

Textual Analysis and Detection of AI-Generated Academic Texts

A Study of ChatGPT Output, User Instructions, and Machine-Learning Classifiers

Adnan Al Medawer

Bachelor of Science in Engineering - Computer Engineering, Bachelor Thesis

Main field of study: Computer Science

Credits: 15 hp

Semester/year: Spring, 2023

Supervisor: Kyi Thar

Examiner: Patrik Österberg

Course code/registration number: DT099G

Programme: bachelor's programme

At Mid Sweden University, it is possible to publish the thesis in full text in DiVA (see appendix for publishing conditions). The publication is open access, which means that the work will be freely available to read and download online. This increases the dissemination and visibility of the degree project.

Open access is becoming the norm for disseminating scientific information online. Mid Sweden University recommends both researchers and students to publish their work open access.

I/we allow publishing in full text (free available online, open access):

- ☒ Yes, I/we agree to the terms of publication.
- ☐ No, I/we do not accept that my independent work is published in the public interface in DiVA (only archiving in DiVA).

Sundsvall, 2023-05-24

Location and date

Bachelor's Thesis DT099G.....

Programme/Course

Adnan Al Medawer

Name (all authors names)

1994.....

Year of birth (all authors year of birth)

Abstract

This study explores the textual resemblance between AI-generated texts by ChatGPT and original academic texts, compares the performance of AI-detection tools and machine-learning classifiers, including SVM, Logistic Regression, and Random Forest, in detecting AI-generated content, and investigates the influence of user instructions on text quality. A range of metrics such as stylometry, sentiment, text similarity, readability, and relevance were utilized to analyze text characteristics. Findings reveal that while AI-generated texts do exhibit textual characteristics like original texts to some extent, there are clear differences. Machine-learning classifiers, trained on DistilBERT embeddings, achieved an F1 score of 99% for SVM and Logistic Regression, and 96% for Random Forest, surpassing the performance of the AI detection tool, which scored between 64-83% in F1 measure. Detailed instructions to ChatGPT were found to improve the resemblance to original texts and reduce the effectiveness of detection tools. This study contributes to the understanding of AI-generated content and aids the development of more efficient identification methods.

Keywords: AI-generated texts, ChatGPT, Machine-learning, Text characteristics, Language models, Text Analysis, Detection tool.

Sammanfattning

Den här studien utforskar den textmässiga likheten mellan AI-genererade texter av ChatGPT och ursprungliga akademiska texter, jämför prestandan hos AI-detekteringsverktyg och maskininlärningsklassificerare, inklusive SVM, Logistic Regression och Random Forest, vid detektering av AI-genererat innehåll, och undersöker hur användarinstruktioner påverkar textkvaliteten. En rad mätvärden som stilometri, sentiment, textlikhet, läsbarhet och relevans användes för att analysera textegenskaper. Resultaten visar att även om AI-genererade texter uppvisar textegenskaper som originaltexter i viss utsträckning, finns det tydliga skillnader. Maskinlärande klassificerare, tränade på DistilBERT-inbäddningar, uppnådde ett F1 Score på 99 % för SVM och Logistic Regression och 96 % för Random Forest, vilket överträffade prestandan för AI-detektionsverktyget, som fick mellan 64–83 % i F1 Score. Detaljerade instruktioner till ChatGPT visade sig förbättra likheten med originaltexter och minska effektiviteten hos detektionsverktyg. Denna studie bidrar till förståelsen av AI-genererat innehåll och hjälper till att utveckla mer effektiva identifieringsmetoder.

Nyckelord: AI-genererade texter, ChatGPT, Maskininläring, Textegenskaper, Språkmodeller, Textanalys, AI-Detektion Verktyg.

Acknowledgements

I would like to express my gratitude to my supervisor, Kyi Thar, for his valuable insights and suggestions that have influenced the direction of my work. To my friends, I owe many thanks for their steady support and camaraderie. They have provided a comforting and encouraging presence, pushing me forward during challenging moments.

Contents

Sammanfattning	3
Acknowledgements	4
Terminology / Notation	7
1 Introduction	1
1.1 Background and motivation	1
1.2 Problem statement and overall aim	2
1.3 Scientific goals/Research questions	2
1.4 Scope	3
1.5 Outline	3
2 Theory	4
2.1 Text representation	4
2.1.1 Tokenization	4
2.1.2 Word embedding	4
2.2 Large Language models	6
2.2.1 Transfer learning and pretraining	7
2.2.2 ChatGPT and GPT-3.5 series	7
2.2.3 BERT	8
2.2.4 DistillBERT	8
2.2.5 RoBERTa	9
2.3 Supervised machine Learning Classifiers	9
2.3.1 Logistic Regression	9
2.3.2 Support vector Machine (SVM)	10
2.3.3 Random forest	10
2.4 OpenAI GPT-2 detection tool	10
2.5 Text analysis	12
2.5.1 Scientific abstracts	12
2.5.2 Readability Score	13
2.5.3 Stylometry	13
2.5.4 N-gram frequencies	14
2.5.5 Passive voice	14
2.5.6 Part of speech	14
2.5.7 Sentiment analysis	14
2.6 Libraries and dataset	15
2.6.1 ArXiv dataset	15
2.6.2 Libraries	15
2.7 Related work	16
3 Methodology	18

3.1	Scientific method description.....	18
3.2	Work method description.....	19
3.3	Evaluation metrics	23
3.3.1	Precision, Recall, Accuracy and F1 score	23
3.3.2	Cosine similarity	25
3.4	Project evaluation method	26
4	Implementation	27
4.1	Data creation.....	27
4.2	Data cleaning	28
4.3	Quantifying key characteristics in data	29
4.4	Preprocessing and training ML classifiers	30
5	Results	33
5.1	Text characteristics.....	33
5.1.1	Lengths and sentence structure.....	33
5.1.2	N-gram lists	35
5.1.3	Sentiment	37
5.1.4	Readability score.....	39
5.1.5	Syntactic features.....	39
5.1.6	Similarity.....	40
5.2	Measurement results	42
6	Discussion.....	45
6.1	Analysis and discussion of results	45
6.2	Work method discussion	49
6.3	Scientific discussion.....	52
6.4	Ethical and societal discussion.....	52
7	Conclusions	54
7.1	Future Work.....	55
7.1.1	Bigger dataset with more categories and hypered ML solution ...	55
7.1.2	Fine-tuning DistillBERT or similar models.....	55
	References	56
	Appendix A: Distributions of textual features and additional plots..	61

Terminology / Notation

BERT	Bidirectional Encoder Representations from Transformers
GPT	Generative pre-trained transformer
LLM	Large language model
ML	Machine-learning
MLM	Modelling language mask
NLP	Natural language processing
NSP	Next sequence prediction
POS	Part-of-speech
RoBERTa	Robustly optimized BERT approach
SVC	Support vector classifier
SVM	Support vector machine
Std	Standard deviation
TF-IDF	Term Frequency-Inverse Document Frequency

1 Introduction

This chapter serves as a starting point, providing the reader with an overview of the background, problem motivation, and the overall purpose and context of this study.

1.1 Background and motivation

ChatGPT is an AI chatbot developed by OpenAI, that has gotten lot of attention in recent months, due to its impressive capabilities of simulating human-like conversations across various domains. Some key features of ChatGPT are that it can answer questions, assist users with troubleshooting, translate text between different languages, and generate high-quality text on various topics and areas, like human written content. Despite these impressive capabilities, ChatGPT raises concerns about its ability to generate fake facts, and texts like academic content without references or validity. For example, by asking ChatGPT to write an academic paper or assignment on a certain topic. This ability may be abused to create misleading and false academic content.

The traditional way to detect plagiarism is by using text-matching tools that search for text-matching between the target text and some huge database with texts. There are also AI-based tools that claim that it can detect if a text is generated or not. One of these tools is GPT-2 output detector [8], which was developed by OpenAI to detect generated texts. But how effective this tool is in detecting generated text especially for specific domain such academic content.

One way to test the effectiveness of GPT-2 output detector, is to create a dataset containing generated texts that are based on original academic texts and evaluate how much it hits the right prediction. But this way alone does not give a meaningful view of effectiveness without comparing the detector to other methods such as other machine learning (ML) classifiers. Further, the quality of generated contents may differ based on the instructions given to the ChatGPT, which may result in undetectable texts. Therefore, comparing the textual characteristics between texts generated with different instruction and original texts gives a better picture of how these texts differ, and what limitations these differences or similarities can cause in detecting the generated texts. These issues will be further addressed and analyzed in this study.

1.2 Problem statement and overall aim

The increasing popularity and sophistication of AI language models, like ChatGPT, bring about both opportunities and challenges. While these models can generate realistic, academic-like text, there is a risk of misuse, including the spread of invalid or misleading information. Recognizing this, OpenAI developed a GPT-2 output detector, aimed at identifying machine-generated text. However, the distinction between human and AI-produced text is an ongoing research problem, with the newer GPT-3.5 and GPT-4 models posing additional challenges due to their enhanced capabilities.

The investigated problem in this study is to extract textual characteristics from generated data and compare them to the one found in original data, to identify whether there are significant features specific to GPT-3.5 model that are independent of detailed instructions given to GPT-3.5 model. Further, a set of machine learning classifiers will be trained to classify generated data, and the results will be compared to the results of the OpenAI's GPT-2 output detector. This investigated problem will be achieved by generating two datasets, containing scientific abstracts by utilizing GPT-3.5 model. These datasets will be generated based on the titles of scientific papers.

The overall aim of this thesis is to provide an overall picture of the similarity and differences between generated and original contents. Hopefully, the findings of this study contribute to understanding the generated texts better, which may contribute to the development of more robust, and efficient methods for identifying generated content.

1.3 Scientific goals/Research questions

The objectives of this thesis are:

1. Comparing the textual characteristics of generated and original content in terms of readability, stylometry, sentiment, semantic similarity, and relevance to the title.
2. Evaluating whether more detailed instructions given to ChatGPT contribute to significantly higher quality texts and how the details challenges AI-detection tool.

3. Evaluating simpler machine-learning classifiers for classifying generated text and comparing them to OpenAI's GPT-2 output detector.

The research questions are the following:

1. Do generated texts resemble the original texts in terms of textual features?
2. Do machine-learning classifiers trained in academic writing styles give better detection than GPT-2 output detector?
3. Does the detailed instruction given to ChatGPT contribute significantly to the quality of generated texts?

1.4 Scope

This study focuses on generating and analyzing only the abstract part of science papers written in formal English. However, the same procedure taken in this study can be applied when generating long documents, but generating long documents is a more complex task and ChatGPT is not able to do it currently. Additionally, while there are many textual features that can be extracted from texts, this study focusses on a subset of these features. This study will not focus on providing a framework for classifying texts. Due to the cost of using the OpenAI API, the generated datasets for this study are relatively small, and only employ two prompts. The newer model GPT-4 will not be used due to the API not being available directly. Therefore, the dataset is generated by using GPT-3.5 for one topic, which is computer science.

1.5 Outline

Chapter 2 presents relevant backgrounds and information that help the reader to understand the context and choices of the methods and materials in the rest of this report. Chapter 3 describes the procedures and methodology used to perform this study. Chapter 4 explains further the actual implementation that is a complement to the methodology. Chapter 5 presents the outcomes of this study. Chapter 5 discusses the findings and the methodology used, and Chapter 6 concludes the report and suggests future work.

2 Theory

The theory chapter provides necessary background information for understanding the study, including text representation, language models, supervised machine learning classifiers, and text analysis methods.

2.1 Text representation

Text representation [1] in natural language processing (NLP) is an important step in transforming text sequences into a format that can be understood and analyzed by machine learning and language models. In NLP, converting text into numerical representations involves two steps tokenization and word embeddings which are described below.

2.1.1 Tokenization

Tokenization is the process of breaking down text into a list of smaller units, these units are then called tokens which can be words, characters, or any other length. The tokenizer used in language models uses subwords tokenization scheme, which is a combination of word and character tokenization. The technique used in subword tokenization is to train a tokenizer to split words into smaller units from a large collection of texts. The training procedure uses statistical rules and algorithms, which help the model to identify frequently occurring character combinations, sequences within the words and whole words. The frequent words are treated as individual tokens as well as subword units. This approach enables the model to effectively deal with rare or complex words, and misspellings by breaking them down into recognizable subword atomic units, while retaining frequent words as whole tokens for efficient text representation.[1]

2.1.2 Word embedding

Word Embeddings in NLP are numerical representation of words in semantic space, where each word is represented as vector in the space. The semantic space is created based on the distribution of the word's neighboring words, or the context for which a word appears in. These vectors in semantic space capture the semantic meaning and relationships between words, meaning that words with similar meanings or functions will have similar vector representations (their points are close to each other). Figure 2.1 shows a visualization of embeddings learned for sentiment analysis projected into a two-dimensional space. [2]



Figure 2.1: Word embeddings projection from 60 dimensions into 2 dimensions, similar words are clustered close to each other in semantic space, blue color consider neutral words, red color negative words and green positive [2]

Consider these words “bad”, “worst”, “good” and “nice” from figure 2.1. Both “bad” and “worst” share some similarities as both are related to negative emotion, while “good” and “nice” are related to positive emotion and are opposite to negative words. To capture the similarities between these words, one can represent these words in a semantic space as shown in figure 2.1, where their points are close to each other in semantic space if they have similar meaning, and far apart if they have different meanings. The relationships between words can be captured based on their positions in semantic space.

Word Embedding representation in language models are dense (containing no zeros) with continuous values. This type of embedding reduces the vector dimension to a typical range between 50-1000 dimension compared to other types of embedding. This embedding is obtained by training models on large text corpora in a self-supervised manner (not hand-labeled) to learn and capture patterns, and relationships between words in context. Some popular word embedding models are Word2Vec for static embedding and language model BERT (Bidirectional Encoder Representations from Transformers) for dynamic contextual embedding.[2]

In static embedding the model learns one fixed embedding for each word in vocabulary. For instance, consider these two sentences: “We trained a deep neural network to recognize patterns in the data” “We designed a network to enhance the data transfer rate”. In static word embeddings, “network” would have the same vector representation irrespective of its context. In the first sentence, “network” refers to a

structure in deep learning, while in the second sentence, “network” refers to a communication system. Thus, static embeddings wouldn't differentiate between these meanings.[2]

In contextual embedding the model considers the context in which a word appears and maps each word to a dynamic embedding that is different in different contexts. For instance, contextual embeddings would generate different vectors for “network” based on the context. The model understands that “network” is part of the term “neural network” in the first sentence. The generated vector for “network” would be closer to vectors of other AI-related terms.[2]

There is also another type of embedding that is sparse embedding, which means it contains zeros in its vector, one of the popular methods to obtain sparse embedding is Term Frequency-Inverse Document Frequency (TF-IDF) weighting method. This method works in three stages [2]:

First stage is counting the number of times term t occurs in document d in some training corpus, and then taking the log of the count plus 1 to avoid undefined behavior when the count is 0, the equation is given by:

$$TF(t, d) = \log_{10}(\text{count}(t, d) + 1) \quad (2.1)$$

Second stage is to count the number of document DF in which contains the term t , then calculating the IDF as:

$$IDF = \log_{10}(N/DF) \quad (2.2)$$

where N is the total number of documents in training corpus.

Third stage is to calculate the weighting $TF-IDF$ as:

$$W_{t,d} = TF_{t,d} * IDF_t \quad (2.3)$$

The limitation of this method is the dimension of the vectors representing words grows quickly e.g., if there is $|V|=1000000$ unique words in corpus then the dimension is $|V|$, also this methods dose not capture the semantic, context and other structures of words and sentences.[2]

2.2 Large Language models

This section provides a brief description of Large Language Models (LLMs) and transfer learning.

2.2.1 Transfer learning and pretraining

Pretraining refers to the process of training a LLM on large dataset in self-supervised manner to learn the meaning, semantic, and structures of words and sentences. The knowledge learned by the model can then be used on specific tasks through fine-tuning the model with small, labelled data on downstream task, or by using learned word embedding as features in another model, see Figure 2.2.[29]

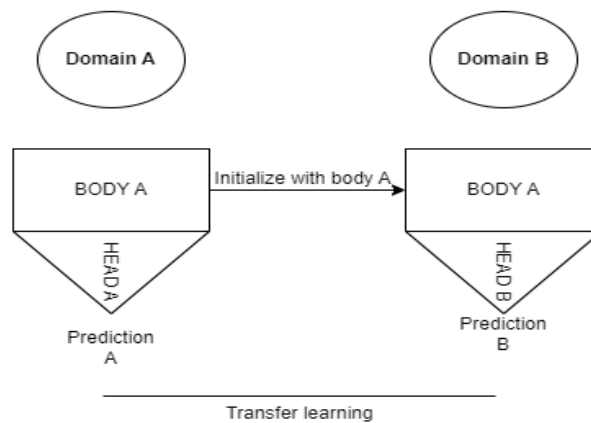


Figure 2.2: Domain B uses the learned knowledge from domain A and applies a specific downstream fine-tuning [29].

2.2.2 ChatGPT and GPT-3.5 series

ChatGPT [3] is a LLM built on the Generative Pre-trained Transformer (GPT) architecture; currently there are two versions of ChatGPT GPT-3.5 and GPT-4, the latter is OpenAI latest mode.

