Artificial text detection via magnitude functions

Hassan Iftikhar Pavel Gurevich

Abstract

Rapid advancement of generative language models has blurred the distinction between human- and machine-generated text, posing significant challenges for fakes detection, verification, and academic integrity enforcement. Current detection methods often rely on lexical or statistical features, which struggle to generalize as the models evolve. To address this, we propose a novel geometric framework that utilizes magnitude functions, a tool from topological data analysis (TDA), to quantify structural differences in text representations derived from language model embeddings.

By mapping texts into high-dimensional embedding spaces, we analyze their geometric "shape" through the magnitude function, which captures the multiscale topological properties of point-cloud distributions. We hypothesize that human-authored texts exhibit distinct organizational patterns (e.g., coherence, semantic consistency) compared to AI-generated outputs, which may manifest as quantifiable disparities in magnitude function profiles.

This work bridges machine learning and topological analysis to advance AI-generated text detection, offering a scalable, model-agnostic solution. Our findings underscore the potential of geometric features as interpretable markers for synthetic text, with implications for trustworthiness in online social, scientific and educational content.

1. Introduction

Modern generative Large Language Models (LLMs), such as GPT-4, Mistral, and DeepSeek, performed a revolution in the text creation area. Now, generated texts are extremely similar to human-made ones. LLMs are used for text translation, polishing, grammar check, style correction and many other tasks. However, such models may also be used for some bad purpose, like online bullying, cheating, fraud activities, or spam. That leads to the need of some effective and robust detection technique, which could allow to distinguish LLM-generated and human-written texts with high accuracy.

Three major text detection approaches could be highlighted:

1. Perplexity-based algorithms (Kushnareva et al., 2024)
2. Statistics/stylistic based techniques (Juzek & Ward, 2025)
3. Supervised models trained on text embeddings (Hu et al., 2023)

Unfortunately, all of these methods have drawbacks that will be discussed in Section 2.

We propose the paradigm shift: instead of analyzing text through lexical or probabilistic lenses, we interrogate the geometric organization of language in embedding spaces. Here we introduce magnitude functions (Leinster & Meckes, 2017), a tool that quantifies multi-scale geometric complexity in point cloud data. By treating a text's embedding sequence as a topological space, we compute its magnitude function, a curve encoding the "shape" of the text's semantic trajectory at varying scales. We posit that magnitude profiles capture intrinsic differences in how humans and LMs organize information.

This work bridges machine learning and computational topology, offering insights into the geometric fingerprints of machine intelligence. Beyond detection, our findings illuminate fundamental differences in how humans and LLMs structure language — a step toward more interpretable AI accountability frameworks.

2. Related work

Many researchers have tried to figure out how to spot AI-generated text. The most significant methods are listed below:

Linguistic Feature-Based Methods: (Ippolito et al., 2020) discovered how often common words (like "the" or "and") appear, how long sentences are, or how punctuation is used. They found AI texts often overuse certain words, which can be a giveaway. This method is extremely fast and easy to get, but it does not work well with newer AI models that seem more human. It is great for catching obvious fakes but not so good for tricky ones.

Supervised Learning: (Hu et al., 2023) proposed RADAR,

Table 1. Comparison of Detection Methods

Method	Strengths	Weaknesses
Magnitude Function	- Checks the shape of word points, a new idea. - Looks at multiple scales for small and big differences. - Can mix with other methods for better results.	- Needs a lot of computer power for math. - Hard to explain what the numbers mean. - Relies on a good word-to-point model.
Linguistic Feature-Based	- Quick and easy to use. - Clear why it flags a text. - Catches simple AI errors well.	- Struggles with human-like AI. - Misses deeper text structure.
Supervised Classification	- Very accurate when trained well. - Spots complex text patterns. - Can learn new AI tricks with retraining.	- Needs tons of data and computer power. - May fail on new AI models. - Ignores word shapes.

a framework to improve AI-generated text detection by adversarially training a paraphraser and a detector iteratively, enhancing robustness against paraphrasing attacks. They evaluated RADAR across 8 diverse LLMs and 4 datasets, demonstrating superior detection accuracy compared to existing methods, especially when paraphrasing is used.

