The most effective features to distinguish humanly-authored and ChatGPT-generated text

Rijnders, Mike

Citation

Rijnders, M. (2023). *The most effective features to distinguish humanly-authored and ChatGPT-generated text*.

Version:	Not Applicable (or Unknown)
License:	License to inclusion and publication of a Bachelor or Master Thesis, 2023
Downloaded from:	https://hdl.handle.net/1887/3631287

Note: To cite this publication please use the final published version (if applicable).

Universiteit Leiden
The Netherlands

The most effective features to distinguish humanly-authored and ChatGPT-generated text

Mike Rijnders

PwC Thesis advisor: Tom Peters
University Thesis advisor: Ewout Steyerberg
University Thesis advisor: Marieke van Buchem

Defended on 14.07.2023

MASTER THESIS

STATISTICS AND DATA SCIENCE
UNIVERSITEIT LEIDEN

Abstract

The developments in Artificial Intelligence have resulted in the emergence of large language models such as ChatGPT. The development of such models has led to an increased risk of fraudulent activities, therefore this research wants to determine the most effective features for distinguishing between humanly-authored and ChatGPT-generated text within the scientific domain. This research has constructed a text corpus consisting of humanly-authored and ChatGPT-generated abstracts based on the titles of scientific papers. This research build three different XGBoost classifiers, the first based on Doc2Vec vector embeddings, the second on text-extracted features and the third combining both. The results underscore the superiority of models incorporating Doc2Vec vector embeddings while reading time emerged as the most influential feature in accurately predicting whether a text is humanly-authored or ChatGPT-generated in both the text-extracted feature and the combined model. The combined model had the best performance in terms of accuracy. Nevertheless, the model based on Doc2Vec vector embeddings and text-extracted features was still outperformed by the GPTZero model, emphasizing the necessity for further refinement before its application in assessing whether a text is humanly-authored or ChatGPT-generated.

Contents

1 Introduction

The advancements in Artificial Intelligence (AI) have led to the emergence of several powerful models. An example of the implementation of AI in daily life includes the incorporation of virtual assistants and home robots [Rawassizadeh et al., 2019]. In the paper by Rawassizadeh [Rawassizadeh et al., 2019], it is stated that both the virtual assistants and home robots share the characteristics that they both interact with users through conversation and make an effort to replicate human behavior. Recently, the AI powered models ChatGPT [OpenAI, 2022b] and DALL-E [OpenAI, 2022a] by OpenAI have garnered significant attention.

DALL-E, for instance, has found applications in domains as art, illustration and tattooing [OpenAI, 2022a]. The model is utilised to generate visualisations based on textual prompts. ChatGPT has attracted attention because of its ability to simplify complex problems and generate novel ideas [Grant, 2023]. The conversational format of ChatGPT enables the model to respond to follow-up questions, challenge incorrect premises and reject inappropriate requests [OpenAI, 2022b]. These models show the recent advancements in deep learning that are contributing to the development of intelligent systems with human-like capabilities.

This research focuses on text generation using the ChatGPT model. The goal of this research is to identify the most effective features to distinguish between texts generated by ChatGPT in comparison to those authored by humans and to employ these to differentiate between the two. While the use of ChatGPT appears to be promising, it also poses an additional threat of fraudulent activities. Educational institutions have expressed their concerns regarding the use of ChatGPT by students and assessing their work [Mhlanga, 2023]. The concerns regarding fraudulent activities have already led to the development of models which are able to distinguish between humanly-authored and AI-generated text.

For instance, GPTZero [Tian, 2023] employs a logistic regression algorithm with perplexity and burstiness to assess whether a text is humanly-authored or AI-generated. In contrast, the GLTR algorithm, found by Gehrmann et al. [Gehrmann et al., 2019], utilises statistical baseline methods to distinguish between the two. OpenAI [Kirchner et al., 2023] has also introduced its own algorithm, based on a neural network, for identifying humanly-authored and AI-generated texts. This research intends to extract features from both humanly-authored and ChatGPT-generated texts to develop an XGBoost classifier. The classifier will be analysed using SHapley Additive exPlanations (SHAP) [Lundberg and Lee, 2017] to identify the most important features in distinguishing between humanly-authored and ChatGPT-generated texts.

As has been stated above, the arrival of ChatGPT concerns educational institutions [Mhlanga, 2023] as it poses an increased risk of fraudulent activities. In light of this concern, this research wants to identify the most effective features to distinguish between humanly-authored and ChatGPT-generated texts in the scientific domain. The existing approaches mentioned above rely on features, statistics or a neural network. This research aims to expand research on the differences in characteristics of AI-generated and humanly-authored text by constructing an XGBoost classifier that incorporates text-extracted features and advanced machine learning techniques by utilising Doc2Vec. Specifically, this research aims to determine the characteristics that distinguish ChatGPT-generated text from humanly-authored text. To determine the most important characteristics, this research will evaluate the classifier using SHAP [Lundberg and Lee, 2017]. By implementing this approach, this research aims to answer the following research question: *What features are most effective in distinguishing ChatGPT-generated and humanly-authored texts in the scientific domain?*

This research is organised into several sections. First, a section on related literature which provides insight into how ChatGPT is trained, a concise overview of existing models to distinguish between humanly-authored and AI-generated text and features that can be used to differentiate. In the third section, the methods, it is described how the data is obtained and the classification method is discussed. The fourth section discusses the exploration of the data, the differences between the humanly-authored text corpus and the ChatGPT-generated corpus and the classification results. The fifth and final section, the discussion, provides an analysis of the results, provides a discussion on further research and the limitations of this research.

2 Related Literature

This section outlines an analysis of the research that has been conducted to distinguish between humanly-authored and AI-generated text. The goal of this section is to define characteristics and features that can be extracted from texts to differentiate between the two. First, the training process and limitations of the ChatGPT model are discussed [OpenAI, 2022b], highlighting the challenges and limitations of this model in generating text.

In the second subsection, an overview of existing AI-generated text classifiers and their underlying principles and techniques are discussed. This subsection explores the different approaches and algorithms used by these classifiers to differentiate between humanly-authored and AI-generated text.

The third subsection, which discusses features to distinguish humanly-authored and ChatGPT-generated text, is further broken down into subsections. The first subsection describes different approaches to pre-process text before extracting features. The next subsection discusses the different vector representations which can be used to represent text and its characteristics to accurately differentiate between humanly-authored and AI-generated text. In the subsequent subsections, different features regarding text informativeness, text readability and text characteristics are discussed to distinguish between humanly-authored and ChatGPT-generated text.

2.1 The ChatGPT model

The ChatGPT model developed by OpenAI has been trained using reinforcement learning from human feedback [OpenAI, 2022b]. The training data consisted out of two parts. Firstly, the data used to train the predecessor of ChatGPT, InstructGPT, was converted into a dialogue format. Next to that, human conversations were added to the training data. OpenAI made use of so-called ”AI trainers” which played both the user, the one who asked the question, and the AI assistant. The trainers used model-written suggestions to formulate their response. To enable reinforcement learning, a comparison dataset was created which consisted of two or more model-generated responses. Model responses were re-sampled after which AI-trainers ranked the responses, these rankings were used to fine-tune the model.