GPT-3.5 are series of models improved on GPT-3 series, developed by OpenAI. These models can generate text based on prompts of some instructions. These models were trained on large datasets up to Sep 2021. The latest models in GPT-3.5 series are text-davinci-003 which is used for text completion, and gpt-3.5-turbo fine-tuned version of text-davinci-003, the latter is optimized for chat and can be used also for text completion. The difference between gpt-3.5-turbo and text-davinci-003 is the cost of using them via API, where the cost of gpt-3.5-turbo is 1/10th the cost of text-davinci-003. [4]

2.2.3 BERT

BERT is a language model, pretrained on both BooksCorpu dataset which contains 800M words and English Wikipedia with 2,500M words. BERT uses WordPiece embeddings in which it contains 30 thousand vocabularies for tokenizing the input sequences. Each sequence of arbitrary length starts and end with special token which are classification (CLS) token that marks the beginning of the sequence, and separation (SEP) token which marks the end of the sequence. The embedding of the classification token represents the whole texts, this embedding is used to tasks such as text classifications.[5]

The key components of BERT training process are masked language modelling (MLM) and next sequence prediction (NSP). In the MLM task, the model tries to guess the masked tokens in a sentence based on surrounding unmasked tokens e.g. "The masked [MASK] is....", here the model tries to predict the [MASK] based on left words "The masked" and right words "is...". This task helps the model to understand and learn the context of words from both directions - left-to-right and right-to-left. While the NSP task makes the model understand relationships between sentences. In this task the model is trained to predict if a sentence is likely to follow a given sentence, thereby learning the context and relationship between sentences. this helps the model learn relationships between sentences and capture higher-level textual structures.[5]

Together, these tasks enable BERT to better understand the semantic meaning of words and the relationship between sentences, which leads to improved performance in a wide variety of language understanding tasks. Thus, Training BERT on these two tasks makes the model learns rich contextualized word representations that can be fine-tuned for downstream tasks such as sentiment analysis, question answering, or text classification. There are two versions of BERT one is BERT base with 110 million parameters, and one is BERT large with 340 million parameters. [5]

2.2.4 DistillBERT

DistilBERT is a smaller mini version of BERT, which was developed by Hugging Face. The main goal behind DistilBERT was to provide a version of BERT that was less resource-intensive, thus easier to deploy in real-world applications, especially where there are constraints on computational resources, such as low memory.[6]

DistilBERT is pretrained using a strategy called knowledge distillation which is a process in which a smaller model called the student model which in this case is DistilBERT, learns from a larger and more complex model called the teacher model which is in this case BERT. DistilBERT model was trained to mimic BERT's outputs and behavior and capture its knowledge. This process makes the student model maintain a simpler architecture which leads to faster processing and less memory requirements. DistilBERT retains approximately 97% of BERT's performance while using 40% less memory and being 60% faster.[6]

2.2.5 RoBERTa

RoBERTa which stands for “Robustly optimized BERT approach” is an optimized version of BERT model that incorporates several modifications to the pretraining process to achieve better performance than BERT on downstream tasks. These modifications in the pretraining process led to improvements in Roberta's performance compared to the original BERT model. [7]:

2.3 Supervised machine Learning Classifiers

Supervised machine learning classifiers are a subset of machine learning algorithms that are trained using labeled data. This section gives a brief description of three machine learning classifiers.

2.3.1 Logistic Regression

Logistic Regression is a machine learning model used for binary classification tasks. The model estimates the probability for which a data point in the dataset belongs to a certain target class. The model uses logistic function called logic, which maps any real number in the input to a continuous value between 0 and 1 which is like the probability of some outcome. These values are mapped to binary classes based on some threshold e.g., if the threshold is 0.5 and the model output 0.7 then it will predict class 1 otherwise predict class 0. [16]

2.3.2 Support vector Machine (SVM)

SVM classifier is a linear machine learning model used for classification tasks of linear data, and for non-linear data it uses what is called kernel trick which works by adding more features. The objective of SVM is to find optimal decision boundaries that best separates two classes in the training data points. These decision boundaries are chosen in way that they maximize the margin between the nearest data points of two classes that lie on, or near decision boundaries. SVM works well for high dimensional small to medium size datasets that have all datapoints in the same scale. [17]

2.3.3 Random forest

Random Forest Classifier is a machine learning model based on ensemble method that combines the predictions of multiple base models as a final prediction. These base models consist of decision trees, with each tree in the forest being constructed and trained independently through a random subset of the training data and a random subset of features for each split, this procedure makes random forest deal with high dimensional data. It also performs features importance by default.[18]

2.4 OpenAI GPT-2 detection tool

OpenAI's researchers have created a dataset where half of the data is outputs from the GPT-2 models and half are samples from the WebText dataset. Thus, any datapoint in the dataset has an equal chance of being real or fake.[8]

The researchers tested and compared different approaches for classifying the generated content, with different sampling methods which can be set as parameter in GPT-2 model, these methods are [8]:

- 1- Temperature: This parameter ranges from 0 to 2, and it controls the randomness of picking the next word. When the temperature parameter is close to zero, the model is more deterministic and likely to pick the most probable word at each step, which makes the completion more repetitive and less diverse. Likewise, When the temperature is closer to 2, the model's output becomes more random and diverse.[9]
- 2- Top-k: This parameter constrains the number of words to pick from. When generating the next word in a sequence, the model calculates probabilities for all possible words, then picks the next

word from the top K probable words. If K is large, the model has more options to choose from, leading to more diverse and less predictable text. If K is small, the model's choices are more limited, leading to more predictable text.

- 3- Nucleus: also known as top-p, this parameter sets a probability threshold and decides the next word from the cumulative probability set of top words that exceeds this threshold.

The approaches used by the researchers for classifying the generated content are [8]:

- Approach 1: they generated data with temperature equal to 1, and trained logistic regression classifier to detect whether the text was generated by a machine or a human. This classifier was trained on features extracted from the text using TF-IDF method for both unigram and bigram combinations features. This classifier was able to detect AI-generated text with accuracies ranging from 88% for GPT-2 with 124 million parameters to 74% for GPT-2 with 1.5 billion parameters.
- Approach 2: they used same approach 1 but constraining the top-k parameter to 40 words (when generating the data for training). The classifier was able to detect AI-generated text with accuracies ranging from 97% for 124 million parameters to 93% for 1.5 billion parameters. They also found that shorter texts were more difficult to detect than longer texts. In addition, the detection would become more challenging if nucleus sampling were used when generating the data.
- Approach 3: the researchers used the GPT-2 model to try to detect its own generated outputs, without any additional training or fine-tuning to improve its detection capability. This approach was found to be less successful with respect to complexity than the TF-IDF based method in approach 1. The accuracy was between 83% and 85% for the model with 1.5 billion parameters.
- Approach 4: same as approach 3 but with fine-tuning the model on an Amazon reviews dataset. This approach led to a decrease in detection accuracy, where the accuracy drops to 74%.
- Approach 5: they fine-tuned RoBERTa models (125 million parameters and 356 million parameters) and achieved approximately 95% accuracy on detecting the output of GPT-2

with 1.5 billion parameters. The researchers also found that the detector's accuracy varies depending on the sampling methods used when generating the text. Training the detector on outputs from larger GPT-2 models improves its ability to classify outputs from smaller GPT-2 models, but it performs less well when classifying outputs from larger models if it was trained on smaller ones. They also found that using a mixed dataset with outputs from different sampling methods and training with random-length sequences of texts improved the accuracy of detection. They suggest that a more diverse and flexible training approach results in a more robust classification, especially for shorter inputs.

The final models for detecting AI generated texts are those fine-tuned RoBERTa models. These models output a probability whether a given text considered generated or not. The models are hosted online on [10] under the name GPT-2-Output-Detector. The models are also available on Huggingface platform under the name "roberta-base-openai-detector" [32].

2.5 Text analysis

This section provides a description of textual characteristics of texts and the characteristics of scientific abstracts.

2.5.1 Scientific abstracts

There are many definitions of what an abstract is, but the general definition is that an abstract is a concise summary of a large document, this summary allows readers to quickly determine the relevance of a document to their needs. It mentioned that writing an abstract can be challenging, as it requires being concise and using dense and diverse language with many compound words, which makes reading abstracts difficult.[19]

The structure of abstracts consists of five moves: Introduction, Problem, Method, Evaluation, and Conclusion [19]:

- 1- The Introduction sets the context and often hints at the problem.
- 2- The Problem presents the issue dealt with in the paper, it can be also merged with the Introduction move.
- 3- The Method explains how the problem is addressed and resolved.
- 4- The Evaluation validates the solution and is crucial for judging the paper's relevance.

5- The Conclusion places the results in a broader context.
All mentioned moves are not strictly followed by researcher.[19]

2.5.2 Readability Score

Text readability as described at [13] has various definitions where each author has defined readability in different way, but in general readability refers to how easily and efficiently a piece of text can be read by target audience. Text readability is measured by a readability formula, which is an analytical method used to measure the readability of written materials. There are many readability formulas, but the most popular readability formula that is widely used and used in Microsoft word is the Flesch Reading Ease (FRE) Formula which is calculated as following [13]:

$$FRE = 206.835 - (1.015 \times ASL) - (84.6 \times ASW)$$

The average sentence length (ASL) is the number of words divided by the number of sentences. The average number of syllables per word (ASW) is the number of syllables divided by the number of words. A syllable is a unit of sound in a word that consists of a single and uninterrupted sound e.g., “comfortable” consists of three syllables “com”, “for” and “able” while “cat” consists of a single syllable.[13]

The Flesch Reading Ease formula gives a score between 0 and 100, with 0 representing the highest reading difficulty and 100 representing the lowest reading difficulty, see Figure 2.3 for the interpretation of the Flesch Reading Ease Score [13].

FLESCH READING EASE SCORE			
Reading Ease Score	Description	Predicted Reading Grade	Estimated Percentage of U.S. Adults
0-30	very difficult	college graduate	4.5%
30-40	difficult	college grade	33%
50-60	fairly difficult	10 th -12 th grade	54%
60-70	standard	8 th -9 th grade	83%
70-80	fairly easy	7 th grade	88%
80-90	easy	6 th grade	91%
90-100	very easy	5 th grade	93%

Figure 2.3: Flesch Reading Ease scores interpretation [13].

2.5.3 Stylometry

Stylometry features help in identifying writing style in texts which are not dependent on the topic. These features include word count, n-gram, part-of-speech tags, passive voice, and other meta-features such average sentence length and number of sentences. [20]

2.5.4 N-gram frequencies

N-gram is all combinations of n consecutive words in each sentence without considering punctuation [19]. For instance, in the given sentence "The cat eats the mouse.", the 2-grams are "The cat", "cat eats", "eats the", "the mouse" and 3-grams are "The cat eats", "cat eats the", "eats the mouse", these sequences represent consecutive word combinations of length 2 and 3.

2.5.5 Passive voice

Passive voice is a grammatical rule used to emphasize the object that receives the action, rather than the subject that do the action, while active voice is the opposite where the emphasizing is on the subject. For instance, consider these two sentences; active voice: "The man eats the food" and passive voice: "The food was eaten by the man." in the active voice sentence the subject "the man" is performing the action of eating the food. In the passive voice sentence, the focus goes to the object "the food" that is receiving the action "eaten by the man" so the emphasize is on the process of the food being eaten.[28]

2.5.6 Part of speech

Part-of-speech (POS) is a grammatical category of words in a language. These categories are used to classify words based on their function within a sentence, and they help to determine the structure and meaning of a sentence. Each POS plays a specific role in each sentence, and a word's POS is determined by its context and usage, see Figure 2.4 shows some POS in English language.[15]

Tag	Definition	Example
NOUN	A word that represents a person, place, thing, or idea.	dog, city,data
VERB	A word that represents an action or a state of being.	run, is, try
ADV	A word that modifies a verb, adjective, or other adverb.	quickly, very, bad
ADJ	A word that describes a noun or pronoun.	quick, brown, badly

Figure 2.4: POS tags and their meaning [15].

2.5.7 Sentiment analysis

Sentiment analysis is a metric that can be used to determine the feelings and emotions expressed in texts toward some topic. The analysis helps identify whether the writer has a positive or negative attitude towards a

particular subject, like a movie, book, or product. Sentiment subjectivity is the degree to which a piece of text expresses personal opinions, feelings, or beliefs, which is the opposite to objectivity that expresses facts or information. Sentiment polarity is a classification of the sentiment expressed in some text as positive, negative, or neutral. This classification is based on the words, phrases, and their associated sentiment scores, which are derived from sentiment lexicons or learned through machine learning models. Also, sentiment is not dependent on the type of texts.[14]

2.6 Libraries and dataset

This section gives a short description of the dataset and some of the libraires used in this project.

2.6.1 ArXiv dataset

The arXiv dataset [34] is a collection of research papers and preprints from various scientific fields, which are available on the website arXiv.org. This platform allows researchers to share and publish their work under certain conditions with others in the scientific community. The dataset contains over 2.3 million papers in various scientific topics, such as physics, mathematics, computer science, and quantitative biology. Each paper is assigned a top-level category and one or more subcategories, which helps in organizing the content see Figure 2.5.

Computer Science		
cs.AI (Artificial Intelligence)	Covers all areas of AI except Vision, Robotics, Machine Learning, Multiagent Systems, and Computation and Language (Natural Language Processing), which have separate subject areas. In particular, includes Expert Systems, Theorem Proving (although this may overlap with Logic in Computer Science), Knowledge Representation, Planning, and Uncertainty in AI. Roughly includes material in ACM Subject Classes I.2.0, I.2.1, I.2.3, I.2.4, I.2.6, and I.2.11.	
cs.AR (Hardware Architecture)	Covers systems organization and hardware architecture. Roughly includes material in ACM Subject Classes C.0, C.1, and C.5.	
cs.CC (Computational Complexity)	Covers models of computation, complexity classes, structural complexity, complexity tradeoffs, upper and lower bounds. Roughly includes material in ACM Subject Classes F.1 (computation by abstract devices), F.2.3 (tradeoffs among complexity measures), and F.4.3 (formal languages), although some material in formal languages may be more appropriate for Logic in Computer Science. Some material in F.2.1 and F.2.2, may also be appropriate here, but is more likely to have Data Structures and Algorithms as the primary subject area.	
cs.CE (Computational Engineering, Finance, and Science)	Covers applications of computer science to the mathematical modeling of complex systems in the fields of science, engineering, and finance. Papers here are interdisciplinary and applications oriented, focusing on techniques and tools that enable challenging computational simulations to be performed, for which the use of supercomputers or distributed computing platforms is often required. Includes material in ACM Subject Classes J.2, J.3, and J.4 (economics).	
cs.CG (Computational Geometry)	Roughly includes material in ACM Subject Classes I.3.5 and F.2.2.	
cs.CL (Computation and Language)	Covers natural language processing. Roughly includes material in ACM Subject Class I.2.7. Note that work on artificial languages (programming languages, logics, formal systems) that does not explicitly address natural-language issues broadly construed (natural-language processing, computational linguistics, speech, text retrieval, etc.) is not appropriate for this area.	
cs.CR (Cryptography and Security)	Covers all areas of cryptography and security including authentication, public key cryptosystems, proof-carrying code, etc. Roughly includes material in ACM Subject Classes D.4.6 and E.3.	
cs.CV (Computer Vision and Pattern Recognition)	Covers image processing, computer vision, pattern recognition, and scene understanding. Roughly includes material in ACM Subject Classes I.2.10, I.4, and I.5.	
cs.CY (Computers and Society)	Covers impact of computers on society, computer ethics, information technology and public policy, legal aspects of computing, computers and education. Roughly includes material in ACM Subject Classes K.0, K.2, K.3, K.4, K.5, and K.7.	

Figure 2.5: Top level category for computer science [Cs] and subcategories.

2.6.2 Libraries

Huggingface Transformers is a library that offers pre-trained models and datasets for various tasks such translation, tokenization, classifications, and sentiment analysis. [21]

Textstat is a library that handles various tasks related to text analysis. It has a set of function for calculate readability scores, lexical diversity, and other text attributes.[22]

Sentence-transformers is a library built on Huggingface Transformers and offers pre-trained models and functions for semantic search, text clustering, and similarity measurements.[23]

PassivePy is a library that analyzes sentences and identifies passive voices in texts. [24]

TextBlob is library designed to simplify common NLP tasks such as text analysis, part-of-speech tagging and sentiment analysis.[25]

Seaborn is a data visualization library in Python built on Matplotlib, it offers an interface for creating statistical graphics, and it simplifies the process of making complex visualizations.[26]

Sklearn is a library for machine learning and data science. It includes various tools and models for data preprocessing, evaluation metrics, machine learning classifiers and other tools related to machine learning.[27]

2.7 Related work

This study [12] aimed to evaluate the capability of ChatGPT in generating abstracts for 50 scientific medical papers from five high-impact journals. The authors assessed the generated abstracts and original abstracts using three methods: text-matching plagiarism detection tool, OpenAI's output detector and human reviewers. The results showed that all the ChatGPT-generated abstracts appeared to be in the format of a scientific abstract, but only 16% correctly used the journal-specific headings. The OpenAI's output detector found a high probability of generated text in many of the generated abstracts ranging between 12.73 % to 99.98 % where 17 corresponding to 34% of datapoints were identified as generated with less than 50%. The detector gave low probabilities for the original abstracts ranging between 0.02% to 0.09%. The plagiarism checker reported that almost all the generated abstracts were completely original.

Human reviewers correctly identified 68% of the generated abstracts as being generated and 86% of the original articles as being original.[12]

The similarities between the study at [12] and this study is that both generate the abstract part of scientific papers and evaluate OpenAI's detection tool. However, this study delves deeper into textual characteristics of generated data, utilizes ML classifiers, and evaluates OpenAI's detection tool more systematically.

3 Methodology

This chapter presents the methodology used to perform this study to achieve the objectives of this study and answer the research questions. It outlines the research approach, data collection procedures, data analysis strategies, workflow stages, and evaluation methods adopted in this study.

3.1 Scientific method description

In this study, quantitative research methodology will be employed. This method was chosen due to its strengths in allowing for the systematic collection and interpretation of numerical data, thus enabling an understanding of patterns and relationships. Additionally, this study will incorporate elements of comparative and experimental research designs, allowing for a structured approach to addressing the research questions.