Watermarking: (Kirchenbauer et al., 2023) developed a watermarking framework for proprietary LLMs to embed imperceptible, algorithmically detectable signals into generated text, enabling misuse detection without compromising output quality. Their method selectively promotes "green" tokens during text generation and uses an open-source detection algorithm with statistical tests to identify watermarked content, even without access to the model's API or parameters.

Statistical and Entropy-Based Methods: (Gehrmann et al., 2019) made a tool called GLTR that looks at how predictable words are in a text. AI tends to pick "safe" words that are very likely, making its writing less surprising than human text. This is a smart idea and works for some AI models, but advanced ones that mix things up can fool it, just like they fool word-based methods.

To see how our approach stacks up, here's a comparison with two of these methods.

3. Project plan

The main steps of our further work are listed below:

1. **Data Collection**: Collect the same number of real texts (like articles or essays by people) and AI-generated texts.
2. **Embedding Texts**: Use a BERT-like model to turn tokens to vectors. This makes a "point cloud" for each text, representing them in some text vector space.
3. **Magnitude Function**: Make a matrix Z_{tA} where each entry is $e^{-t\cdot d(x_i,x_j)}$ (distance between points, scaled by t). Solve $Z_{tA}w = 1$ to get a vector w, and the magnitude $|tA|$ is the sum of w_i. Do this for different t values to see the shape at different zoom levels.
4. **Feature Extraction**: Pick numbers from the magnitude function, like $|tA|$ at $t = 1, 5, 10$, to use as features for the classifier.
5. **Classification**: Train a simple model to guess if a text is human or AI based on these features.

3.1. Promises

Here we select the major points why we believe that our approach will work:

- **Geometry-Based**: This method looks at how words are spread out in a math space, not just what words are used. Real texts might have words that "cluster" together, like ideas that connect, while AI texts could be more random.
- **Multi-Scale Analysis**: By checking the shape at different t values, we can see both tiny details and the big picture. Maybe AI texts look weird when you zoom in or out.
- **Can Complement Other Methods**: We could combine my shape features with word counts or other tricks to make a stronger classifier.

3.2. Challenges

- *Computational Cost*: Direct inversion of Z_t scales cubically in the number of tokens; we mitigate this via iterative solvers.
- *Interpretability*: While magnitude curves are mathematically grounded, connecting specific geometric features to linguistic phenomena remains an open challenge.
- *Embedding Quality*: Detection efficacy depends on the representational power of the embedding model.

3.3. Goals

Here's what we are planning next:

- Build a dataset with real and AI-generated texts.
- Obtain text embeddings using the BERT-like model.
- Implement the magnitude function calculation.
- Make a simple classifier to guess real vs. generated texts.
- Compare our method with other approaches.
- Prepare Final Report and store the code of all our experiments into the Github repo.

4. Theoretical Background

Consider the finite metric space A. Write Z_A for the $A \times A$ matrix as:

$$Z_A(a, b) = e^{-d(a,b)}, \tag{1}$$

where $a, b \in A$, and d is some distance function.

If Z_A is invertible, the magnitude of A is:

$$|A| = \sum_{a,b \in A} Z_A^{-1}(a, b) = \mathbf{1}^\top Z_A^{-1} \mathbf{1} \in \mathbb{R} \tag{2}$$

Magnitude may be understood as the *effective number of points*.

Magnitude assigns to each metric space not just a *number*, but a *function*.

For $t > 0$, write tA for A scaled by a factor of t.

For $t > 0$, the scaled space tA induces

$$Z_{tA}(a, b) = \exp\big(-t\, d(a, b)\big),$$

and the *magnitude function* is

$$\Phi_A(t) = |tA| \quad : (0, \infty) \to \mathbb{R}.$$

A magnitude function has only finitely many singularities. It is increasing for $t \gg 0$, and $\lim_{t\to\infty} |tA| = \text{cardinality}(A)$.

The magnitude function also contains information about the geometrical features of the space and encodes its dimension. Deeper math could be found in the paper (Leinster & Meckes, 2017).

5. Results and discussion

We conducted extensive experiments to evaluate the effectiveness of using magnitude-based features for detecting AI-generated texts. Below, we summarize the major findings and their implications.

5.1. Embedding Preparation and Dataset

For our task we choose a dataset with human-written and AI-generated texts from theDAIGT V2 dataset. The whole corpus (Kaggle) containing 50k human and GLM-generated texts across academic, creative, and technical domains.

