Text Graph Neural Networks for Detecting AI-Generated Content

Andric Valdez-Valenzuela
Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas
CDMX, Mexico.

Helena Gómez-Adorno
Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas
CDMX, Mexico.

Manuel Montes-y-Gómez
Instituto Nacional de Astrofísica, Óptica y Electrónica
Puebla, Mexico.

Abstract

The widespread availability of Large Language Models (LLMs) such as GPT-4 and Llama-3, among others, has led to a surge in machine-generated content across various platforms, including social media, educational tools, and academic settings. While these models demonstrate remarkable capabilities in generating coherent text, their misuse raises significant concerns. For this reason, detecting machine-generated text has become a pressing need to mitigate these risks. This research proposed a novel classification method combining text-graph representations with Graph Neural Networks (GNNs) and different node feature initialization strategies to distinguish between human-written and machine-generated content. Experimental results demonstrate that the proposed approach outperforms traditional machine learning classifiers, highlighting the effectiveness of integrating structural and semantic relationships in text.

1 Introduction

Large language Models (LLMs) are now widely available and easily accessible, resulting in increased machine-generated content across various platforms, including Q&A forums, social media, educational resources, and academic contexts. Recent advancements in LLM technology, such as Llama-3 and GPT-4, have enabled these models to generate coherent responses to most user inquiries, making them increasingly attractive for replacing human labor in multiple fields. Moreover, this increased accessibility has led to concerns about misuse, including the creation of fake news, financial and legal issues, and education disruptions. Given the difficulty people have distinguishing between machine-generated and human-written text, there is a growing need for automated systems that can detect machine-generated content to mitigate and address the risks associated with its misuse (Nitu and Dascalu, 2024).

Viewing this problem as a classical text classification task, there are plenty of approaches to tackle it nowadays, from traditional methods such as training machine learning or deep learning models using Bag of Words or stylometric patterns as features to more advanced approaches based on the Transformer architecture, such as BERT (Devlin, 2018) or RoBERTa (Liu, 2019).

However, in recent years, a new area of research known as Graph Neural Networks (GNNs), or graph embeddings, has gained significant interest (Battaglia et al., 2018). These networks have proven to be highly effective in text classification tasks involving complex relational structures as they retain the global structure of a graph within their embeddings.

A text can be appropriately represented as a graph using the words/documents as nodes and the edge representing the significant relationship between the nodes. Also, it is possible to assign different attributes or weights to the graph's edges/nodes to add more significant information. Moreover, graph-based methods capture complex connections and dependencies that traditional methods might miss. Understanding these relationships between words/nodes is crucial for exploiting the text's best features. It could also help capture semantic and syntactic nuances in the text, distinguishing between human and machine-generated texts.

The contributions of our paper are summarized as follows:

- A classification method that combines Text-Graph representation and GNNs with different node feature initialization strategies to detect machine-generated text [1].
- A detailed comparison of the performance of our method against baseline and state-of-the-art methods on English texts.

[1] https://github.com/andricValdez/GraphDeepLearning

Proceedings of the 1st Workshop on GenAI Content Detection (GenAIDetect), pages 134–139
January 19, 2025.

2 Background

Several approaches to identifying automatically generated text have been proposed in recent years. Many of these works have been published in the framework of a series of shared tasks that focus on this specific problem, such as PAN-CLEF 2024 (Bevendorff et al., 2024) or IberAutextification 2023 (Sarvazyan et al., 2023). In (Abburi et al., 2023), the authors proposed an ensemble neural model combining probabilities from pre-trained LLMs as features and then applying a traditional machine learning classifier. Another relevant work assessed text predictability by leveraging features like grammatical accuracy, word frequency, linguistic patterns, and fine-tuned LLM representations (Duran-Silva, 2023). As we will see in the next sections, this last system ranks 1st in the binary classification task (for English text) on the IberAutextification shared task