The first research question is to assess whether ChatGPT resembles the textual characteristic in abstract part in scientific papers. To answer this question various metrics will be extracted from the texts. These metrics include the number of unique words, sentence count, average sentence length, passive sentence count, frequency of noun and verb usage, frequency of adjectives and adverbs, sentiment polarity and subjectivity, title relevance to text, text similarity, n-gram lists frequencies and readability. Each metric will be computed using statistical and NLP tools. All these metrics and measurements will be subjected to descriptive statistics to provide an overall picture of generated and original texts. The resulting analysis will provide insights into the textual characteristics found in the texts, which will be used to perform the comparison between these texts.

The second research question is to assess whether machine-learning classifier trained on academic writing style give better detection than GPT-2 output detector. This will be achieved by training three candidate machine learning classifiers on the datasets, the output of these classifiers will be then compared to the output of GPT-2 output detector. The comparison will be based on four metrics: F1 score, precision, recall and accuracy.

The third research question is to assess how the detailed instructions given to ChatGPT affect the quality of the generated texts. In addressing this question, an experimental research approach will be employed. The level of detail in the instructions given to ChatGPT will be manipulated, and the quality of the generated text will be evaluated. Before proceeding further, let define what meant with “quality” in this context. The quality is defined in terms of which set of the two generated sets resemble closer the characteristics of the original texts and challenge the detection tools. The answer to this question will be based on both first and second questions, where the analysis obtained from the first question will be used to determine the quality, together with the evaluation of ML classifiers and OpenAI’s detection tool.

The subsequent sections in this chapter will provide a detailed description of the data collection procedures, the data analysis strategy, and the specific steps undertaken in the experimental design.

3.2 Work method description

This project consists of seven phases in total, phase one is literature study, phases 2-6 related to planning and implementation, phase 7 is related to results and conclusion. The phases will be carried out using a sequential approach, where each phase will be completed in order before moving onto the next. However, in some phases may return to earlier phases for revisions or improvements, so the work method may partially include iterative methodology.

3.2.1 Phase 1: Literature study

The first phase consists of literature study. During this phase relevant research papers, e-books, books, articles, and other materials were collected to gain a better understanding of the current research area. The literature study covered works related to text classification, language models, text analysis, plagiarism detection and paraphrasing. The literature study helped in choosing candidate methods and metrics that will be used in this study as well as knowing their limitations. This phase provides the necessary context and theoretical basis for completing this study. At this phase Google Scholar has been used mainly to get relevant information and scientific papers.

3.2.2 Phase 2: Data collection and generation

The second phase is dataset creation, during this phase the ArXiv dataset will be downloaded from Kaggle platform [34], this dataset will be then prepared for sampling. This dataset was chosen because it is suitable for the problem since it contains academic abstracts with other meta data. The preparation include that abstracts that is shorter than 200 words and longer than 300 words will be dropped, this step is taken because this length is the typical abstracts length, and the models used in this study has maximum token equal to 512 tokens, which corresponds to around 200-350 words. Also, abstracts that have long title or short titles will be dropped a well, the length of the tiles is chosen to be between 5-20 words, this range is chosen because shorter length will not provide enough information for GPT-3.5 models on the topic, while longer length may make the models use the title as it with some modifications and without diversity. After preparation, a sample of 800 datapoints will be taken.

Based on the titles of the abstracts in the sampled ArXiv dataset, two separate abstract datasets will be generated using OpenAI's API. This will be achieved by providing the API with two different prompts, which are instructions given to GPT-3.5 models to generate data. The models that will be used to generate data are "text-davinci-003" and "gpt-3.5-turbo". Both data sets will be generated using 0.7 temperature setting (see section 2.4) and only one version will be generated. The chosen temperature is based on the actual temperature used in ChatGPT.

First dataset will be generated using this prompt **'Write a scientific abstract between 200 to 300 words on the following topic: {title}'** the chosen prompt is like the prompt used in study [12] but not identical, and the model used to generate the data is "text-davinci-003". **This dataset will be called Prompt A and the instructions given to the model will be referred to as prompt A.**

The second dataset will be generated using this prompt **'Write a coherent, detailed and well-structured scientific abstract of {text length} words on the following topic: "{title}"'. The abstract should be written in rich, clear, and academic language.'**, and the model used is "gpt-3.5-turbo", the chosen prompt gives the GPT-3.5 model more instruction, the length given to the model is the same length as the original abstract. **This dataset will be Prompt B and the instructions given to the model will be referred to as prompt B.**

For prompt B will explicitly request a “coherent, detailed and well-structured” abstract. This would put more emphasis on logical flow, detailed explanation, and overall structure in the output. In contrast, prompt A doesn't mention these specific qualities, which means there could be more flexibility in the structure and detail of the abstract. Additionally, prompt B specifically asks for the abstract to be written in "rich, clear, and academic language." This would suggest the output should be more diverse, employ complex sentence structures, and use academic jargon relevant to the topic. Prompt A, on the other hand, doesn't specify the level of language, which could allow for a simpler, more accessible style.

Each of the generated datasets will be combined separately with the original sampled ArXiv dataset. The purpose of this combination is to form a dataset consisting of both human-created and AI-generated data, which will subsequently be used to train a classifier and quantify text characteristics.

3.2.3 Phase 3: Data cleaning

The third phase includes data cleaning, during this phase outliers found within generated data will be identified and removed, along with their corresponding original abstracts. This step is essential to ensure reliable results when comparing the generated and original texts, as well as when training machine learning classifiers. All abstracts will also be cleaned by truncating any leading and trailing spaces. Additionally, all abstracts will be converted to lowercase to ensure consistency and ease of analysis.

3.2.4 Phase 4: Quantifying textual characteristics

The fourth phase consists of quantifying key characteristics of the generated and original abstracts. These characteristics include readability, text similarity between generated and original abstracts, relevance to the title, stylometry, and sentiment. Various NLP tools and libraries will be utilized to measure these attributes, see section 2.6 for more details about the libraries.

Readability will be evaluated using the Flesch-Reading-Ease score (see section 2.5). Although there are other popular metrics for readability, this one was chosen to limit the scope of the analysis. The weakness of this

method is that it gives a negative score which is hard to interpret in terms of which target audience the text is suited for. Also, this metric doesn't assess the complexity of word meanings.

Text similarity and relevance to the title will be measured using Cosine Similarity (see section 3.3). The model 'all-MiniLM-L12-v2' provided by Sentence Transformer library at [23] will be used to measure the semantic similarity. The chosen model is trained on large and diverse labeled datasets specifically made for semantic text comparison. The strategy to measure title relevance is by splitting abstract into a set of sentences where each sentence is measured against abstract's title, the final score will be the mean of all scores. The weakness of this method is when the score is negative this will cancel out positive scores for example if 3 out of 5 sentences have value 2 and the other two sentences has -2 then the final score is 0 which is not correct. Stylometry will be evaluated using statistical methods. Sentiment scores will be measured using NLP tools designed for this purpose.

3.2.5 Phase 5: ML training and evaluation

The fifth phase is training, hyper parameters search, and evaluating machine learning classifiers. Three candidate classifiers have been selected from among many: SVM, Logistic Regression, and Random Forest (see section 2.3). In this phase, the texts will be tokenized using the DistilBERT tokenizer, and the tokens will be passed into DistilBERT to extract embeddings corresponding to (CLF) token for classification (see section 2.2.3 and 2.2.4), which will be used as features in training the machine learning classifier. DistilBERT was chosen because it is lighter and suited for limited hardware compared to other models. It also produces dynamic contextual embeddings, which helps in generalization when training the machine learning classifier. These classifiers will be evaluated using four metrics: accuracy, precision, recall and F1 score. These metrics are described further in section 3.3. The evaluation will be done by using cross validation with 5 folds. As base line Dummy classifier provided by Scikit-learn will be used with uniform strategy this is like flipping a coin and guessing the head.

3.2.6 Phase 6: Evaluating OpenAI's detection tool.

The sixth phase is evaluating OpenAI's output detector. During this phase, the detector will be downloaded from the Hugging Face platform at [32]. The texts will first be tokenized using the model's tokenizer and

truncated to the maximum number of tokens that the model accepts to avoid any errors during the process. Since the model outputs a probability, this probability will be mapped to a binary label using a threshold of 0.5.(same threshold in ML classifiers used) where label '1' is generated and '0' is original, for example, if the model detects that the text is fake with a probability greater than or equal to 0.5, then the label will be 1, indicating a generated text. Otherwise, the label will be 0, meaning the text is classified as 'real'. These values will be evaluated using the F1, precision, recall and accuracy.

3.2.7 Phase 7: Presenting the results.

The seventh phase consists of plotting the key characteristics and comparing the scores obtained from the evaluation metrics for ML classifiers and detector. In this step will be visualizing the data using appropriate graphical representations, such as histograms, bar charts, scatter plots and boxen plots. These visualizations will help to identify patterns in the data, providing a clear comparison of the characteristics and overall picture of the AI-generated abstracts and the original abstracts. The analysis includes comparing average scores, median distributions, and correlations among different variables. This phase is crucial for understanding the implications of the results and for drawing meaningful conclusions from the study.

3.3 Evaluation metrics

This section describes the evaluation metrics used for evaluating OpenAI's detector, machine learning classifier, similarity between original abstracts and generated abstracts and title relevance to the abstract.

3.3.1 Precision, Recall, Accuracy and F1 score

Precision, recall and F1 are metrics that measure how well a classifier performs on predicting the correct target class. The metrics are derived from confusion matrix which is for binary classification problem a 2×2 matrix, where the rows are the true classes, and the columns are the predicted class. The upper left cell in confusion matrix is called true negative (TN) which contains the instances of negative class that has been correctly classified as negative, the lower right cell is true positive (TP) which is the target class in interest and contains the instances that

has been correctly classified as positive, the lower left cell is false negative (FN) cell and contains the instances that has been incorrectly classified as negative, The last cell is the upper right cell called false positive (FP) that contains the instances that has been misclassified as positive while they belong to the negative class, see Figure 3.3-1 that shows confusion matrix for arbitrary example.[30]

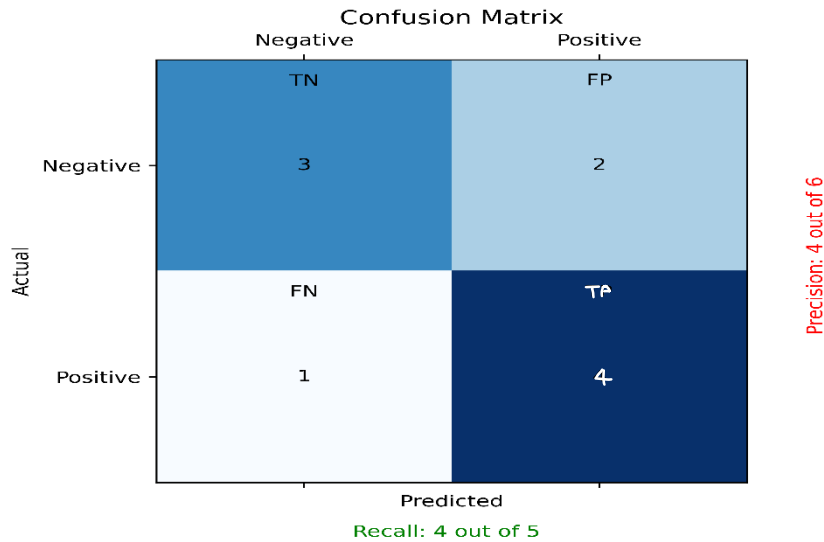


Figure 3.3-1: Confusion matrix example, the classifier has missed 1 instance of 5 positive instances, and it misclassified 2 instances as positive out of 6 positive instances [30].

These metrics can be calculated based on the confusion matrix as following [30]:

Accuracy: is the ratio of correctly classified instances out of the total instances, this metric is not suitable for imbalanced dataset and dose not gives how well the classifier perform on specific class, but it gives the performance for classifying both negative and positive classes.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (3.1)$$

Precision: is the ratio of true positive class out of all positive prediction. Precision measures the quality of the classifier when it predicts the positive class where high precision indicates a low rate of false positive errors, but precision does not give how much of all positive classes the classifier could identify therefore precision is combined with another metrics such Recall.

$$Precision = \frac{TP}{TP+FP} \quad (3.2)$$

Recall: is the ratio of true positives out of all actual positive instances, it measures how well the classifier identifies positive instances, where high recall indicates a low rate of false negative errors, recall is often combined with precision for more precise measurement.

$$Recall = \frac{TP}{TP+FN} \quad (3.3)$$

F1 Score: this metric is called the harmonic mean of both precision and recall, this combines precision and recall into a one single metric that evaluates the trade-off between precision and recall, it can be used to compare classifiers. The value of F1 score is high if both precision and recall are high while if precision or recall has low value then F1 score will be low.

$$F1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (3.4)$$

3.3.2 Cosine similarity

Cosine similarity is a metric that measures how similar two vectors are by calculating the cosine of the angle between these vectors. In NLP cosine similarity is used to measure the similarity between words that are represented as vectors. This metric is widely used in NLP compared to other metrics. [11][32]

The metric is based on the dot product in linear algebra, the definition of dot product of two vectors **a**, **b** is given by the formula [31]:

$$\mathbf{a} \cdot \mathbf{b} = ||\mathbf{a}|| \ ||\mathbf{b}|| \cos \theta \quad (3.5)$$

Thus, the normalized form of dot product in terms of cosine θ is:

$$\cos \theta = \frac{\mathbf{a} \cdot \mathbf{b}}{||\mathbf{a}|| \ ||\mathbf{b}||} \quad (3.6)$$

If vectors **a** and **b** are pointing in the same direction and the angle between them is close to 0, then the cosine of this angle is close to 1 which indicates high similarity. If the vectors are orthogonal meaning the angle is 90 degrees which gives cosine 90 equal to 0 this indicates that these

vectors are unrelated. If these vectors are pointing in the opposite direction and the angle between them is close to 180 degrees, then the cosine of this angle is close to -1 which indicates high dissimilarity. [31]

The limitation of cosine similarity arises if the embeddings do not represent the text well. For example, in sparse embeddings the vectors may have many zeros which produces small value in dot product which in turns give low similarity score. This limitation could be reduced if contextual or static embedding is being used or the embeddings is representative for the text type.

3.4 Project evaluation method

The success of this study will be evaluated based on the achievement of the objectives and the degree to which the research questions in chapter 1.3 were answered. In the discussion chapter, each objective will be revisited, and the corresponding results will be critically analyzed. Furthermore, the strengths and weakness of the methods used in this study will also be evaluated. Any challenges encountered during the study will be noted, as they could offer valuable insights for enhancing future research efforts. Possible improvements and future directions will also be suggested. Lastly, the study's success will be assessed by its contribution to understanding of AI-generated texts and their detection.

4 Implementation

The diagram represented in Figure 4-1 describes the overall process from generating dataset to evaluating ML classifiers and comparing the textual characteristics.

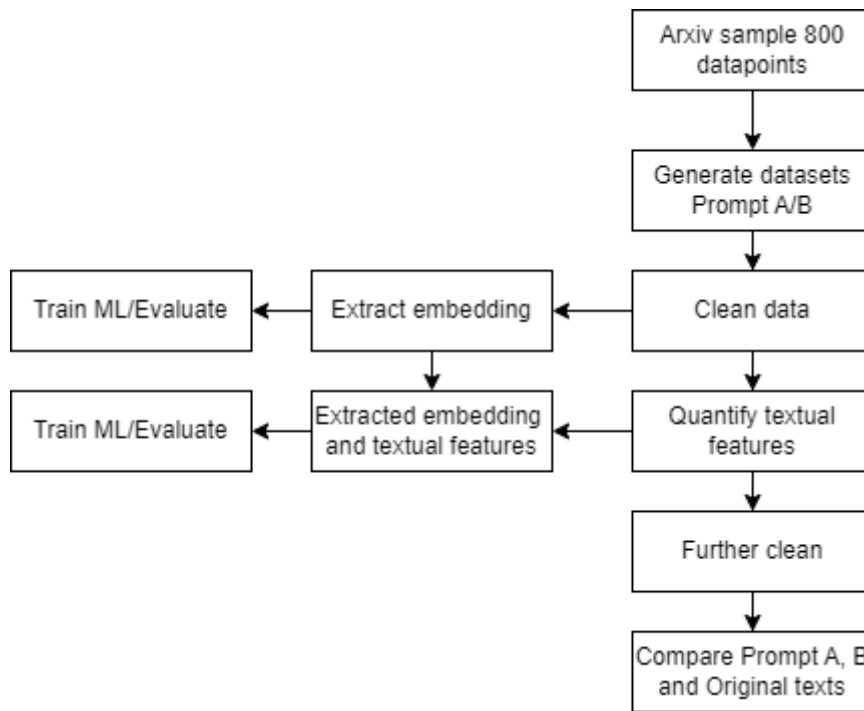


Figure 4-1: Overview of the process from generating the datasets to evaluating and comparing textual features.

4.1 Data creation

The ArXiv dataset was filtered and prepared for sampling, where irrelevant attributes are filtered out, the relevant attributes are shown in Figure 4.1-1, the id attribute is unique for each paper in the dataset. The update date attribute is the publish date of the paper.

```
42  
43     return {  
44         'id': paper['id'],  
45         'title': paper['title'],  
46         'category': paper['categories'].split(' '),  
47         'abstract': paper['abstract'],  
48         'update_date': paper['update_date'],  
49     }  
50
```

Figure 4.1-1: Relevant attribute from ArXiv dataset.

After filtering, 800 datapoints were randomly chosen from the computer science category.