To transform tokens to embeddings BERT model was used. For the experiments, a subset of 1000 texts (500 human; 500 AI-generated) was taken and transformed to matrices with embedding size equal to 768 and maximal number of tokens per text was chosen in range of [64, 320] with step of 32. Each document was thereby represented as a matrix of shape $max\_text\_length \times 768$, interpreted as a point cloud in high-dimensional space.

5.2. Magnitude Function Computation

Magnitude function calculation was implemented, Manhattan metric was taken as a distance.

Magnitude functions were computed using the Manhattan metric across 50 uniformly spaced t-values ranging from 0.005 to 0.04. Initial visualizations (Figure 1), indicated the separation trend in the magnitude function curves for human and AI-generated texts.

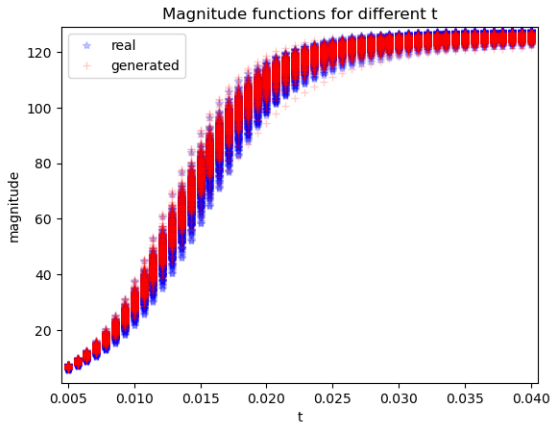


Figure 1. Magnitude function

However, any single value of t was insufficient to linearly separate the two classes with high confidence. As presented in Figure 2, the magnitudes of real and generated texts have shifted distributions, but they are close to each other.

Interestingly, that for the higher values of t, the distributions

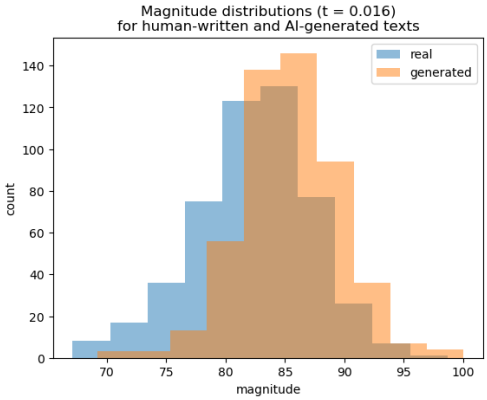


Figure 2. Magnitude distributions for $t = 0.016$

"swap", so the real texts become have higher magnitudes (Figure 3).

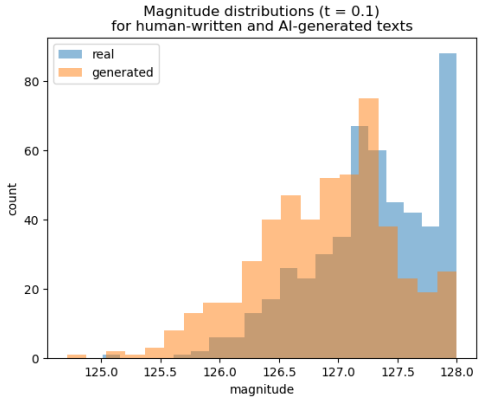


Figure 3. Magnitude distributions for $t = 0.1$

All this means, that the magnitude function values for different properly selected t may be valuable feature vector for real and generated texts distinguishing.

5.3. Generated text detection with magnitude function

5.3.1. Experimental setup

To compare different magnitude function parameters and our approach with embeddings-based one the logistic regression classifier with default ScikitLearn hyperparameters was used. Model performance evaluation employed the ROC-AUC metric, selected for its robustness in binary classification tasks. For more trustworthy evaluation, the Stratified K-fold (k=6) cross-validation (CV) was performed.

5.3.2. Maximal tokens number optimization

The first step in our pipeline is the obtaining embeddings from texts using some encoder, which is BERT in our study. Given the computational complexity ($O(n^2)$) of tokenization and subsequent embedding processes, we investigated optimal maximum token lengths to balance computational efficiency with model performance. An empirical evaluation examined token limits $\in$ 64, 96, 128, 160, 192, 224, 256, 288, 320, with magnitude functions computed at 200 uniformly spaced t-values (t $\in$ [0.001, 0.1]).

