

Article

Towards AI-Generated Essay Classification Using Numerical Text Representation

Natalia Krawczyk ^{1,†} , Barbara Probiez ^{1,2,*}  and Jan Kozak ^{1,†} 

¹ Department of Machine Learning, University of Economics in Katowice, 1 Maja 50, 40-287 Katowice, Poland; natalia.krawczyk@ue.katowice.pl (N.K.); jan.kozak@ue.katowice.pl (J.K.)

² Łukasiewicz Research Network–Institute of Innovative Technologies EMAG, Leopolda 31, 40-189 Katowice, Poland

* Correspondence: barbara.probiez@ue.katowice.pl

† These authors contributed equally to this work.

Abstract: The detection of essays written by AI compared to those authored by students is increasingly becoming a significant issue in educational settings. This research examines various numerical text representation techniques to improve the classification of these essays. Utilizing a diverse dataset, we undertook several preprocessing steps, including data cleaning, tokenization, and lemmatization. Our system analyzes different text representation methods such as Bag of Words, TF-IDF, and fastText embeddings in conjunction with multiple classifiers. Our experiments showed that TF-IDF weights paired with logistic regression reached the highest accuracy of 99.82%. Methods like Bag of Words, TF-IDF, and fastText embeddings achieved accuracies exceeding 96.50% across all tested classifiers. Sentence embeddings, including MiniLM and distilBERT, yielded accuracies from 93.78% to 96.63%, indicating room for further refinement. Conversely, pre-trained fastText embeddings showed reduced performance, with a lowest accuracy of 89.88% in logistic regression. Remarkably, the XGBoost classifier delivered the highest minimum accuracy of 96.24%. Specificity and precision were above 99% for most methods, showcasing high capability in differentiating between student-created and AI-generated texts. This study underscores the vital role of choosing dataset-specific text representations to boost classification accuracy.

Keywords: natural language processing; numerical text representations; text classification; large language models



Citation: Krawczyk, N.; Probiez, B.; Kozak, J. Towards AI-Generated Essay Classification Using Numerical Text Representation. *Appl. Sci.* **2024**, *14*, 9795. <https://doi.org/10.3390/app14219795>

Academic Editor: Malik Yousef

Received: 23 September 2024

Revised: 18 October 2024

Accepted: 23 October 2024

Published: 26 October 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The generation of content by artificial intelligence is becoming increasingly prevalent. Large language models such as GPT (Generative Pre-training Transformer) and BERT (Bidirectional Encoder Representations from Transformers) are gaining significance, offering capabilities for generating complex texts. Recent advances in natural language generation have significantly improved the diversity and quality of these texts, making them almost indistinguishable from human-written texts. This has raised concerns about potential misuse, such as spreading misinformation or disrupting educational systems [1]. This AI-generated content has applications across various domains, including education, media, and business. Consequently, it presents new challenges, particularly in the realm of academic integrity.

In the context of education, academic integrity is paramount. According to Tauginien et al. (2018), it is defined as “compliance with ethical and professional principles, standards, practices, and a consistent system of values that serves as guidance for making decisions and taking actions in education, research and scholarship” (p. 8) [2]. The ability to accurately detect AI-generated texts is crucial to maximizing their potential while minimizing severe consequences. This capability can increase user trust in available solutions and help researchers

prevent the unauthorized use of generated texts [1]. A significant threat to academic integrity is unauthorized content generation (UCG). According to Foltyniek et al. (2023), “UCG is the production of academic work, in whole or in part, for academic credit, progression or award, regardless of whether or not a payment or other favour is involved, using unapproved or undeclared human or technological assistance” (p. 2) [3]. This practice undermines the foundational values of honesty and fairness in academia. Many existing market solutions, such as OpenAI Text Classifier and ZeroGPT, are designed to handle a broad range of texts on diverse topics. However, an approach that focuses on the development of similar tools tailored to specific types of content may prove to be more effective.

This paper explores various numerical text representation methods to enhance the classification process for detecting AI-generated essays. Using a comprehensive dataset and implementing a variety of pre-processing steps, we aim to evaluate the effectiveness of different text representation techniques. These include traditional methods such as Bag of Words (BoW) and TF-IDF, as well as more advanced embeddings such as fastText, MiniLM, and distilBERT. Specifically, for advanced embedding techniques like fastText, both pre-trained models and models trained on our own dataset are utilized. The results highlight the importance of selecting appropriate text representation techniques tailored to the dataset’s characteristics and aim to bridge the gap by focusing on dataset-specific text representations and their impact on classification accuracy.

This paper is organized as follows: Section 1 introduces the topic of this paper. Section 2 presents an overview of the work related to the problem of AI-generated text detection. In Section 3, we describe the theoretical background of natural language processing, and in Sections 3.1 and 3.2, we present numerical representations of text and large language models. In Section 4, we present our methodology for conducting research on AI-generated text detection. In Section 5, we present an analysis of our research on real-world datasets, and in Section 5.3, we discuss the experimental results. Finally, in Section 6, we conclude with general remarks on this work and possible directions for future research.

2. Related Works

Detection tools for AI-generated texts can be classified into black-box detection and white-box detection [1]. Black-box detection methods access large language models at the API level, collecting samples of human- and machine-written texts to train a classification model. Initially, these detectors were highly effective, as AI-generated texts often exhibited linguistic or statistical patterns. However, as the models evolved, their precision decreased [1]. White-box detection, on the other hand, provides full access to the target language model, allowing for the monitoring and control of its behavior. Black-box detectors are typically external solutions, while white-box detectors are often tools developed by the language model’s creators [1].

The three main stages in black-box detection are data acquisition, feature selection, and the construction of classification models. The effectiveness depends largely on the quality of the acquired data. Feature selection techniques detect variables considering statistical differences, linguistic patterns, or fact verification. Commonly used classifiers include logistic regression, support vector machines, naive Bayes classifiers, decision trees, and random forests [4]. Basic methods offer interpretability, allowing researchers to analyze the significance of the input variable and understand the classification decisions of the model [1]. Recent studies explore neural network-based classification techniques using language models such as RoBERTa fine-tuned for specific human- and AI-generated text mixtures, achieving greater accuracy than human responses [5,6].

In white-box detection, identification strategies use watermarking, encoding hidden messages or identifiers in text to monitor suspicious or unauthorized activities [1]. These watermarks must be effectively embedded, verifiable, discreet, and resistant to removal through common text modification techniques, such as synonym replacement. Two main watermarking methods are post hoc and inference-time watermarking. Post hoc watermark-

ing embeds identifiers in pre-generated text, while inference-time watermarking modifies the text generation process to embed watermarks during decoding [1].

A notable study in this field [7] focuses on tools for detecting AI-generated texts in academic publications, highlighting the potential risks of unfair use of such content in academic settings. The authors emphasize that existing detection tools are neither accurate nor reliable, often misclassifying AI-generated texts as human-written. They propose the implementation of a prototype to detect AI-generated texts, comparing their solution with existing tools such as the OpenAI Text Classifier and ZeroGPT. The proposed model returns a percentage probability that indicates whether a text was generated by AI. It includes a user interface, a text detection tool, a survey for user feedback, and a connected database. The model was trained on a dataset comprising sections and paragraphs from research papers on Google Scholar, manually classified as human written or AI generated using ChatGPT. The dataset consisted of 1200 records, labeled 0 for human-written and 1 for AI-generated text.

The authors used a word embedding technique based on simple frequencies for numerical text representation, followed by training an artificial neural network. They compared the accuracy of their model with the OpenAI and ZeroGPT classifiers. The OpenAI classifier provided results in the form of five phrases that indicated the likelihood of AI generation. The phrases 'very unlikely' and 'unlikely' indicated that the text was considered human written, while 'possibly' and 'very likely' indicated AI generation. For some records, the phrase 'unclear' was displayed, indicating ambiguous results. The OpenAI classifier required a minimum of 1000 characters to return a result and achieved an accuracy of 42.08%, with 40% of the results being unclear or not meeting the character requirement. The proposed model, with no character count constraints, achieved an accuracy of 89.09%, outperforming the OpenAI classifier. ZeroGPT, using a perplexity score, achieved an accuracy of 87.50% [7]. The authors noted that their model had similar false positive and false negative rates compared to ZeroGPT. However, the model's limitation was the use of basic techniques, and different approaches for numerical text representation were not compared. The authors plan to explore various preprocessing techniques in future work.

The need for effective anomaly detection in textual data has prompted extensive comparisons between various text representation methods, particularly focusing on the effectiveness of frequency-based versus prediction-based models [8]. The authors have utilized pre-trained models, fastText and Word2vec, using Skip-gram and Continuous Bag of Words (CBoW) approaches to classify legal emails and phishing attacks. They used three datasets, embedding words using frequency, TF-IDF, Word2vec (CBoW and Skip-gram), and fastText (CBoW and Skip-gram) methods. Various vector lengths were tested, and classification was performed using random forests, decision trees, XGBoost, logistic regression, and support vector machines. The highest accuracy for the first dataset was 99.50% using fastText (CBoW), random forests, and a vector length of 100. For the second dataset, the highest accuracy was 99.39% using TF-IDF, random forests, and a vector length of 200. The highest accuracy of the third dataset was 99.18% using word frequency, random forests, and a vector length of 150.