On the other hand, regarding the usage of GNNs, many research works implement this approach to solve different types of text classification tasks (Wang et al., 2024). In (Yao et al., 2019), the authors explored the use of GNNs with convolutional layers (called TextGCN) for text classification, building a single text graph for a corpus based on word co-occurrence and document-word relationships, then used the graph embeddings as input for a final classification layer. Experimental results show that the TextGCN, even without external word embeddings, outperforms state-of-the-art methods and is particularly robust when training data is reduced. Another research work combines BERT and GCN for text classification tasks (called BertGCN) (Lin et al., 2021). BertGCN constructs a heterogeneous graph where documents are represented as nodes and uses BERT embeddings as feature representation. Their experiments demonstrate that BertGCN achieves state-of-the-art performance across multiple text classification datasets.

Moreover, regarding using Text Graph and GNNs to solve the machine-generated content identification task, in (Valdez-Valenzuela and Goméz-Adorno, 2024) it is proposed an ensemble architecture that combines Text Graph and GNNs, LLMs embeddings, and stylometric features achieving better performance than baselines.

3 Method Overview

This section describes our classification methodology for detecting machine-generated text, combining text-graph and GNNs. Section 3.1 describes the text-to-graph construction process, and section 3.2 explains all the pipeline architecture.

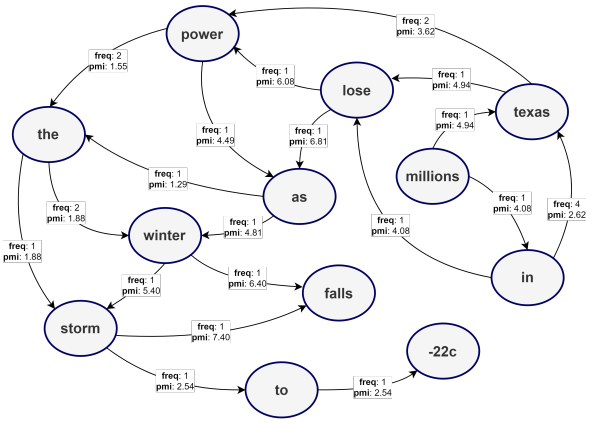

Figure 1: CoOccurrence Graph for the text: millions in texas lose power as the winter storm falls to -22c

3.1 Text Graph Representation

We implemented the Co-Occurrence graph to represent the text documents; to apply this transformation in a simple and flexible way, we used a Python library called text2graphAPI (Valdez-Valenzuela and Gómez-Adorno, 2024). In the Co-Occurrence graph, each word is represented as a node, and an edge connects two nodes if the corresponding words co-occur within the same text document (in a predefined window size). In addition, different weights can be assigned to the edges, including the frequency of co-occurrences between words and the Point-wise Mutual Information (PMI) for each word pair. Considering this, we built one Co-Occurrence graph representation for each text document in the corpus [2].

For instance, consider the sentence 'Millions in Texas lose power as the winter storm falls to 22°C' [3] as shown in Figure 1. In the corresponding graph, each node represents a unique word from the sentence, such as 'power', 'lose', and 'Texas'. These nodes are connected by edges, which indicate that the words co-occur within a certain context or proximity in the text (within a window size of 2). The frequency weight on an edge indicates how often the connected words appeared together. For instance, if an edge is labeled 'freq: 2', it means that the two words appeared together twice.

[2]Edge weights were calculated for each text document separately.

[3]This sentence is part of a longer document; thus, the weights of the edges may not correspond to what is observed in it

Additionally, the PMI weight measures the association strength between two words, highlighting how often these words co-occur more than would be expected by chance. Higher PMI values suggest a strong contextual relationship between the word pairs, even if their co-occurrence is infrequent. This metric helps reveal significant word associations that might not be immediately apparent from frequency alone.

3.2 Pipeline Architecture

Figure 2 shows the proposed pipeline architecture for identifying machine-generated text. As a first step, each document in the corpus is transformed into a Co-Occurrence graph, as described in Section 1. This transformation captures the relational structure of the texts, which is crucial for further processing.