As demonstrated in Figure 4,classifier performance (mean ROC-AUC) increases with growing maximal text length, but after 288-tokens text length it reaches plateau and even loose some quality. So, we choose 288 as the optimal number of tokens.

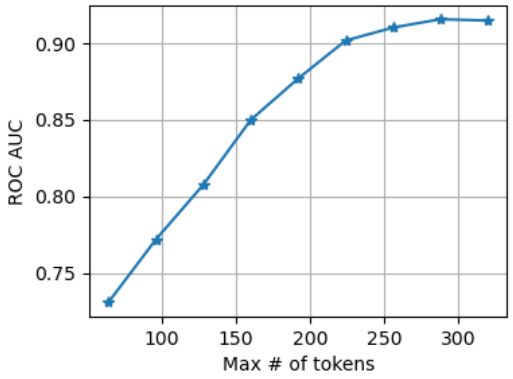


Figure 4. Model performance for different max # of tokens

It seems logical, that higher number of tokens gives more information about the whole text. But for better understanding, we compared the normalized (divided by the maximal # of tokens) curves obtained from magnitude functions of texts with max token length of 64 and 288 (Figure 5).

As it shown, for max token length of 288, for both real and generated texts magnitudes have much more variance and more differences between each other, which allows the classifier to learn more complex dependencies.

First two principle components of magnitude functions (Fig. 5, second row) are even more illustrative: for max token length 64 all dots (both real and generated) are mixed, while for length 288 dots corresponding for different classes are more clustered and distinct.

5.3.3. Magnitude Function Parameterization

Single value of the embedding matrix magnitude does not contain enough information to be the good feature. Instead, calculation of the magnitude function in some range of the parameter t may give different behavior for the human-made and AI-generated texts, as it shown in the Figure 1. However, magnitude function calculation is the expensive procedure. So, it is essential to determine the optimal range and points number of t.

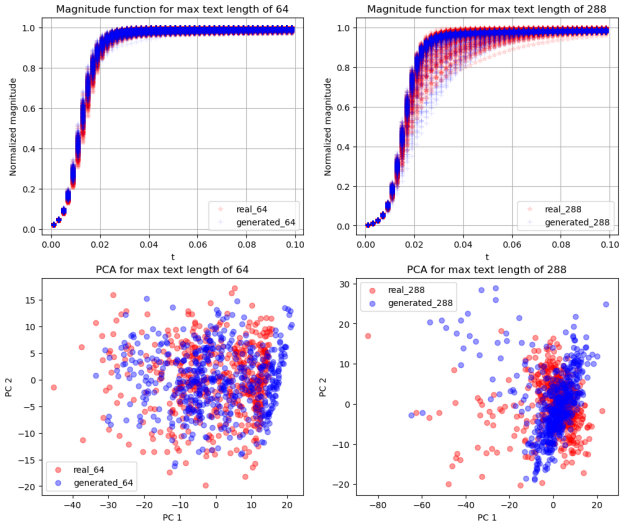


Figure 5. Magnitude function comparison for different # of tokens

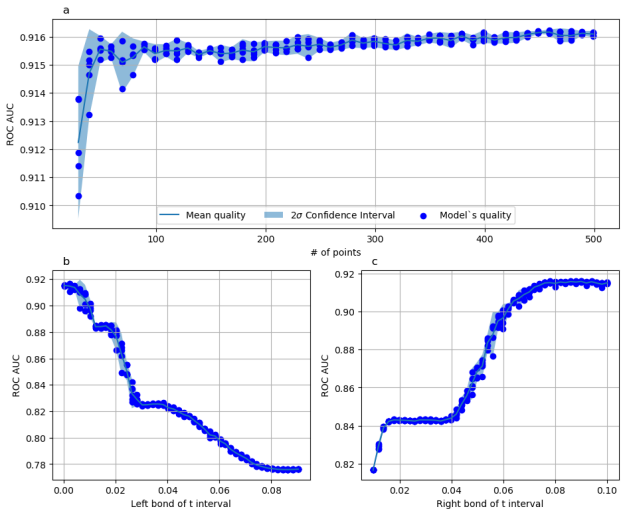


Figure 6. Optimization of range and points count of magnitude function parameter t. **a** – Prediction quality for different number of t points in the interval $[10^{-8}, 0.1]$. **b** – Prediction quality for different lower bounds of t interval with fixed number of points (50). **c** – Prediction quality for different upper bounds of t interval with fixed number of points (50).