In addition to the challenges posed by AI-generated text detection, recent advancements have highlighted the growing threat of adversarial attacks in the field of text classification, where subtle modifications to the text can deceive classifiers into making incorrect predictions. In this context, the robustness of text representation techniques becomes critical. For instance, Kwon and Lee [9] proposed a defense against textual backdoor attacks by using class differences for detection. Their method compares a detection model trained on secure, partial data with a target model trained on the entire dataset, demonstrating the importance of high-quality text representations to accurately identifying manipulated samples. The study reported detection rates above 79% on the MR and IMDB datasets, demonstrating how tailored detection models can mitigate backdoor attacks without needing full access to the training dataset.

Furthermore, Kwon and Lee [10] explored defenses against adversarial examples in text classification systems, which involve modifying important words in a text to cause misclassification. Their proposed method leverages text modification by replacing key words with semantically similar alternatives. This approach exploits the fact that adversarial examples are more sensitive to such modifications than original sentences, enabling the model to distinguish between the two. Their experiments on datasets like IMDB and MR showed that this technique can achieve an adversarial detection rate of over 71%, underscoring the importance of advanced text representations in improving model robustness against adversarial manipulations.

3. Natural Language Processing

Natural language processing (NLP) is a rapidly evolving field with numerous applications and a promising future. Recent breakthroughs, including advances in voice assistants, real-time language translation, and AI-driven customer support, are revolutionizing the way we interact with technology today. NLP, a vast interdisciplinary field at the intersection of linguistics and computer science [11], continues to evolve. Its essence lies in accurately understanding language structures and formal rules while applying this knowledge in technological contexts. ‘Natural language’ refers to a way of communication that is not strictly governed by formalized constraints [12].

Early approaches to NLP focused on automating language analysis in terms of its linguistic structure and developing fundamental technologies like machine translation and speech synthesizers. Over time, significant advances in existing tools have expanded their applications. Innovative technologies such as dialogue systems, tools to explore social networks for various social insights, and sentiment analysis solutions that identify user emotions toward products and services have emerged [13].

Natural language can be analyzed at various levels corresponding to the degrees of detail with which it can be represented. From the perspective of natural language processing, the key levels include morphology, syntax, semantics, and pragmatics.

Morphology is a branch of linguistics that focuses on the description of words and their inflections. It includes word formation and inflection. The morphological level is associated with the morpheme, defined as the smallest communicative unit in the morphological plane of the language. In other words, it is the smallest indivisible part of a word that carries meaning. The morpheme forms the basis for the construction of more complex linguistic units [14].

‘Syntax’ refers to the grammatical structure and organization of words in sentences. The term derives from the Greek word *syntaxis*, which means arrangement or ordering. These concepts are often used interchangeably. Nowadays, syntax refers to the subsystem of language that focuses on utterances, which are units in the syntactic plane. ‘Syntax’ is also defined as the field of linguistics that describes the structure of sentences and their classification, the relationships between words that make up a sentence, and the ways of expressing individual parts of sentences and their functions [14]. This enables the analysis of sentence structure and identification of the role of individual words in the context of a sentence.

Semantics deals with the study of the meaning of words, phrases, and sentences. At this level, the meaning of words in specific sentences and the relationships between them are analyzed. Currently, semantics focuses on the relationships between the meaning of words and their form, as well as the relationships between the basic definition of a word and its actual meaning in a specific sentence [14]. This involves analyzing how the structure and context of a word or phrase can affect its meaning. For instance, the same word may carry different meanings depending on its grammatical form or how it is used in a sentence, requiring careful interpretation to understand its specific meaning in context. An additional challenge in natural language processing is the existence of homonyms (words that sound the same or are spelled the same but have different meanings), polysemes (words with

multiple meanings), and synonyms (different words with the same or similar meanings). Semantic research also encompasses these areas [14].

‘Pragmatics’ refers to the analysis of language meaning in the context of linguistic communication. Pragmatics includes studies on the ways language is used in interactions, the purpose and effectiveness of utterances, the interpretation of the speaker’s intentions, as well as non-linguistic factors, such as social or contextual aspects [14].

By analyzing natural language at these various levels, NLP systems can better understand and generate human language, making interactions with machines more intuitive and effective.

3.1. Numerical Text Representations

Text, one of the most common types of sequential data, can be treated as a sequence of characters or words, with most operations being performed at the word level [15]. Before analysis, text or text sets must be preprocessed and cleaned, by converting raw text into a structured form that models can interpret while identifying key words to reinforce contextual understanding [16]. Several methods exist for converting raw text into a model-understandable structure, ranging from basic to advanced techniques.

3.1.1. Basic Numerical Text Representation Methods

The fundamental technique is to create a text representation in the form of a vector, where each word is a key, and its value is the number of its occurrences in the document. This representation is known as Bag of Words (BoW) [16]. However, BoW has certain limitations because it does not consider the order of words or their context. Moreover, it focuses solely on the number of occurrences, which can assign a greater weight to frequently occurring words despite their low semantic significance [17].

Another similar technique is the one-hot encoding of tokens, which uses binary values to indicate the presence of the word [18]. Both the Bag of Words approach and one-hot encoding represent each word by a vector whose length corresponds to the size of the vocabulary, i.e., the list of all unique words in the corpus. The key difference between these methods is how non-zero elements are recorded in the vectors. The bag of words representation counts the number of occurrences of words in a document, while one-hot encoding indicates whether a given word appears in the document through binary values [18]. A more refined method for numerical text representation is using TF-IDF weights (term frequency–inverse document frequency). This method assigns high weight to words that frequently appear in a specific document but not across many documents in the corpus. If a particular word occurs with high frequency in a document while rarely appearing in the entire corpus, it is likely very descriptive of the document’s content. This mitigates the issue of assigning high weights to insignificant but frequently occurring words, a problem in the Bag of Words technique [8].

TF (term frequency) is the number of occurrences of a term (word) in a document, usually divided by the total number of words in the document [8]:

$$TF(t, d) = \frac{x}{y} \quad (1)$$

where:

- x —the number of occurrences of the word t in the document d ,
- y —the number of words in the document d .

IDF (inverse document frequency) measures the uniqueness of a word in the context of the entire corpus:

$$IDF(t, d) = \log\left(\frac{N}{n}\right) \quad (2)$$

where

- N —the total number of documents in the corpus,
- n —the number of documents in which the word t appears.

TF-IDF weight is the product of the above measures:

$$TF - IDF = TF \cdot IDF \quad (3)$$

All of these methods share one common feature: They generate sparse matrices. A sparse matrix is characterized by the fact that most of its elements are zeros. This typically results from an extensive set of words in the vocabulary, leading to the creation of high-dimensional vectors, many of which contain no information about the given text. This can be particularly problematic with large datasets, as it significantly increases memory load [18].

3.1.2. Embedding Techniques

An alternative to high-dimensional data generation methods is word embedding, also known as vector-based distributional word representation [19]. This technique allows for the creation of dense vectors, that is, low-dimensional vectors. These representations are created by algorithms that assign each word in the corpus a set of numerical values. The advantage of word embedding techniques is that they reflect semantic relationships between words, enabling models to process text more efficiently [8]. Words with similar meanings are represented by vectors that are close to each other in space, which not only helps the models detect patterns in the text but also alleviates the problem of sparse matrices, thus increasing the efficiency of the model [19].

A popular algorithm for creating word vector representations is Word2vec, based on neural networks, which was proposed by Mikolov et al. [19]. It was developed in two architectures aimed at minimizing computational complexity. The authors concluded that such complexity mainly arises from the non-linear hidden layer in neural networks. Therefore, they decided to explore simpler models and their effectiveness [19].

The first architecture is the Continuous Bag of Words (CBoW). Its name stems from the fact that the order of words does not affect the results, similar to the classical Bag of Words. This is mainly because a single weight matrix is used for all words during training instead of creating separate weights for each word. The term ‘continuous’ refers to the continuous and distributed representation of the surrounding context [19]. In the standard Bag of Words model, word representation is discrete and sparse. Each word has a separate index in the vocabulary, the vector length equals the vocabulary length, and most elements are zeros. Continuity and dispersion in continuous bags of words imply that words are represented by dense vectors of a specified length, with each element potentially containing a real floating-point number, indicating the context of the given word. Additionally, in the CBoW method, both preceding and succeeding words of the target word are used. This is achieved by using a logistic classifier that takes four preceding and four succeeding words as input and attempts to correctly classify the target word [19].

