# Distinguishing Human and ChatGPT Text

**Likith Kumar Dundigalla**
School of Information
University of Arizona
Tucson, AZ 85721
likithkumard@arizona.edu

## Abstract

In this project, three distinct approaches are explored to distinguish between human-generated and AI-generated text. The existing approach involves generating TD-IDF vectors and comparing their accuracies and F1 scores across multiple models. The prediction is made based on the model that achieves the highest score.The N-grams approach focuses on transforming text data into n-gram features using tokenization and CountVectorizer. This allows the representation of word sequences as features for classification using Multinomial Naive Bayes. Additionally, the Word Embeddings approach leverages Word2Vec models to embed text data into numerical vectors, capturing semantic information. It utilizes Logistic Regression for classification based on these embedded representations. These approaches were designed to encapsulate different aspects of text representation and feature extraction. They aim to capture nuanced patterns and semantic information within the text, improving classification accuracy compared to the baseline TF-IDF approach. The exploration of N-grams and Word Embeddings provides a more comprehensive understanding of text data and its underlying structures. This exploration potentially offers enhanced performance in distinguishing between human and AI-generated text.

## 1 Introduction

Have you ever wondered how to tell if a text was written by a human or an AI? Do you think you could tell the difference? 'Can you tell me about the history of the Kohinoor (Koh-i-Noor) Diamond?'. Try to guess who generated this question: A human or an AI. With the advances in natural language generation, it is becoming harder and harder to tell the difference. In this project, we will build a machine-learning models that will tell if a human or ChatGPT generated the text. Sounds interesting, right? Let's get started.

The project revolves around the task of discerning between human-generated and AI-generated text. The primary objective is to create machine learning model capable of accurately classifying text into these two categories.

To achieve this, the project employs various Natural Language Processing (NLP) techniques and machine learning algorithms. Three distinct methods are proposed and implemented[8]:

### 1.1 TF-IDF Approach

This approach focuses on extracting features from the provided text dataset and training multiple classifiers, such as Logistic Regression, Support Vector Machines (SVM), Decision Trees, and others. The method involves transforming the text data, extracting relevant features, training classifiers, and evaluating their accuracy using confusion matrices. This approach utilizes techniques like TF-IDF vectorization and various classification algorithms to distinguish between human and AI-generated text.

### 1.2 Ngrams Approach

This approach employs Ngrams, which are continuous sequences of words, to tokenize and create features from the text data. By utilizing CountVectorizer to generate Ngrams features and employing a Multinomial Naive Bayes classifier, this method aims to classify text into human-generated or AI-generated categories.

### 1.3 Word Embeddings Approach

Utilizing Word2Vec, a technique for learning word embeddings, this approach transforms text data into numerical vectors to capture semantic similarities between words. The Word2Vec model converts text into vectors, which are then used to train a Logistic Regression classifier to differentiate between human-generated and AI-generated text.

Each method undergoes a similar process of data transformation, feature extraction, model training, and evaluation. The evaluation metrics primarily include accuracy scores and confusion matrices to assess the performance of the models.

These methods are implemented with the goal of demonstrating the effectiveness of different NLP techniques and machine learning algorithms in correctly categorizing text as either human-generated or AI-generated.

## 2 Related Work

**How to Build a Machine Learning Model to Distinguish If It's Human or ChatGPT?**: The objective entails constructing a machine-learning model capable of distinguishing between human-generated and ChatGPT-produced text across various genres, including questions, essays, stories, jokes, code, and more, aiming to develop a versatile classifier capable of handling diverse textual formats. [12]

**Check Me If You Can: Detecting ChatGPT-Generated Academic Writing using CheckGPT**: The paper explores detecting AI-generated academic writing using the GPABenchmark dataset of 600,000 human-written and GPT-generated abstracts. Existing detectors struggle with polished GPT text, confirmed by a user study. CheckGPT, a new LLM-content detector, achieves 98-99% accuracy across disciplines and 90% accuracy in new domains, reaching 98% with domain-specific tuning. Insights from CheckGPT shed light on key behaviors in AI-generated texts. [6]

**AI vs. Human – Differentiation Analysis of Scientific Content Generation**: This study dissects AI-generated scientific content versus human-written text using a multi-feature framework, revealing discrepancies in depth, quality, and a distinct "writing style" gap. It aims to enhance AI models for better content quality, navigating ethical concerns, and optimizing detection methods across domains. [7]

## 3 Procedure

### 3.1 TF-IDF Approach

The project employs a diverse set of machine learning models to distinguish between human and AI-generated text. The process begins with the extraction and transformation of text data from the provided dataset. The models used include Logistic Regression, Support Vector Machines (SVM), Decision Trees, K-Nearest Neighbors (KNN), Random Forest, Extra Trees, AdaBoost, Bagging, and Gradient Boosting classifiers.