The first dataset was generated using this prompt ‘Write a scientific abstract between 200 to 300 words on the following topic: {title}’ see Figure 4.1-2.

```
257
258 def generate_text_completion(
259     title, max_tokens=4000, model='text-davinci-003', retries=3, initial_delay=1, backoff_factor=2):
260     #Text completion
261     prompt = f"Write a scientific abstract between 200 to 300 words on the following topic: {title}"
262     delay = initial_delay
263
264     for attempt in range(retries):
265         try:
266             response = openai.Completion.create(
267                 engine=model,
268                 prompt=prompt,
269                 max_tokens=max_tokens,
270                 n=1,
271                 stop=None,
272                 temperature=0.7,
273             )
274             return response["choices"][0]["text"].strip(), None
275         except Exception as e:
```

Figure 4.1-2: This figure shows the parameters used when generating dataset Prompt A.

The second dataset was generated using this prompt is ‘Write a coherent, detailed and well-structured scientific abstract of {text length} words on the following topic: “{title}”. The abstract should be written in rich, clear, and academic language.’, see Figure 4.1-3.

```
228 def generate_text_ChatCompletion(
229     title, text_length, max_tokens=4000, model='gpt-3.5-turbo',
230     retries=3, initial_delay=20, backoff_factor=2):
231     # ChatCompletion
232     prompt_user = f"Write a coherent, detailed and well-structured scientific abstract of {text_length} words on the following topic: “{title}”. The abstract should be written in rich, clear, and academic language."
233     delay = initial_delay
234
235     for attempt in range(retries):
236         try:
237             response = openai.ChatCompletion.create(
238                 model=model,
239                 messages=[
240                     {"role": "user", "content": prompt_user},
241                 ],
242                 max_tokens=max_tokens,
243                 n=1,
244                 stop=None,
245                 temperature=0.7,
246             )
247             return response["choices"][0]["message"]["content"].strip(), None
```

Figure 4.1-3: This figure shows the code snippet for generating dataset Prompt B.

4.2 Data cleaning

GPT-3.5 models did not pay much attention to the provided length for the abstracts, the generated abstracts have different lengths some are near the length of the original abstracts, and some are much longer or shorter. Figure 4.2-1 shows the distribution of dataset Prompt A on the left and the distribution for dataset Prompt B on the right.

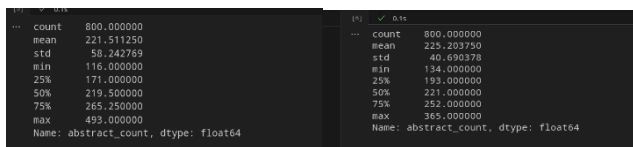


Figure 4.2-1: On left the distribution for abstract length of dataset Prompt A, and on the right for Prompt B.

All generated texts in dataset Prompt A that are shorter than 170 words or longer than 350 words are removed from the dataset with their corresponding original abstracts. This yields around 80% of the data have text length between 200 and 300 words. The ids of the removed texts are used to drop corresponding rows in dataset Prompt B; thus, A and B contain the same original abstracts. The datasets are converted to lowercase and leading and trailing spaces are removed. Each of the final datasets contains 600 data points generated texts and 600 data points original texts. This gives balanced labels; these datasets are used for training ML classifiers.

For comparing the statistical distribution further cleaning has been done, where all abstracts that are longer than 300 words or shorter than 200 words are dropped. Figure 4.2-2 shows the distribution of abstract length, this dataset is used to compare the distributions.

	count	mean	std	min	25%	50%	75%	max
written_by								
Original	253.0	239.241107	23.096785	200.0	222.0	237.0	255.0	300.0
Prompt A	253.0	250.695652	26.974236	202.0	231.0	248.0	274.0	299.0
Prompt B	253.0	240.660079	25.999601	200.0	219.0	238.0	259.0	298.0

Figure 4.1-2: The distribution of abstract length in the dataset which are used to compare the distributions of text characteristics.

4.3 Quantifying key characteristics in data

At this step syntax, readability, sentiment, and similarity features are added to the datasets, see Figure 4.3-1.

```
df_a.columns|
Index(['id', 'title', 'abstract', 'category', 'title_count', 'abstract_count',
      'year', 'label', 'written_by', 'unique_words', 'sentence_count',
      'avg_sentence_length', 'lexical_diversity_ratio', 'readability_score',
      'passive_count', 'noun_freq', 'verb_freq', 'adjective_freq',
      'adverb_freq', 'sentiment_polarity', 'sentiment_subjectivity',
      'passive_ratio', 'title_relevance', 'similarity_org_gen_score'],
      dtype='object')
```

Figure 4.2-1: key characteristics are quantified.

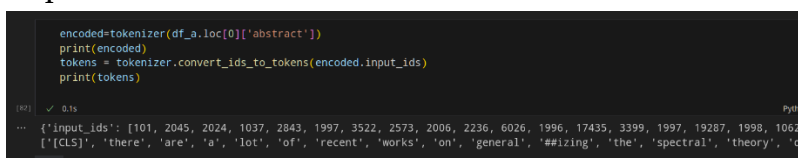
The syntactic features (pos) are divided into 4 categories Noun, Verb, Adjective and Adverb. Each word in the texts that belong to one of these categories has been considered.

For N-gram lists, the following procedure has been taken:

- Stop word, which is words that do not give important information e.g., “a”, “an” has been removed.
- The n-gram has been calculated for the whole dataset.
- The frequency of n-gram is calculated.
- The frequencies are sorted, and indices of top k n-gram are sliced, where these indices are used to get the corresponding n-gram sequence for each label in the dataset.

4.4 Preprocessing and training ML classifiers

First the abstracts are tokenized using DistillBERT tokenizer, see Figure 4.4-1, the tokenizer maps each token to integer called input_ids this mapping is used to convert tokens back to strings. Because abstracts vary in length, each abstract is truncated or padded with zeros so that the total length of tokens list is 512. The first token [CLS] indicates the start of the sequence and used for classification see section 2.2.3.



```

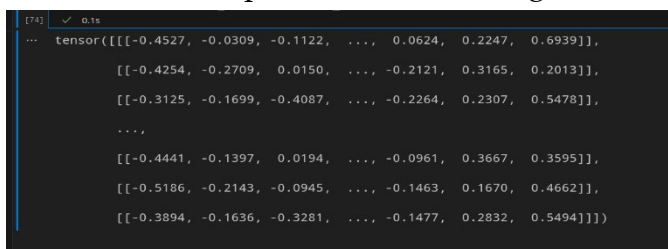
encoded=tokenizer(df_a.loc[0]['abstract'])
print(encoded)
tokens = tokenizer.convert_ids_to_tokens(encoded.input_ids)
print(tokens)

[101, 2045, 2024, 1037, 2843, 1997, 3522, 2573, 2006, 2236, 6026, 1996, 17435, 3399, 1997, 19287, 1998, 10629]
['[CLS]', 'there', 'are', 'a', 'lot', 'of', 'recent', 'works', 'on', 'general', '#izing', 'the', 'spectral', 'theory', 'of']

```

Figure 4.3-1: Example of how texts are tokenized.

The tokenized texts (input_ids) are fed into DistillBERT which output a 768-dimensional embedding for each token, only the embedding corresponding to [CLS] are extracted to be used as features to train ML classifiers, these embedding represents the whole text, see Figure 4.4-2 that shows example of the embeddings for dataset Prompt A.



```

... tensor([[[[-0.4527, -0.0309, -0.1122, ..., 0.0624, 0.2247, 0.6939]],
             [[-0.4254, -0.2709, 0.0150, ..., -0.2121, 0.3165, 0.2013]],
             [[-0.3125, -0.1699, -0.4087, ..., -0.2264, 0.2307, 0.5478]],
             ...,
             [[-0.4441, -0.1397, 0.0194, ..., -0.0961, 0.3667, 0.3595]],
             [[-0.5186, -0.2143, -0.0945, ..., -0.1463, 0.1670, 0.4662]],
             [[-0.3894, -0.1636, -0.3281, ..., -0.1477, 0.2832, 0.5494]]]])

```

Figure 4.4-2: Embeddings for dataset Prompt A which will be used as features to train ML classifiers, same procedure is taken for dataset Prompt B.

After extracting the embedding for both Prompt A and B, a hyper parameters search has been done using Randomized Search. The parameter's space used in this method are shown in Table 1. This parameter search has been performed on dataset Prompt A only and the best-found parameters are used to train on dataset Prompt B. The reason to search for parameters for only one dataset is that if the datasets for Prompt A and Prompt B are similar or related in some way, the same set of hyperparameters might perform reasonably well across all of them. This can be particularly true if the datasets are different subsets or different views of the same underlying data. This provides indication of changes in dataset Prompt B if the evaluation metrics gives poor results. Furthermore, this procedure avoids overfitting both datasets.

Table 1: Hyper parameter search space for ML classifiers.

Classifier	Parameter space
Logistic Regression	'C': np.linspace(0.1, 4, 20), 'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'], 'penalty': ['l1', 'l2']
Random Forest	'n_estimators': [100, 200, 300, 400, 500], 'max_depth': [10, 20, 30, 40, None], 'max_features': ['sqrt', 'log2', None], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4]
SVM	"C": np.logspace(-3, 3, 7), "kernel": ["linear", "poly", "rbf", "sigmoid"], "gamma": ["scale", "auto"] + list(np.logspace(-3, 3, 7))

After getting best parameters the classifiers have been evaluated using cross validation with 5 folds. The same best parameters have been used for dataset Prompt B.

The classifiers have been evaluated in two phases one with only embeddings extracted from DistilBERT and the second phase with

embeddings and characteristic features (only numerical features) see Figure 4.4-3, where the latter have been scaled using MinMaxScalar with range [-1,1] which corresponds to the range in the embeddings. The scaling was performed on only training sets not on test sets.

```
le
['abstract_count', 'unique_words', 'sentence_count', 'avg_sentence_length',
'lexical_diversity_ratio', 'readability_score', 'passive_count', 'noun_freq', 'verb_freq',
'adjective_freq', 'adverb_freq', 'sentiment_polarity', 'sentiment_subjectivity']
```

Figure 4.4-3: the selected columns that have been added as features in training ml classifiers.

5 Results

This chapter presents the results of all the measurements performed. The results for textual characteristics are presented in the first section in the form of plots and summaries of statistical measurements. The second section presents the evaluation metrics for both the machine learning classifiers and the GPT-2 output detector, including confusion matrices plots.

5.1 Text characteristics

This section introduces statistical measurements and distributions of texts characteristics. See Appendix A that provides more plots about the distributions and n-grams.

5.1.1 Lengths and sentence structure

Figure 5.1-1 shows a scatter plot of the number of unique words on y-axis and abstract length on x-axis, abstract length ranges between 200 to 300 words, see implementation chapter for more details.

- Prompt A texts have a median value of 106 unique words, with the range spanning from a minimum of 70 to a maximum of 146 unique words, the standard deviation (std) is 13.74, with a mean of 105 unique words.
- Prompt B texts has a median of 119 unique words, it ranges from a minimum of 82 unique words to a maximum of 170, the std is 12.80, with a mean of 120 unique words.
- The original texts have a median of 139 unique words, the data spans from a minimum of 103 unique words to a maximum of 182, with a std of 15.37 and a mean of 141 unique words.

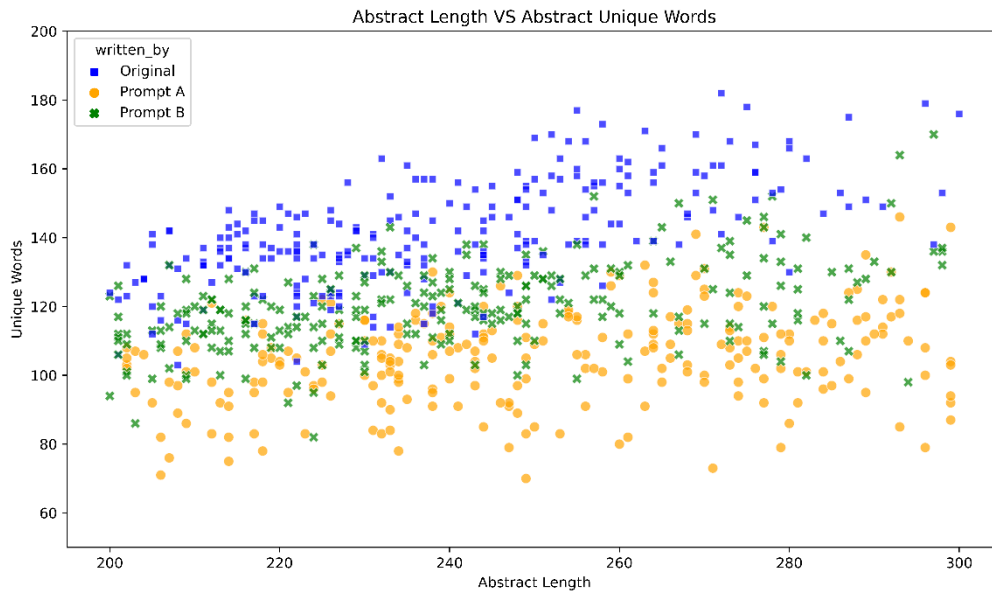


Figure 5.1-1: Scatter plot shows the number of unique words in y-axis and abstract word count in x-axis for the generated texts both prompts and original texts.

Figure 5.1-2 shows boxen plot of sentence structure: sentence count and average sentence length per abstract.

Sentence count:

- Prompt A texts have a median value of 13 sentences, with a range from a minimum of 8 to a maximum of 19 sentences, The std is 2.11, with a mean of 12.85 sentences.
- Prompt B texts have a median of 12 sentences, the sentence count spans from a minimum of 8 to a maximum of 19. The std is 1.82, with an average of 11.67 sentences.
- The original texts have a median of 11 sentences, the sentence count spans from a minimum of 6 sentences to a maximum of 19 sentences, with a std of 2.06 and a mean of 10.78 sentences.

Average sentence length:

- Prompt A texts have a median value of 19.8 words per sentence, the sentence length spans from a minimum of 10.8 to a maximum of 27.3 words per sentence, the std is 2.64, with a mean of 19.83 words per sentence.
- Prompt B texts have a median value of 20.8 words per sentence, it ranges from a minimum of 13.3 to a maximum of 31.4 words per sentence, the std is 2.72, with a mean of 20.93 words per sentence.

- The original texts have a median value of 22.2 words per sentence, the sentence length spans from a minimum of 13.9 to a maximum of 40.6 words per sentence, with a std of 4.21 and a mean of 22.82 words per sentence.

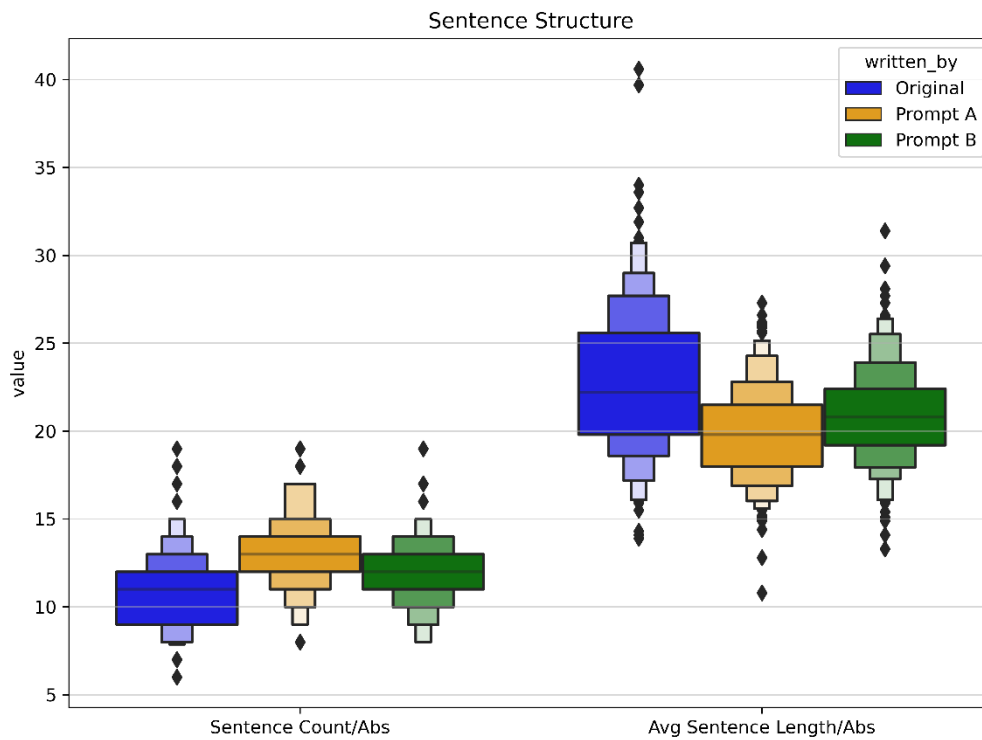


Figure 5.1-2: Boxen plot for sentence count and average sentence length.

5.1.2 N-gram lists

Figures 5.1-3 shows bar charts of top 20 2-gram frequencies in Prompt A and in original texts.

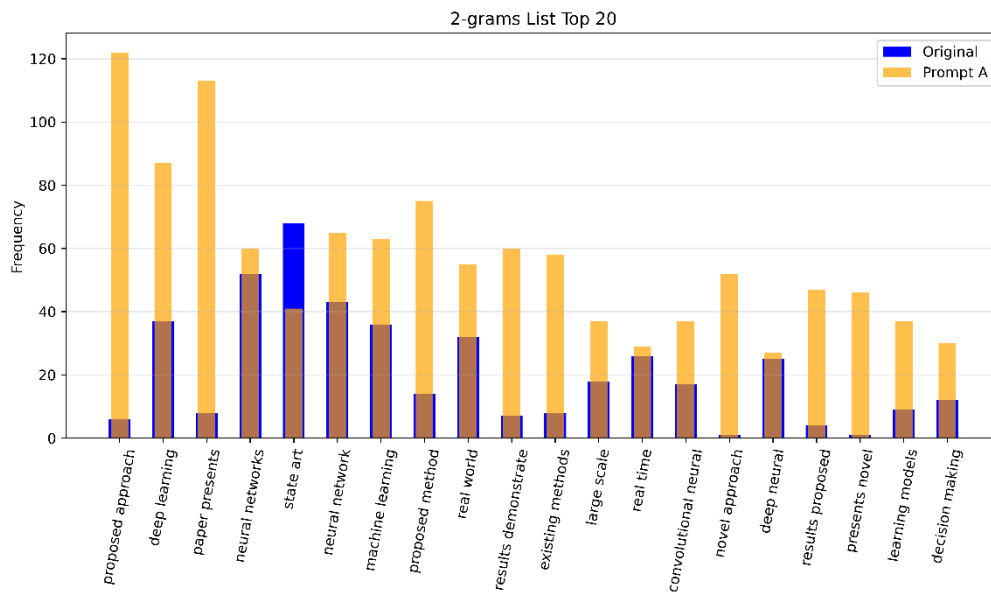


Figure 5.1-3: List of top 20 frequent 2-grams in Prompt A and the corresponding frequencies in original data.