The second architecture of the Word2vec algorithm is Skip-gram. It works in the opposite way to the CBoW method, where the target word is predicted based on its surrounding context. In this case, contextual words are predicted, that is, words surrounding the target word. In other words, Skip-gram analyses a specific word and tries to predict which words might surround it. Given a sequence of training words $w_1, w_2, \dots, w_T$, the goal of the Skip-gram model is to maximize the average log probability [19]:

$$\frac{1}{T} \sum_{t=1}^T \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} | w_t), \quad (4)$$

where

- c —the size of the training context (number of words surrounding the target word). A higher value of c means a larger training context, resulting in a more complex model. A lower value of c results in faster training, but the model may be less precise in predicting the context.

FastText is an extension of Mikolov’s algorithm. This approach is based on the Skip-gram model, whose original concept was word-based. In the case of fastText,

an innovative approach using character n-grams was introduced, where each n-gram is treated as a separate “word-like” fragment [20]. This allows the model to flexibly recognize words, including rare ones, those with typos, or even those not present in the training set, by analyzing their n-gram fragments [21]. It can be trained on custom corpora, providing specialized embeddings tailored to specific data. However, this requires a sufficiently large dataset to ensure robustness. Alternatively, pre-trained models on large corpora offer general embeddings suitable for various tasks, but may lack domain-specific nuances.

A highly accurate algorithm for word and context representation is BERT (Bidirectional Encoder Representations from Transformers) [22]. It utilizes transformer neural networks, commonly used in natural language processing. These networks are distinguished by their ability to process data sequences considering the context. They use the attention mechanism, which allows the model to assign different weights to individual elements in the sequence. This enables the analysis of dependencies between different parts of the text, considering not only their distance, but also their mutual connections [23]. This mechanism will be explained in more detail in Section 3.2. Additionally, BERT employs word-piece tokenization, which means that words are split into smaller components. This is particularly useful for rare words or those that are not directly present in the vocabulary. As a result, the model can generate representations for smaller word fragments, even despite typos or their absence in the training corpus. This is similar to the fastText mechanism, where words are represented using character fragments [21].

3.2. Large Language Models

Language models are computational systems capable of understanding and generating text in natural language. They possess the transformative ability to predict the likelihood of a given sequence of words or generate new text based on specified input data [24]. The most commonly encountered type of language models are those based on n-grams. These models estimate word probabilities based on context to predict the next element in a sequence.

Large language models (LLMs) are advanced language models with a large number of parameters and characterized by high learning capabilities [24]. The core of most large language models, such as GPT-3 [25] and GPT-4 [26], is the neural network transformer architecture [23]. This architecture eliminates the sequential nature of the problem. Instead of processing data word by word, transformers capture dependencies in the text by considering the entire sequence at once [24]. Another common feature of LLMs is contextual learning, where the model is trained to generate text based on a specific context or prompt, which serves as an instruction or command [27]. This allows for generating more coherent and contextually appropriate responses, which is particularly important for interactive and conversational applications.

A significant aspect of LLMs is reinforcement learning from human feedback (RLHF). This technique involves fine-tuning the model based on human-generated responses, significantly improving the quality of answers and performance over time [28]. Most LLMs also utilize transfer learning as a fine-tuning technique. The pre-trained models are adapted to specific tasks using task-specific data [29].

There is a growing trend towards models fine-tuned for specific tasks and open-source models. It is worth noting that the first pre-trained models, such as T5 [30] and mT5 [31], utilized transfer learning. The advent of GPT-3 demonstrated that large language models could be used for various tasks without additional fine-tuning [27]. Consequently, the innovative technique of zero-shot learning emerged, allowing models to sample unseen classes during training. Few-shot learning, where the model has a limited number of samples for a task not included in its original training, has also gained popularity. Few-shot learning often results in higher accuracy for solutions provided by LLMs compared to zero-shot learning. Moreover, fine-tuning models that employ both zero-shot and few-shot learning significantly enhances their ability to handle new tasks [29].

Sequential learning is a crucial aspect of LLMs, enabling them to generate fluent text. This forms the foundation for autoregressive models which improve themselves without human intervention. An example of such a model is GPT-3 [24]. The concept of sequential learning is to predict the next tokens in a sequence based on the previous context. Given a context sequence X , the model's task is to predict the next token y . The model is trained by maximizing the probability of a given token sequence conditioned on the context [24]:

$$P(y|X) = P(y_t|x_1, x_2, \dots, x_{t-1}) \quad (5)$$

where

- $x_1, x_2, \dots, x_{t-1}$ —the tokens in the context sequence,
- t —the current position.

Using the chain rule, this conditional probability can be decomposed into the product of probabilities for previous tokens [24]:

$$P(y|X) = \prod_{t=1}^T P(y_t|x_1, x_2, \dots, x_{t-1}) \quad (6)$$

where

- T is the length of the sequence.

In this way, the model predicts each token at each position autoregressively, generating a complete text sequence. In summary, the model analyses previous text fragments (the context) and then predicts which token should appear next based on what it has already “seen”. This process occurs sequentially, with each subsequent prediction being based on previous text fragments.

Another important aspect of large language models is the use of attention mechanisms. These mechanisms enable the model to focus on different parts of the text or word sequences, which are then combined to create more precise representations. There are many ways to implement this mechanism [29]:

- Self-Attention: This mechanism is used to combine all positions of a given sequence while preserving their complexity. It is essential for learning long-term dependencies between input data.
- Cross-Attention: In this process, the information representing the input data (often called keys and values) is also used as a query within the component that interprets and processes this information. This allows the model to focus on the most relevant parts of the input data.
- Full Attention: This type of mechanism refers to the calculation of self-attention using the naive method, which assumes that each element of the sequence is directly related to the other elements. This approach is termed naive because, in practice, it can be very computationally expensive.
- Sparse Attention: This is an approximation method for the attention mechanism that reduces the number of relationships between individual elements in the sequence. This technique is used, for example, in GPT to reduce computational complexity.

Attention mechanisms do not consider the order of words in input sequences, so it is essential to use techniques known as positional encoding. There are two main types of positional encoding: absolute and relative positional encoding. Absolute positional encoding involves simply assigning unique identifiers to each position in the sequence before passing them to the attention mechanism [29]. The second method, relative positional encoding, aims to convey information about the mutual dependencies of tokens appearing at different positions in the sequence. One example of this technique is the Alibi method, which reduces the influence of the distance between two tokens when calculating attention. The greater the distance, the lesser the influence. This causes the model to focus on the most recent tokens, ignoring earlier ones in the sequence [32]. Another well-known method of relative positional encoding is rotary positional embedding (RoPE), which operates by

rotating the representation of the query and key proportionally to their absolute positions in the input sequence. The greater the distance between the tokens, the less their influence on each other in this technique. In other words, rotary positional embedding allows for considering the positions of tokens, but their significance diminishes with distance [33].

The use of attention mechanisms, along with appropriate positional encoding, enhances the ability of models to capture complex relationships and dependencies within the text, leading to more accurate and contextually aware language representations. This is crucial for tasks that require a deep understanding of context, such as machine translation, text summarization, and question answering.

4. Research Methodology

The basis for starting this research was the need to find a comprehensive approach to evaluating various numerical text representation techniques. One of the key elements of natural language processing is the way text is represented, so our objective was to compare traditional and advanced methods of representing text. Finding the most effective text representation method could significantly improve not only text classification but also other NLP tasks such as clustering or summarizing texts.

Our research methodology involved exploring multiple text representation techniques, focusing on their effectiveness in classifying AI-generated essays versus those written by students. The innovation of our approach lies in the comparative analysis of different methods, ranging from basic techniques like Bag of Words (BoW) to sophisticated methods such as embeddings created using fastText, MiniLM, and distilBERT, including those using pre-trained models and those trained from scratch. Figure 1 shows a diagram that illustrates the three main stages of the proposed approach.

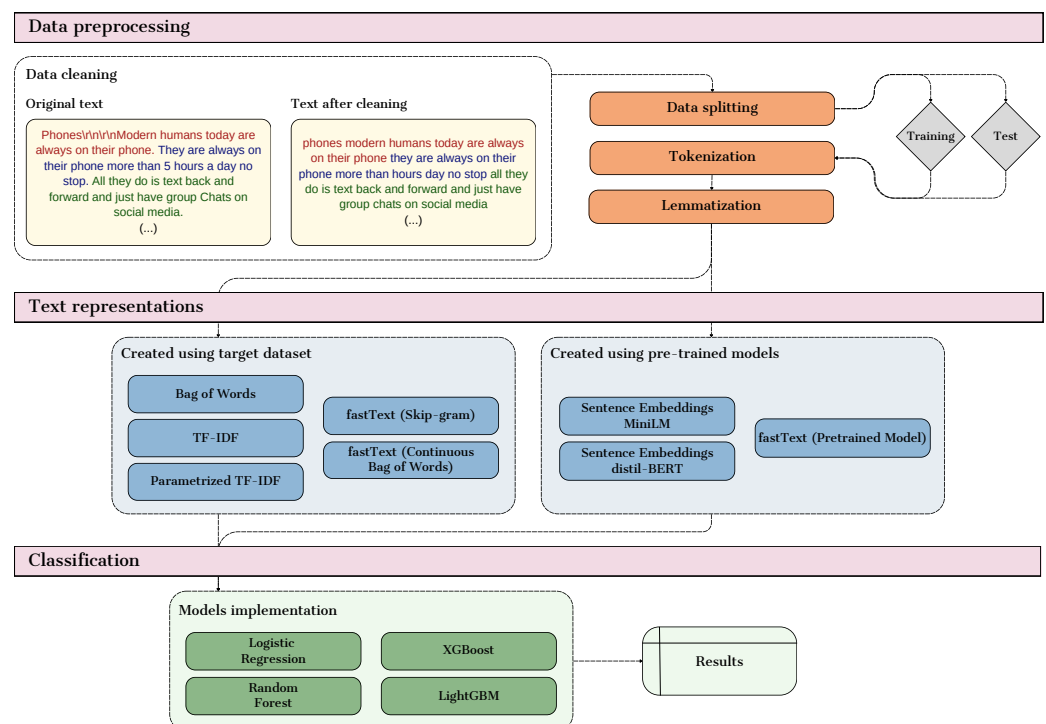


Figure 1. Diagram of the proposed approach.