After the graph has been built, we apply fine-tuning using pre-trained transformer models, such as BERT-Base-Uncased or RoBERTa-Base. These models are used to initialize node features in the graph, enhancing the semantic understanding of the text. Moreover, for the sake of comparison in the performance, we tested using different node features, such as the Word2Vec model (Mikolov, 2013) and random features.

The processed graph is then fed into a Graph Neural Network using a Graph Attention Network (GAT) layer (Veličković et al., 2017). This GAT layer uses attention mechanisms to focus on the most important nodes and edges, capturing relationships between words in a more subtle way. This attention-based learning enables the model to understand complex dependencies and associations within the text.

We implemented the GAT layer using the PyTorch Geometric library [4], providing as inputs the node features, the co-occurrence graph (as a sparse matrix, in COO format) and the edge weights (freq or PMI metrics). Additionally, we set and test with different kinds of hyperparameter related to this GNN, such as the number of convolutions (message passing layers), head attentions, hidden channels, pooling layers (add, mean, max), and normalization layers (such as BatchNorm1d or Dropout).

Lastly, the graph document embedding is fed into a final classification model. This classifier, typically handled by a dense neural network (but it could be any classification algorithm), determines whether the text was machine-generated based on the learned representations.

[4]This library provides various methods for deep learning on graphs from a variety of published papers: https://pytorch-geometric.readthedocs.io/en/latest/

Partition	***Autextification 2023***		
	human	**machine**	**total**
Train	11,963	11,728	**23,691**
Validation	5,083	5,071	**10,154**
Test	10,642	11,190	**21,832**

Table 1: Summary stats for the Autextification 2023 English dataset used in the experiments.

4 Experiments

This section shows all the experiment settings, datasets used, and the performance and results obtained for the proposed method.

4.1 Dataset

To evaluate the effectiveness of our proposed method, we utilized the Autextification2023 (Sarvazyan et al., 2023) dataset, a publicly available corpus specifically designed for machine-generated text detection. This dataset contains text in English and Spanish (In our case, we only used English texts). It comprises human and LLM-generated texts across five domains: tweets, reviews, how-to articles, news, and legal documents, representing a range of writing styles from formal to informal. Human texts were sourced from publicly available datasets like MultiEURLEX, XSUM, XL-SUM, MLSUM, Amazon Reviews, WikiLingua, and more. Machine-generated texts were produced using BLOOM and GPT-3 models, chosen for their multilingual capabilities and accessibility. Table 1 shows the Train, Validation, and Test sets, with a balanced distribution between human-generated and machine-generated text samples.

4.2 Results

Table 2 compares the performance of various models on the datasets for detecting machine-generated text using the Accuracy and F1-Score (macro) measures. The models evaluated include traditional machine learning classifiers, such as Support Vector Machine (using TF-IDF unigrams for vector representation), a fine-tuned BERT and RoBERTa model [5], and GNNs with different node initialization strategies.

[5]Using a 16 batch size, five training epochs, 2e-5 as learning rate and 0.01 of weight decay.

