International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 21 No. 3 (1) 2025

ijerst.editor@gmail.com

editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 21, No. 3(1), 2025

Research Paper

HYBRID CNN-DRIVEN RANDOM FOREST AND FASTTEXT EMBEDDINGS FOR UNMASKING AI-GENERATED TWEETS

K. Manohar Rao[1], Manaswi Ramidi[2], Nikitha Manga[2], Mythili Koppula[2]
[1]Assistant Professor, [2]UG Student, [1,2]Department of Computer Science and Engineering
[1,2]Kommuri Pratap Reddy Institute of Technology, Hyderabad, Telangana, India.
[1]Email: manoharkat@gmail.com.

Received: 05-6-2025 Accepted: 06-7-2025 Published: 14-7-2025

ABSTRACT

This study introduces a robust framework for detecting fake tweets using a dedicated Twitter fake tweet dataset, combining natural language processing (NLP) and machine learning to improve classification accuracy. Traditional manual detection methods are limited by scalability issues, subjectivity, and the inability to effectively identify subtle linguistic or contextual signals in vast volumes of social media data. To overcome these limitations, the proposed approach employs a multi-stage pipeline. It begins with comprehensive NLP preprocessing to clean and normalize the tweet content, followed by the application of FastText embeddings to convert textual information into meaningful numerical vectors. The data is then partitioned into training and testing sets using a train-test split strategy to ensure reliable evaluation. A deep learning convolutional neural network (DLCNN) is used for sophisticated feature extraction, uncovering complex patterns within the text. These features are subsequently classified using a random forest algorithm, which determines whether tweets are real or fake. The model's performance is thoroughly evaluated using key metrics to validate its accuracy and applicability in real-world misinformation detection scenarios.

Keywords: Deepfake Tweet Detection, AI-generated Tweets, Social Media Misinformation, Convolutional Neural Network, Random Forest Classifier.

1. INTRODUCTION

Online Social Networks (OSNs) have revolutionized digital communication, providing a platform for instant information sharing, discussion, and interaction. Among these, Twitter stands out as one of the most influential social media platforms due to its vast user base and real-time content dissemination. However, the rapid growth of OSNs has also led to the increasing spread of misinformation and deep fake content, particularly in the form of machine-generated tweets. This issue has significantly impacted various sectors, including politics, education, and public trust.

India, with one of the highest numbers of social media users, is particularly vulnerable to the negative consequences of deep fakes. Political misinformation, fake news campaigns, and manipulated narratives have influenced public opinion and even national elections. Students, who rely on OSNs for news, academic discussions, and social interactions, are especially susceptible to such deceptive content, leading to misinformation, academic dishonesty, and misinformed decision-making. Furthermore, the rampant spread of synthetic media has eroded public trust in digital communication, making it difficult for users to distinguish between real and fake content.

Given Twitter's dominance as a primary communication tool for news, politics, and social interactions, detecting machine-generated

tweets is critical. Traditional moderation techniques and manual fact-checking are insufficient to tackle the scale and sophistication of AI-generated content. Therefore, leveraging deep learning models combined with Fast Text embeddings can significantly enhance the accuracy and efficiency of detecting synthetic tweets. Fast Text embeddings allow models to capture contextual and sub-word-level information, making them highly effective in identifying unnatural patterns in text.

2. LITERATURE SURVEY

Khalid et al. [1] analyze AI-generated texts in the form of deepfake "tweets". These are evaluated using the logistic regression model, for which a maximum accuracy of 98.9% was achieved. Sentiment classification was validated through k-fold cross-validation [2]. The article by Al-Khazrajı et al. [3] also describes research on the impact of deepfakes on social networks. The discussions on this topic conclude by stating the necessity of collaboration between researchers and deeper education of the population. The analysis by Ahmed et al. [4] was conducted in eight countries. The factors analyzed were perception of accuracy, fear of missing out (FOMO), deficient self-regulation (DSR), and cognitive ability. The results showed that individuals with high FOMO or DSR are prone to sharing deepfakes, regardless of their cognitive level. Users with reduced cognitive abilities tend to distribute misinformation.

Mubarak et al. [5] classify the deepfakes into visual, audio, and textual deepfakes. Patel et al. [6] analyze different models and datasets used, implementation challenges, and certain future directions. These models are specific to image-based deepfakes [7].