While the utilisation of ChatGPT appears to be promising, its reliability is hindered by certain inadequacies as reported by OpenAI [OpenAI, 2022b]. ChatGPT occasionally provides incorrect or nonsensical answers. This is difficult to improve as this is a consequence of the limitations of the training data. An adoption to this problem could be to make ChatGPT more cautious to answer questions, however, this would make the model also decline to answer a question to which it does know the answer. Moreover, ChatGPT's capability to answer a question accurately can be impacted by its formulation, the model's current inability to ask for clarification on a question represents a significant limitation. Furthermore, reusing certain sentences and writing long-winded sentences is considered one of the models shortcomings. The ”AI trainers” preferred longer extensive answers which caused for bias in the training data. Despite efforts from OpenAI to make the model decline improper requests, ChatGPT may still respond to detrimental prompts or display biased behavior.

2.2 Existing AI-generated text classification models

The first model introduced in the introduction to detect whether text is AI-generated or humanly-authored is called GPTZero [Tian, 2022]. This model is trained using a corpus of humanly-authored and AI-generated texts. The initial version of the model relied on linear regression, but it has since been updated to logistic regression [Tian, 2023]. The updated model utilises the same variables and inputs, but it provides a more nuanced classification. The model utilises perplexity, burstiness, and other variables in a simple calculation to determine the probability of the text being generated by AI [Tian, 2023]. Perplexity measures the degree of familiarity of the text to a Large Language Model (LLM), whereas burstiness is a measure of variation in the text.

GPTZero returns a document-level score, this indicates the probability of a document being AI-generated. GPTZero [Tian, 2022] recommends to use a threshold of 0.65 or higher to determine whether a document is AI-generated. The algorithm considers documents with scores above this threshold as AI-generated. Furthermore, sentence based classification should only be used to indicate which parts of a document classified as AI-generated could be generated. It is not advisable to rely on sentence classification to decide whether a document is partially generated as it may not provide accurate results. Testing the model, GPTZero managed to correctly classify 99% of the humanly-authored texts and 85% of the AI-generated texts. Notably, it has been observed that the performance of the GPTZero model improves when the size of the input text increases or the input closely resembles text from the training data.

GPTZero [Tian, 2022] also provides insight into the limitations of the current model. As AI is developing, AI-generated content is constantly changing. Therefore, GPTZero also wants to create awareness that the classification of GPTZero should only be a small part on assessing a students work.

The second algorithm, with the aim of detecting AI-generated text is developed by Gehrmann et al. [Gehrmann et al., 2019]. The authors emphasized the importance of the classification method being accurate, explainable to non-experts and easy to set up. The GLTR model is based on the foundation that text generators tend to use a limited subset of the natural language with a high probability of usage. Gehrmann et al. [Gehrmann et al., 2019] use color-coded visualisations to show the likelihood of the occurrence of a word. Words that are highly probable, like ”in”, ”and” or ”example”, are colored green. Less likely words are colored yellow and the least likely words are colored red and purple respectively. In their experiments, Gehrmann et al. [Gehrmann et al., 2019] found that humans tend to use words that are colored purple more often than AI-text generators.

Furthermore, Gehrmann et al. [Gehrmann et al., 2019] conducted an experiment that showed that humans were able to differentiate between AI-generated and humanly-authored text in 54% of the cases. The GLTR algorithm proposed by their research was able to achieve an accuracy of over 72% without any prior training. As the authors assume that AI-generated text stems from biased sampling techniques to produce fluent output, the research performs three different tests.

- The likeliness of the occurrence of a word.
- Where the word ranks as compared to other words.
- How predictable the next word is based on its context.

The first two tests are used to determine whether a word is sampled from the top of the distribution. The last test establishes whether the previous context is well-known to the detection system, this is used to establish what word occurs next.

The third classifier mentioned in the introduction was developed by OpenAI. OpenAI has developed a classifier using a neural network trained on a dataset comprising both humanly-authored and AI-generated texts on the same subject [Kirchner et al., 2023]. The dataset was generated in two different ways. The first method involved truncating text at 1,000 tokens, after which language models generated a completion of the truncated text. This completion was then paired with the original continuation. The second method used human answers to prompts which were used to train the predecessor of ChatGPT, InstructGPT. The prompts were submitted to language models trained by OpenAI, as well as models trained by other organisations and then paired to the human answers. The resulting dataset included texts generated by 34 different language models.

The OpenAI classifier [Kirchner et al., 2023] uses the input text and yields whether the text is ”very unlikely”, ”unlikely”, ”unclear if it is”, ”possibly” or ”likely AI-generated”. As noted by the authors, it remains challenging to accurately detect AI-generated text. Therefore, as was the case with GPTZero, the threshold is adjusted to keep the false positive rate low. This results in only classifying text as AI-generated if the classifier is confident. The OpenAI classifier demonstrated a 26% success rate in correctly identifying AI-generated texts as ”likely AI-generated”. In 9% of the cases, the classifier incorrectly classified humanly-authored texts as ”likely AI-generated”.

OpenAI considers the model to be unreliable when the input text is below 1,000 characters [Kirchner et al., 2023]. This is similar to the classifier of GPTZero [Tian, 2022], where it was also found that when the input text was larger, the accuracy of the classifier increased. Additionally, it is recommended to solely use the classifier on English texts since this was the language of the training data. The authors noted that the performance of the classifier deteriorated considerably when applied to texts in other languages. Furthermore, the authors cautioned that whenever an AI-generated text is altered, it can evade being detected as AI-generated. Moreover, the authors acknowledged the limitations of using a neural network to train a classifier of AI-generated and humanly-authored text. If the input text deviates from the training data, the classifier can occasionally demonstrate high confidence in an incorrect prediction, resulting in ”very unlikely” or ”likely AI-generated” classifications.

2.3 Features

2.3.1 Pre-proccesing of text

In the article authored by Gharehchopogh and Khalifelu [Gharehchopogh and Khalifelu, 2011] textual data is characterised as: ”unstructured, amorphous, and complicated to deal with”. Therefore, text should be pre-processed to have it in a usable format. Avinash and Sivasankar [Avinash and Sivasankar, 2019] pre-processed the data in their study by removing stop words and performing tokenisation. Tokenisation is used to break a text into meaningful data while retaining the information about the text. As stop words do not posses meaningful information about a text, Avinash and Sivasankar therefore remove stop words from a text.

These pre-processing techniques are also discussed in the paper by Hickman et al. [Hickman et al., 2022]. This research outlines different text pre-processing techniques. In their paper, Hickman et al. provide recommendations for pre-processing textual data, it is however noted that specific cases may require a different approach.

According to the research of Hickman et al. [Hickman et al., 2022], all text should be converted to lowercase. This should be done because this decreases the dimensionality of the text while maintaining semantic information. Lowercase conversion decreases dimensionality because ”dog” and ”Dog” are now represented in the same way, namely ”dog”.