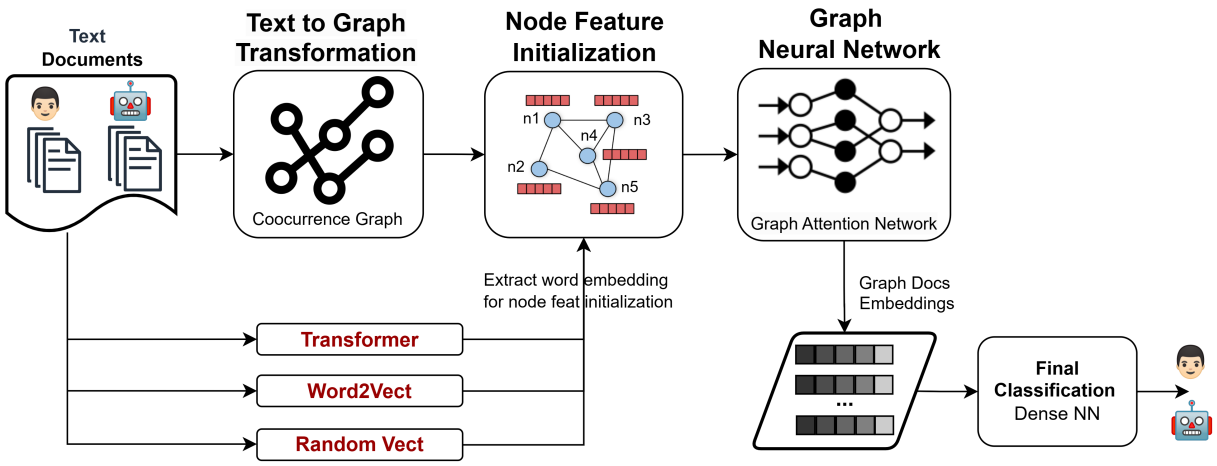

Figure 2: Pipeline Architecture for generated text identification.

Approach	Node Feat Init	Val Acc	Val F1Score	Test Acc	Test F1Score
Linear-SVM	-	0.7419	0.7419	0.5944	0.5624
Word2Vec	-	0.7325	0.7320	0.6040	0.5982
FT-BERT	-	0.8924	0.8916	0.6197	0.5515
FT-RoBERTa	-	**0.8974**	**0.8965**	0.6184	0.5481
GNN Cooc-Graph	Random	0.6469	0.6434	0.5738	0.5737
GNN Cooc-Graph	Word2Vec	0.7896	0.7896	0.6618	0.6592
GNN Cooc-Graph	FT-BERT	0.8889	0.8882	**0.7448**	**0.7441**
GNN Cooc-Graph	FT-RoBERTa	0.8812	0.8805	0.7447	0.7370

Table 2: Accuracy and Macro F1-Score on classification tasks for validation and test sets in the Autext 2023 dataset.

For the nodes feat initialization, we considered the following strategies, varying the feature vector size from 128 to 768:

- **Random.** In this approach, node features are initialized randomly using the PyTorch Embedding Layer [6], taking values from -1 to 1.
- **Word2Vec**. We trained the model (on train set) using the Continuous Bag of Words method and then obtained the word embeddings for each node/word in the graph. We applied a random vector initialization for those out of the vocabulary words.
- **Transformer**. Fine-tuned the BERT and RoBERTa models using the training data and then extracted the word embeddings. The tokenizer of this model generates some fine-grained tokens for certain words. To handle and match this with the graph's nodes, we obtain the average embedding for each token and assign the result as the node feature.

[6]https://pytorch.org/docs/stable/nn.html

Table 2 highlights the best-performing models for text classification in the Autextification 2023 dataset, showcasing significant differences in validation and test performances across approaches. The Linear-SVM and Word2Vec baselines achieved a validation accuracy (Val F1 Macro Score) of 0.7419 and 0.7320, respectively. Among baseline models, the fine-tuned BERT and RoBERTa achieve the highest validation F1 score at 0.8965 and 0.8916, respectively, but a decline in test performance (showing high overfitting). Now, regarding the GNN Cooc-Graph, using a random feature node initialization yields moderate results, while Word2Vec improves validation and test scores (Val F1: 0.6618, Test F1: 0.6592). Using fine-tuned transformer-based features (BERT and RoBERTa models) further enhances GNN performance, achieving the best overall test results: Test F1 Score: 0.7370 for FT-RoBERTa and 0.7441 for FT-BERT. This demonstrates the effectiveness of leveraging pre-trained transformer features within a graph-based framework and how the node feature initialization significantly impacts performance.