One special category of deepfakes is audio deepfakes. Sunkari and Srinagesh [8] propose a model for combating audio deepfakes. The model uses a convolutional neural network (CNN) architecture to extract spectral features and a recurrent neural network (RNN) to analyze temporal dynamics.

Other models, such as large language models (LLMs), are used to identify text-based deepfakes. The article by De Angelis et al. [9] aims to study the Chat Generative Pre-Training Transformer (ChatGPT) phenomenon with respect to the spreading of deepfakes using LLMs. Koike et al. [10] proposed a model that identified text-based deepfakes. Following the research, the OUTFOX model achieved a performance of 96.9%.

3. PROPOSED SYSTEM

The rapid advancement of deepfake technology has significantly reshaped the digital landscape, particularly on social media platforms where misinformation can spread at an alarming rate. Deepfakes leverage sophisticated artificial intelligence (AI) techniques to generate highly realistic synthetic content, posing a severe threat to public trust, media integrity, and cybersecurity. Among the various forms of deepfake-generated content, machine-generated tweets—crafted using deep learning models—are increasingly being used to manipulate public opinion, spread propaganda, and deceive users. As AI-generated text becomes more convincing, distinguishing between authentic and synthetic content presents a growing challenge for social media users, policymakers, and cybersecurity experts.

To address this challenge, deep learning techniques combined with natural language processing (NLP) approaches, such as FastText embeddings, have emerged as effective tools for detecting machine-generated content. Advanced deep learning models, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transformers, have demonstrated remarkable performance in text classification and pattern recognition. FastText embeddings, in particular, excel at capturing contextual relationships in text, making them valuable for distinguishing human-written

tweets from AI-generated ones. This study explores the integration of these techniques to enhance detection accuracy and mitigate the impact of AI-driven misinformation. By analyzing linguistic patterns, semantic inconsistencies, and structural features, our research contributes to the broader field of digital forensics and online security.

With the increasing spread of deepfake technology, detecting AI-generated tweets has become a critical task in ensuring the integrity of social media content. This system leverages Django's Model-View-Template (MVT) architecture to build a scalable and efficient deep fake detection tool. The framework integrates FastText embeddings, deep learning (CNN), and a Random Forest classifier to accurately classify tweets as either human-written (Normal) or AI-generated (Fake).

Django follows the Model-View-Template (MVT) structure, which separates the application logic into three main components: Model (M): Handles database management and machine learning logic. View (V): Processes user input, calls the AI model, and returns the classification result. Template (T): Displays the output in a structured and user-friendly web interface.

The proposed system architecture as shown in Figure 4.1. for AI-based deep fake detection follows a structured Django MVT framework, integrating user authentication, database management, and AI-driven text classification. Users can sign up and log in, with their credentials securely stored in an SQL database. Once authenticated, they can submit tweets for analysis, which undergo NLP preprocessing, FastText embedding conversion, deep feature extraction using CNNs, and final classification via a Random Forest model to determine if a tweet is human-written (Normal) or AI-generated (Fake). The results are stored in the database and displayed on a user-friendly HTML interface, showing the original tweet, classification label, and confidence score. The system ensures efficient data retrieval, history tracking, and real-time deepfake detection, helping users combat misinformation effectively while maintaining a seamless and secure experience. The system follows a structured algorithm to detect AI-generated (fake) tweets using FastText embeddings, Deep Learning CNN (DLCNN), and Random Forest Classifier (RFC) within a Django-based web application.

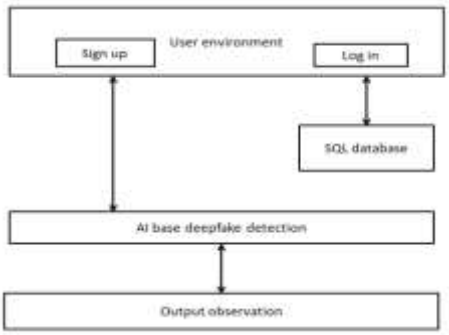

Fig. 1: Proposed Django architecture.