Each stage plays a crucial role in transforming raw text data into numerical formats suitable for machine learning models and applying various classifiers to assess their effectiveness. The first part of the diagram, which covers data preprocessing tasks such as data cleaning, tokenization, and lemmatization, is explained in detail in Section 4.1. The second part, which focuses on the transformation of textual data into numerical representations, is discussed in Section 4.2, while the final stage, which involves the application of classi-

fication algorithms, is detailed in Section 4.3, and the subsequent evaluation metrics are covered in Section 4.4.

4.1. Data Preprocessing

The first stage of our approach involved pre-processing the content of the documents, which included several key processes. The first was data cleaning, which involved removing unnecessary characters, punctuation, and digits, as all of these can disrupt text analysis. Unnecessary characters, such as special symbols or non-standard text elements, can introduce noise without contributing meaningful information. Similarly, punctuation, while important for sentence structure, can interfere with tasks like tokenization or frequency-based analysis. Digits, unless specifically relevant to the analysis, can skew results by distorting word frequency patterns or adding irrelevant data to processes such as sentiment analysis or topic modeling.

Next, tokenization was used—i.e., dividing the text into individual tokens, which can take the form of individual words or phrases—which allows for more detailed and precise text processing in later stages of analysis. Tokenization is crucial because it allows for breaking down the text into elements that can then be analyzed individually or in context.

The last step of text pre-processing was lemmatization. The aim of this process is to reduce words to their basic forms, or lemmas. This made it possible to ensure consistency in the data and reduce dimensionality, which facilitated analysis. Lemmatization allows one to avoid problems related to the diversity of grammatical forms of words, which could lead to misinterpretations in text analysis.

4.2. Text Representations

Further analysis requires transforming textual data into appropriate numerical structures. After lemmatization, the data consist of lists of tokens for individual essays. To proceed, these token lists are converted back into sequences of text, allowing for the restoration of sentence structure.

The first technique used for numerical text representation was the Bag of Words (BoW). The texts were converted into a matrix where each column represented a unique word in the corpus, and each row represented an essay, showing the frequency of each word. This simple and commonly used technique in natural language processing tasks was applied only to training data at this stage. Transformation of the test data occurred directly before evaluating the classifiers due to the method's speed.

Next, the TF-IDF (term frequency–inverse document frequency) technique was applied. Similar to BoW, it represents texts as vectors, but it accounts for both the frequency of words and their significance across the entire corpus. In addition to the basic TF-IDF matrix, a modified version with selected parameters was configured to improve text representation:

- L2 normalization: adjusts vectors so their Euclidean length is 1, making vector components proportionally scalable, which is especially useful for documents of varying lengths;
- IDF smoothing: adds 1 to the frequency of each term, preventing division-by-zero errors when calculating IDF values for terms not present in any training document;
- Sublinear term frequency transformation: uses a logarithmic scale:

$$TF = 1 + \log(TF) \quad (7)$$

to reduce the influence of terms that appear very frequently, making them less informative.

Like BoW, this transformation was applied only to the training data. The same vectorizers will be used to transform test data during classification.

The fastText model was used to create the first numerical representations in the form of word embeddings. Two models—Skip-gram and Continuous Bag of Words (CBOW)—were trained using the lemmatized training data with the gensim library. Training parameters included:

- vector dimensionality (vector_size) of 300, based on common values in pre-trained models;
- context window size (window) of 5, considering the context of five words before and after a given word;
- minimum word occurrence (min_count) of 3 to exclude rare words;
- training algorithm (sg): 0 for CBoW and 1 for Skip-gram;
- range of n-gram sizes (min_n and max_n) for constructing word vectors, improving model handling of rare words or spelling errors;
- number of buckets (bucket) for hashing n-grams during training, affecting model accuracy and training time.

Additionally, a pre-trained fastText model was used, which was trained on Wikipedia and Common Crawl data. The English model (cc.en.300) uses CBoW with position weighting, considering both context words and their positions. This resulted in three forms of word embeddings, transformed first on the training set and similarly on the test set.

The final stage involved creating sentence embeddings using transformer-based neural network models. Models were accessed via Hugging Face using API interfaces like transformers and sentence_transformers. Selected models were MiniLM (all-MiniLM-L6-v2) and distilBERT (distilbert-base-nli-stsb-mean-tokens), both modified versions of Sentence-BERT (sBERT), designed for semantic sentence embeddings.

MiniLM focuses on minimizing model size while maintaining efficient embeddings. It employs knowledge distillation from a larger model, enabling faster data processing with lower computational requirements [34]. DistilBERT, also a result of BERT model distillation, retains similar-quality embeddings with faster processing and lower resource demands. These models were trained on tasks solved by the larger model, effectively transferring knowledge to a smaller, more efficient model [35].

4.3. Classification

After creating numerical representations, an analysis of various classification methods using these representations was conducted. The classifiers tested include logistic regression, random forests, XGBoost, and LightGBM. Each of these models has different mechanisms of operation and varying efficiency depending on the characteristics of the input data. This allows for a more detailed assessment of the quality of individual numerical representations.

Logistic regression is widely used in binary classification. It is based on the concept of linear regression and uses the logistic function to predict the probability of an observation belonging to one of two classes. Logistic regression is particularly effective when there is a strong linear relationship between the data. It can effectively handle numerical representations based on simple frequencies, such as Bag of Words or TF-IDF weights, where each row of the input matrix is a linear combination of features describing individual documents. However, this model has limitations with more complex data, such as word or sentence embeddings, where data are characterized by more complex, non-linear patterns.

Random forests are a complex structure composed of many decision trees, where each tree is trained on a random subset of input data and a random set of features. The final prediction in a classification problem is the result of majority voting. The use of a set of randomly constructed trees gives this method high resistance to overfitting. It is also effective for data with a large number of features and relatively non-linear patterns, which is typical for extensive textual data.

For logistic regression and random forest, no extensive parameter tuning was performed. Both classifiers were run with the default settings, with the maximum number of iterations set to 1000. This ensures that the algorithms have sufficient iterations to converge. Logistic regression, in particular, can sometimes require more iterations to reach convergence when handling large datasets or complex models, hence the choice of 1000 iterations.

Methods such as XGBoost (Extreme Gradient Boosting) and LightGBM (Light Gradient Boosting Machine) are based on iterative training of decision trees. Unlike random forests, which train many trees independently, these models build successive trees in a sequence,

where each subsequent tree corrects the errors made by the previous ones. XGBoost creates trees sequentially, while LightGBM uses more optimized techniques. Instead of fully creating each tree level sequentially, it only expands the tree leaves that have the greatest impact on improving model accuracy. Both methods are highly effective and scalable on large datasets and tasks with complex data structures, including text data, due to their ability to handle nonlinearities and capture complex patterns.

For XGBoost, the following parameters were used:

- evaluation metric (eval_metric): log loss, which is appropriate for binary classification tasks;
- number of boosting rounds (n_estimators) set to 100. This controls how many trees are built during the training process. A moderate number of estimators was selected to avoid overfitting while ensuring that the model had enough complexity to capture patterns in the data.

For LightGBM, the following parameters were used:

- objective set to binary in order to specify that the task is binary classification;
- evaluation metric (eval_metric): log loss, similar to XGBoost;
- boosting type: gradient boosting decision trees;
- learning rate set to 0.1, a commonly used value which balances the speed of convergence with performance;
- number of leaves (num_leaves) in the trees, where a larger value can lead to better accuracy but may risk overfitting. Based on testing, 40 was chosen as a middle ground to balance these concerns;
- minimum number of data points a leaf can have (min_data_in_leaf) set to 150. A higher value helps prevent overfitting;
- maximum depth of the trees (max_depth): deeper trees capture more complex patterns but risk overfitting, so a value of 8 was chosen as optimal.