To tackle the problem, the set of experiments was performed. As a preparation, we calculated the magnitude function for 500 values of t uniformly distributed in the interval of 10^{-8} and 0.1.

First, we took a number of uniform subsamples with different sizes to understand how dense should be the t-grid (Fig. 6**a**). The experiment setup is following:

1. Fix the t-range: from 10^{-8} to 0.1
2. Five times sample uniformly distributed t-values of selected sample size (from 30 to 500 with the step of 10).
3. For each sample train logistic regression 6-fold CV, evaluate the ROC AUC, calculate mean and standard deviation (σ) of the quality values.

As it shown, the ROC AUC slightly grows with increasing of sample size. However, all the values are in the range of 0.91-0.92. So, there is no dramatic quality changes. That means, we can pick, for example, 50 points of t in the appropriate interval and obtain good quality.

Next, we estimated the optimal lower and upper bonds for t interval with the fixed 200 t points (Fig. 6 **b-c**). We found that the lower bound should be as small as possible, while the optimal upper bound starts from approximately $t = 0.08$ where the quality reaches plateau. Interestingly, that in the interval of 0.02-0.04 the quality remains the same for both lower and upper bond optimization, which may mean that this interval is lack of significant information about the nature of text.

5.3.4. Comparison with text embedding-based classification

We conducted a systematic comparison between magnitude function-based classification and conventional text embedding approaches using equivalent experimental conditions. Both methodologies employed logistic regression (Scikit-learn default parameters) with six iterations of stratified k-fold cross-validation (k=6) on texts truncated to 288 tokens. Magnitude functions were computed across 300 linearly distributed t-points in the range of $[10^{-8}, 0.09]$, while text embeddings were generated through mean pooling of final-layer BERT token embeddings. An additional fusion model combined both feature spaces through vector concatenation. Results are presented in the Table 2.

Table 2. Magnitude: Inversion vs. CG

Magnitude function	Embeddings	ROC AUC
√	×	0.91541
×	√	0.99629
√	√	**0.99648**

The embedding-based approach demonstrated near-perfect discriminative capacity (AUC = 0.9963), suggesting the corpus exhibits strong stylometric differentiability in latent semantic space. In contrast, magnitude functions

alone achieved substantially lower performance (ΔAUC = -0.0809), indicating insufficient standalone discriminative power for this classification task.

Notably, feature concatenation produced a slight performance improvement. This suggests that while magnitude functions contain non-redundant signal orthogonal to semantic embeddings, their informational contribution remains subordinate to primary embedding features in this experimental context.

The observed performance hierarchy implies that magnitude functions may serve as supplementary features rather than primary discriminators for AI-generated text detection in stylometrically homogeneous corpora. Future work should investigate nonlinear fusion architectures to better leverage cross-modal feature interactions.

5.4. Efficient Magnitude Computation via Conjugate Gradient Optimization

Magnitude is a cardinality-like property of a metric space. For N objects represented as points in a metric space, we define a similarity matrix Z (1).

The magnitude is then given by 2.

This requires computing the inverse of Z, which is computationally expensive with complexity $\mathcal{O}(n^3)$. As the number of tokens per text grows, inversion becomes slow and impractical.

To address this, we solve the linear system:

$$Zw = \mathbf{1} \tag{3}$$

using the Conjugate Gradient (CG) method (see Algorithm 1). It's easy to notice, that

$$w = Z^{-1}\mathbf{1}, \quad \mathbf{1}^\top w = \mathbf{1}^\top Z^{-1}\mathbf{1} = |A|$$

And the resulting magnitude is simply:

$$|A| = \sum_i w_i \tag{4}$$

Algorithm: Magnitude-by-Conjugate-Gradient

We evaluated the CG method against direct inversion across random matrices with sizes $n \in 5, 10, 20, 50, 100, 1000$, measuring absolute error. Table 3 demonstrates the same results with negligible error across all scales.

For $t = 0.1$, we computed magnitudes using both the exact and CG methods for real and generated text.

Both methods produce the same results, confirming the validity of the CG approach (see A Fig. 7).