The text data is vectorized using the TF-IDF (Term Frequency-Inverse Document Frequency) technique, which represents the importance of words in a document. Each model is then trained on the TF-IDF vectorized data to learn the patterns distinguishing between human and AI-generated text. After training, the models make predictions on test data, and their accuracy is evaluated using confusion matrices. This comprehensive approach aims to leverage the strengths of various classifiers and TF-IDF vectorization to achieve accurate classification of text into human or AI-generated

categories. The entire process involves extracting meaningful features from text, training multiple classifiers, and evaluating their performance in distinguishing between the two text sources.

### 3.1.1 TF-IDF Explanation

TF-IDF (Term Frequency-Inverse Document Frequency) is a technique used in natural language processing to evaluate the importance of a word in a document relative to a collection of documents.[3] It consists of two components:

**Term Frequency (TF):** Measures the frequency of a term (word) within a document. Mathematically represented as:

$$\textbf{TF(t, d)} = \frac{\text{Number of times term } word \text{ appears in document}}{\text{Total number of words in document}}$$

**Inverse Document Frequency (IDF):** Measures the rarity of a term across all documents in a dataset. Mathematically represented as:

$$\textbf{IDF(t, D)} = \log\left(\frac{\text{Total number of documents in the dataset}}{\text{Number of documents containing the word}}\right)$$

The overall TF-IDF score for a term in a document combines these two components by multiplying the TF and IDF scores together:

$$\textbf{TF-IDF(t, d, D)} = TF(t, d) \times IDF(t, D)$$

This score represents the importance of a term in a specific document within a collection of documents. Higher TF-IDF scores indicate that a term is more important or relevant to the document.

### 3.1.2 TF-IDF in TF-IDF Approach

In the context of the TF-IDF approach, TF-IDF vectorization is used to convert the text data into numerical vectors. These vectors represent the importance of words within each document relative to the entire dataset. This process helps in transforming the textual data into a format suitable for training machine learning models like Logistic Regression, Support Vector Machines, and others, allowing these models to learn from the TF-IDF weighted features and effectively distinguish between human and AI-generated text.

## 3.2 Extra Trees Classifier(ETC)

ETC is an ensemble learning method based on decision tree classifiers. It creates a forest of randomized decision trees and aggregates their predictions. "Extra" in Extra Trees refers to the fact that it chooses random thresholds for each feature, hence adding more randomness compared to regular decision trees or random forests. This randomness often helps in reducing overfitting and can be beneficial when dealing with high-dimensional data like text.
Decision trees are the fundamental building blocks of ETC. Each tree is constructed by recursively splitting the data based on features using thresholds that optimize certain criteria, often aiming to maximize information gain or decrease impurity measures like Gini index or entropy.[1]
Information Gain for ETC -

$$\textbf{Gain(S, A)} = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \cdot \text{Entropy}(S_v)$$

Where:
S represents the current dataset at a node.
A denotes the feature under consideration for splitting the dataset (S).
Values(A) refers to the potential values that feature A can take.
$S_v$represents the subset of dataset S corresponding to the value v of feature A.

$Entropy()$represents the calculation of entropy for a given dataset or subset.

$$\textbf{Entropy(S)} = -\sum_{i=1}^{c} P_i \log_2(P_i)$$

Where:
c is the number of classes or outcomes in the dataset S.
$P_i$is the probability of occurrence of class i in the dataset S.

The ETC model here is trained using the TF-IDF transformed training data.

### 3.3 Ngrams Approach

N-grams play a crucial role in text classification tasks by capturing language patterns, differentiating features, and improving model understanding. The choice between unigrams and bigrams affects the granularity of the features extracted.[2][11]

**Unigrams (1-grams):** These are single words considered individually.
**Bigrams (2-grams):** These consist of pairs of adjacent words.

#### 3.3.1 Ngrams in Ngrams Approach

In our approach, `CountVectorizer` is configured with `ngram_range=(1, 2)`, enabling the extraction of both unigrams and bigrams, covering both individual words and pairs of words in the text data for classification.

#### 3.3.2 Multinomial Naive Bayes (MNB)

The Ngrams approach utilizes the Multinomial Naive Bayes (MNB) classifier to predict the category (human-generated or AI-generated) of a given text based on the frequency distribution of Ngrams. The mathematics behind this classifier involves probability estimation using Bayes' theorem and assumes conditional independence between features.