In addition to lowercase conversion, Hickman et al. [Hickman et al., 2022] suggest that negation handling should also be implemented in the pre-processing process of textual data. Despite increasing the dimensionality of the data, this technique is crucial for a better understanding of the semantic information in a text. In their paper, an example is given of separately tokenising ”not” and ”honest” or tokenising them as one, ”not honest”. Separately tokenising the words gives the sentence a different meaning as opposed to tokenising them as one.

Hickman et al. [Hickman et al., 2022] also discuss the importance of spelling correction, addressing the use of abbreviations and punctuation. They state: ”Unless idiosyncratic language differences, such as spelling errors, abbreviations, and use of punctuation, are useful for the research question at hand, researchers should make these corrections to standardize the language”.

In case of a small corpus, it is argued that stemming or lemmatisation should be used [Hickman et al., 2022]. Stemming is a process that reduces the vocabulary of a corpus by ”removing morphological affixes from words, leaving only the word stem” [Bird et al., 2009]. As an example Hickman et al. [Hickman et al., 2022] show that using the Porter stemmer [Porter, 1980], *organ, organs, organic, organism, organize* and *organization* would all be stemmed to *organ* while *organizational* would be stemmed to *organiz*.

A lemmatiser is another technique that reduces the vocabulary of a corpus by using morphological information to get the base form of a word [Schütze et al., 2008]. The Natural Language Toolkit Wordnet Lemmatiser [Bird et al., 2009] would lemmatise *organized* and *organize* to *organize* while *organization* and *organizations* would be transformed to *organization*.

Hickman et al. [Hickman et al., 2022], argue that lemmatisation is preferable to stemming as it reduces the likelihood of distinct words being conflated, thus enhancing the validity of the analysis. Furthermore, lemmatised words remain recognisable, thus keeping their meaning and therefore being more interpretable than stems.

2.3.2 Vector embeddings

In their research, Gharehchopogh and Khalifelu [Gharehchopogh and Khalifelu, 2011] state that text is an unstructured form of data. This poses a challenge when using machine learning techniques, which typically require fixed-length feature vectors. To address this challenge, researchers have explored different techniques for feature extraction. In the study by Avinash and Sivasankar [Avinash and Sivasankar, 2019], the Term Frequency-Inverse Document Frequency (TF-IDF) and Doc2Vec (known as the paragraph vector [Le and Mikolov, 2014]) techniques are used to discern the polarity of textual data. In their paper it is discussed that whenever patterns in the texts show resemblance, discriminating between classes becomes a challenging task.

In the paper of Avinash and Sivasankar [Avinash and Sivasankar, 2019], the TF-IDF and Doc2Vec document representations are employed to extract features from textual data. However, there are more methods available, Basarkar [Basarkar, 2017] for instance mentions the use of the TF-IDF vectoriser but also discusses the use of the Binary and Count vectoriser. Kim et al. [Kim et al., 2019] proposed another method while comparing document classification methods using different document representations. In their research TF-IDF, Doc2Vec and Latent Dirichlet Allocation (LDA) are used.

The Binary, Count and TF-IDF vectorisation all are based on the Bag-Of-Words (BOW) approach [Harris, 1954]. These models are popular because of its simplicity and often good performance. Binary vectorisation [Basarkar, 2017] is a text representation technique that uses binary values to encode a text document. The document vector is representative of the number of unique words in the text corpus, referred to as the vocabulary. A binary encoding assigns a value of 1 to a word that exists in a document and 0 to a word that is absent. A variation on this approach is the use of Count vectorisation [Basarkar, 2017]. This technique uses the frequency of each word to represent a document. Words appearing twice in the document will have value 2, followed by 1 for words that appear once and 0 for words that do not appear in the document. Count vectorisation captures more detail than Binary vectorisation as it considers the frequency of each word, allowing for identification of important words in a document.

One limitation of the Count vectoriser is that it does not take into account how rare or common a word is within the corpus of documents [Basarkar, 2017]. To overcome this limitation, an alternative approach called Term Frequency-Inverse Document Frequency (TF-IDF) [Jurafsky and Martin, 2023] has been suggested. The technique assigns a weight to each word in a document based on its frequency and importance in the corpus of documents, with the aim of capturing the informative power of less frequent words.

The intuition behind this approach is that the most frequent terms are not informative [Jurafsky and Martin, 2023]. A document that contains word t 10 times is not 10 times more relevant than a document that only contains word t once. To take this into account the Log Term Frequency can be used as opposed to the regular Term Frequency. Using this approach, when there are two documents, A and B, where document A only contains word t once and document B contains word t twice, document B is regarded as more relevant. However, when word t occurs one million times in document A and two million times in document B, both are regarded to be highly relevant. In this case B is not considered to be much more relevant than document A. The Log Term Frequency is calculated as:

$$TF_{t,d} = log_{10}(tc_{t,d} + 1)$$

- $tc_{t,d}$: The term count of term t in document d

The second factor in the TF-IDF score, the IDF, is used to assign a higher weight to terms that are present in a limited number of documents [Jurafsky and Martin, 2023]. This is because such terms are considered to have high discriminatory power and are more informative than words which occur frequently across the entire corpus of documents. The Inverse Document Frequency is calculated by:

$$IDF_t = log_{10} \frac{N}{df_t}$$

- df_t: The document frequency, this is the number of documents that contain the word t.
- N: The total number of documents in the collection.

The TF-IDF score for a specific term t in document d is computed as $TF_{t,d} * IDF_t$.

In the paper by Blei et al. [Blei et al., 2003] it is stated that the TF-IDF approach reveals little in the way of inter- or intradocument statistical structure. According to the research by Blei et al., several approaches have been introduced to address these shortcomings. Latent Semantic Indexing (LSI) [Deerwester et al., 1990] is the foremost approach according to Blei et al.. LSI works on the following principle: "LSI uses a singular value decomposition of the X matrix to identify a linear subspace in the space of TF-IDF features that captures most of the variance in the collection" [Blei et al., 2003]. Nonetheless, LSI still follows the BOW approach. Therefore, in order to generate exchangeable representations for both words and documents, mixture models should be considered to capture the ex-changeability of both words and documents. This has lead to the development of the Latent Dirichlet Allocation (LDA) by Blei et al. [Blei et al., 2003].

LDA is a generative probabilistic model of a corpus [Blei et al., 2003]. Blei et al. describe the intuition as: "The basic idea is that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words". For each document d in a text corpus D, LDA assumes the following generative process:

- Choose $N \sim Poisson(\zeta)$.
- Choose $\theta \sim Dir(\alpha)$.
- For each of the N words d_n:
  - Choose a topic $z_n \sim Multinomial(\theta)$
  - Choose a word w_n from $p(d_n|z_n, \beta)$, a multinomial probability conditioned on the topic z_n.