Timing Comparison

Algorithm 1 Conjugate Gradient for Magnitude

Input: Pairwise distances d_{ij},
tolerance ε, max iterations K
Output: Magnitude $|A|$

$Z_{ij} \leftarrow \exp(-d_{ij})$
$b \leftarrow (1, \ldots, 1)^\top$
$w \leftarrow 0$, $r \leftarrow b$, $p \leftarrow r$, $\rho \leftarrow r^\top r$, $k \leftarrow 0$
repeat
  $q \leftarrow Z \cdot p$
  $\alpha \leftarrow \rho/(p^\top q)$
  $w \leftarrow w + \alpha \cdot p$
  $r \leftarrow r - \alpha \cdot q$
  $\rho_{\text{new}} \leftarrow r^\top r$
  if $\sqrt{\rho_{\text{new}}} \leq \varepsilon \cdot \sqrt{n}$ **then**
    break
  end if
  $\beta \leftarrow \rho_{\text{new}}/\rho$
  $p \leftarrow r + \beta \cdot p$
  $\rho \leftarrow \rho_{\text{new}}$
  $k \leftarrow k + 1$
until $k = K$
return $|A| = \sum_i w_i$

Table 3. Magnitude: Inversion vs. CG

SIZE	$\lvert A\rvert_{inv}$	$\lvert A\rvert_{CG}$	ϵ_{abs}
5	1.5414	1.5414	5.8×10^{-14}
10	1.6546	1.6546	6.6×10^{-13}
20	1.7175	1.7175	2.4×10^{-8}
50	1.9088	1.9088	3.2×10^{-8}
100	1.9802	1.9802	3.6×10^{-7}
1000	2.0675	2.0675	3.2×10^{-7}

We measured the total and average computation times for computing the magnitude of 1000 text samples (Table 4).

Table 4. Magnitude computation time

METHOD	TOTAL TIME (S)	TIME PER SAMPLE (S)
INVERSION	17.77	0.0178
CG	11.59	0.0116

The CG method achieves a 1.53 × speedup with identical numerical outputs. This acceleration stems from CG's $O(n^2)$ complexity per iteration versus inversion's $O(n^3)$, making it scalable for large-scale text analysis. Figure 7 (Appendix) confirms equivalent magnitude profiles for real and synthetic texts across both methods.

This optimized approach enables efficient magnitude computation for high-dimensional embedding spaces, facilitating

practical applications in AI-generated text detection at scale.

6. Conclusions and Future Directions

6.1. Conclusions

This work establishes an optimized framework for AI-generated text detection through systematic analysis of magnitude functions and semantic embeddings. Key contributions include:

1. **Computational Efficiency**: The Conjugate Gradient method achieves 1.53× acceleration over direct matrix inversion (Table 4) while maintaining numerical equivalence ($\epsilon < 10^{-7}$ across n $\leq$ 1000, Table 3), enabling scalable magnitude computation for high-dimensional token sequences.

2. **Feature Optimization**: Token length optimization revealed an information saturation threshold at 288 tokens (Figure 4), with magnitude functions demonstrating enhanced class separability in PCA-reduced space.

3. **Comparison with Embedding-based Classification**: While semantic embeddings alone achieved near-perfect discrimination (AUC = 0.9963), magnitude functions provided complementary signal that marginally improved fused models (ΔAUC = +0.0002, Table 2), suggesting their role as supplementary features rather than primary classifiers.

4. **Accurate parametrization**: Magnitude functions require dense sampling in t $\in [10^{-8}, 0.08]$ to capture discriminative dynamics, with computational savings achieved through linear spaced sampling without performance loss (Fig. 6).

6.2. Future Work

Three strategic directions emerge for advancing magnitude-based text analysis:

1. Feature Fusion Architectures
    - Develop nonlinear fusion models (e.g., ANNs, kernel methods) to better leverage magnitude-embedding synergies
    - Investigate magnitude derivatives ($d|A|/dt$, or at logarithmic scale) as temporally resolved features
2. Algorithmic Scaling
    - Extend CG solver with preconditioning techniques for n ¿ 1000 token sequences
    - Implement GPU-accelerated magnitude computation via batched linear solves
3. Theoretical Enhancement
    - Formalize the relationship between magnitude trajectories and text perplexity metrics
    - Explore alternative metric spaces for magnitude computation
4. Generalization Validation
    - Benchmark performance on multi-lingual corpora and modern LLMs (GPT-4, Claude 3)
    - Analyze magnitude invariance to paraphrasing attacks and adversarial perturbations

The demonstrated efficiency gains and feature complementarity position magnitude functions as a viable auxiliary modality for AI text detection. However, their full potential requires addressing current limitations in discriminative power and theoretical interpretability through the outlined research trajectory.