**Bayes' Theorem:**

$$\text{P}(c \mid x) = \frac{P(x \mid c) \cdot P(c)}{P(x)}$$

Where:
$P(c \mid x)$ is the posterior probability of class c given the feature vector x.
$P(x \mid c)$ is the likelihood, the probability of observing the feature vector x given the class.
$P(c)$ is the prior probability of class c.
$P(x)$ is the probability of observing the feature vector x.

In the case of the Ngrams approach, the MNB classifier estimates $P(x \mid c)$, the likelihood of observing a particular Ngram sequence given a class. This calculation involves the frequency distribution of Ngrams in the training data for each class (human-generated or AI-generated).

### 3.4 Word Embeddings Approach

In the Word Embeddings approach, the primary model utilized is Word2Vec. This technique is designed to represent words as high-dimensional vectors in a continuous space where the relationships between words are captured based on their context in the text corpus.[5][4][10]

#### 3.4.1 Word2Vec in Word Embeddings Approach

Word2Vec utilizes two primary architectures: Continuous Bag of Words (CBOW) and Skip-gram.[9]

**Continuous Bag of Words (CBOW):** CBOW predicts a target word from its neighboring context words. Mathematically, CBOW maximizes the probability of predicting the target word given its context:

$$\max \frac{1}{T} \sum_{t=1}^{T} \log p(w_t \mid \text{context}(w_t))$$

**Skip-gram:** In contrast, Skip-gram predicts context words using a target word. It maximizes the likelihood of predicting the context words around a target word:

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

During training, both CBOW and Skip-gram use a softmax function to compute the probability distribution over the vocabulary for predicting the target or context words.

#### 3.4.2 Logistic Regression

Once Word2Vec learns the word embeddings, Logistic Regression is employed as a classifier. The logistic function computes the probability of a text belonging to a category using the word embeddings:

$$P(y = 1 \mid x) = \frac{1}{1 + e^{-(b + \sum_{i=1}^{n} w_i \cdot x_i)}}$$

Where: $P(y = 1 \mid x)$ represents the probability of the text being in the "AI-generated" category.
$b$ is the bias term.
$w_i$ are the weights.
$x_i$ are the word embeddings/features.
$n$ is the number of features (dimension of word embeddings).

The Word2Vec model captures semantic relationships between words based on their context in the text corpus. These word embeddings are then utilized by Logistic Regression for accurate text classification based on the learned relationships.

## 4 Evaluation

### 4.1 TF-IDF Approach Evaluation

The TF-IDF approach initially focused on extracting features and training various classifiers, including Logistic Regression, Support Vector Machines (SVM), Decision Trees, and others. The TF-IDF vectorization technique was employed to represent text data numerically. The models were evaluated primarily based on their accuracy scores and confusion matrices.

**Model Performance** - Each classifier exhibited varied performance in distinguishing between human-generated and AI-generated text. Extra Tree Classifier demonstrated considerable accuracy, achieving notable discrimination between the two categories. However, KNeighbors and Decision Trees Classifiers showed relatively lower accuracy scores.

**Confusion Matrices** - The confusion matrices provided a detailed understanding of the models' performance, indicating the number of true positives, true negatives, false positives, and false negatives. These matrices depicted the classifiers' ability to correctly classify human and AI-generated text, offering insights into misclassifications and error patterns.

### 4.2 Ngrams Approach Evaluation

The Ngrams approach aimed to tokenize text data and leverage Multinomial Naive Bayes (MNB) for classification based on Ngrams' frequency distributions.

**Feature Extraction** - The utilization of `CountVectorizer` with `ngram_range=(1, 2)` enabled the extraction of both unigrams and bigrams. This approach enhanced the granularity of features by capturing sequences of words, potentially improving the models' ability to discern subtle linguistic patterns.

Table 1: Model Evaluation Scores

| Model | Accuracy | F1 Score |
|---|---|---|
| Logistic Regression | 80.67 | 81.21 |
| Support Vector Machine | 77.62 | 77.98 |
| Multinomial Naive Bayes | 69.16 | 68.80 |
| Decision Tree Classifier | 70.13 | 69.21 |
| KNeighbors Classifier | 60.39 | 48.90 |
| Random Forest Classifier | 79.60 | 79.81 |
| Extra Trees Classifier | 81.26 | 81.12 |
| AdaBoost Classifier | 75.70 | 74.92 |
| Bagging Classifier | 77.73 | 77.83 |
| GradientBoosting Classifier | 76.34 | 77.12 |

*Average of 5 executions is taken.