This AI-based deep fake detection system integrates FastText embeddings for effective word representation, DLCNN for deep feature extraction, and RFC for final classification. Implemented within a Django web application, it ensures high accuracy, real-time classification, and secure user interaction. By following this structured algorithm, the system effectively identifies AI-generated misinformation on social media, contributing to a safer and more reliable digital environment.

AI based Deepfake Detection

The flowchart such as Figure 2 represents the deepfake tweet detection system using deep learning and NLP techniques**.** The proposed deepfake tweet detection system successfully integrates Natural Language Processing (NLP), FastText embeddings, Deep Learning Convolutional Neural Networks (DLCNN), and a Random Forest Classifier (RFC) to identify AI-generated tweets with high accuracy. By leveraging FastText embeddings, the system effectively captures linguistic structures and contextual relationships, enhancing the

differentiation between real and fake tweets. The DLCNN model extracts deep textual features, while the RFC classifier ensures robust and efficient final classification. This approach is implemented within a Django-based web application, providing secure user authentication, real-time detection, and a user-friendly interface for analyzing tweets. The system demonstrates strong performance in detecting AI-generated misinformation, contributing to digital forensics, cybersecurity, and social media integrity. By deploying this solution, we take a significant step towards combating misinformation and ensuring the authenticity of online information, ultimately fostering a more trustworthy digital environment

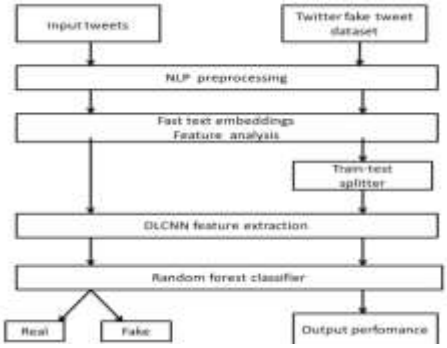

Fig. 2: Internal structure of a deepfake detection

NLP PREPROCESSING

In the second step, text pre-processing is performed to clean the tweet data. This involves removing unnecessary punctuation, special characters, and irrelevant symbols from the text. By standardizing the text, we make it easier for the model to analyze and learn from the relevant content, ensuring that only the meaningful information is considered during the training phase.

Step 1-Splitting Operation: The first step in NLP preprocessing involves splitting a text document into meaningful units. This could be splitting into sentences or words, depending n the task. Tools like NLTK's sent_tokenize() or Spacy's sentencizer help in sentence splitting, while word-tokenize () helps in word-level tokenization.

Step 2-Punctuations Removal: Punctuations are often removed to standardize text input, especially when they do not add meaning to the analysis. Python libraries like re (regular expressions) or NLTK's Regexp Tokenizer() can be used for this step.

Step 3-Tokenization (Sentence-Level Generation): Tokenization converts a string of text into smaller components, usually words or sentences. splitting text into meaningful chunks or tokens, typically words or phrases.

Step 4- Is Alpha (Special Character Removal): This step removes non-alphabetic characters such as numbers, symbols, and other special characters. It helps in focusing on meaningful words.

Step 5-Stop Words Removal: Stop words are common words (e.g., is, the, and, but) that do not contribute significantly to meaning and are removed to reduce noise. Libraries like NLTK (stopwords words()) and SpaCy provide predefined lists of stop words.

Step 6-Word-Level Tokenization: Word tokenization splits sentences into individual words. Tools are NLTK: nltk.word_tokenize(text) and SpaCy: nlp(text).

Step 7- Lemmatization: Lemmatization Converts words to dictionary forms (running → run, better → good). Stemming such as uses rules to chop off word endings (running → runn).

Step 8- Grouping Output Words: After tokenization and preprocessing, words are grouped based on linguistic properties, synonyms, or topic modeling.

Step 9 Output Preprocessed Data: The last step in nlp preprocessing is saving the output. Finally, the cleaned and structured text data is saved for further NLP tasks like classification, sentiment analysis.

DLCNN Feature Extraction

A Hybrid Convolutional Neural Network (CNN) as shown in Figure 4.6 integrates various deep learning techniques, combining CNNs with

other machine learning models or neural network architectures to enhance performance. The hybrid approach aims to improve feature extraction, increase accuracy, and handle more complex patterns in the data. Hybrid CNNs are often used in advanced applications such as image classification, object detection, and other domains that require high levels of accuracy.