Using LDA, the algorithm iteratively updates the topic assignments considering the words in the document and the distribution of topics in the entire corpus [Blei et al., 2003]. LDA produces a set of probability distributions over topics for each corpus document and a set of probability distributions over words for each topic. These will help to identify the most probable topic for each document and the most probable words for each topic. As a result, the most probable words for each topic can be used as a document representation method for a classification task.

As is already stated by Blei et al. [Blei et al., 2003], the TF-IDF approach has certain shortcomings. This is also highlighted by Le and Mikolov [Le and Mikolov, 2014]. In their paper several limitations of the BOW approach are emphasized. The use of Binary vectorisation, Count vectorisation, TF-IDF vectorisation or LDA, models based on the BOW approach, the word order is ignored. Consequently, sentences containing identical words may be assigned the same representation despite carrying different meaning. Models using the BOW approach lack a semantic understanding of the words, which leads to a sub-optimal performance. Le and Mikolov [Le and Mikolov, 2014] state that models using the BOW approach treat words as "powerful", "strong" and "Paris" equally while semantically, "powerful" should be closer to "strong" than to "Paris" since they have a similar meaning.

In their paper, Le and Mikolov [Le and Mikolov, 2014] introduce the concept of the Paragraph vector used in the research by Avinash and Sivasankar [Avinash and Sivasankar, 2019] and Kim et al. [Kim et al., 2019]. This model learns continuous distributed vector representations for different pieces of text. The model's framework for learning paragraph vectors mirrors the approach employed in learning word vectors (Word2Vec) [Le and Mikolov, 2014]. Word vectors are build while constructing a neural network to predict a word given its contextual words. After the training converges, words that share similar meaning are positioned closely together within the vector space. "strong" and "powerful" would thus be close to each other while "Paris" would be more distant.

The same construct is adapted in the context of the Paragraph Vector Model [Le and Mikolov, 2014], wherein the subsequent word is predicted based on contexts sampled from the paragraph. These contexts are sampled using a sliding window of a predetermined length over the paragraph. As apposed to Word2Vec, where a single vector representation is learned for each word, Doc2Vec learns two vectors for each document, a document vector (known as the Paragraph Vector) and a set of word vectors. Just as was the case with the representation of similar words in the vector space, similar documents can now also be positioned closely together within the vector space. Moreover, the paragraph vectors constructed can be used as features in machine learning models.

This research has decided to use Doc2Vec as this method has the ability to capture the semantic meaning of a text and is therefore able to capture more detail than the other methods mentioned above. The ability of Doc2Vec to capture the semantic meaning allows to compare the content of the different abstracts. As each paper title has one humanly-authored and one ChatGPT-generated abstract, it will be interesting to see whether the vector representations of these abstracts, and thus it contents, are most similar.

2.3.3 Text informativeness

Jiang and Srinivasan [Jiang and Srinivasan, 2023] propose three features to quantify the contextual awareness of a text. The first measure, the boilerplate evaluates the informativeness of a text. The boilerplate refers to the words in a sentence that can be removed without altering its meaning. The degree of informativeness is given by the boiler score which is determined by comparing the number of sentences containing boilerplate language to the total number of words in the text. The higher the score, the lower the informativeness of a given text. This is mathematically defined as:

$$Boilerplate = \frac{W_s}{W_d}$$

- W_s: The word count of the sentence that has a boilerplate.
- W_d: The word count of the whole document.

Next to the boilerplate score, the informativeness of a text can also be evaluated using the redundancy as proposed by Jiang and Srinivasan [Jiang and Srinivasan, 2023]. Redundancy is a prevalent issue in text generation. In the paper by Lloret and Palomar [Lloret and Palomar, 2013], it is argued that an effective summarisation technique strives to combine the central themes of a piece of text while maintaining completeness, readability and conciseness. However, redundancy poses a well-established challenge to text summarisation as it introduces noise and undermines the quality of the resulting summary when information is duplicated.

Therefore, the measure of redundancy, as proposed by Jiang and Srinivasa [Jiang and Srinivasan, 2023], reflects the usefulness of a text. This can be quantified by calculating the proportion of large sentences that appear more than once in a text. To determine the redundancy, Jiang and Srinivasan [Jiang and Srinivasan, 2023] make use of n-grams. N-grams can be defined as ”consecutive sequences of n characters)” [Damashek, 1995]. When specific n-grams are repeatedly used in a text, the resulting duplicate information is non-useful. Similar to the boiler score, higher redundancy scores indicate lower informativeness of the text.

Third, Jiang and Srinivasan [Jiang and Srinivasan, 2023] propose to measure the specificity of a text. This measures how much a text relates to a specific subject. It is described as the number of specific entity names, numerical values and time or date specifications scaled by the overall word count of a text. To determine the entities in a text, Jiang and Srinivasan make use of the Natural Language Processing Python Package spaCy [Honnibal et al., 2020].

Measure	**Meaning**	**Scale**
Boilerplate	The informativeness of a piece of text. Measured as the ratio of sentences that contain words that can be left out without altering the meaning of the sentence, relative to the total word count of a text.	0% (Not informative) - 100% (Very informative)
Redundancy	A measure of usefulness of text, defined by the n-grams that occur more than once.	0% (Not informative) - 100% (Very informative)
Specificity	How much a text relates to a specific subject, defined as the number of specific entity names, numerical values and time or date specifications scaled by the overall word count of a text	0% (Not informative) - 100% (Very informative)

Table 1: Measures on informativeness.

To quantify the informativeness of a text, this research uses the boilerplate, redundancy and specificity measure proposed by Jiang and Srinivasan [Jiang and Srinivasan, 2023].

2.3.4 Text readability

In recent years the application of neural networks to summarise text has gained increasing attention. One of the main problems concerning such generated texts, was its readability [Zhuang and Zhang, 2019]. Different metrics, based on different foundations as syllables, polysyllables, word length and lists of similar words, are available to determine the readability of a text.

The Flesch Reading Ease Test [Flesch, 1979], Flesch Kincaid Reading Ease Test [Kincaid et al., 1975] and Linsear Write approach [O'hayre, 1966] are readability measures based on the number of words, sentences and syllables. Flesch [Flesch, 1979] asserted that mastering the use of a readability formula is a fundamental skill for writing English text that is comprehensible to the target audience. A high score indicates that a text is easily understandable while a low score indicates that the content of a text is complex.

$$\text{Flesch Reading Ease Score} = 206.835 - 1.015 * \frac{\#words}{\#sentences} - 84.6 * \frac{\#syllables}{\#words}$$

A variation to this approach is the Flesch-Kincaid grade level test [Kincaid et al., 1975]. This alternative to the Flesch Reading Ease Test is designed to assess the readability of a text within the context of the US education system. The resulting score is a grade level indicating the number of years of education required to comprehend a text. A high score on the Flesch-Kincaid Readability Test thus indicates a difficult text.

$$\text{Flesch Kincaid Grade Level} = 0.39 * \frac{\#words}{\#sentences} + 11.8 * \frac{\#syllables}{\#words} - 15.59$$

The Linsear Write [O'hayre, 1966] readability metric was found specifically for assessing the readability of technical manuals. Based on a 100 word sample of a text, the Linsear Write formula evaluates the readability based on sentence length and the number of ”easy” and ”hard” words. A word is seen as ”easy” whenever the word consists out of less than three syllables. A word is seen as ”hard” if it consists out of three or more syllables. For each word in the sample, it is determined whether it is ”easy” or ”hard”, ”easy” words are assigned a value of 1, while ”hard” words are assigned a value of 3. The Linsear Write Score is then calculated as:

1. The starting score of the text sample is 0.
2. For each ”easy” word in the text, add 1 point to the score.
3. For each ”hard” word in the text, add 3 points to the score.
4. Divide the score by the number of sentences in the sample.
    - If the score is bigger than 20, the Linsear Write Score is equal to the score divided by 2.
    - If the score is smaller or equal to 20, the Linsear Write Score is equal to the score.