Rank	Approach	Run	Macro-F1
1	TALN-UPF	HB plus	0.8091
-	GNN (our)	FT-BERT	0.7418
2	TALN-UPF	HB	0.7416
3	CIC-IPN	run2	0.7413
23	BOW+LR	baseline	0.6578
52	Transformer	baseline	0.5710
77	UAEMex	run1	0.3387

Table 3: Final Ranking on the IberAutextification 2023 shared task (English text, subtask 1).

On the other hand, Table 3 shows the final ranking on the IberAutextification 2023 shared task for subtask 1 (binary classification on English texts). The team TALN-UPF ranks 1st and 2nd with a Macro-F1 with 0.8091 and 0.7416, respectively (the system is described in section 2). As we can see, the 1st rank system outperformed our proposed approach; however, our method achieved better results than the other 76 systems, including the baselines.

5 Conclusions and Future work

This paper addresses the increasing prevalence of machine-generated content due to advancements in LLMs. With their rising accessibility, concerns about their misuse have grown. To tackle this, we proposed a model architecture that combines text-graph representations and GNNs to detect machine-generated text; specifically, we implemented a co-occurrence graph where each word is represented as a node, and if two words co-occur within the same text document, it is linked with an edge. Then, this graph is fed into a GNN (GAT), generating the graph document embeddings as output, which are used to train a final classification model to distinguish between human and machine text documents.

Based on the experiments in the Autextification 2023 Engish dataset, our approach demonstrated superior performance compared to baselines and traditional approaches, highlighting the effectiveness of integrating structural and semantic features in identifying machine-generated content.

Moreover, future studies could enhance these approaches further and investigate their applicability across various languages and datasets with different domains. Also, different text graph representations (e.g. Heterogeneous Graphs) and GNN architectures should be tried using a combination of different node feature initialization strategies.

6 Limitations

The results are based on a specific dataset, which may not fully represent the diversity of machine-generated content across different domains or languages. Moreover, combining GNNs and transformer-based features can be computationally expensive, making the approach less feasible for real-time and large-scale applications without further optimization.

References

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Janek Bevendorff, Matti Wiegmann, Jussi Karlgren, Luise Dürlich, Evangelia Gogoulou, Aarne Talman, Efstathios Stamatatos, Martin Potthast, and Benno Stein. 2024. Overview of the "Voight-Kampff" Generative AI Authorship Verification Task at PAN and ELOQUENT 2024. In *Working Notes of CLEF 2024 - Conference and Labs of the Evaluation Forum*, CEUR Workshop Proceedings. CEUR-WS.org.

Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Yuxiao Lin, Yuxian Meng, Xiaofei Sun, Qinghong Han, Kun Kuang, Jiwei Li, and Fei Wu. 2021. Bertgcn: Transductive text classification by combining gcn and bert. *arXiv preprint arXiv:2105.05727*.

Yinhan Liu. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 364.

Tomas Mikolov. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 3781.

Melania Nitu and Mihai Dascalu. 2024. Beyond lexical boundaries: Llm-generated text detection for romanian digital libraries. *Future Internet*, 16(2).

Areg Mikael Sarvazyan, José Ángel González, Marc Franco Salvador, Francisco Rangel, Berta Chulvi, and Paolo Rosso. 2023. Overview of autextification at iberlef 2023: Detection and attribution of machine-generated text in multiple domains. In *Procesamiento del Lenguaje Natural*, Jaén, Spain.

Andric Valdez-Valenzuela and Helena Goméz-Adorno. 2024. The iimasnlp team at iberautextification 2024: Integrating graph neural networks, multilingual llms, and stylometry for automatic text identification.

Andric Valdez-Valenzuela and Helena Gómez-Adorno. 2024. text2graphapi: A library to transform text documents into different graph representations. *SoftwareX*, 28:101888.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Kunze Wang, Yihao Ding, and Soyeon Caren Han. 2024. Graph neural networks for text classification: A survey. *Artificial Intelligence Review*, 57(8):190.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 7370–7377.