The LightGBM parameters were carefully selected to balance performance and overfitting. The learning rate and number of leaves were chosen to prevent the model from overfitting while ensuring it captured important patterns. The maximum depth and minimum data in the leaves were also tuned to control complexity and maintain generalizability.

4.4. Model Evaluation

The evaluation of classification models is crucial for understanding their performance and effectiveness in detecting AI-generated essays. While accuracy is a primary metric for assessing model performance, relying solely on accuracy can be misleading, especially in cases of class imbalance. Therefore, additional metrics such as precision, recall, specificity, F1-score, and Matthews correlation coefficient (MCC) are also considered to provide a comprehensive evaluation.

Accuracy (*Acc*) measures the proportion of correctly classified instances among the total instances and is calculated as follows [8]:

$$Acc = \frac{(TP + TN)}{(TP + FP + TN + FN)} \cdot 100 \quad (8)$$

where

- $(TP + TN)$ —the number of correctly classified AI- and student-written essays,
- $(TP + FP + TN + FN)$ —the total number of AI-generated essays and student-written essays together.

In addition to the model accuracy, additional metrics were also calculated and compared [8].

Sensitivity—*TPR* (true positive rate), calculated as follows:

$$TPR = \frac{TP}{(TP + FN)} \cdot 100 \quad (9)$$

where

- TP —the number of correctly classified AI-generated essays,
 - $(TP + FN)$ —the total number of AI-generated essays.
- Specificity— TNR (true negative rate), calculated as follows:

$$TNR = \frac{TN}{(TN + FP)} \cdot 100 \quad (10)$$

where

- TN —the number of correctly classified student-written essays,
- $(TN + FP)$ —the total number of student-written essays.

Precision— P , calculated as follows:

$$P = \frac{TP}{(TP + FP)} \cdot 100 \quad (11)$$

where

- TP —the number of correctly classified AI-generated essays,
- $(TP + FP)$ —the total number of essays classified as AI generated.

F1-Score— $F1$, calculated as follows:

$$F1 = 2 \cdot \frac{P \cdot TPR}{P + TPR} \quad (12)$$

Matthews Correlation Coefficient— MCC , calculated as follows:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (13)$$

This coefficient indicates the correlation between the predicted and actual classifications, ranging from -1 to 1 , where 1 indicates perfect classification.

5. Experiments and Results

The primary objective of this paper is to compare methods of numerical text representation using various classifiers in the detection of content generated by large language models. This section presents the data used in the comparison process insights from their exploration, which significantly influenced the selection of appropriate data processing and cleaning methods. Selected results of numerical text representation methods and classifiers used for comparison are introduced.

5.1. Dataset Description

Advancements in large language models have achieved remarkable results, enabling the generation of text that closely resembles human-written content. A notable example of such data is the “DAIGT V2” training set [36]. It consists of various prompts, which serve as instructions, and includes several models for AI-generated content. The columns used in the analysis are

- text: the content of the essay;
- label: indicates whether the essay was written by a student (0) or generated using large language models (1);
- prompt_name: the name of the prompt in response to which the essay was written.

The dataset contains 44,868 essays, including 27,371 written by students and 17,497 generated by large language models. Table 1 presents the prompts used along with the corresponding number of essays, divided into those generated by large language models and those written by students. Given that the dataset was supplemented by the community with essays from various sources, it was also verified for duplicate content. Each essay has a unique text.

Table 1. Prompts used along with the corresponding number of essays.

Prompt	Total Number of Essays	Number of Essays Generated by LLM	Number of Essays Written by Students
Distance learning	5554	3397	2157
Seeking multiple opinions	5176	3624	1552
Car-free cities	4717	2051	2666
Does the electoral college work?	4434	1720	2714
Facial action coding system	3084	917	2167
Mandatory extracurricular activities	3077	1407	1670
Summer projects	2701	951	1750
Driverless cars	2250	364	1886
Exploring Venus	2176	314	1862
Cell phones at school	2119	463	1656
Grades for extracurricular activities	2116	490	1626
Community service	2092	550	1542
A Cowboy Who Rode the Waves	1896	524	1372
The Face on Mars	1893	310	1583
Phones and driving	1583	415	1168

5.2. Data Preprocessing

To transform the raw essay content into a numerical representation, it must first be processed and cleaned appropriately. The text must be in a form suitable for analysis, free of digits, punctuation marks, or errors that could hinder the NLP process. The first step involved thoroughly verifying the contents of the essays to find any unusual characters that could not be fully eliminated using standard text-cleaning libraries and functions. The main challenge in the cleaning process was non-standard links present in the text as part of bibliographies. Most links were complex and long, and nearly all contained randomly placed spaces. Constructing a regular expression to eliminate them without risking the removal of text following the links proved problematic. As a result, most links were removed using a single, complex regular expression, applied carefully to avoid deleting any text following the links. The remaining links were analyzed, and a custom list of words to be removed from the essay content was created, including ["www", "org", "html", "http", "https", "edgarsnyder", "pittsburgcriminalattorney", "thezebra", "fivethirtyeight", ...].

It was also noted that the texts contained many newline characters, such as "\n", "\r", and their combinations. Additionally, non-standard punctuation marks, like quotation marks in three different forms (or "instead of the standard"), were found in the text, which were undetectable by punctuation removal functions in NLP libraries. Another characteristic of essays generated by large language models was placeholders that students were supposed to replace, such as "STUDENT_NAME", "TEACHER_NAME", or "Mrs/Mr.{insert principal's name here}". Automatically removing punctuation marks through ready-made functions would result in words like "STUDENTNAME" and similar combinations, likely not found in dictionaries. Therefore, characters in these characteristic forms (like newline characters and non-standard quotation marks) were replaced with spaces using regular expressions.

Subsequent steps included converting the text to lowercase, removing remaining punctuation marks, multiple spaces, and digits using the gensim library. Additionally, words consisting of two or fewer letters were eliminated to reduce data sparsity without risking the removal of words that could affect the meaning and context of the entire sentence. The final step was eliminating essays that were incorrectly generated responses by large language models. One example of such a record was an essay identified as code for the prompt "Facial action coding system". Other incorrectly generated essays were identified through character count analysis. The dataset contained six records with fewer than 400 characters, all of which were AI-generated essays with truncated or unfinished content or forms atypical for essays, possibly due to misinterpretation of the prompt by

the model. Table 2 presents examples of the described records, which were removed from the dataset.

Table 2. Examples of essays removed from the dataset.

Text	Label	Prompt_Name
In recent years, there has been a growing trend	1	Distance learning
Passage 1:\r\n\r\nPassage 2:\r\n\r\nPassage 3:\r\n\r\n\r\nPassage 4:\r\n\r\n\r\nPassage 5:\r\n\r\n\r\nPassage 6:\r\n\r\n\r\nPassage 7:\r\n\r\n\r\nPassage 8:\r\n\r\n\r\nPassage 9:\r\n\r\n\r\nPassage 10:\r\n\r\n\r\nPassage 11:\r\n\r\n\r\nPassage 12:\r\n\r\n\r\nPassage 13:\r\n\r\n\r\nPassage 14:\r\n\r\n\r\nPassage 15:\r\n\r\n\r\nPassage 16:\r\n\r\n\r\nPassage 17:\r\n\r\n\r\nPassage 18:\r\n\r\n\r\nPassage 19:\r\n\r\n\r\nPassage 20:\r\n\r\n\r\nPassage 21:\r\n\r\n\r\nPassage 22:\r\n\r\n\r\nPassage 23:\r\n\r\n\r\nPassage 24:\r\n\r\n\r\nPassage 25:	1	Seeking multiple opinions
Limiting car usage is a concept that has gained significant attention in recent years due to its potential benefits. Advocates of this movement argue that by reducing car usage, society can experience numerous advantages, such as improved air quality and reduced greenhouse gas emissions. One example of	1	Car-free cities
<pre>#include <stdio.h>\n#include <stdlib.h>\n#define N 3\n// A utility function to swap x and y\nvoid swap(int* a, int* b)\n{\n int t = *a;\n *a = *b;\n *b = t;\n}\n// A utility function to partition and get pivot\nint partition (int arr[], int low, int high) (...)</pre>	1	Facial action coding system

The cleaned text was placed in a new column. A fragment of the dataset after the preprocessing is presented in Table 3. With the cleaned dataset, the word count distribution for AI-generated essays and student-written essays was verified using the column with cleaned content. The results are presented in Figures 2 and 3.

Table 3. Examples of essays and their cleaned versions.