A comparable, but slightly different approach for assessing readability is the use of the number of words, sentences and polysyllables, words with three or more syllables. These metrics are the foundation for the Gunning Fog Index [Robert, 1952] and the Simple Measure of Gobbledygook (SMOG) grade [Mc Laughlin, 1969]. The Gunning Fog Index [Robert, 1952] has been used to determine the grade level required to comprehend a text. The Gunning Fog Index is calculated by taking a sample of approximately 100 words of the input text. The average number of words per sentence is determined by calculating the ratio between the number of words and sentences in the selected sample. The number of difficult words, or polysyllables, is then specified as the words consisting out of three or more syllables that are not proper names, combinations of easy words or verbs that are prolonged because of suffixes.

$$\text{Gunning Fog Index} = 0.4 * (\frac{\#words}{\#sentences} + 100 * \frac{\#polysyllables}{\#words})$$

A comparable approach, the SMOG [Mc Laughlin, 1969], estimates the years of education needed to understand a text. The intuition behind the SMOG grade is that longer words are more difficult to read and understand. The formula uses ten sentences at the beginning of a text, ten in the middle and ten sentences at the end of a text. With a total sample size of 30 sentences, the SMOG grade is an ill advised metric for texts consisting of less than 30 sentences. Needless to say, the higher the estimate of the years of education needed to understand a text, the more difficult a text is to comprehend.

$$\text{SMOG grade} = 1.043 * \sqrt{\#polysyllables * \frac{30}{\#sentences}} + 3.1291$$

Instead of using syllables to assess the difficulty of a text, other approaches incorporate the number of characters to determine the readability. This approach is incorporated by the Automated Readability Index (ARI) [Senter and Smith, 1967], the Coleman Liau Index (CLI) [Coleman and Liau, 1975] and the McAlpine approach [McAlpine, 2012. Using those linguistic features, the ARI [Senter and Smith, 1967] gives an indication of the US grade level needed to comprehend a text. When the ARI score is low, it indicates that the text is simple and easy to comprehend. Conversely, a high ARI score indicates that the text is complex and difficult to understand.

$$\text{ARI} = 4.71 * \frac{\#characters}{\#words} + 0.5 * \frac{\#words}{\#senctences} - 21.43$$

A variation to this is approach is the CLI [Coleman and Liau, 1975], the CLI uses the average numbers of letters and sentences per 100 words. This approach was found because word length in letters was seen as a better predictor of readability than word length in syllables. A score of a 6 corresponds to 6th grade in the US schooling system, thus, the higher the score the higher the level of education that should have been obtained to understand a text.

$$\text{CLI} = 0.0588 * L - 0.296 * S - 15.8$$

- L: Is the average number of letters per 100 words.
- S: Is the average number of sentences per 100 words.

The McAlpine approach [McAlpine, 2012] on the other hand uses the length of a word to define whether a word is difficult to understand for a foreigner. Rachel McAlpine proposed the McAlpine EFLAW readability score to determine the difficulty of a text for non-native English speakers. Familiar words or mini-words, as McAlpine calls them, are words of a length smaller or equal to 3 characters. The lower the score, the easier it is to comprehend a text.

$$\text{McAlpine Readability Score} = \frac{\#words + \#miniwords}{\#sentences}$$

The Dale Chall Index [Dale and Chall, 1948] and Spache [Spache, 1953] approach, as opposed to other methods, determine readability based on word similarity. These methods are rooted in the finding that readers exhibit improved reading comprehension when they encounter familiar words in a text. Both use a count of "hard" words to determine the reading difficulty. Therefore, the methods use a list to categorise words as "familiar". If the word is not present in this list, the word is categorised as "hard". In the first version of the Dale Chall formula, the list of "familiar" words, consisted of 763 words which was later updated to 3,000 words. The Space approach uses a list of 769 "familiar" words for kids up to third grade.

$$\text{Dale Chall Index} = 0.1579 * (\frac{\#hardWords}{\#words} * 100) + 0.0496 * \frac{\#words}{\#sentences}$$

$$\text{Spache Index} = (0.121 * averageSentenceLength) + (0.082 * \frac{\#hardWords}{\#words}) + 0.659$$

A different way of quantifying a texts readability is by looking at the reading time of a text [Demberg and Keller, 2008]. Demberg and Keller use a constant value of 14.69 milliseconds to read one character. This indicates that for each extra character in a word, the reading time increases by approximately 15 milliseconds.

There has also been research on increasing the readability of a text. Kaushik et al. [Kaushik et al., 2020] have performed a study on the use of grammar checkers in increasing the readability of a text. Although, the results show that grammar tools show a significant reduction in error rates, it does not increase the readability of a text.

The development of tools that can evaluate textual inaccuracies holds great importance in the domain of NLP research. D. Naber [Naber et al., 2003] has made a significant contribution to this domain by developing LanguageTool, a tool which can be used for assessing errors in grammar, punctuation and spelling. The tool, which was developed in 2003 using Python, uses a rule set to detect errors in English texts. Since its development, volunteer maintainers have expanded the tool by adding support for multiple languages and different rule sets.

Measure	**Meaning**	**Scale**
Flesch Reading Ease Test	Readability measure based on the number of words, sentences and syllables.	1 (Complex text) - 100 (Easily understandable)
Flesch Kincaid Reading Ease Test	Readability measure based on the number of words, sentences and syllables.	US grade level, indicating the years of education needed to comprehend a text. A higher score thus indicating a more complex text.
Linsear Write Approach	Readability measure based on the number of words, sentences and syllables.	US grade level, indicating the years of education needed to comprehend a text. A higher score thus indicating a more complex text.
Gunning Fog Index	Readability measure based on the number of words, sentences and polysyllables.	US grade level, indicating the years of education needed to comprehend a text. A higher score thus indicating a more complex text.
SMOG Index	Readability measure based on the number of words, sentences and polysyllables.	Years of education needed to comprehend a text, a higher score thus indicating a more complex text.
Automated Readability Index (ARI)	Readability measure based on the number characters, words and sentences.	US grade level, indicating the years of education needed to comprehend a text. A higher score thus indicating a more complex text.
Coleman Liau Index (CLI)	Readability measure on the average number of letters and sentences per 100 words.	US grade level, indicating the years of education needed to comprehend a text. A higher score thus indicating a more complex text.
McAlpine EFLAW	Readability measure based on word length.	1 (Very easy to comprehend)- 30+ (Very difficult to comprehend).
Dale Chall Index	Readability measure based on word similarity using a list of 3,000 familiar words.	US grade level, indicating the years of education needed to comprehend a text. A higher score thus indicating a more complex text.
Spache Approach	Readability measure based on word similarity using a list of 769 words.	US grade level, indicating the years of education needed to comprehend a text. A higher score thus indicating a more complex text.
Reading time	Readability measure based on the time it takes (in seconds) to read text.	Time in seconds, a higher score indicating a longer reading time and thus a more difficult text.
Grammatical and syntactical errors	Readability measure based on the number of grammatical and syntactical errors in a text.	The number of grammatical and syntactical errors in a text, a higher score thus indicating more errors and thus a text that is more difficult to comprehend.