Convolution: The network applies filters to small regions of the image, scanning the image in both horizontal and vertical directions.

Activation: The output from the convolution operation is passed through an activation function, such as ReLU (Rectified Linear Unit).

Pooling: The output is down sampled to reduce spatial dimensions and retain important information.

Flattening: The output is flattened into a one-dimensional vector.

Fully Connected Layers: The flattened vector is passed through fully connected layers to produce the final output.

Dropout: Dropout is used to randomly drop out neurons during training, which helps to prevent overfitting.

SoftMax: The final output is passed through a SoftMax activation function to produce a probability distribution over the classes.

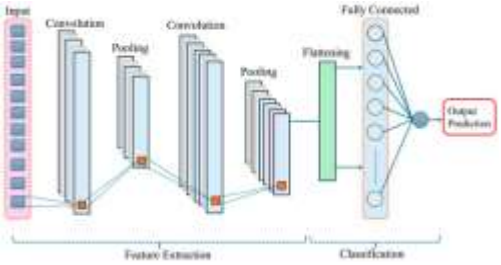

Fig. 3: Convolutional Neural Networks

RANDOM FOREST CLASSIFIER

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

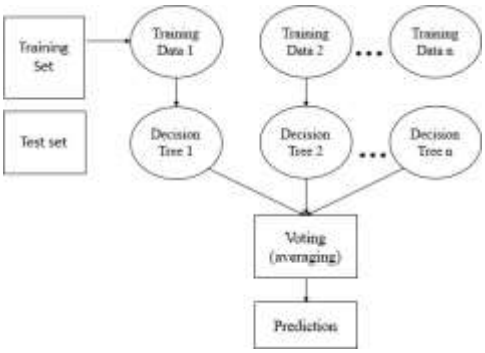

Fig. 4: RFC Block Daigram.

Step 1: In Random Forest n number of random records are taken from the data set having k number of records.

Step 2: Individual decision trees are constructed for each sample.

Step 3: Each decision tree will generate an output.

Step 4: Final output is considered based on Majority Voting or Averaging for Classification and regression respectively.

ADVANTAGES:

DLCNN

CNNs automatically learn hierarchical text features, making them highly effective in fake tweet detection. Can handle large-scale textual data with complex patterns, reducing false positives in classification. Can be extended to handle other NLP tasks beyond fake tweet detection, such as sentiment analysis and spam detection.

Random Forest Classifier (RFC)

Uses multiple decision trees to reduce overfitting and improve classification performance. Works effectively even when fake tweets are much fewer than real tweets. Compared to deep learning models, RFC requires less computational power and provides quick predictions

4. RESULTS AND DISCUSSION

The above fig; 5 shows the FastText embeddings are applied to social media tweets, converting textual data into numerical vectors. These embeddings are then used as input for machine learning models to classify tweets as either bot-generated or human-written, enhancing the accuracy of deepfake detection.

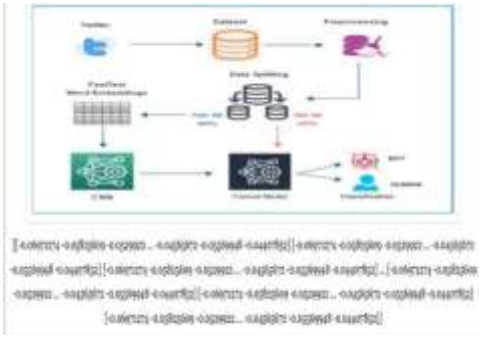

Fig. 5: Fast Text Embeddings

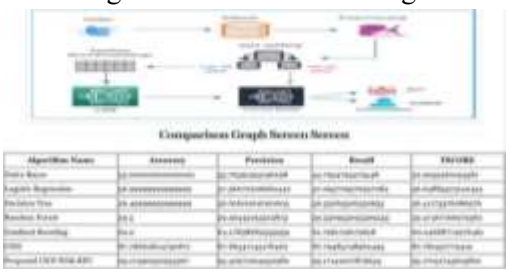

Fig. 6: All Algorithms

The fig; 6 compares various machine learning algorithms for classifying social media tweets as human- or bot-generated, evaluated using Accuracy, F1-score, Precision, and Recall. The Proposed CNN with RFC model achieves the highest performance, leveraging deep learning's feature extraction and ensemble learning's robustness. CNN also performs well due to its ability to capture contextual relationships in text embeddings. Traditional models like Decision Tree, Gradient Boosting, Logistic Regression, and Naïve Bayes show moderate performance, with Naïve Bayes being the weakest. Random Forest performs well but slightly below the proposed hybrid model. The results indicate that deep learning, particularly CNN combined with RFC, is the most effective approach for deepfake detection in social media.