**Classification Performance** - The evaluation of the Ngrams approach highlighted its effectiveness in categorizing text. The Multinomial Naive Bayes classifier demonstrated reasonably good performance, indicating promising capabilities in distinguishing between human and AI-generated text based on Ngrams' frequency distribution.

### 4.3 Word Embeddings Approach Evaluation

The Word Embeddings approach utilized Word2Vec to transform text into numerical vectors, capturing semantic relationships between words for classification via Logistic Regression. **Semantic Representations** - Word2Vec facilitated the creation of word embeddings that encapsulated semantic information. The embeddings were utilized as features to train the Logistic Regression classifier, enabling it to capture nuanced semantic relationships between words. **Classification Accuracy** - The Word Embeddings approach showed promising accuracy in discerning between human-generated and AI-generated text. The model's performance highlighted the significance of leveraging semantic context encoded within word embeddings for accurate text classification.

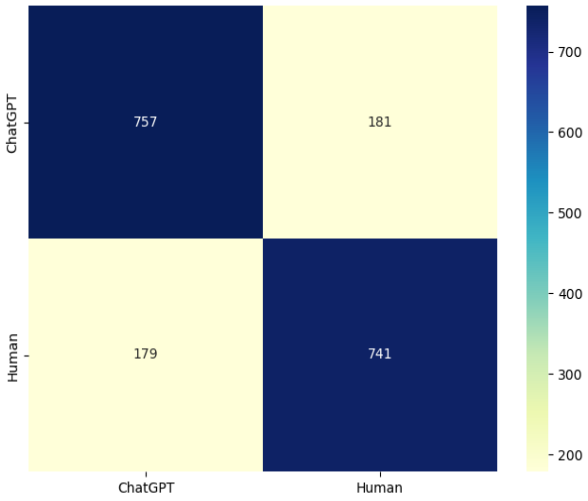
## 5 Results

### 5.1 TF-IDF Approach

The TF-IDF approach showcased commendable performance with an accuracy of approximately 80% for 5000 records. Among the ensemble of classifiers used, Extra Trees Classifier (ETC) emerged as the top performer within this approach, achieving the highest accuracy among the models at 80%. The confusion matrices provided insightful details about the distribution of accurate predictions across both human and AI-generated text categories.

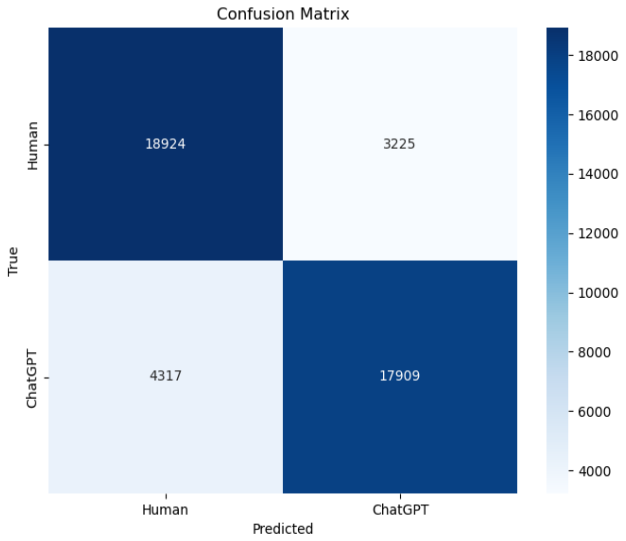
Table 2: Sample Results after Prediction

| Category | ID | Text | Predicted Category |
|---|---|---|---|
| chatgpt | 21 | The boo | chatgpt |
| chatgpt | 22 | As a second-year student of PESU's EEE progra | chatgpt |
| human | 23 | How can I convince my family for marriage? | human |
| chatgpt | 24 | Will Narendra Modi be elected as the Prime Min... | chatgpt |
| human | 25 | What is actual situation of petroleum engineer... | chatgpt |
| human | 26 | He will also talk about new ways NATO and the ... | chatgpt |
| human | 27 | I want to build one marriage hall in my land, ... | human |
| chatgpt | 28 | What is the root of misogyny? | human |
| human | 29 | Cavanaugh also presided over the construction ... | human |
| chatgpt | 30 | What is causing my dog to shake excessively? | chatgpt |




### 5.2 Ngrams Approach

Utilizing Ngrams (both unigrams and bigrams) and Multinomial Naive Bayes, the Ngrams approach aimed to classify text based on the frequency distribution of Ngrams. The accuracy achieved by this approach was notably higher compared to the TF-IDF approach, hovering around 83%. The Ngrams approach revealed intriguing patterns in the confusion matrices, indicating specific strengths and weaknesses in classifying certain types of text.