Text	Label	Prompt_Name	Clean_Text
Phones\r\n\r\nModern humans today are always on their phone. They are always on their phone more than 5 hours a day no stop. All they do is text back and forward and just have group Chats on social media. They even do it while driving. They are some really bad consequences when stuff happens when it comes to a phone. (...)	0	Phones and driving	phones modern humans today are always on their phone they are always on their phone more than hours day no stop all they do is text back and forward and just have group chats on social media they even do it while driving they some really bad consequences when stuff happens when it comes to a phone (...)
This essay will explain if drivers should or should not be able to use electronic devices while operating a vehicle. Using a phone while driving can cause very bad wrecks, because they are putting themselves and other people at risk. People should not be able to use cell phones while operating a vehicle is because bad wrecks, putting others' lives in danger, and may cause death.\r\n(...)	0	Phones and driving	this essay will explain if drivers should or should not be able to use electronic devices while operating vehicle using phone while driving can cause very bad wrecks because they are putting themselves and other people at risk people should not be able to use cell phones while operating vehicle is because bad wrecks putting others lives in danger and may cause death (...)

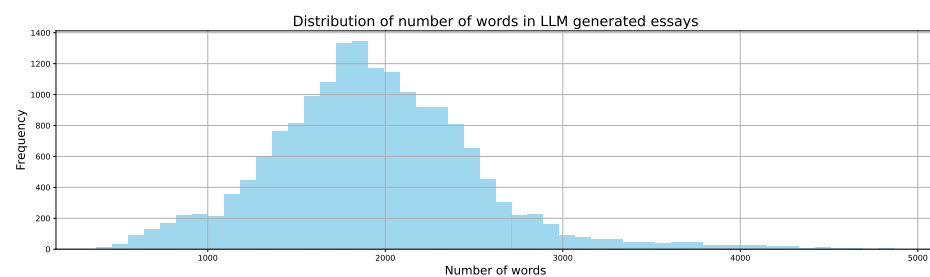


Figure 2. Distribution of number of words in LLM-generated essays.

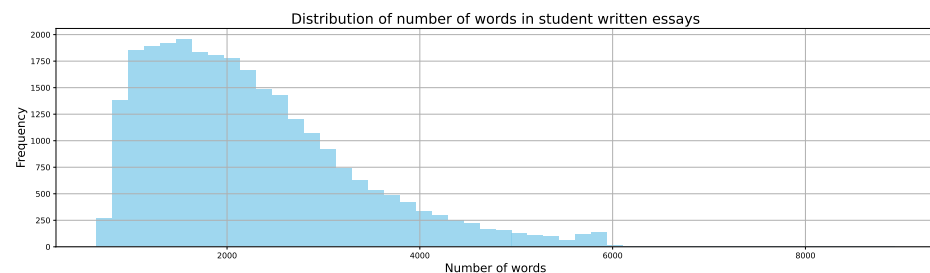


Figure 3. Distribution of number of words in student-written essays.

After filtering and cleaning, the dataset contained only valid essays. Figure 2 presents the distribution of the number of words in AI-generated essays. It can be observed that the word count distribution is similar to a normal distribution. This indicates that while most AI-generated essays center around a particular word count, there is still considerable variation, with some essays having slightly fewer or more words. Also, shorter essays are much more common than in student-written content.

Figure 3 shows the distribution of the number of words in student-written essays. Here, the word count distribution resembles a Poisson distribution, with a higher occurrence of shorter essays. However, despite the higher frequency of shorter essays, student-written essays have a higher minimum word count compared to AI-generated ones, which suggests that students likely perceive an essay as a formal task with a required or expected minimum length. Additionally, the distributions suggest that students tend to write essays with a wider range of lengths, possibly influenced by individual writing styles, interpretations of prompts, or external constraints such as time limits.

Table 4 presents statistical results regarding the number of words in the series. Student-written essays generally have a higher word count, averaging 404, compared to AI-generated essays, with an average of 320 words. The minimum word count for students is 136, and the highest is 1611, more than double the maximum word count in AI-generated essays, where the minimum word count is 62, and the maximum is 764.

The next step in the experiments was to analyze different stop word lists. It was decided not to remove stop words before further analysis due to the presence of words on the lists whose elimination could disrupt the context of entire sentences, such as “against”, “nobody”, or “not”, whose occurrence in a sentence can completely change its meaning. The stop word lists also included many words whose removal would not significantly affect the sentence context, such as “eg”, “be”, “kg”, “ie”, “a”, or “km”. Such expressions and abbreviations were filtered out in earlier stages by removing words with fewer than three letters.

Table 4. Statistics on the number of words in essays.

Statistics on the Number of Words	Essays Generated by LLM	Essays Written by Students
Minimum	62	136
Maximum	764	1611
Average	320	404

The final steps of processing involved tokenization and lemmatization, i.e., dividing essays into tokens and then reducing them to their base form. Space characters were used as the criterion for tokenization. Before lemmatization, the dataset was split into test and training sets. This split helps avoid accidental information leakage between the sets. If lemmatization were applied to the entire dataset before the split, there is a risk that patterns learned during training could be influenced by the test set, compromising the integrity of the evaluation process. By splitting the data first, we ensure that each set is processed independently, maintaining a clear distinction between training and test data and thus preventing potential bias or overfitting. This is important because lemmatization can reveal information about the characteristics of the entire dataset, which could influence how the test data is interpreted by the model. Independent processing of these sets ensures that models are trained and tested objectively, based only on their designated data, and the procedure mimics real-world model usage. A key step in splitting the dataset was balancing the number of essays for each label. The number of essays from the majority class (student-written essays, label 0) was reduced by randomly selecting an equal number of essays as the minority class (AI-generated essays, label 1). This reduced the number of records from 44,860 to 34,978. The balanced dataset was split into test and training sets using stratification by labels, ensuring an even distribution of data in both sets. The test set comprised 25% of the data. The final record counts and label distributions are presented in Table 5.

Table 5. Number of records in training and test sets.

Training Set		Test Set	
0	1	0	1
13,116	13,117	4373	4372

It was also necessary to verify that all the prompts used to generate essays were present in both sets. The analysis did not show omissions in the test or training sets, indicating that each prompt was adequately represented in the data for the training and testing models. In addition, a frequency analysis was performed to ensure that its presence was sufficiently diverse in terms of quantity. The results are presented in Table 6.

Table 6. Prompts used along with the corresponding number of essays in training and test sets.

Prompt	Training Set	Test Set
Distance learning	3557	1181
Seeking multiple opinions	3453	1155
Car-free cities	2848	895
Does the electoral college work?	2568	892
Mandatory extracurricular activities	1868	622
Facial action coding system	1740	548
Summer projects	1550	513
Cell phones at school	1185	366
Community service	1158	388
Driverless cars	1128	431
Exploring Venus	1128	398
Grades for extracurricular activities	1121	405
A Cowboy Who Rode the Waves	1059	341
The Face on Mars	995	321
Phones and driving	875	289

The `en_core_web_sm` language model, a pre-trained English model offered by spaCy, was used to transform tokens into their base forms. This model contains information not only about base forms but also about part-of-speech tagging and syntactic dependencies,

thus considering the structure of the analyzed texts. The process was conducted separately for training and test data.

5.3. Results of Classification

To evaluate the effectiveness of the text representations, we used various classifiers, including:

- Logistic Regression: a baseline model effective for linearly separable data;
- Random Forest: an ensemble method that combines multiple decision trees to improve robustness and accuracy;
- XGBoost and LightGBM: advanced boosting algorithms that iteratively improve model performance by focusing on misclassified instances.

Experiments were conducted on a dataset comprising essays labeled as either AI generated or student written. The dataset was split into training and test sets to objectively evaluate model performance. The division ensured that 75% of the data was used for training and 25% for testing, providing a reliable assessment of each model's performance. Various metrics such as accuracy, precision, recall, and F1-score were used to assess the classifiers.

The comprehensive methodology employed in this study aimed to provide a robust comparison of various text representation techniques, highlighting their strengths and weaknesses in the context of text classification. This approach contributes to a better understanding of how different methods can be leveraged to improve natural language processing tasks. Table 7 shows the accuracy comparison for each classifier, highlighting the highest results for each.

Table 7. Accuracy comparison for individual embeddings and classifiers.

Embedding Technique	Logistic Regression	Random Forest	XGBoost	LightGBM
Bag of Words	99.05	97.90	98.96	98.63
TF-IDF	99.82	97.78	98.78	98.55
Parameterized TF-IDF	98.98	97.72	98.78	98.51
fastText (Skip-gram)	97.80	98.03	98.91	98.83
fastText (Continuous Bag of Words)	97.87	96.59	98.17	97.98
fastText (Pretrained Model)	89.88	95.41	97.42	96.87
Sentence Embeddings MiniLM	95.73	95.06	96.47	96.13
Sentence Embeddings distilBERT	96.63	93.78	96.24	95.60

For logistic regression, the highest accuracy was observed with TF-IDF weights, achieving 99.82%. Similarly high values were obtained with Bag of Words (99.05%) and parameterized TF-IDF weights (98.98%). For random forests, the best performance was with word embeddings created using the fastText model (Skip-gram) trained on the essay dataset, achieving an accuracy of 98.03%. LightGBM also performed well with these embeddings, reaching an accuracy of 98.83%. For XGBoost, the highest accuracy was noted for Bag of Words (98.96%), with closely matching accuracies for TF-IDF weights, parameterized TF-IDF weights, and fastText word embeddings (Skip-gram), at 98.78%, 98.78%, and 98.91%, respectively.