Figures 5.1-4 shows bar charts of top 20 2-gram frequencies in Prompt B and in original texts.

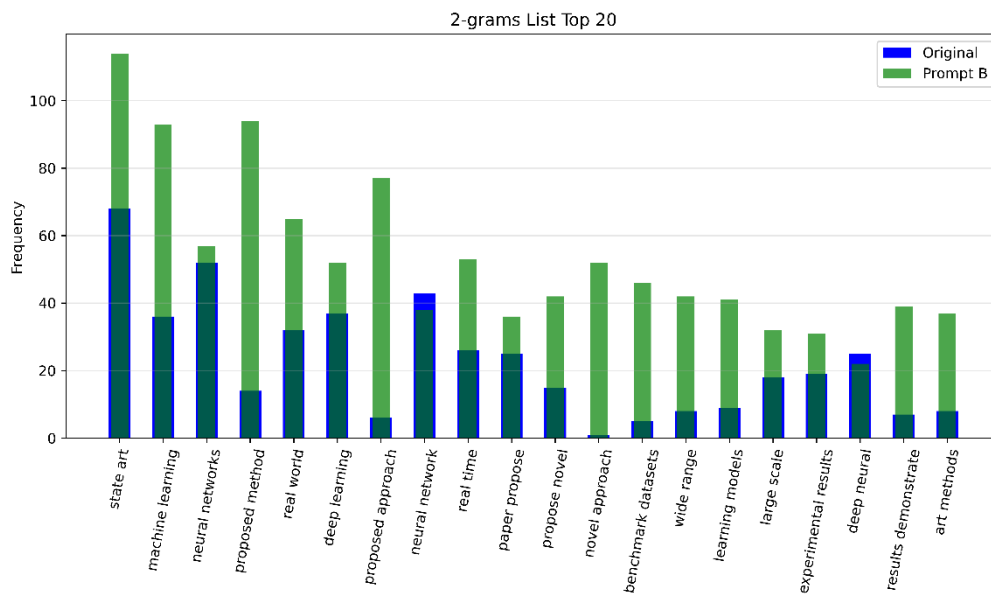


Figure 5.1-4: List of top 20 2-grams frequencies in Prompt B and the corresponding frequencies in original data.

Figures 5.1-5 show the top 20-word frequencies for generated data Prompt A and B.

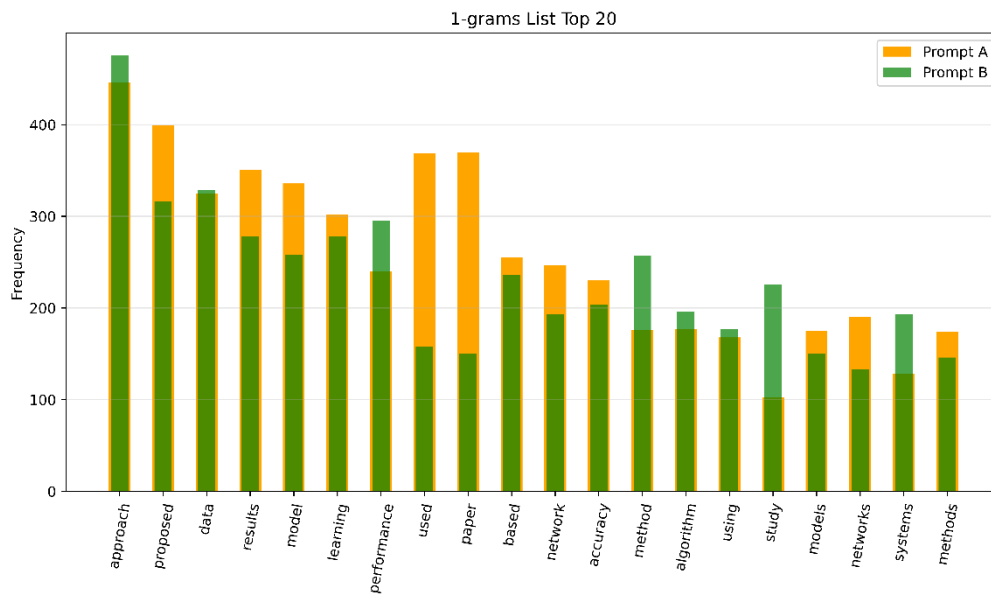


Figure 5.1-5: Top 20-word list frequencies in Prompt A and B.

Figure 5.1-6 shows top 20-word list frequencies of original texts.

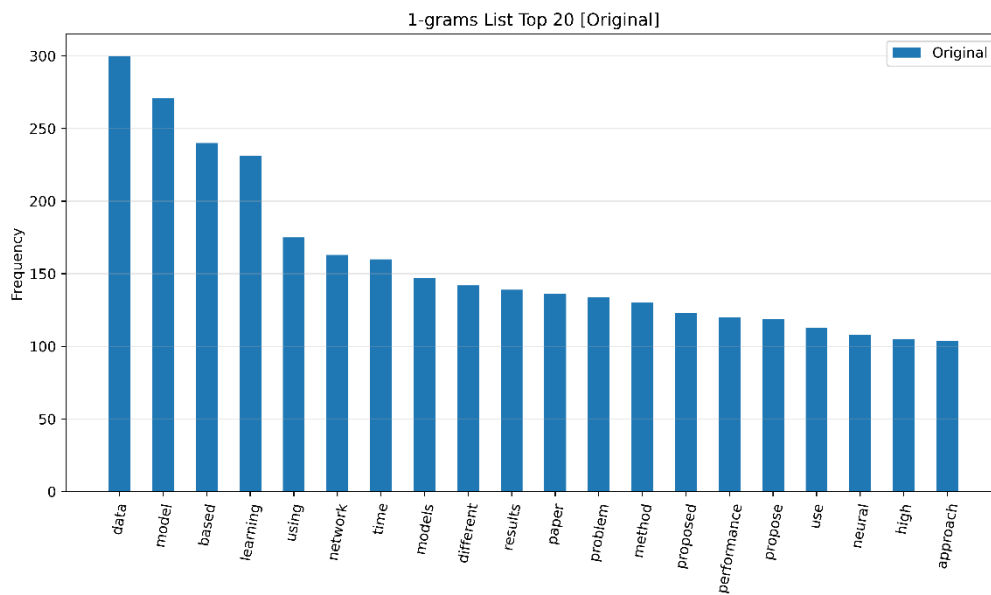


Figure 5.1-6: Top 20-word list frequencies in original texts.

5.1.3 Sentiment

Figure 5.1-7 shows boxen plot for sentiment metrics:

Sentiment polarity:

- Prompt A texts have a median value of 0.14, spanning from a minimum of -0.19 to a maximum of 0.57, the std is 0.10, with a mean of 0.15.
- Prompt B texts have a median value of 0.11, ranging from a minimum of -0.15 to a maximum of 0.34, with a std of 0.08 and a mean of 0.11.
- The original texts have a median value of 0.09 in sentiment polarity, ranging from a minimum of -0.16 to a maximum of 0.27, with a std of 0.08 and a mean of 0.09.

Sentiment subjectivity:

- Prompt A texts show a median value of 0.49, with a range from a minimum of 0.13 to a maximum of 0.88, the std is 0.10, with a mean of 0.48.
- Prompt B texts have a median value of 0.47, ranging from a minimum of 0.19 to a maximum of 0.70, with a std of 0.09 and a mean of 0.47.
- The original texts present a median value of 0.44 for sentiment subjectivity, spanning from a minimum of 0.22 to a maximum of 0.70, with a std of 0.08 and a mean of 0.45.

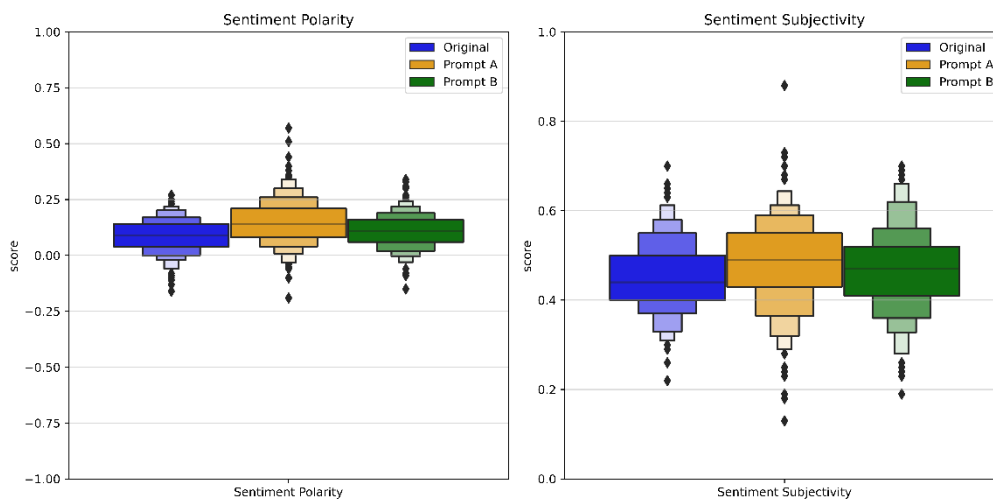


Figure 5.1-7: On the left is sentiment polarity, on the right is sentiment subjectivity for all data.

5.1.4 Readability score

Figure 5.1-8 shows histogram for readability scores where the left plot is for Prompt A and the right plot for Prompt B:

- Prompt A texts show a median readability value of 34.97, spanning from a minimum of -7.55 to a maximum of 68.97, the std is approximately 11, with a mean of 35.33.
- Prompt B texts have a median readability of 31.31, with a range from a minimum of 0.31 to a maximum of 64.61, the std is approximately 9.79, with a mean of 30.37.
- The original texts have a median readability of 30.30, extending from a minimum of 0.45 to a maximum of 60.89, with a std of 11.93 and a mean of 30.36.

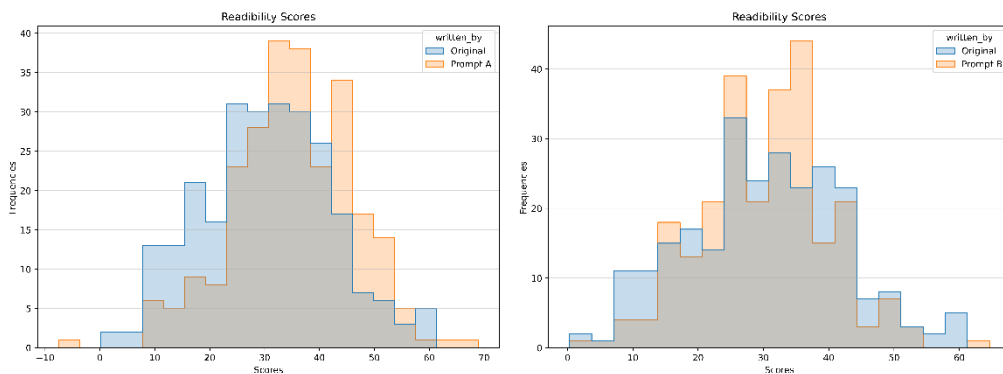


Figure 5.1-8: Histogram for readability scores. Left histogram is for Prompt A and original, while the right histogram is for Prompt B and original.

5.1.5 Syntactic features

Figure 5.1-9 shows boxen plot for the syntactic features and passive voice counts found in the texts.

Nouns: Prompt A texts have a median of 32%, with a std of 4% and a mean of 32%. Prompt B texts have a median of 34%, with a std of 3% and a mean of %34. The original texts have a median of 33%, with a std of 3% and a mean of 32%.

Verbs: Prompt A texts have a median of 17%, with a std of 3% and a mean frequency of 16%. Prompt B texts have a median of 15% for verbs, with a

std of 2% and a mean of 15%. The original texts have a median of 15% for verbs, with a std of 2% and a mean of 15%.

Adjectives: Prompt A texts have a median of 12% with std is 3.13% and a mean of 12.12%. Prompt B texts have a median of 12%, with a std of 2.83% and a mean of 12.44%. The original texts have a median of 14%, with a std of 3.03% and a mean of 13.94%.

Adverbs: Prompt A texts show a median of 2%, the std is 1.39%, with a mean of 2.57%. Prompt B texts have a median of 2% for adverbs, with a std of 1.13% and a mean of 2.39%. The original texts show a median of 3%, with a std of 1.58% and a mean of 3.54%.

Passive count: Prompt A texts have a median value of 4 passive sentences, the std is 2.60, with a mean of 4.16. Prompt B texts show a median value of 2, the std is 1.99, with a mean of 2.49. The original texts have a median value of 2, with a std of 2.14, the mean stands at 2.93.

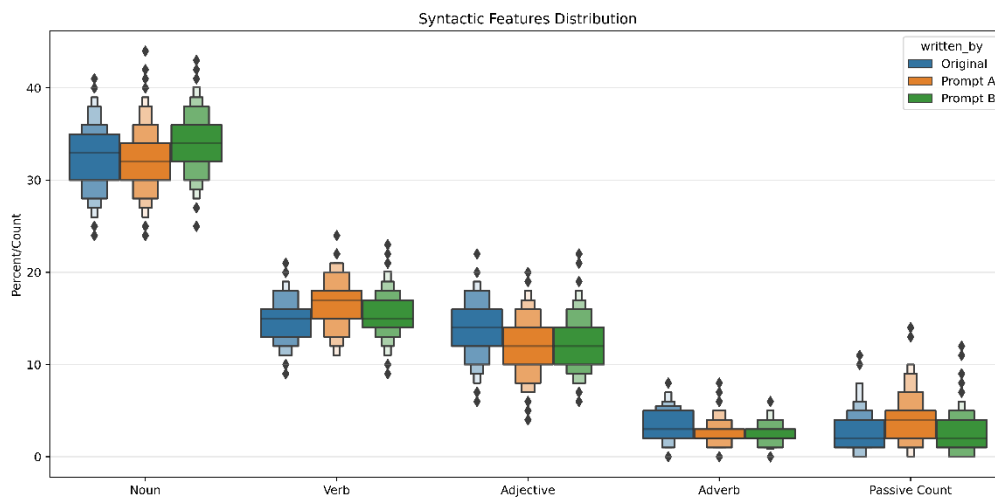


Figure 5.1-9: Syntax features and passive count distribution. The median for adverbs is not shown because it lays on the upper edge of the box, however the distribution of adverbs is listed in Appendix.

5.1.6 Similarity

Figure 5.1-10 shows boxen plot for the similarity between original-generated pairs.

Prompt A/original pairs have a median similarity of 0.71, with a range from a minimum of 0.16 to a maximum of 0.90, the std is approximately 0.10, with a mean similarity of 0.70.

Prompt B/original pairs have a median similarity of 0.74, ranging from a minimum of 0.34 to a maximum of 0.89, the std is approximately 0.09, with a mean similarity of 0.72.

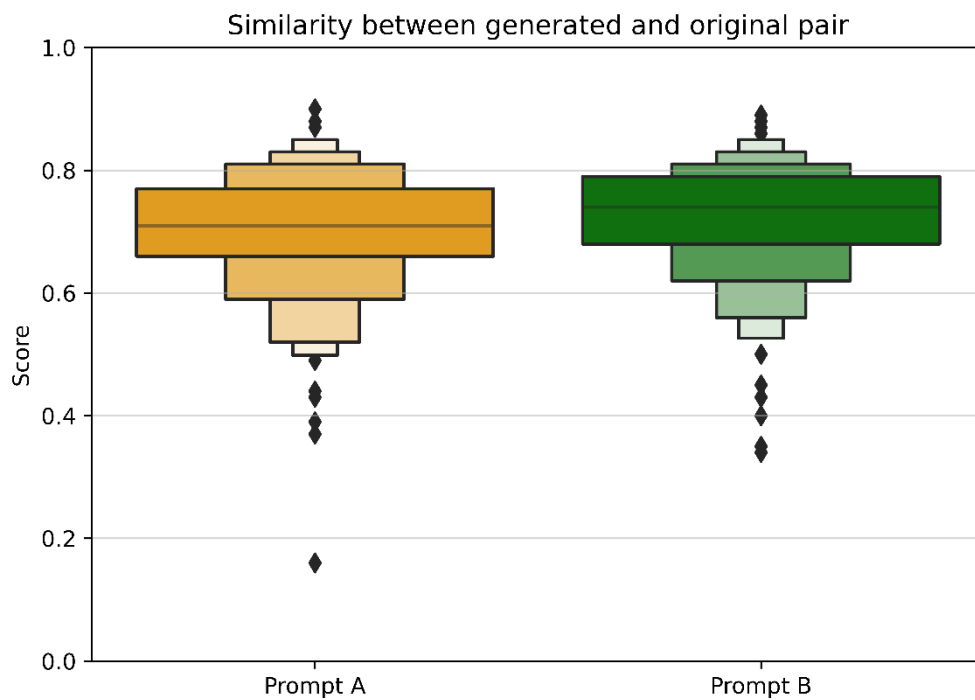


Figure 5.1-10: Similarity scores between original and generated texts.