### 5.3 Word Embeddings Approach

The Word Embeddings approach, utilizing Word2Vec models and Logistic Regression, attained an accuracy of approximately 57%. Although the accuracy was comparatively lower than the other approaches, it provided valuable insights into semantic relationships between words, enhancing the understanding of text distinctions between human and AI-generated sources.

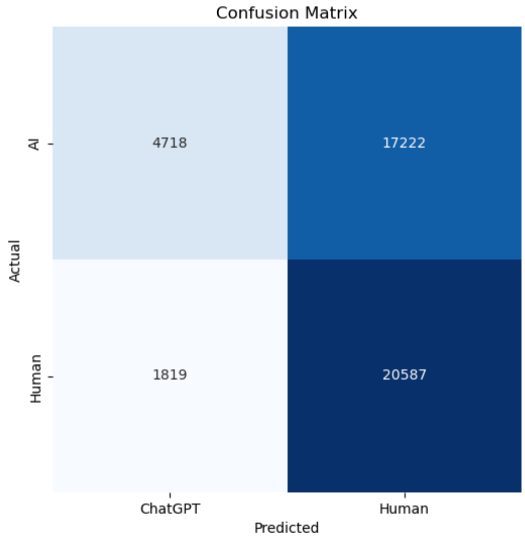



Table 3: Accuracies Comparision of 3 Models

| Model | Records | Accuracy |
|---|---|---|
| TF-IDF Approach | 5000 | 80 |
| N-grams Approach | 125000 | 83 |
| Word Embeddings Approach | 125000 | 57 |

*Average of 5 executions is taken.

## 6 Conclusion

The evaluation of the three distinct approaches illustrated varying levels of accuracy in distinguishing between human and AI-generated text. The Ngrams approach surpassed the other methods, demonstrating the highest accuracy at 83%. The TF-IDF approach followed closely, showcasing competitive accuracy at 80% for just 5000 records, particularly with the Extra Trees Classifier. It shows that as the number of records increases, it could yield better results. Meanwhile, the Word Embeddings approach lagged behind in accuracy; however, its emphasis on capturing semantic nuances within text offered supplementary insights into classification tasks.

These results underscore the significance of different feature extraction techniques and classification algorithms in text classification tasks, paving the way for further exploration and refinement of these approaches.

## References

[1] GeeksforGeeks. *ML | Extra Tree Classifier for Feature Selection*. URL: https://www.geeksforgeeks.org/ml-extra-tree-classifier-for-feature-selection/.

[2] GeeksforGeeks. *N-gram Language Modelling with NLTK*. URL: https://www.geeksforgeeks.org/n-gram-language-modelling-with-nltk/.

[3] GeeksforGeeks. *Understanding TF-IDF (Term Frequency-Inverse Document Frequency)*. URL: https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/.

[4] GeeksforGeeks. *Word Embeddings in NLP*. URL: https://www.geeksforgeeks.org/word-embeddings-in-nlp/.

[5] Kaggle. *Word Embeddings - Logistic Regression*. URL: https://www.kaggle.com/code/kstathou/word-embeddings-logistic-regression.

[6] Zeyan Liu et al. "Check Me If You Can: Detecting ChatGPT-Generated Academic Writing using CheckGPT". In: (2023). URL: https://arxiv.org/abs/2306.05524.

[7] Yongqiang Ma et al. "AI vs. Human – Differentiation Analysis of Scientific Content Generation". In: (2023). URL: https://arxiv.org/abs/2301.10416.

[8] OpenAI Chat. *OpenAI Chat - Shared Conversation*. URL: https://chat.openai.com/share/1954395d-66ba-4d60-8bb7-68aec664fe23.

[9] TensorFlow. *Word2Vec*. URL: https://www.tensorflow.org/text/tutorials/word2vec.

[10] Turing. *Guide on Word Embeddings in NLP*. URL: https://www.turing.com/kb/guide-on-word-embeddings-in-nlp.

[11] Analytics Vidhya. *What are N-grams and How to Implement Them in Python*. URL: https://www.analyticsvidhya.com/blog/2021/09/what-are-n-grams-and-how-to-implement-them-in-python/#:~:text=N%2Dgrams%20are%20continuous%20sequences,(Natural%20Language%20Processing)%20tasks..

[12] Amrutha K (Analytics Vidhya). "How to Build a Machine Learning Model to Distinguish If It's Human or ChatGPT?" In: (2023). URL: https://www.analyticsvidhya.com/blog/2023/04/how-to-build-a-machine-learning-model-to-distinguish-if-its-human-or-chatgpt/#Implementation.