Fig. 7: Predict Deep Fake

Figure 7 represents the Deepfake Detection System for social media, which uses deep learning and FastText embeddings to classify tweets as human-generated or bot- generated. The prediction interface allows users to input a tweet into the system, which then processes it through several stages: preprocessing, FastText word embeddings, and CNN-based classification. The trained model analyzes the input tweet and determines whether it originates from a bot or a human. This automated system ensures efficient deepfake detection by leveraging contextual word representations and neural network- based classification, helping mitigate misinformation on social media platforms.

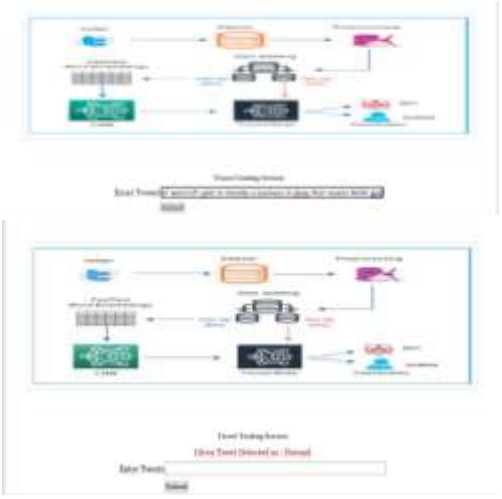

Fig. 8: Output for the given label

The fig: 8 represents the Deepfake Detection System's output for a given tweet. The system takes an input tweet, processes it through FastText embeddings and a CNN-based classification model, and determines whether the tweet is generated by a bot or a human. In this specific case, the system has classified the given tweet as "Normal," indicating that it is likely human-generated rather than a deepfake or bot-generated content. The detection workflow involves data preprocessing, feature extraction using FastText, training with CNN, and final classification, ensuring an accurate prediction based on learned patterns.

Table 1. Result Comparison for Twitter Bot Fake Tweet Dataset

Algorithm	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
Naïve Bayes	55.50	54.66	53.56	51.57
Logistic Regression	59.00	59.49	59.46	58.99
Decision Tree	64.00	64.29	64.33	63.99
Random Forest	63.50	63.39	63.47	63.39
Gradient Boosting	64.00	66.13	65.14	63.71
CNN	87.81	87.88	87.80	87.80
Proposed CNN with RFC	94.69	94.85	94.62	94.68

The table presents a comparative analysis of seven algorithms evaluated based on four performance metrics: Accuracy, Precision, Recall, and F1 Score, all expressed as percentages. The Naïve Bayes algorithm performs the least effectively, with an accuracy of 55.50%, precision of 54.66%, recall of 53.56%, and an F1 score of 51.57%, indicating relatively poor performance across all metrics. Logistic Regression shows a slight improvement, achieving an accuracy of 59.00%, precision of 59.49%, recall of 59.46%, and an F1 score of 58.99%. The Decision Tree and Gradient Boosting algorithms both achieve an accuracy of 64.00%, but they differ in other metrics: Decision Tree has a precision of 64.29%, recall of 64.33%, and F1 score of 63.99%, while Gradient Boosting excels in precision at 66.13% and recall at 65.14%, though its F1 score is slightly lower at 63.71%. Random Forest falls slightly behind with an accuracy of 63.50%, precision of 63.39%, recall of 63.47%, and F1 score of 63.39%, showing consistent but not standout performance among the traditional machine learning models.