Table 2: Measures on readability.

To capture the readability of an abstract this research has opted to use one readability metric from each group in combination with the reading time and grammatical errors. Regarding the readability metrics, considering the measures in the first category, this research has decided to use the Flesch Reading Ease Test as this is the most widely used and well-established formula [Zamanian and Heydari, 2012]. Regarding the second category, the SMOG index is an ill advised metric for texts consisting of less than 30 sentences, therefore, as an abstract is a short text, this research opted to use the Gunning Fog Index. Considering the third category, the research has chosen to use the ARI readability measure. This approach was chosen as the use of this method has been validated on technical materials by Smith and Kincaid [Smith and Kincaid, 1970]. Regarding the fourth category, as the Spache approach is developed for assessing the readability of a text by children up to third grade, the use of the Spache approach is not suitable to capture the readability of an abstract. Therefore, in this this research, the Dale Chall Index is used.

2.3.5 Text characteristics

According to Lippi et al. [Lippi et al., 2019], natural languages show remarkable statistical properties in their word statistics. Zipf's and Heaps' law [Zipf, 1999, Heaps, 1978] are two statistical principles that show these remarkable properties. Both are used to analyse the frequency distribution and vocabulary of a text corpus. The laws have been found to provide valuable insight into the structure and complexity of natural language. Lippi et al. [Lippi et al., 2019], have used both laws to evaluate the frequency distribution and vocabulary complexity of the LSTM and Markov generated texts. Deviations suggest that a generated text is not similar to human language.

Zipf's law [Zipf, 1999] describes the relationship between the frequency and rank of words in a text corpus. According to this law, the most frequent word in a text corpus occurs approximately twice as often as the second most frequent word, three times as often as the third most frequent word, and so on. Zipf's law suggests that the word frequency in a text corpus follows a power-law distribution where a small number of words occur frequently while others occur rarely.

Heaps' law [Heaps, 1978] is used to describe the relationship between the size of a text corpus and its vocabulary. Heaps' law states that when the size of the corpus increases, the size of the vocabulary also increases. However, when the corpus becomes bigger and bigger the vocabulary will increase at a decreasing rate. This thus means that the number of new unseen words in a new document decreases. Heaps' law does suggest that language is infinitely complex and always evolving as it suggests that no matter how big the text corpus is, there will always be new words to discover.

Both Zipf's and Heaps' law [Zipf, 1999, Heaps, 1978], can thus be used to measure the language complexity of a text. Visualising a text corpus according Heaps' law can show the rate at which the vocabulary grows as a function of its size. The growth rate thus is an indicator of the lexical diversity in a text corpus. The lexical diversity of a text can be assessed using the TRUNAJOD library released by Palma et al. [Palma et al., 2021].

TRUNAJOD, the library developed by Palma et al. [Palma et al., 2021], provides a function to capture the complexity of a text. The lexical diversity of a text is assessed by dividing the unique tokens by the total number of tokens. This ratio is indicative of the extent of repetition within a text, with a higher ratio indicating a more diverse range of vocabulary.

Measure	Meaning	Scale
Lexical diversity	The ratio of unique tokens and the total number of tokens.	0% (Non diverse vocabulary) - 100%(Very diverse vocabulary)

Table 3: Measure on text characteristics.

This research will visualise both the humanly-authored and ChatGPT-generated text corpus according Zipf's and Heaps' Law to give insight into the complexity of both corpora. The complexity of the text will be used as a feature by leveraging the library developed by Palma et al. [Palma et al., 2021]. Palma et al. state that lexical diversity is not effective when comparing texts of different lengths, however, as abstracts are of a comparable length, lexical diversity is a useful measure.

3 Methods

This section outlines the approach taken to determine the most effective features in distinguishing ChatGPT-generated and humanly-authored texts in the scientific domain. To achieve this, a collection of humanly-authored scientific texts and ChatGPT-generated texts in the scientific domain is required. In this section, the development of a PubMed ["PubMed", n.d.] scraper tool is described. This tool extracts the URL to the article, the PubMed ID, title and abstract.

Using the paper titles extracted by the PubMed scraper, this research uses the OpenAI API [Brockman et al., 2020] to generate new abstracts. Features will be extracted from both the humanly-authored and ChatGPT-generated abstracts. The features were based on those found in the related literature and were selected to capture key text characteristics, such as sentence structure, vocabulary and informativeness. These extracted features are used to build an XGBoost classifier. To establish which are the most effective features to determine whether a text is humanly-authored or ChatGPT-generated the SHapley Additive exPlanations (SHAP) library will be used [Lundberg and Lee, 2017].

In the first two subsections the step-by-step approach involved in constructing the PubMed scraper and the generation of abstracts using the ChatGPT API is presented. Third, the approach to exploring the humanly-authored and ChatGPT-generated text corpora is explained. Additionally, the classification procedure is elucidated also explaining how the most important features will be identified.

3.1 PubMed scraper

The PubMed scraper consists out of 3 functions. The first function, check_pmids_page, of the scraper takes a URL as input. This URL contains the query used on PubMed and the page number. If you would search for articles on "data science" on PubMed, the URL for the first page would be "https://pubmed.ncbi.nlm.nih.gov/term=data+science\&page=1".

The purpose of this function is to return a list of unique PubMed IDs on a given page. To return this list, the function first extracts the HTML code of the search page using a Python library called Requests [Reitz, n.d.]. This function returns the HTML code of a search page which is parsed by the Beautiful Soup library [Richardson, 2007] to extract the unique PubMed IDs on a page. The function then returns the list of the unique PubMed IDs found on a page.

The second function, collect_pmids is designed to retrieve 12,499 unique PubMed IDs. This research has opted to scrape 12,499 PubMed IDs due to the possibility that some papers associated with the PubMed IDs may not adhere to the filtering criteria discussed in Section 3.2. As a result, to ensure an adequate amount of a data, the substantial number of 12,499 PubMed ID's has been selected for scraping.