Figure 5.1-11 shows boxen plot for similarity scores between title and abstract. Prompt A texts have a median value of 0.55, the std is 0.11, with a mean of 0.55. Prompt B texts have a median of 0.54, the std is approximately 0.09, with a mean of 0.54. The original texts have a median title relevance of 0.46, with a std of 0.09 and a mean of 0.45.

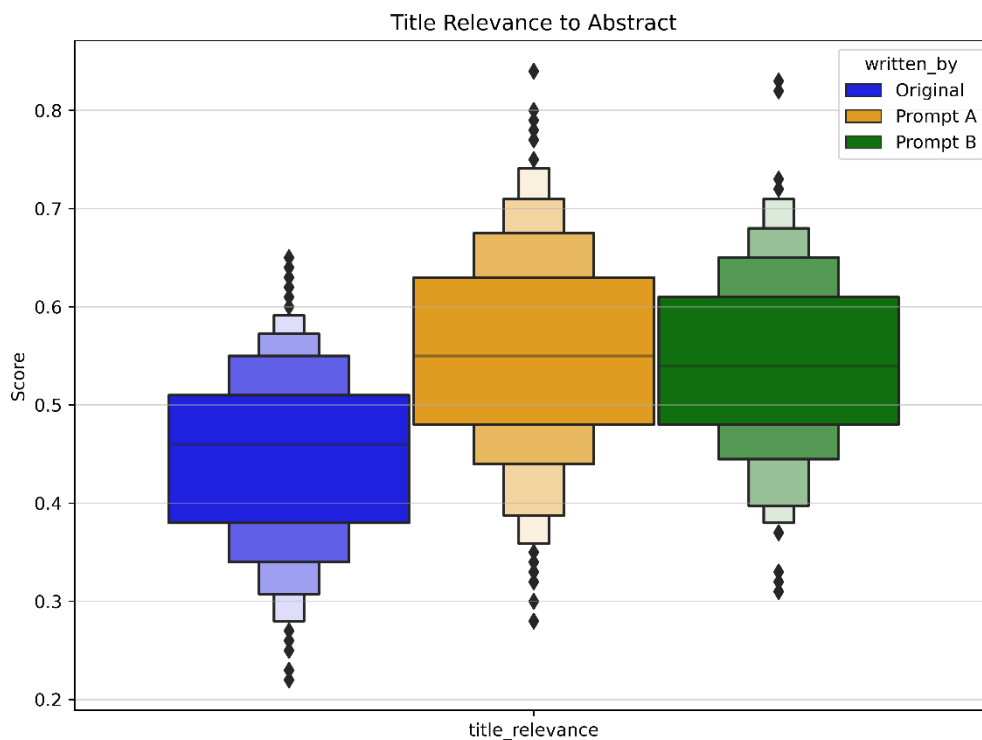


Figure 5.1-11: Scores for the title relevance to the text.

5.2 Measurement results

This section presents the results obtained from ML evaluation metrics.

Table 2 shows the best parameters found for ML classifiers.

Table 2: Best parameters found for machine learning classifiers.

Model name	Best parameters
SVC	C=100, kernel='sigmoid'
Logistic Regression	max_iter=3000,solver='liblinear', C=3.1578947368421044
Random Forest	n_estimators=400,min_samples_leaf=4, min_samples_split=10,max_features='sqrt', max_depth=10

Figure 5.2-1 shows the performance of GPT-2 detector on predicting the generated texts from Prompt A and B.

	accuracy	Precision	Recall	F1 Score
OpenAI Detector Prompt A	0.856	0.995	0.715	0.832
OpenAI Detector prompt B	0.736	0.993	0.475	0.643

Figure 5.2-1: GPT-output detector evaluation results.

Figure 5.2-2 shows confusion matrix for GPT-2 output detector.

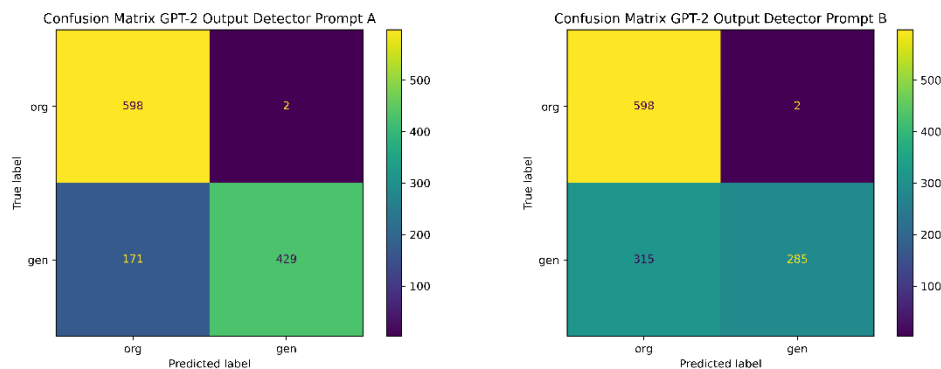


Figure 5.2-2: Confusion matrix for GPT-output detector.

Figure 5.2-3 shows the results of training ML classifiers with only embeddings produced by DistillBERT.

Classifier Scores With Only Embeddings Features [Pompt A]

	Accuracy	Precision	Recall	F1 Score
SVC	0.99	0.992	0.988	0.99
LogisticRegression	0.989	0.99	0.988	0.989
RandomForestClassifier	0.943	0.956	0.93	0.942
DummyClassifier	0.516	0.516	0.52	0.518

Classifier Scores With Only Embeddings Features [Pompt B]

	Accuracy	Precision	Recall	F1 Score
LogisticRegression	0.987	0.988	0.985	0.987
SVC	0.984	0.987	0.982	0.984
RandomForestClassifier	0.933	0.923	0.947	0.934

Figure 5.2-3: Performance of ML classifiers trained on embeddings.

Figure 5.2-4 shows confusion matrix for SVM and logistic regression trained on embeddings.

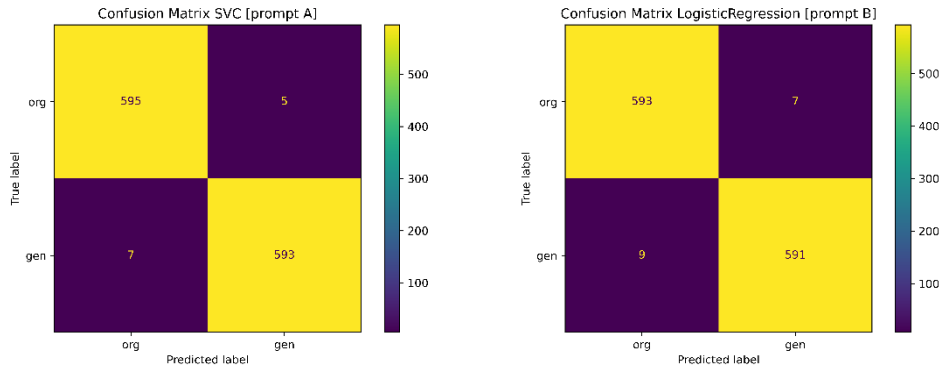


Figure 5.2-4: Confusion matrix for best classifiers trained on embeddings only.

Figure 5.2-5: Performance of ML classifiers trained on both embeddings and the features obtained from text analysis.

Classifier Scores With All Features [Prompt A]				
	Accuracy	Precision	Recall	F1 Score
LogisticRegression	0.991	0.992	0.99	0.991
RandomForestClassifier	0.973	0.988	0.958	0.973
SVC	0.957	0.958	0.955	0.957
DummyClassifier	0.516	0.516	0.52	0.518

Classifier Scores With All Features [Prompt B]				
	Accuracy	Precision	Recall	F1 Score
LogisticRegression	0.986	0.987	0.985	0.986
RandomForestClassifier	0.958	0.95	0.968	0.959
SVC	0.932	0.936	0.928	0.932

Figure 5.2-5: ML classifiers trained on both embeddings and text features.

Figure 5.2-6 shows the results of training Logistic regression on one dataset and predicting the generated text from the other dataset.

Fit A/B Predict A/B				
	accuracy	Precision	Recall	F1 Score
Fit whole A, Predict generated from B	0.967	1.0	0.967	0.983
Fit whole B, Predict generated from A	0.985	1.0	0.985	0.992

Figure 5.2-6: Performance of logistic classifier fitted on one set to predict the generated text in the second set.

6 Discussion

This chapter delves into a detailed discussion and thorough analysis of the results obtained, interpreting their significance, which will be used to answer the research question later. The discussion includes interpretation of the textual characteristics, discussing the behavior of machine learning classifiers and GPT-2 output detector; discussing the work method; the knowledge gained from this work; and finally the ethical aspects and the risks that may occur when relying on detection tools.

6.1 Analysis and discussion of results

The results indicate some kind of pattern in the generated texts, where the texts show tighter distributions compared to original texts. The generated texts demonstrate more uniformity in the writing style despite the detailed instructions in Prompt B.

The number of unique words in the abstracts generated using Prompt A is less diverse compared to those from Prompt B. One explanation for why set B is richer, could be the inclusion of words such as “detailed”, and “rich” in Prompt B. In contrast, original abstracts are dense and concentrated, which can be related to the fact that abstracts typically summarize an entire document. Each part of the document may need to be represented in the abstract so repeated or non-functional words would waste space in the abstract section without providing useful information. The number of unique words may increase with a higher temperature setting or by providing additional information in the prompt. Generally, texts produced by prompt B are closer to original abstracts in terms of vocabulary richness than prompt A.

The distribution of average sentence length for generated texts suggests a standardized approach to the models' writing style. The median and mean between Prompt A and Prompt B differ with one word and the std is almost equal. This contrasts with original abstracts which display a wider spread in their distributions, possibly due to researchers' individual writing styles with not all adhering to a standard sentence length. The texts generated by Prompt A have more sentences as indicated by a higher median sentence count compared to Prompt B, this is due to the abstracts' length on average is longer for Prompt A. Further,

both Prompt A and B have similar distributions where Prompt B is slightly closer to the original text's distribution than texts in Prompt A.

The readability score distribution shows a large overlap between generated and original texts, where most texts have scores between 25 and 45. According to the formula these scores are interpreted as 'difficult' to 'very difficult'. The readability scores also show negative values in Prompt A, this is due to the limitations of the formula discussed methodology section, but these negative scores indicate that the average number of syllables per word is high in those texts, which indicates the use of longer and more complex words. The distribution of Prompt B aligns more closely with that of the original text than does Prompt A as indicated by the median and mean.

The median of sentiment polarity found for Prompt A and B is higher than in the original texts, indicating that GPT-3.5 models generally express more positive words in the texts. From the distribution it is seen that Prompt B and original texts are closer to each other more than Prompt A. This may be related to the inclusion of keyword "academic language" in prompt B which changed the tone to be more neutral in texts for Prompt B. In terms of sentiment subjectivity, the distribution indicates that the texts generated by Prompt A tend to be slightly more subjective on average than those generated by Prompt B or by original, although the differences are small between all texts.

The median of nouns of both original and Prompt B texts are higher than Prompt A texts. Moreover, Prompt B has 1% higher median than original texts and 2% higher than Prompt A which has 1% median lower than original texts. Thus, both Prompt A and B can consider close to original texts at same degree with respect to the median.

The distributions of verbs and passive voice in Prompt A are higher than those in Prompt B and original texts. This is related to the fact that constructing passive voice required auxiliary verbs such as "be, am, is, are, was, were, being, been". But also, may indicate that prompt A puts more weight on the methodology section more than other sections. However, texts in Prompt B show similar distribution to original texts in terms of passive count, that could be related to the inclusion of keyword "clear" in prompt B.

The use of adjectives and adverbs in generated texts is lower than in the original texts. This may indicate that the generated texts contain fewer details and provide more general views of the subject, as adjectives used for describing things and adverbs for answering questions [33] such “how, when, where, why, or to what extent—how often or how much.”, therefore we find that original contents have more adjectives and adverbs this because the focus is on what have been done in the work.

The 1-gram, 2-gram and 3-gram lists show a clear pattern in generated texts. They frequently use phrases such as “paper presents novel,” “proposed approach”, “proposed method” “presents novel” and “overall paper presents” these phrases with others are related to Introduction, Methodology, Results and Conclusion. This pattern along with others indicates a standard writing style used in the generated texts, which is less followed in original texts as indicated by less frequencies. This pattern makes clear separation between each section in abstract structure. Also, there are high frequencies of sequences such as “state art”, “machine learning”, this indicates that the sample selected from ArXiv data set contains many texts from artificial intelligence category. Also, there is overlaps between generated and original wordlists which may explain why similarity scores obtained are on average high. Generally, there is overuse of words in generated texts, where Prompt A has more repeated sequences than Prompt B, this due to the unique words is higher in Prompt B and the length of texts are on average longer in Prompt A.

The relevance of titles is higher in generated texts than in original texts, and this is due to two main reasons. First, the generated texts frequently repeat title words in each sentence, and they also tend to have a higher sentence count. This results in a larger number of sentences that are semantically like their titles. Second, if the original texts contain many sentences that are dissimilar or have low similarity scores with their titles, the overall score will be smaller. The high similarity scores between original and generated texts indicate that they share some characteristics such those discussed above.

The GPT-2 detector misclassified approximately 30% of the generated texts from prompt A and 53% from prompt B as original. These differences may be due to two possible reasons: first one, the model may not be trained on abstracts in scientific papers; second one, the detector

was trained on data generated by an older version of text completion. Newer versions of text completion are more akin to human-written texts making it harder for the detector to classify them correctly. The notable thing about the detector is that it has high precision around 99.5 % on both Prompt A and B, it only misclassified 2 original texts out of 600.

The high scores obtained from training ML classifiers on Prompt A and B using only embeddings looks overfitting. But after multiple evaluations the scores remained high. This consistent performance is related to the embeddings produced by DistilBERT. This model can capture complex patterns in the text and thus the embeddings already include the relational dependencies presented in the texts. Both logistic regression and SVM achieved the highest scores ~99% on all metrics, followed by the random forest model with ~95%. This outcome could be because the produced embeddings are linearly separable, for which logistic regression and SVM are well suited. Possible reason why random forest model get lower scores may be due the complexity of the model and the features selection mechanism which splits features in across multiple decision tress, and some of the pattern found in the embedding are split in these trees, which affected the performance. Precision and recall scores are a bit higher in Prompt A compared to Prompt B might be due to some generated texts in Prompt B being very similar to the original texts or having a high percentage of overlap. This similarity couldn't be adequately identified by the classifiers. However, training the ML on specific type of texts yields high performance.

In the second phase of training, when the ML models were trained on all features including the embeddings, logistic regression and the random forest model performed best. This could be because the additional features introduced some non-linearity to the data, which the random forest model could capture. The logistic regression was less affected by these additional features. The performance of SVM dropped dramatically in the second phase. This may be because these features add more noise to the data which is not suitable for SVM. One more possible explanation why the added features did not increase the performance could be that these features are already included in the embedding. The final best model based on the evaluation metrics that give good prediction on both Prompt A and B is logistic regression. Finally, fitting

the classifier on Prompt B to predict generated texts from Prompt A, gives 100% with 99% recall which is very good performance.

6.2 Work method discussion

The adopted research method for this study was a hybrid of quantitative, comparative, and experimental methodologies, ideal for making comparisons, measurements, and statistical analyses. The study's approach employed a blend of sequential and partially iterative strategies, providing a balance between a systematic progression and the flexibility to revisit and refine certain aspects as necessary. This methodology proved to be effective because it facilitated a thorough investigation of the research questions, allowing for the extraction of rich numerical data and comprehensive comparisons. The sequential aspect ensured a structured, step-by-step process, while the iterative component allowed for adjustments and improvements which were mostly encountered in data generation. Consequently, the chosen approach allowed for a robust and dynamic exploration of the research objectives.

All the project phases for this study were met successfully. During the data creation phase 2, the dataset was chosen to suit the problem of analyzing academic writing style found in abstracts. The samples taken weren't diverse, as many of the data points are from the Artificial Intelligence category. Two GPT-3.5 models were used in this study, the first one is "text-davinci-03" model which was employed to generate dataset Prompt A. This model was fast and easy to use but costly, it generated 800 datapoints in two hours. The second model is GPT-Turbo which was used to generate dataset Prompt B. While this model was much cheaper, it was extremely slow and prone to errors when sending requests to the server, this model took 10 hours to generate 800 datapoints. The reason why two models are used is that I started with the cheaper one, but because it was so slow and many errors occurred when connecting to OpenAI's API, I switched to the faster model.

The first prompt A used in this study was inspired by other relevant studies such the one in section 2.7. The second prompt B was particularly improved to get more realistic texts and to see how it impacts the generated content. However, there were many keywords included in

prompt B which made the interpretation a bit harder. One more suitable approach is to generate many datasets each with only one or two added keywords. This gives a better view of which keyword led to which response. However, this procedure was not taken to limit the scope of this study. However, as seen in the result both prompt A and B have similar distributions despite the detailed instructions, the differences were small between generated texts.

The length of the generated texts varied, and quite a large portion of the generated datasets were extremely short or long, which was unexpected. This issue could be resolved by generating n number of texts and then selecting the best one, or by setting a higher number for text length in the prompts. The chosen temperature setting (see section 3.2.2) for text generation should have been slightly higher, such as 1, which may result in more diversity in the generated outputs.

In the third phase, the data was cleaned in two stages due to the presence of extreme outliers in the datasets. The first stage involved selecting only data points from sets A and B that had lengths ranging from 200 to 300 words, which is a similar range to the original data. This was done to compare the statistical measurements and distributions, although these distributions may be slightly affected by the varied lengths of the texts. The second stage prepared the data for training. During this phase only the extreme outliers were removed from the sets, the varied length helps the ML to generalize better than taking all texts of the same length.