To provide a more detailed comparison, additional metrics were calculated from the confusion matrix. Besides accuracy, precision, recall, specificity, F1-score, and MCC (Matthews correlation coefficient) were considered. The values of these indicators were calculated assuming class 1 (essays generated by large language models) as the positive class. This allows for a precise evaluation of how well the embeddings and classifiers perform in detecting this class. The results are presented in Table 8, with the best results for each numerical representation highlighted.

Table 8. Comparison of metrics for various embeddings and classifiers.

Embedding Technique	Classifiers	Accuracy	Precision	Recall	Specificity	F1-Score	MCC
Bag of Words	Logistic Regression	99.05	99.33	98.76	99.34	99.05	98.10
	Random Forest	97.90	99.25	96.52	99.27	97.87	95.83
	XGBoost	98.96	99.54	98.38	99.54	98.95	97.93
	LightGBM	98.63	99.08	98.17	99.09	98.62	97.26
TF-IDF	Logistic Regression	99.82	99.35	98.28	99.36	98.82	97.65
	Random Forest	97.78	99.22	96.32	99.25	97.75	95.60
	XGBoost	98.78	99.31	98.24	99.31	98.77	97.56
	LightGBM	98.55	99.10	97.99	99.11	98.54	97.10
Parameterized TF-IDF	Logistic Regression	98.98	99.54	98.42	99.54	98.98	97.97
	Random Forest	97.72	99.06	96.36	99.09	97.69	95.48
	XGBoost	98.78	99.24	98.31	99.25	98.77	97.56
	LightGBM	98.51	99.05	97.96	99.06	98.51	97.03
fastText (Skip-gram)	Logistic Regression	97.80	98.76	96.82	98.79	97.78	95.63
	Random Forest	98.03	98.91	97.14	98.93	98.02	96.08
	XGBoost	98.91	99.42	98.40	99.43	98.91	97.83
	LightGBM	98.83	99.35	98.31	99.36	98.83	97.67
fastText (Continuous Bag of Words)	Logistic Regression	97.87	98.49	97.23	98.51	97.86	95.75
	Random Forest	96.59	97.62	95.52	97.67	96.55	93.21
	XGBoost	98.17	98.75	97.58	98.77	98.16	96.35
	LightGBM	97.98	98.68	97.26	98.70	97.96	95.96
fastText (Pretrained Model)	Logistic Regression	89.88	91.56	87.85	91.90	89.67	79.83
	Random Forest	95.41	96.14	94.62	96.20	95.38	90.84
	XGBoost	97.42	98.16	96.69	98.17	97.42	94.87
	LightGBM	96.87	97.45	96.25	97.48	96.85	93.74
Sentence Embeddings MiniLM	Logistic Regression	95.73	96.23	95.20	96.27	95.71	91.47
	Random Forest	95.06	96.55	93.46	96.66	94.98	90.17
	XGBoost	96.47	97.17	95.72	97.21	96.44	92.94
	LightGBM	96.13	96.84	95.38	96.89	96.11	92.28
Sentence Embeddings distilBERT	Logistic Regression	96.63	96.70	96.55	96.71	96.62	93.25
	Random Forest	93.78	95.29	92.11	95.45	93.67	87.61
	XGBoost	96.24	97.31	95.11	97.37	96.19	92.50
	LightGBM	95.60	96.39	94.74	96.46	95.56	91.21

For Bag of Words, TF-IDF weights, parameterized TF-IDF weights, and fastText word embeddings (Skip-gram), accuracies for all classifiers were equal to or greater than 97.72%. For fastText embeddings and Continuous Bag of Words, accuracies were equal to or greater than 96.59%. The lowest performance was observed for embeddings created using the pre-trained fastText model with logistic regression—this was the only instance where accuracy fell below 90% (89.88%). For the same embeddings and other classifiers, accuracies did not exceed 97.42%. For MiniLM sentence embeddings, the highest accuracy was 96.47%, and the lowest was 95.06%. For distilBERT sentence embeddings, the lowest and highest accuracies were 93.78% and 96.63%, respectively.

TF-IDF weights and parameterized TF-IDF weights achieved the highest accuracy with logistic regression. However, for Bag of Words, the fastText model (both Skip-gram and continuous Bag of Words), and MiniLM embeddings, the best performance was with the XGBoost classifier. Similarly, for distilBERT sentence embeddings, the highest accuracy was achieved with logistic regression, but the precision and specificity were higher for the XGBoost classifier. This indicates that, despite slightly lower accuracy, XGBoost better detected the positive class in the test set and made fewer errors regarding negative cases.

6. Conclusions and Future Work

Based on the conducted analyses, it can be observed that the selected numerical text representation techniques combined with various classifiers generally exhibit very high accuracy. The primary criterion for evaluating their effectiveness was the analysis of accuracy. Methods such as Bag of Words, TF-IDF weights (both with default parameter values and those adjusted to the dataset's characteristics), and embeddings created using the fastText model trained on the analyzed training set (for Continuous Bag of Words and Skip-gram) showed accuracies exceeding 96.50% with all classifiers. Among these, the highest accuracy was observed for TF-IDF weights combined with logistic regression, reaching 99.82%.

Regarding sentence embedding methods like MiniLM and distilBERT, accuracies ranged from 93.78% to 96.63%, indicating the potential need for further optimization of classifiers or data processing techniques before applying these methods. The lowest performance was noted for embeddings created using the pre-trained fastText model. In the analysis, one instance of accuracy below 90% was observed: 89.88% for logistic regression using these embeddings. The Matthews correlation coefficient (MCC) value of 79.83% also indicated generally lower performance compared to other methods. This might be due to the model's lower adaptation to the specificity of the analyzed dataset.

It should also be noted that the XGBoost classifier achieved the highest minimum accuracy among the classifiers at 96.24%. This highlights its adaptability and effectiveness when combined with various numerical representation techniques.

Analyzing the specificity values, all the results are equal to or greater than 91.90%. Excluding embeddings from the pre-trained fastText model, all others are equal to or greater than 95.45%, while all basic numerical representations (Bag of Words, TF-IDF weights, and parameterized TF-IDF weights) exceed 99%. This indicates that the models rarely classify negative cases (i.e., student-written essays) as positive (generated by large language models). This is crucial in applications where it is important not to falsely label positive content as AI-generated.

A similar situation is observed for precision values. Here, too, for basic numerical representation methods, all results exceed 99%. Excluding the pre-trained fastText model (where the lowest value was 91.56%), precision is generally equal to or greater than 95.29%. This indicates the high effectiveness of the models at correctly classifying positive cases according to their actual class.

Each text representation method has its inherent strengths and weaknesses. For example, TF-IDF is effective at capturing word frequency and importance within a corpus but lacks the ability to consider word context. BERT embeddings, on the other hand, provide contextualized word representations, which can better capture the meaning of a word in relation to its sentence. However, BERT embeddings are computationally expensive and may not significantly outperform simpler methods like TF-IDF or fastText in smaller datasets. BERT is also trained on large, general corpora, which may not perfectly align with the style and language patterns found in student essays or AI-generated texts. Furthermore, the domain-specific nature of educational texts may require fine-tuning or the use of specialized embeddings, which was not performed in this study. This trade-off between computational complexity and performance is a key limitation, particularly when applying these methods to real-world scenarios with limited resources.

Another important limitation is the interpretability of these methods. While advanced models like BERT capture more complex relationships within the text, they lack transparency compared to traditional methods like TF-IDF. This lack of interpretability can be a disadvantage, especially in educational or academic contexts where understanding the reasoning behind model decisions is important. The simplicity of methods such as TF-IDF makes them easier to interpret, as they rely on well-understood metrics like word frequency, whereas BERT embeddings operate on abstract, high-dimensional representations, which makes it more difficult to trace how specific words or sentences influence the final classification.

The computational costs of generating these representations also vary significantly between methods. TF-IDF and fastText embeddings are computationally efficient and quick to generate, making them suitable for large-scale or resource-constrained environments. In contrast, BERT embeddings require much more time and computational power due to their complex architecture and the need to process entire sequences in context. For example, generating TF-IDF embeddings for this study took a few seconds, while fastText embeddings took several minutes. In comparison, creating BERT embeddings was considerably slower, requiring up to an hour due to the model's size and complexity. This significant difference in time and resource demands further emphasizes the trade-offs between simpler methods and more complex deep learning-based approaches like BERT, depending on the available computational resources and the specific needs of the task at hand.

The studies also confirmed that techniques targeted at the specific nature of the problem, such as those tailored for essays, significantly enhance the effectiveness of further analyses. Comparing information from selected articles regarding the performance of market solutions for detecting text generated by large language models with the results obtained in the classification process, it was observed that focusing the developed solution on a specific problem contributes to more accurate source detection, with all models demonstrating relatively high accuracy. This was also influenced by the appropriate preparation of the text. The choice of suitable preprocessing methods has a significant impact on classification efficiency and is crucial for achieving high model accuracy.