To scrape the 12,499 PubMed IDs, this function uses a list of URLs, which are used as input to the check_pmids_page function. The URLs in the list provided are formatted without a page number, which is provided by a counter. The concatenation of the URL and the page number provided by the counter are then passed as input to the check_pmids_page function. The PubMed IDs extracted by collect_pmids_page are appended to a list if they have not been seen before. If the PubMed IDs are added to the list, the value of the counter is incremented by one to proceed on to the next page.

PubMed only allows scraping of PubMed IDs for papers up to the first 1,000 pages on a query. Therefore, if the 1,000th page is reached, the counter is reset to one after which the next URL in the list of URLs is used in combination with the counter until the 1,000th page is reached. This way, the unique PubMed IDs extracted by the check_pmids_page function are added to a complete list, facilitating the collection of unique PubMed IDs across all the URLS provided to the collect_pmids function.

As PubMed only allows the first 1,000 pages on a query to be scraped, a list of thirteen URLs in the domain of data science is provided to the collect_pmids function. The queries used in this list are: "data science", "artificial intelligence", "statistics", "machine learning", "data statistics", "probability theory", "deep learning", "information science", "computer science", "data mining", "data engineering", "data visualisation" and "business intelligence". The list of unique PubMed IDs is then saved using the pickle library [Van Rossum, 2020] to enable reloading it at a later moment.

The third function, collect_info, uses the list of unique PubMed IDs to extract the title and abstract of a scientific paper. The unique PubMed IDs are concatenated to "https://pubmed.ncbi.nlm.nih.gov/". This combination results in a URL to a paper specific page on PubMed. This page contains the title of a paper and the abstract. The URL is passed to the Requests library [Reitz, n.d.] to extract the HTML code from a page. Using BeautifulSoup [Richardson, 2007] the HTML code can be parsed.

The extraction of the title and abstract is performed by utilising the HTML ID tag, a unique identifier assigned to specific elements on a HTML page. This ID tag allows for the identification and extraction of a specific element from a webpage. Using the HTML div-tag in combination with "abstract", the abstract can be extracted. The title of a paper can be retrieved using "soup.select(h1.heading-title)", here the title is retrieved using a text element in a h1 container with the class "heading-title".

3.2 Abstract generation

By using the PubMed scraper, this research has obtained a collection of 12,499 paper titles and its corresponding abstracts. The paper titles are going to be used as input for ChatGPT in order to generate new abstracts.

Research by H. Else [Else, 2023] has shown that ChatGPT is capable of writing believable abstracts. In her research she found that abstracts generated by ChatGPT passed a plagiarism check with a median score of 100%. This indicates that none of the abstracts generated by ChatGPT was considered to contain plagiarism. The abstracts were also posed to an AI output detector which was able to correctly identify 66% of the generated abstracts. Humans did not perform better when classifying ChatGPT-generated abstracts and humanly-authored abstracts. Humans were able to correctly identify 68% of the generated abstracts as generated while classifying the other 32% as humanly-authored. Humans did perform better on classifying humanly-authored abstracts, correctly classifying 86% of the humanly-authored abstracts as humanly-authored and 14% as ChatGPT-generated.

However, prior to generating new abstracts, the existing abstracts are subjected to a filtering process. This was done in order to ensure that both the ChatGPT-generated and humanly-authored abstracts were in the same format.

The decision to filter the abstracts was made in order to avoid potential biases that could arise from variations in abstract length. If the length of the abstracts from humanly-authored text and ChatGPT-generated text would greatly differ, it could serve as an indicator of which abstracts were humanly-authored or ChatGPT-generated. Filtering the abstracts before generating new abstracts, papers of which the abstracts did not conform to the preferred format could already be filtered out, leading to a reduction in the number of abstracts that needed to be generated.

In accordance with the guidelines outlined by Nagda [Nagda, 2013], an informative abstract should contain approximately 200 words. To ensure that only informative abstracts are used as input for the classification algorithm, this research uses a 15% margin and only considers abstracts that have a length of 170 to 230 words.

In addition to the length-based filtering, this research also addresses the language in which the abstract is written. This research will use abstracts that are all written and generated in English. Therefore, to exclude abstracts that are not written in English, this research uses langdetect [Danilak, 2021]. This Python library is used to determine the language in which an abstract is written. Papers of which the abstract is not written in English are filtered out. By applying both length and language based filtering, this research aims to create a collection of humanly-authored and ChatGPT-generated texts in the same format. This leads to a collection of 3,648 paper titles which can be used for the generation of abstracts by ChatGPT.

The OpenAI API [Brockman et al., 2020] is available for use through a paid API key. The API can be used in combination with a variety of different models [OpenAI, n.d.]. The most recent and advanced version, GPT-4, is currently only available in a limited beta version. Access to this version is restricted to those who have been granted permission to use it. Therefore, this research will use the GPT-3.5-turbo model, this is also the version which is currently used by ChatGPT [OpenAI, 2022b, *at least until 16.04.2023*]. According to OpenAI [OpenAI, n.d.], this is "the most capable and cost effective model in the GPT-3.5 family" which is the latest version that is publicly available.

Using the GPT-3.5-turbo model of the API, abstracts for a list of titles from scientific papers are generated. To achieve this, the "openai.ChatCompletion.create" command is used. This command allows the creation of text based on the message provided to the API. The role is specified as "user" and the content of the request is filled using a for loop to create an abstract for each paper title. This research had posed different prompts to ChatGPT, this included the prompts "write an abstract for a paper on the title: ***title***" and "write an abstract for a scientific paper on the title: ***title***". It was found that the more detail provided in the prompt, the better ChatGPT was able to generate an abstract in the preferred format. Ultimately the prompt was formulated as "write an abstract for a scientific paper of 170 to 230 words on the title: ***title***". This prompt was used as the API content where ***title*** was equal to a title in the filtered list of paper titles extracted by the PubMed scraper.

Given that an abstract is a required element in a scientific article and it is specified that the text should be for a scientific paper, ChatGPT will generate a text using a scientific writing style. Therefore, the style and tone of the abstracts generated by ChatGPT will be similar to the style and tone of the humanly-authored abstracts. The generated abstracts are appended to an empty list which is transformed into a dataframe containing the paper title and its generated abstract.

The generation of abstracts has resulted in a collection of 3,648 humanly-authored and ChatGPT-generated texts which can be used for classification purposes. As the presence of shorter abstracts may introduce a length bias into the classification process, generated abstracts were again subjected to filtering. Despite instructing the OpenAI API to generate abstracts ranging between 170 to 230 words, a filter is once again applied to determine whether the generated abstracts are within this length. Filtering the generated abstracts to be of a length in between 170 and 230 words, this leads to a total of 2,744 of the generated abstracts to be retained. These filtered generated abstracts' titles are used to filter the human-authored abstracts, resulting in an equal number, 2,744 each, of humanly-authored and ChatGPT-generated abstracts. Each title now has one abstract that is humanly-authored and one abstract that is generated by ChatGPT.

