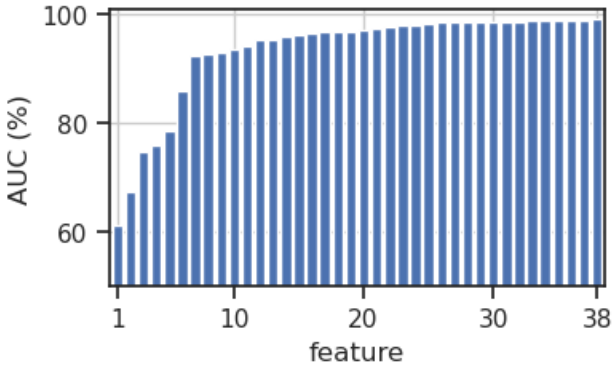
The entire process of feature extraction and training of the OneClass SVM takes approximately 20 minutes.

3. Feature Engineering

The used features are synthetic statistics (e.g., median, standard deviation, mean, percentiles) of two vectors. The first vector contains the expected amount of information (entropy) for each word in the essay, while the second vector contains the information content (surprisal) associated with each word in the essay. Additionally, I included the number of words and the average number of characters per word as features.

4. Feature Selection

Starting from a set of 38 features, I used the performance on DAIGT V2 Train Dataset (https://www.kaggle.com/datasets/thedrcat/daigt-v2-train-dataset) to identify the most effective ones. The following bar chart illustrates the performance, measured in terms of AUC (Area under the ROC Curve), using each feature individually on DAIGT V2 Train Dataset.



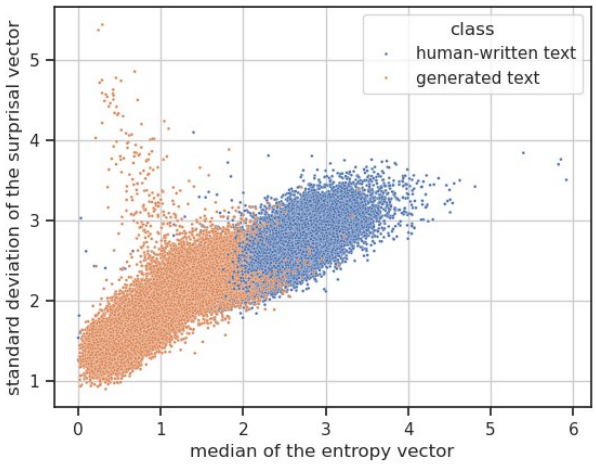
During the competition, I tried various subsets of features. Below are the results of my submissions with the corresponding number of used features.

#Features	Public Score	Private Score
7	0.746763	0.627362
2	0.766609	0.662438
4	0.766812	0.693086
4	0.789550	0.746763
2	0.819771	0.709330
3	0.872820	0.901595
5	0.873312	0.968132

My second-best solution uses 3 features: median of the entropy vector; median of the surprisal vector; average number of characters per word.

My best solution uses 5 features: median of the entropy vector; median of the surprisal vector; average number of characters per word; standard deviation of the surprisal vector; 5th percentile of the entropy vector.

In the following, there is the scatter plot of two used features on DAIGT V2 Train Dataset:



5. Training Method

I employed One-Class SVM with an RBF kernel as classifier. The classifier was trained exclusively using the human-written essays provided by the organizers as training-set.

6. Interesting findings

I believe the most crucial aspect of my approach is the use of the one-class training strategy. This seems to represent the main difference between my solution and those of most of the other participants in this competition.

7. Simple Features and Methods

The solution utilizing only two features (median of the surprisal vector and average number of characters per word) provided a private score of 0.709330, which is significantly lower than my best score of 0.968132.

8. Model Execution Time

The feature extraction process takes 18 minutes for 1378 essays using an NVIDIA Tesla P100.
The training of the OneClass SVM is completed in just few minutes.

9. References

[1] Davide Cozzolino, and Verdoliva Luisa. "Noiseprint: A CNN-based camera model fingerprint." IEEE Transactions on Information Forensics and Security, 2020.

[2] C. Huang, et al. "Approximating Human-Like Few-shot Learning with GPT-based Compression." arXiv preprint arXiv:2308.06942, 2023.

[3] M. Javaheripi, et al. "Phi-2: The surprising power of small language models." https://huggingface.co/microsoft/phi-2