In contrast, the deep learning-based algorithms significantly outperform the traditional models. The Convolutional Neural Network (CNN) achieves a much higher accuracy of 87.81%, with precision at 87.88%, recall at 87.80%, and an F1 score of 87.80%, demonstrating balanced and strong performance across all metrics. The standout performer is the Proposed CNN with RFC (Random Forest Classifier), which achieves the highest scores: an accuracy of 94.69%, precision of 94.85%, recall of 94.62%, and an F1 score of 94.68%. This hybrid approach, combining CNN's feature extraction capabilities with the robustness of Random Forest, yields superior results, with all metrics exceeding 94%, making it the most effective algorithm in this evaluation. The significant gap between the traditional models (Naïve Bayes, Logistic Regression, Decision Tree, Random Forest, and Gradient Boosting) and the deep learning models (CNN and Proposed CNN with RFC) underscores the advantage of advanced

neural network architectures in achieving higher accuracy and robustness in classification tasks.

5. CONCLUSION

The rapid rise of deepfake content on social media presents a growing threat to the authenticity of information, particularly within critical domains such as politics and entertainment. This research tackles the detection of AI-generated tweets by employing a deep learning framework combined with FastText embeddings. Unlike traditional methods such as manual moderation—which are time-consuming, error-prone, and lack scalability this approach enables fast and accurate identification of machine-generated content. FastText embeddings effectively transform tweet text into rich word vectors, enabling deep learning models to distinguish between human- and AI-generated tweets. This real-time detection capability is essential for promptly curbing the spread of misleading content. By integrating deep learning with FastText, the proposed model delivers improved accuracy and scalability, significantly outperforming rule-based systems reliant on static keyword sets or manual intervention.

REFERENCES

[1] Khalid, M.; Raza, A.; Younas, F.; Rustam, F.; Villar, M.G.; Ashraf, I.; Akhtar, A. Novel Sentiment Majority Voting Classifier and Transfer Learning-Based Feature Engineering for Sentiment Analysis of Deepfake Tweets. IEEE Access 2024, 12, 67117–67129.

[2] Rosca, C.M.; Ariciu, A.V. Unlocking Customer Sentiment Insights with Azure Sentiment Analysis: A Comprehensive Review and Analysis. Rom. J. Pet. Gas Technol. 2023, 4, 173–182.

[3] Al-Khazraji, S.H.; Saleh, H.H.; Khalid, A.I.; Mishkhal, I.A. Impact of Deepfake Technology on Social Media: Detection, Misinformation and Societal Implications. Eurasia Proc. Sci. Technol. Eng. Math. 2023, 23, 429–441.

[4] Ahmed, S.; Ng, S.W.T.; Bee, A.W.T. Understanding the role of fear of missing out and deficient self-regulation in sharing of deepfakes on social media: Evidence from eight countries. Front. Psychol. 2023, 14, 1127507. [PubMed]

[5] Mubarak, R.; Alsboui, T.; Alshaikh, O.; Inuwa-Dutse, I.; Khan, S.; Parkinson, S. A Survey on the Detection and Impacts of Deepfakes in Visual, Audio, and Textual Formats. IEEE Access 2023, 11, 144497–144529.

[6] Patel, Y.; Tanwar, S.; Gupta, R.; Bhattacharya, P.; Davidson, I.E.; Nyameko, R.; Aluvala, S.; Vimal, V. Deepfake Generation and Detection: Case Study and Challenges. IEEE Access 2023, 11, 143296–143323.

[7] Rosca, C.M. Comparative Analysis of Object Classification Algorithms: Traditional Image Processing Versus Artificial Intelligence—Based Approach. Rom. J. Pet. Gas Technol. 2023, IV (LXXV), 169–180.

[8] Sunkari, V.; Srinagesh, A. Efficient Deepfake Audio Detection Using Spectro-Temporal Analysis and Deep Learning. J. Electr. Syst. 2024, 20, 10–18.

[9] De Angelis, L.; Baglivo, F.; Arzilli, G.; Privitera, G.P.; Ferragina, P.; Tozzi, A.E.; Rizzo, C. ChatGPT and the rise of large language models: The new AI-driven infodemic threat in public health. Front. Public Health 2023, 11, 1166120.

[10] Koike, R.; Kaneko, M.; Okazaki, N. OUTFOX: LLM-Generated Essay Detection Through In-Context Learning with Adversarially Generated Examples. Proc. AAAI Conf. Artif. Intell. 2024, 38, 21258–21266.