In the fourth phase, to quantify the data NLP libraries were utilized. These libraries are well known in the fields of NLP and ML. The approach used to measure readability was based on one readability metric which had limitations as described in the methodology chapter. Many similar metrics have been checked, but all exhibit similar issues.

To measure title relevance, a cosine similarity metric was used, and all scores were averaged. However, this approach does not yield accurate results, especially when sentence count varies. Also, the title may not always reflect the content of the abstracts. Therefore, this method gives possible interpretations. Even with this weakness the result showed that generated texts tend to use much of the title in each section of the abstract. The model used to measure semantic similarity and title relevance provided by Sentence Transformer library as mentioned in section 3.2.4

is suited for semantic search as there are no other alternatives. But with the model's accuracy in producing embeddings being 68.7%, the comparison may not be fully reliable. Also, the model can take a maximum number of words equal to 256 words, thus all texts that are longer than 256 words were cut off. The library continuously improves their models, now it has newer models that take longer texts and produces more accurate embeddings.

In the fifth phase ML classifiers were selected based on their capabilities. Both SVM and Logistic Regression can handle linearly separable data in higher dimensions efficiently, where logistic regression can handle noise better than SVM, but SVM works well for small datasets. Random Forest on other side can deal with both high-dimensional data, linear and non-linear data due to its ensemble learning strategy which builds multiple decision trees, and the results is based on the averages of their predictions, this makes Random Forest robust against the risk of overfitting.

The metrics used for ML evaluation were chosen because the problem at hand is to compare the quality and quantity, which these metrics serve this purpose, where recall was used to compare the quantity of correct predictions, precision for the quality of these predictions, F1 score for comparing the classifiers, and accuracy for the overall performance across both negative and positive classes. The approach used to represent the text was by utilizing DistilBERT embeddings. Other approaches were considered, but this approach was considered most suitable considering quality over performance. Other methods such as TF-IDF produce numerous features and do not capture all the hidden relationships within the text. This approach has been tested by OpenAI researcher and the accuracy was low, (see section 2.1, 2.4).

At sixth phase the approach used to evaluate the OpenAI's detector was to make a mapping between the output of the detector and binary label has been done, this way it becomes easier to compare the detector with ML classifiers using F1, recall, and precision score. There is another approach that is less suitable and hard to interpret such comparing the probability of the output detector and the ML outputs.

At the seventh phase the results were collected, analyzed, and presented. However, it could also be suitable if more plots that show the shape of the distributions of textual features were used.

6.3 Scientific discussion

The scientific knowledge obtained from this study in terms of technical aspects can be summarized in several points:

- 1- The GPT-turbo model is generally not appropriate when generating large data and the time is an important factor.
- 2- The embeddings produced by DistilBERT effectively represent the texts, as indicated by high metrics obtained from training ML classifiers.
- 3- Logistic Regression and SVM show better performance than Random Forest. On bigger dataset probably all ML will show similar performance.
- 4- The texts are linearly separable in higher dimensions, as indicated by the metrics of logistic regression and SVM.

These points cannot be generalized but specific for the problem and datasets in this study. To generalize, a bigger dataset with diverse topics is needed. However, Training machine learning classifiers on high-quality generated data leads to better results in detecting lower-quality generated data. These findings agree with the finding of OpenAI's researcher in approach 5 section 2.4.

In related work, 34% of the generated abstracts had a score under 0.5, when doing same mapping same mapping that been done in this study, it corresponds to 34% abstracts misclassified as original abstracts. Thus, the recall is 64%, the prompt used in related work is to some extent like the prompt A in this study. The obtained recall in this study is 71.5%, the difference seems not large compared to related study. But the comparison is unfair because the dataset size in this study was 12 times bigger than related work.

6.4 Ethical and societal discussion

This study aimed to show the differences and similarities between the generated abstracts and original abstracts, and the effect of the

instructions given to the GPT-3.5 models on the quality of generated abstracts, thus this study is not responsible for any misuse of the results. In addition, this study has focused on training machine learning tools on a specific text style which is academic writing style to detect generated texts of similar style.

These ML tools trained in this study can come with many risks, especially when deploying these tools and other similar tools in education or in academic areas. In the result obtained from this study around 7 original papers were misclassified as generated texts, this was around 1% false positive rate. This false positive could lead to serious issues, this becomes more serious when considering larger scale, for example assume the false positive rate is 1% and 100 million research papers are checked using these detection tools. This would result in one million papers being incorrectly classified as generated and thus rejected. Further, imagine if one of these rejected papers contained a new discovery that could benefit humanity, thus it would be a big loss for humanity. Therefore, when training machine learning models or language models, it is important to minimize the false positive rate and avoid considering the results as absolute answer.

Another issue can be arisen from these detection tools are the bias in the training data, for example assume that correct grammar and language diversity are two important features found in the training data and machine learning classifiers adds more weights for these features. Now, imagine if the selected sample taken from ArXiv dataset in this study consists mostly of texts written by native speakers who use correct grammar and diverse language in its writing, these tools may incorrectly classify texts written by non-native speakers that their texts may contain incorrect grammar or less diverse language as a generated text. Also, these tools can be misused, for example by generating texts and checking if these texts are detectable by these tools and based on the output the generated texts are modified to fool these tools.

7 Conclusions

This chapter summarizes the whole work by giving the answers to the research questions in section 1.3 and concludes the whole work. In addition to providing future work.

Do generated texts resemble the original texts in terms of textual features? Compared to original texts, generated texts usually have a less rich vocabulary, tend to be slightly less objective and neutral, and are often less descriptive and do not dive deeper into the subject. The generated texts generally follow a uniform writing style independent of the detailed instructions given to GPT-3.5 models, as opposite to original texts that show a wide range of varied styles. The answer to this question when looking at the whole picture of the textual characteristics, is yes to some extent, generated texts resemble the original texts.

Do the instructions given to GPT-3.5 model contribute to the quality of generated text? Yes, based on all textual characteristics extracted from the generated texts, the more detailed the instructions given to the GPT model, the closer the generated texts become to resembling the original texts in terms of the overall distributions of the textual characteristics. Additionally, the GPT-2 output detector recall dropped dramatically on the data that has more detailed instructions, which also a sign that the quality of the generated data increases with more instructions.

Do machine-learning classifiers trained in academic writing style give better detection than GPT-2 output detector? When comparing the F1 metric, the ML classifiers, specifically SVM and Logistic Regression, have a 99% score on generated. In contrast, the GPT-2 output detector has an F1 score of 83% on the first dataset and 64% on the second dataset. On the other hand, when comparing precision, the SVM and Logistic Regression have a score that's 0.2% lower than that of the GPT-2 output detector. The recall scores for the GPT-2 output detector are 71.5% and 47.5% on the first and second datasets, respectively, while the recall scores for SVM and Logistic Regression correspond to 99%. Overall, the answer is yes, ML classifiers provide better detection than the GPT-2 output detector.

In conclusion, the study's findings offer valuable insights into the nature of AI-generated academic texts and the methods of their detection. The

textual analysis revealed that texts generated by GPT-3.5 models do mirror original academic texts to some extent in terms of textual features. This finding points towards the fact that while GPT-3.5 can generate academic-sounding text, there is a difference in complexities in human-written text that it has yet to fully capture.

Furthermore, it was found that the instructions given to the GPT-3.5 model do influence the quality of generated text. Detailed instructions not only improved the resemblance of generated texts to the original ones but also challenged the effectiveness of the GPT-2 output detector. This implies the importance of the prompt instructions in leveraging the capabilities of GPT-3.5 models and adds an extra layer of complexity to the detection of such texts.

Lastly, it was noted that ChatGPT generates text in a standard way with no diversity in writing style, irrespective of the specific instructions provided.

The findings of this study add to the growing body of knowledge on AI-generated content and support the development of more efficient identification methods. Future studies could delve deeper into how different models of GPT generate text and further refine the detection techniques.

7.1 Future Work

7.1.1 Bigger dataset with more categories and hypered ML solution

One possible future work is to create a larger and more diverse dataset with many topics such as economics, physics, computer science etc. After this dataset has been created, try to explore, and compare many machine learning classifiers trained on all topics. Another approach is training each classifier on each topic, then select a combination of classifier that could make correct prediction. This approach makes each classifier specialized in specific topic or topics, after that combine these classifiers into hyper classifier, this would enable more generalized learning and prediction.

7.1.2 Fine-tuning DistillBERT or similar models

One more possible future work is to make bigger dataset as explained in 7.1.1 and fine-tune DistillBERT or other similar models. And compare the result either to ML classifiers in 7.1.1 or other language models.

References

- [1] L. Tunstall, L. von Werra, and T. Wolf, "*From Text to Tokens*" in Natural Language Processing with Transformers. Sebastopol, CA, USA: O'Reilly Media, Inc., 2022, pp. 29-34
- [2] D. Jurafsky and J. H. Martin, "*Vector Semantics and Embeddings*" in Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, 3rd ed., Draft. Stanford, USA: Stanford University, 2023, pp. 107-108, 113-116, 119-120. [Online]. Available: <https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf> [Accessed April 1, 2023].
- [3] OpenAI, "ChatGPT" OpenAI Blog. [Online]. Available: <https://openai.com/blog/chatgpt>. [Accessed March. 25, 2023]
- [4] OpenAI, "Models" OpenAI API Reference. [Online]. Available: <https://platform.openai.com/docs/models>. [Accessed Apr. 2, 2023].
- [5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "*BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*" arXiv:1810.04805, May 2019. [Online]. Available: <https://doi.org/10.48550/arXiv.1810.04805>
- [6] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "*DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*" arXiv:1910.01108, Mar. 2020. [Online]. Available: <https://doi.org/10.48550/arXiv.1910.01108>
- [7] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "*RoBERTa: A Robustly Optimized BERT Pretraining Approach*" arXiv:1907.11692, July 2019. [Online]. Available: <https://doi.org/10.48550/arXiv.1907.11692>
- [8] I. Solaiman, M. Brundage, J. Clark, A. Askill, A. Herbert-Voss, J. Wu, A. Radford, G. Krueger, J. W. Kim, S. Kreps, M. McCain, A. Newhouse, J. Blazakis, K. McGuffie, and J. Wang, "*Release Strategies and the Social Impacts of Language Models*" arXiv:1908.09203, Nov. 2019. pp. 12-15 [Online]. Available: <https://doi.org/10.48550/arXiv.1908.09203>

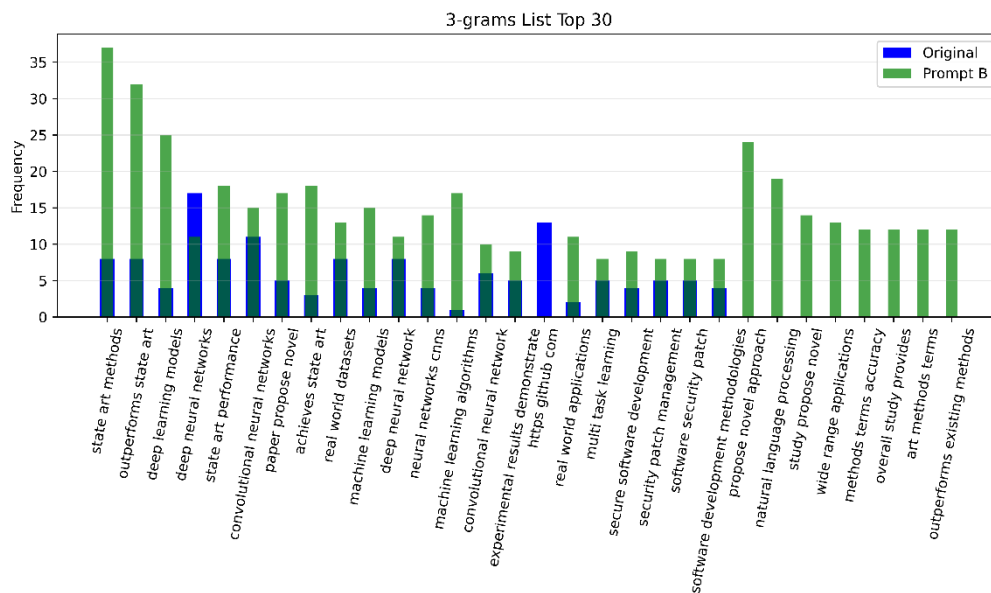
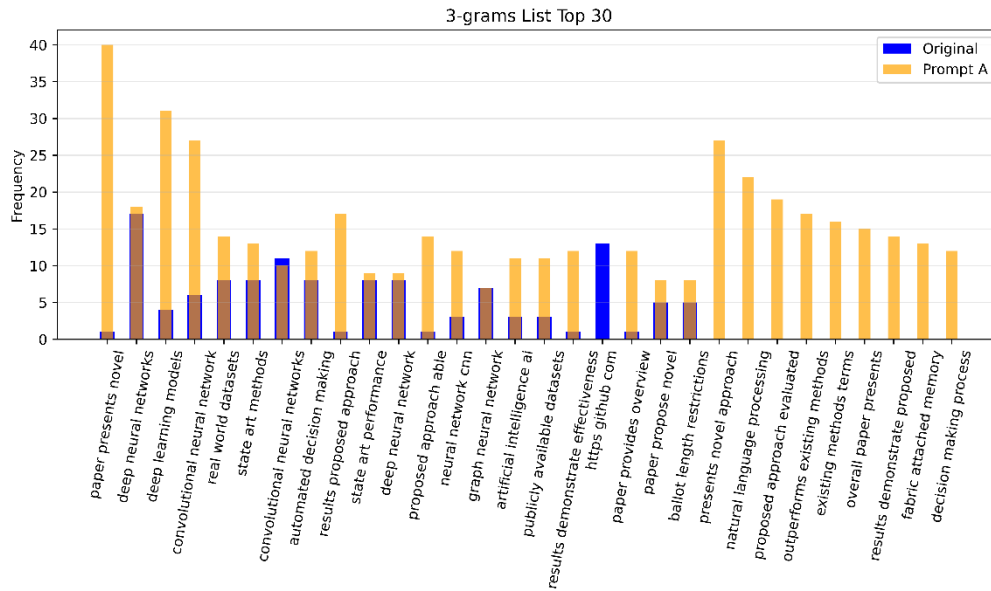
- [9] OpenAI, "Completions" OpenAI API Reference. [Online]. Available: <https://platform.openai.com/docs/api-reference/completions>. [Accessed Apr. 2, 2023].
- [10] OpenAI, "GPT-2 Output Detector Demo" OpenAI Detector Demo. Available: <https://openai-openai-detector-nx2r9.hf.space/>. [Accessed Apr. 5, 2023].
- [11] H. El Mostafa and F. Benabbou, "A deep learning based technique for plagiarism detection: a comparative study" International Journal of Artificial Intelligence, vol. 9, no. 1, Mar. 2020. pp. 83,90 [Online]. Available: <https://dx.doi.org/10.11591/ijai.v9.i1.pp81-90>
- [12] Catherine A. Gao, Frederick M. Howard, Nikolay S. Markov, Emma C. Dyer, Siddhi Ramesh, Yuan Luo, and Alexander T. Pearson, "Comparing scientific abstracts generated by ChatGPT to original abstracts using an artificial intelligence output detector, plagiarism detector, and blinded human reviewers" bioRxiv, Dec. 23, 2022. [Online]. Available: <https://doi.org/10.1101/2022.12.23.521610>
- [13] M. Zamanian and P. Heydari, "Readability of Texts: State of the Art" in Theory and Practice in Language Studies, vol. 2, no. 1, Jan. 2012. pp. 43-45. [Online]. Available: <https://ir.ucc.edu.gh/xmlui/bitstream/handle/123456789/6414/The%20Yutong%20Bus%20Representations%20of%20a%20New.pdf?sequence=1&isAllowed=y#page=45> [Access Apr. 10, 2023]
- [14] D. Jurafsky and J. H. Martin, "Lexicons and Connotation for Sentiment, Affect" in Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, 3rd ed., Draft. Stanford, USA: Stanford University, 2023, pp. 496,502, 510. [Online]. Available: <https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf> [Accessed April 1, 2023].
- [15] D. Jurafsky and J. H. Martin, "Sequence Labeling for Parts of Speech and Named Entities " in Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, 3rd ed., Draft. Stanford, USA: Stanford University, 2023, pp. 160. [Online]. Available:

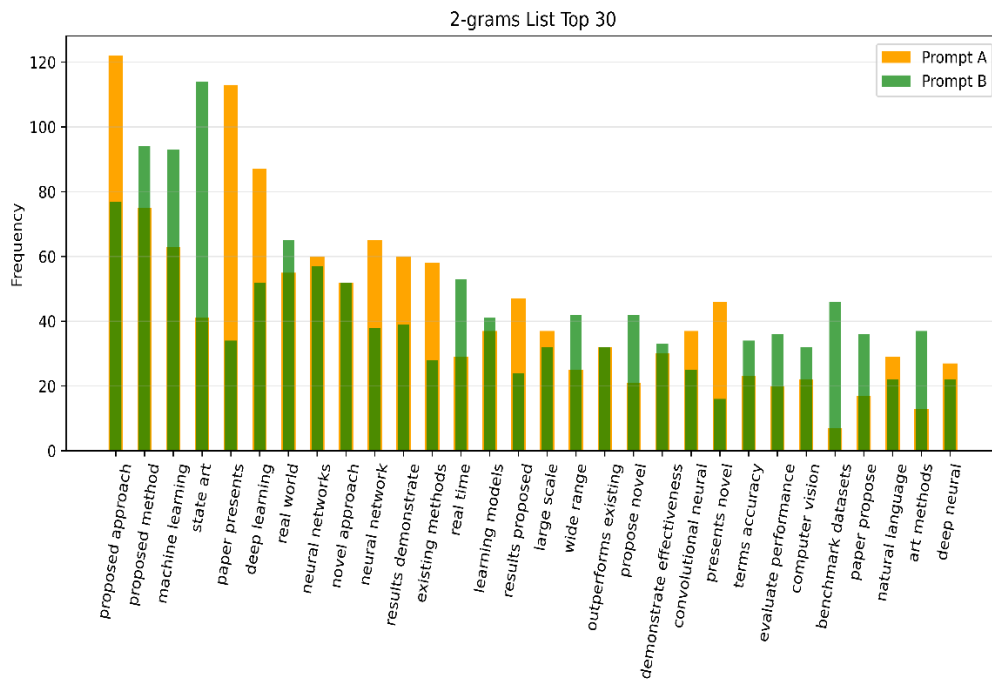
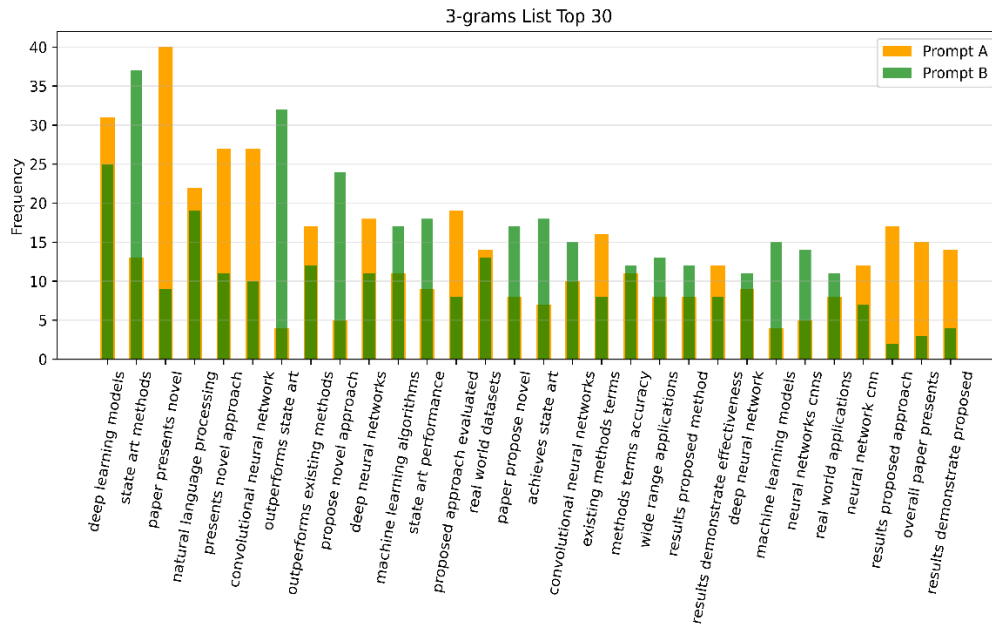
- <https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf>
[Accessed April 1, 2023].
- [16] A. Géron, "*Logistic Regression*" in Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd ed., Sebastopol, CA: O'Reilly Media, Inc., 2019, ISBN: 978-1-492-03264-9, pp. 143-144.
 - [17] A. Géron, "*Support Vector Machines*" in Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd ed., Sebastopol, CA: O'Reilly Media, Inc., 2019, ISBN: 978-1-492-03264-9, pp. 153-155, 157.
 - [18] A. Géron, "*Random Forests*" in Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd ed., Sebastopol, CA: O'Reilly Media, Inc., 2019, ISBN: 978-1-492-03264-9, pp. 197-198.
 - [19] C. Orasan, "*Patterns in scientific abstracts*" in Proceedings of Corpus Linguistics 2001 Conference, School of Humanities, Languages and Social Sciences, University of Wolverhampton, 2001. [Online]. Available:
<https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=91e0cdf92155f1e500ee1f9a3861287a72ec2e5f>
[Access Mar. 28, 2023]
 - [20] H. Gómez-Adorno, J-P. Posadas-Duran, G. Ríos-Toledo, G. Sidorov, and G. Sierra, "*Stylometry-based Approach for Detecting Writing Style Changes in Literary Texts*" *Computación y Sistemas*, vol. 22, no. 1, 2018, pp 1-2 [Online]. Available:
<https://doi.org/10.13053/cys-22-1-2882>
 - [21] Hugging Face, "*Transformers*". [Online]. Available:
<https://huggingface.co/docs/transformers/index> [Accessed Apr. 1, 2023]
 - [22] Textstat, "*Project description*". [Online]. Available:
<https://pypi.org/project/textstat/> [Accessed Apr. 1, 2023]
 - [23] Sentence-transformers. [Online]. Available:
<https://www.sbert.net/docs/installation.html> [Accessed Apr. 1, 2023]

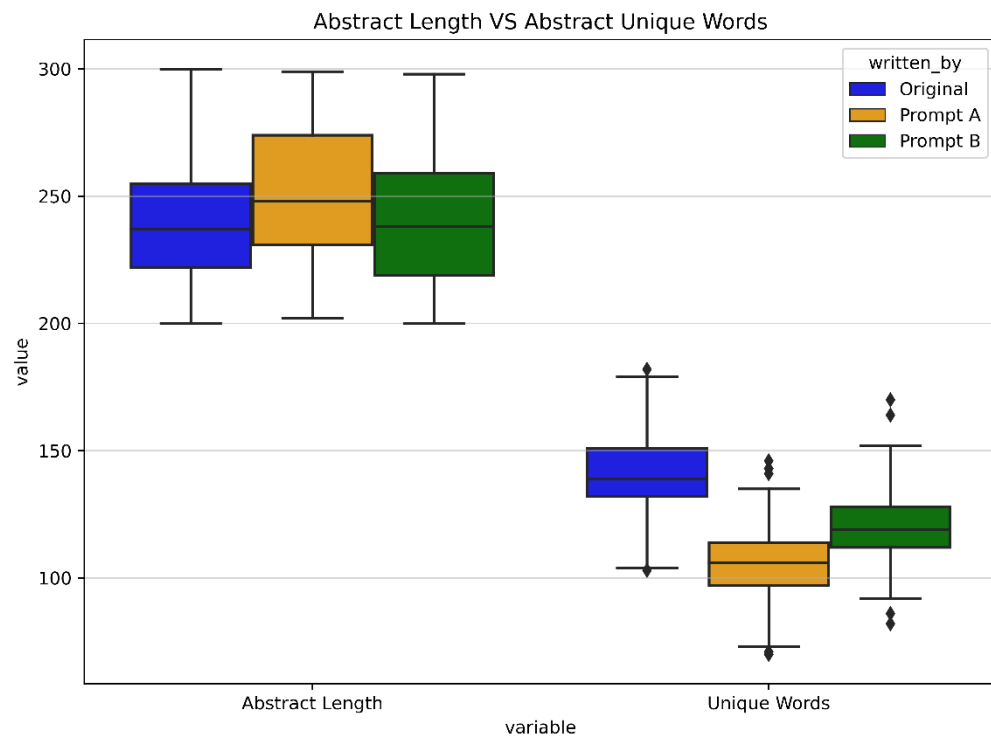
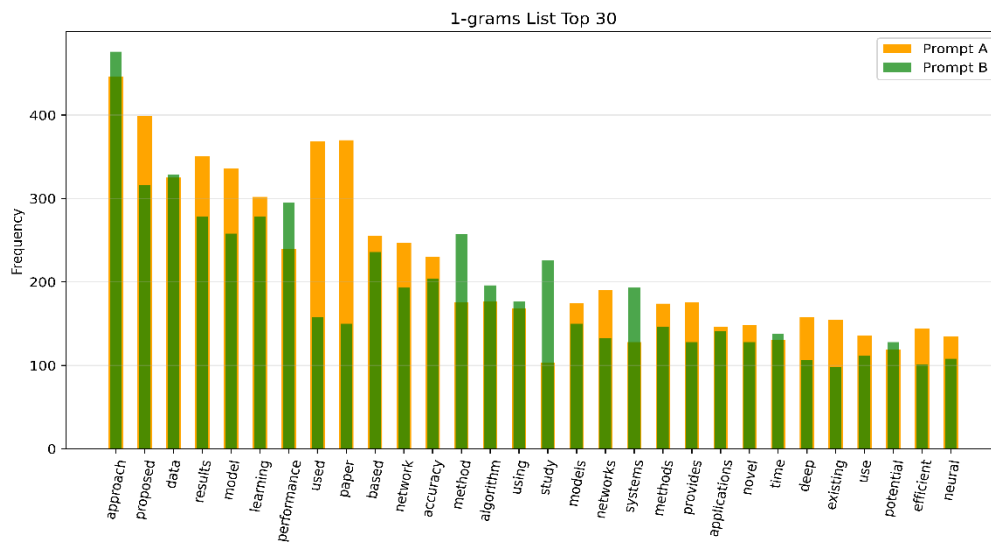
- [24] PassivePy, "A package for processing large corpora and detecting passive voice" [Online]. Available: <https://pypi.org/project/PassivePy/> [Accessed Apr. 1, 2023]
- [25] Text-blob, "TextBlob: Simplified Text Processing" [Online]. Available: <https://textblob.readthedocs.io/en/dev/> [Accessed Apr. 1, 2023]
- [26] Seaborn, "Statistical Data Visualization". [Online]. Available: <https://seaborn.pydata.org/> [Accessed Apr. 7, 2023]
- [27] Scikit-learn, "Machine Learning in Python" [Online]. Available: <https://scikit-learn.org/stable/index.html> [Accessed Apr. 1, 2023]
- [28] Online Writing Support, "ACTIVE / PASSIVE VOICE" Towson University. [Online]. Available: <https://webapps.towson.edu/ows/activepass.aspx> [Accessed Apr. 15, 2023]
- [29] L. Tunstall, L. von Werra, and T. Wolf, "Transfer Learning in NLP" in Natural Language Processing with Transformers. Sebastopol, CA, USA: O'Reilly Media, Inc., 2022, pp. 6-8
- [30] A. Géron, "Performance Measures" in Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd ed., Sebastopol, CA: O'Reilly Media, Inc., 2019, ISBN: 978-1-492-03264-9, pp. 88-93
- [31] D. Jurafsky and J. H. Martin, "Cosine for measuring similarity" in Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, 3rd ed., Draft. Stanford, USA: Stanford University, 2023, pp. 112. [Online]. Available: <https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf> [Accessed April 1, 2023]
- [32] Hugging Face, "RoBERTa Base OpenAI Detector". [Online]. Available: <https://huggingface.co/roberta-base-openai-detector> [Accessed Apr. 1, 2023]
- [33] GrammarBook, "Adjectives and Adverbs", [Online]. Available: <https://www.grammarbook.com/grammar/adjAdv.asp> [Accessed May 15, 2023]

- [34] Kaggle, “*arXiv Dataset*” [Online]. Available:
[https://www.kaggle.com/datasets/Cornell-
University/arxiv?datasetId=612177&sortBy=dateRun&tab=profile](https://www.kaggle.com/datasets/Cornell-University/arxiv?datasetId=612177&sortBy=dateRun&tab=profile)
[Access Apr. 1,2023]

Appendix A: Distributions of textual features and additional plots







```
df_both_cleaned.groupby('written_by')['unique_words'].describe()
```

✓ 0.1s

	count	mean	std	min	25%	50%	75%	max
written_by								
Original	253.0	140.869565	15.369096	103.0	132.0	139.0	151.0	182.0
Prompt A	253.0	105.501976	13.739823	70.0	97.0	106.0	114.0	146.0
Prompt B	253.0	120.035573	12.795814	82.0	112.0	119.0	128.0	170.0

```
df_both_cleaned.groupby('written_by')['title_count'].describe()
```

✓ 0.1s

	count	mean	std	min	25%	50%	75%	max
written_by								
Original	253.0	9.968379	2.77728	5.0	8.0	10.0	12.0	17.0
Prompt A	253.0	9.968379	2.77728	5.0	8.0	10.0	12.0	17.0
Prompt B	253.0	9.968379	2.77728	5.0	8.0	10.0	12.0	17.0

```
df_both_cleaned.groupby('written_by')['abstract_count'].describe()
```

✓ 0.1s

	count	mean	std	min	25%	50%	75%	max
written_by								
Original	253.0	239.241107	23.096785	200.0	222.0	237.0	255.0	300.0
Prompt A	253.0	250.695652	26.974236	202.0	231.0	248.0	274.0	299.0
Prompt B	253.0	240.660079	25.999601	200.0	219.0	238.0	259.0	298.0

```
df_both_cleaned.groupby('written_by')['title_relevance'].describe()
```

✓ 0.1s

	count	mean	std	min	25%	50%	75%	max
written_by								
Original	253.0	0.445059	0.088293	0.22	0.38	0.46	0.51	0.65
Prompt A	253.0	0.552451	0.106917	0.28	0.48	0.55	0.63	0.84
Prompt B	253.0	0.544585	0.090516	0.31	0.48	0.54	0.61	0.83


```
df_both_cleaned.groupby('written_by')['sentiment_subjectivity'].describe()
```

✓ 0.1s

	count	mean	std	min	25%	50%	75%	max
written_by								
Original	253.0	0.447945	0.081551	0.22	0.40	0.44	0.50	0.70
Prompt A	253.0	0.481225	0.102115	0.13	0.43	0.49	0.55	0.88
Prompt B	253.0	0.467154	0.092851	0.19	0.41	0.47	0.52	0.70

```
df_both_cleaned.groupby('written_by')['sentiment_polarity'].describe()
```

✓ 0.1s

	count	mean	std	min	25%	50%	75%	max
written_by								
Original	253.0	0.088142	0.075531	-0.16	0.04	0.09	0.14	0.27
Prompt A	253.0	0.145692	0.103203	-0.19	0.08	0.14	0.21	0.57
Prompt B	253.0	0.107431	0.076330	-0.15	0.06	0.11	0.16	0.34

```
df_both_cleaned.groupby('written_by')['adverb_freq'].describe()
```

✓ 0.1s

	count	mean	std	min	25%	50%	75%	max
written_by								
Original	253.0	0.035415	0.015797	0.0	0.02	0.03	0.05	0.08
Prompt A	253.0	0.025652	0.013890	0.0	0.02	0.02	0.03	0.08
Prompt B	253.0	0.023913	0.011308	0.0	0.02	0.02	0.03	0.06

```
df_both_cleaned.groupby('written_by')['adjective_freq'].describe()
```

✓ 0.1s

	count	mean	std	min	25%	50%	75%	max
written_by								
Original	253.0	0.139407	0.030251	0.06	0.12	0.14	0.16	0.22
Prompt A	253.0	0.121186	0.031348	0.04	0.10	0.12	0.14	0.20
Prompt B	253.0	0.124427	0.028315	0.06	0.10	0.12	0.14	0.22

```
df_both_cleaned.groupby('written_by')['verb_freq'].describe()
```

✓ 0.1s

	count	mean	std	min	25%	50%	75%	max
written_by								
Original	253.0	0.147826	0.023189	0.09	0.13	0.15	0.16	0.21
Prompt A	253.0	0.164664	0.027566	0.11	0.15	0.17	0.18	0.24
Prompt B	253.0	0.153478	0.023916	0.09	0.14	0.15	0.17	0.23

```
df_both_cleaned.groupby('written_by')['noun_freq'].describe()
```

✓ 0.1s

	count	mean	std	min	25%	50%	75%	max
written_by								
Original	253.0	0.323360	0.034906	0.24	0.30	0.33	0.35	0.41
Prompt A	253.0	0.321937	0.035386	0.24	0.30	0.32	0.34	0.44
Prompt B	253.0	0.338379	0.032988	0.25	0.32	0.34	0.36	0.43

```
df_both_cleaned.groupby('written_by')['passive_count'].describe()
```

✓ 0.1s

	count	mean	std	min	25%	50%	75%	max
written_by								
Original	253.0	2.932806	2.143519	0.0	1.0	2.0	4.0	11.0
Prompt A	253.0	4.158103	2.597065	0.0	2.0	4.0	5.0	14.0
Prompt B	253.0	2.494071	1.991292	0.0	1.0	2.0	4.0	12.0

```
df_both_cleaned.groupby('written_by')['readability_score'].describe()
```

✓ 0.1s

	count	mean	std	min	25%	50%	75%	max
written_by								
Original	253.0	30.357352	11.934265	0.45	21.81	30.30	38.55	60.89
Prompt A	253.0	35.330949	11.010646	-7.55	28.43	34.97	43.32	68.97
Prompt B	253.0	30.371581	9.792625	0.31	24.17	31.31	36.52	64.61

```
df_both_cleaned.groupby('written_by')['lexical_diversity_ratio'].describe()
```

✓ 0.1s

	count	mean	std	min	25%	50%	75%	max
written_by								
Original	253.0	0.591739	0.053126	0.460000	0.550000	0.600000	0.630000	0.700000
Prompt A	253.0	0.424017	0.059598	0.266892	0.383178	0.427984	0.467128	0.570755
Prompt B	253.0	0.501897	0.054754	0.330000	0.470000	0.510000	0.540000	0.640000

```
df_both_cleaned.groupby('written_by')['avg_sentence_length'].describe()
```

✓ 0.1s

	count	mean	std	min	25%	50%	75%	max
written_by								
Original	253.0	22.818577	4.205204	13.9	19.8	22.2	25.6	40.6
Prompt A	253.0	19.829644	2.636667	10.8	18.0	19.8	21.5	27.3
Prompt B	253.0	20.926877	2.720321	13.3	19.2	20.8	22.4	31.4

```
df_both_cleaned.groupby('written_by')['sentence_count'].describe()
```

✓ 0.1s

	count	mean	std	min	25%	50%	75%	max
written_by								
Original	253.0	10.782609	2.063519	6.0	9.0	11.0	12.0	19.0
Prompt A	253.0	12.849802	2.106577	8.0	12.0	13.0	14.0	19.0
Prompt B	253.0	11.671937	1.821140	8.0	11.0	12.0	13.0	19.0