References

Gehrmann, S., Strobelt, H., and Rush, A. GLTR: Statistical detection and visualization of generated text. In Costa-jussà, M. R. and Alfonseca, E. (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 111–116, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-3019. URL https://aclanthology.org/P19-3019/.

Hu, X., Chen, P.-Y., and Ho, T.-Y. Radar: robust ai-text detection via adversarial learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Ippolito, D., Duckworth, D., Callison-Burch, C., and Eck, D. Automatic detection of generated text is easiest when humans are fooled. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J. (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1808–1822, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.164. URL https://aclanthology.org/2020.acl-main.164/.

Juzek, T. S. and Ward, Z. B. Why does ChatGPT "delve" so much? exploring the sources of lexical overrepresentation in large language models. In Rambow, O., Wanner, L., Apidianaki, M., Al-Khalifa, H., Eugenio, B. D., and Schockaert, S. (eds.), *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 6397–6411, Abu Dhabi, UAE,

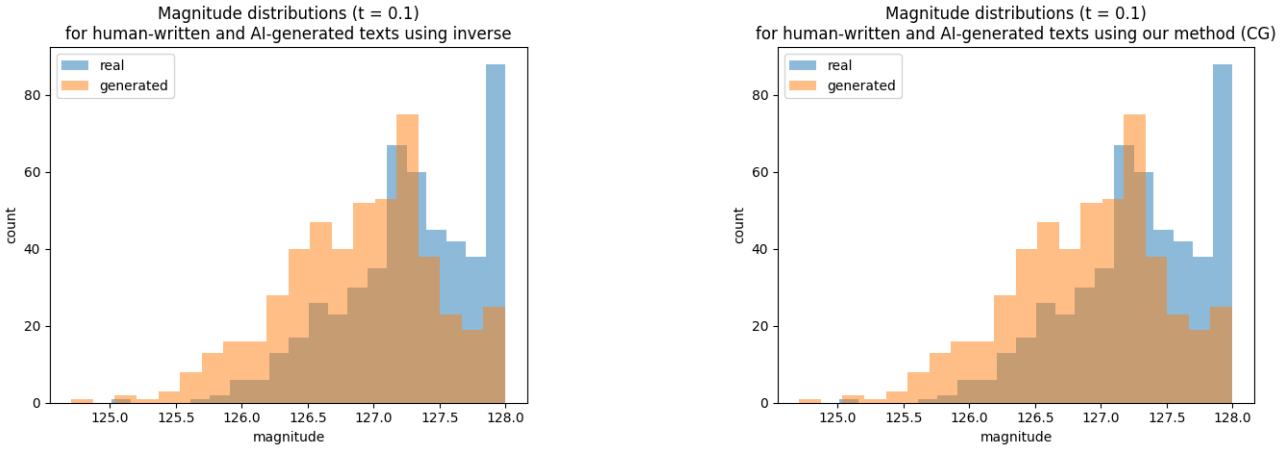
January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.426/.

Kirchenbauer, J., Geiping, J., Wen, Y., Katz, J., Miers, I., and Goldstein, T. A watermark for large language models. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 17061–17084. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/kirchenbauer23a.html.

Kushnareva, L., Gaintseva, T., Magai, G., Barannikov, S., Abulkhanov, D., Kuznetsov, K., Tulchinskii, E., Piontkovskaya, I., and Nikolenko, S. Ai-generated text boundary detection with roft, 2024. URL https://arxiv.org/abs/2311.08349.

Leinster, T. and Meckes, M. W. *The magnitude of a metric space: from category theory to geometric measure theory*, pp. 156–193. De Gruyter Open, December 2017. ISBN 9783110550832. doi: 10.1515/9783110550832-005. URL http://dx.doi.org/10.1515/9783110550832-005.

A. Appendix



(a) (b)

Figure 7. Side-by-side comparison of average magnitude values for real vs. generated texts