The applied solutions have potential for further development and commercialization, especially in the educational sector, where there is a need to verify the authenticity of content. Many authors of scientific publications emphasize the limitations of general classifiers for detecting text generated by large language models. The research conducted in this study focused on a specific task, which allowed for the appropriate selection of processing techniques and the creation of numerical representations, thereby achieving high classification results. This indicates that dedicated solutions, designed for narrower areas of application, can be more effective. Furthermore, this work provides a solid foundation for further research in the field of detecting texts generated by large language models, opening new perspectives for the development of tools and technologies in this area.

Author Contributions: Conceptualization, N.K., J.K. and B.P.; methodology, N.K., J.K. and B.P.; software, N.K.; validation, N.K., J.K. and B.P.; formal analysis, N.K., J.K. and B.P.; investigation, N.K., J.K. and B.P.; resources, N.K.; data curation, N.K.; writing—original draft preparation, N.K., J.K. and B.P.; writing—review and editing, N.K., J.K. and B.P.; visualization, N.K., J.K. and B.P.; supervision, J.K. and B.P.; project administration, J.K. and B.P.; funding acquisition, J.K. and B.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Research conducted on the DAIGT V2 Train Dataset. Available online: <https://www.kaggle.com/datasets/thedrcat/daigt-v2-train-dataset>, accessed on 20 October 2024.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

NLP	Natural Language Processing
GPT	Generative Pre-training Transformer
BERT	Bidirectional Encoder Representations from Transformers
UCG	Unauthorized Content Generation
BoW	Bag of Words
TF-IDF	Term Frequency-Inverse Document Frequency
LLM	Large language model

References

1. Tang, R.; Chuang, Y.-N.; Hu, X. The Science of Detecting LLM-Generated Texts. *arXiv* **2023**, arXiv:2303.07205. [CrossRef]
2. Tauginienė, L.; Gaižauskaitė, I.; Glendinning, I.; Kravjar, J.; Ojstersek, M.; Robeiro, L.; Odineca, T.; Marino, F.; Cosentino, M.; Sivasubramaniam, S.; et al. Glossary for Academic Integrity. ENAI Report (Revised Version), October 2018. Available online: https://www.academicintegrity.eu/wp/wp-content/uploads/2023/02/EN-Glossary_revised_final_24.02.23.pdf (accessed on 6 March 2024).
3. Foltyniek, T.; Bjelobaba, S.; Glendinning, I.; Khan, Z.R.; Santos, R.; Pavletic, P.; Kravjar, J. ENAI Recommendations on the ethical use of Artificial Intelligence in Education. *Int. J. Educ. Integr.* **2023**, *19*, 12. [CrossRef]
4. Fröhling, L.; Zubiaga, A. Feature-based detection of automated language models: Tackling GPT-2, GPT-3 and Grover. *PeerJ Comput. Sci.* **2021**, *7*, e443. [CrossRef]
5. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv* **2019**, arXiv:1907.11692.
6. Ippolito, D.; Duckworth, D.; Callison-Burch, C.; Eck, D. Automatic Detection of Generated Text is Easiest when Humans are Fooled. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; pp. 1808–1822.
7. Sarzaeim, P.; Doshi, A.M.; Mahmoud, Q.H. A Framework for Detecting AI-Generated Text in Research Publications. In Proceedings of the International Conference on Advanced Technologies, Istanbul, Turkey, 17–19 August 2023; pp. 121–127.
8. Somesha, M.; Alwyn, R.P. Classification of Phishing Email Using Word Embedding and Machine Learning Techniques. *J. Cyber Secur. Mobil.* **2022**, 279–320.
9. Kwon, H.; Lee, J. Detecting Textual Backdoor Attacks via Class Difference for Text Classification System. *IEICE Trans. Inf. Syst.* **2024**, 2023EDP7160. [CrossRef]
10. Kwon, H.; Lee, S. Detecting Textual Adversarial Examples through Text Modification on Text Classification Systems. *Appl. Intell.* **2023**, *53*, 19161–19185. [CrossRef]
11. Suszczańska, N.; Szmaj, P. Inżynieria języka dla systemu Thetos. *Stud. Inform.* **2011**, *32*, 135–150.
12. Vetulani, Z. *Komunikacja Człowieka z Maszyną. Komputerowe Modelowanie Kompetencji językowej*; Akademicka Oficyna Wydawnicza EXIT: Warszawa, Poland, 2004.
13. Hirschberg, J.; Manning, C. Advances in natural language processing. *Science* **2015**, *349*, 261–266. [CrossRef]
14. Płóciennik, I.; Podlaska, D. *Słownik Wiedzy o Języku*; Wydawnictwo Szkolne PWN: Warszawa, Bielsko-Biała, Poland, 2011.
15. Chollet, F. *Deep Learning with Python*; Manning Publications Co: New York, NY, USA, 2018.
16. Tabassum, A.; Patil, R.R. A Survey on Text Pre-Processing & Feature Extraction Techniques in Natural Language Processing. *Int. Res. J. Eng. Technol. (IRJET)* **2020**, *7*, 4864–4867.
17. Silva, S.; Pereira, R.; Ribeiro, R. Machine learning in incident categorization automation. In Proceedings of the IEEE 13th Iberian Conference on Information Systems and Technologies (CISTI), Cáceres, Spain, 13–16 June 2018; pp. 1–6.
18. Wang, D. Semantic Representation and Inference for NLP. Ph.D. Thesis, University of Copenhagen, Copenhagen, Denmark, 2020.
19. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; Dean, J. Distributed representations of words and phrases and their compositionality. In Proceedings of the 26th International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–10 December 2013; pp. 3111–3119.
20. Joulin, A.; Grave, E.; Bojanowski, P.; Douze, M.; Jégou, H.; Mikolov, T. FastText.zip: Compressing text classification models. *arXiv* **2016**, arXiv:1612.03651.
21. D’Sa, A.G.; Illina, I.; Fohr, D. BERT and fastText Embeddings for Automatic Detection of Toxic Speech. In Proceedings of the International Multi-Conference on: “Organization of Knowledge and Advanced Technologies” (OCTA), Tunis, Tunisia, 6–8 February 2020.
22. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pretraining of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Seattle, DC, USA, 10–15 July 2019; Volume 1, pp. 4171–4186.
23. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Polosukhin, I. Attention is all you need. *arXiv* **2017**, arXiv:1706.03762.
24. Chang, Y.; Wang, X.; Wang, J.; Wu, Y.; Yang, L.; Zhu, K.; Xie, X. A Survey on Evaluation of Large Language Models. *J. Assoc. Comput. Mach.* **2023**, *15*, 111–153. [CrossRef]
25. Floridi, L.; Chiriatti, M. GPT-3: Its nature, scope, limits, and consequences. *Minds Mach.* **2020**, *30*, 681–694. [CrossRef]
26. OpenAI. GPT-4 Technical Report. *arXiv* **2023**, arXiv:2303.08774.
27. Shyam, P.; Sastry, G.; Askeel, A. Language models are few-shot learners. *arXiv* **2020**, arXiv:2005.14165.
28. Ziegler, D.M.; Stiennon, N.; Wu, J.; Brown, T.B.; Radford, A.; Amodei, D.; Irving, G. Fine-tuning language models from human preferences. *arXiv* **2019**, arXiv:1909.08593.
29. Naveed, H.; Khan, A.U.; Qiu, S.; Saqib, M.; Anwar, S.; Usman, M.; Mian, A. A Comprehensive Overview of Large Language Models. *arXiv* **2023**, arXiv:2307.06435.
30. Raffel, C.; Shazeer, N.; Roberts, K.; Lee, K.; Narang, S.; Matena, M.; Liu, P.J. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* **2020**, *21*, 5485–5551.

31. Xue, L.; Constant, N.; Roberts, A.; Kale, M.; Al-Rfou, R.; Siddhant, A.; Raffel, C. mt5: A massively multilingual pre-trained text-to-text transformer. In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Online, 6–11 June 2021.
32. Press, O.; Smith, N.; Lewis, M. Train short, test long: Attention with Linear Biases Enables Input Length Extrapolation. In Proceedings of the International Conference on Learning Representations, Online, 25–29 April 2022.
33. Su, J.; Lu, Y.; Pan, S.; Murtadha, A.; Wen, B.; Liu, Y. Roformer: Enhanced transformer with rotary position embedding. *arXiv* **2021**, arXiv:2104.09864. [[CrossRef](#)]
34. SBERT.net Sentence-Transformers. Available online: https://www.sbert.net/docs/pretrained_models.html (accessed on 17 April 2024).
35. Sanh, V.; Debut, L.; Chaumond, J.; Wolf, T. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv* **2019**, arXiv:1910.01108
36. DAIGT V2 Training Set. Available online: <https://www.kaggle.com/datasets/thedrcat/daigt-v2-train-dataset> (accessed on 15 February 2024).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.