3.3 Data exploration

Zipf's and Heaps' law [Zipf, 1999, Heaps, 1978] show the distribution of word frequency in a text corpus and can be used to provide insight into the complexity of a text corpus. In this research, both the humanly-authored and ChatGPT-generated corpus will be subjected to Zipf's and Heaps' law. By visualising both corpora, the aim is to identify differences in their distributions. This will provide insight into how a text corpus containing texts generated by ChatGPT compares to a text corpus consisting of humanly-authored texts.

Both the humanly-authored and ChatGPT-generated corpus will be analysed using Zipf's law [Zipf, 1999]. The text corpora will be visualised in three different ways. First, a bar chart depicting the top 20 most commonly occurring words, including stop words, will be constructed. As stop words do not carry any meaning, these will be neglected for the second visualisation. The second visualisation will show the word frequency distribution considering the top 20 most frequent words excluding stop words. Third, a line plot will visualise the frequency distribution of the whole vocabulary. This plot will show the rank, according to the frequency of occurrence and its frequency.

Next to that, both corpora will be analysed using Heaps' law [Heaps, 1978]. To visualise the corpora, the size of each corpus is increased by adding one token from the text corpus at a time and the number of unique words at that time will be determined. Doing so, the growth and richness of the vocabulary in both the humanly-authored and ChatGPT-generated corpus can be assessed. According to Gehrmann et al. [Gehrmann et al., 2019], text generators use a subset of the natural language which is used as the foundation for the GLTR algorithm developed by Gehrmann et al.

In addition to analysing the frequency distribution, the growth and vocabulary richness of both corpora, the differences in scores on the features will be investigated. For each feature, the mean, standard deviation and histograms of the scores will be examined for both corpora. To assess whether a significance difference exists between both corpora a t-test [Student, 1908] will be employed if the scores on a feature exhibit a normal distribution. Alternatively, if the assumption of normality is violated, the Mann-Whitney U test [McKnight and Najab, 2010], which does not assume normality, will be used. The resulting p-value will be compared against an alpha level of 0.05 to determine statistical significance. This could already provide insight into how humanly-authored and ChatGPT-generated texts differ from each other.

3.4 Classification and feature importance

In this research the XGBoost classifier [Chen and Guestrin, 2016] will be used to classify humanly-authored and ChatGPT-generated texts. To determine which features are most effective in distinguishing humanly-authored and ChatGPT-generated texts, three different XGBoost models will be build. The first model will be build on the Doc2Vec vector embeddings as discussed in Section 2.3.2. The second model will be build on text-extracted features, discussed in Sections 2.3.3, 2.3.4 and 2.3.5, derived from the abstracts. The last model will combine both methods to investigate the added value of the different features.

XGBoost [Chen and Guestrin, 2016] has gained popularity in the data science community due to its ability to achieve state-of-the-art results and its scalability and efficiency. XGBoost was used in 17 of 29 winning solutions in a Kaggle competition underwriting the success of using XGBoost in a classification task. Of those 17 solutions, eight solely used XGBoost while the others also incorporated neural networks, as is also done in this research using Doc2Vec.

XGBoost [Chen and Guestrin, 2016] is a classification algorithm which uses gradient boosted decision trees. As opposed to traditional machine learning algorithms, where a single model is trained on the data and used for prediction, XGBoost is an ensemble of iteratively trained models. In each iteration, a new model is trained to correct for errors of the previous models. The boosting algorithm creates new models and sequentially combines their predictions to improve the overall performance of the model. Sub-models are assigned weights based on their performance, models that have a higher contribution to the overall improvement of predictions are given a higher weight in the final ensemble model. To optimise the model, the gradient descent algorithm is used. This is an optimisation method that updates the weights of the model by computing the gradient of the loss with respect to each weight and updating the weights in the opposite direction of the gradient.

As the XGBoost classifier is incapable of processing textual data, whether a text is written or generated is encoded. Written text is encoded as 1 while generated text is encoded as 0. Subsequently, the encoded data is partitioned into training and testing sets, whereby 70% of the data is assigned for training, amounting to 3,841 abstracts. Of this, 1,902 abstracts are humanly-authored while the remaining 1,939 are ChatGPT-generated.

As the dataset exhibits an equal distribution comprising of 2,744 humanly-authored and 2,744 ChatGPT-generated abstracts, this research has chosen to adopt the accuracy as most important performance metric. This allows to evaluate the predictive capabilities of the model in correctly classifying abstracts as either humanly-authored or ChatGPT-generated. To give a more comprehensive insight into the models performance, this research will construct confusion matrices.

To investigate which features are most important this research uses the SHAP [Lundberg and Lee, 2017] library. Using the SHAP library, this research will analyse the contribution of each feature in the XGBoost model build on text-extracted features and the model build on vector embeddings and text-extracted features. Moreover, this research will evaluate which group of features, the vector embeddings or the text-extracted features, is more important. To do this, the XGBoost classifier constructed on both the vector embeddings and text-extracted features will be analysed. By summing the mean absolute SHAP values of this model for each feature group, it can be analysed which group of features has a higher contribution towards the model's output.

Additionally, the contribution of the text-extracted features to the model constructed using both vector embeddings and the text-extracted features will be examined using a leave-one-out approach. The importance of each feature within the model is assessed by excluding it from the model and subsequently measuring the resulting accuracy. The magnitude of decrease in accuracy serves as an indicator of the importance of each feature, with the most most important features expected to exhibit the highest decrease in accuracy when omitted from the model.

3.5 Features

3.5.1 Vector embeddings

To prepare the abstracts for vectorisation, using Doc2Vec, the content is lowercased, lemmatised and tokenised. The words are lowercased because this decreases dimensionality while maintaining semantic information [Hickman et al., 2022]. This is also the reason for lemmatising the words in an abstract, lemmatising decreases the vocabulary while also keeping the meaning of a word [Schütze et al., 2008]. Finally, tokenisation is used to break the abstract into individual units. Spelling errors, stop words, abbreviations and the use of punctuation are all kept in the data as they could expose different patterns in humanly-authored and ChatGPT-generated texts.

The Doc2Vec model utilised in this research is specified by several parameters. Kenter et al. [Kenter et al., 2016] also used the Doc2Vec model in their research and state that they have used "default" parameters. In their research the number of dimensions is specified as 300, which is also used in this research. As rare terms might be effective in distinguishing between humanly-authored and ChatGPT-generated texts, this research uses a minimum count of 0. This number specifies the minimum frequency of a word in the training data to be considered in the vocabulary. Le and Mikolov [Le and Mikolov, 2014] have demonstrated that the PV-DM variant outperforms the PV-DBOW variant, hence this variation is used in this research. For the number of epochs, the iterations over the corpus, the default value of 10, as specified in the documentation of Doc2Vec [Le and Mikolov, n.d.], is chosen. To ensure deterministic results, the seed and workers parameter are to set 1.

To determine the similarity between the documents, the Doc2Vec model is trained using the whole corpus of both the written and ChatGPT-generated abstracts. For each abstract, a vector is inferred based on the trained model. These inferred vectors will then be compared to look at which documents are most similar, thus whether the content of the ChatGPT-generated abstracts is most similar to the content of the humanly-authored abstracts. For classification purposes, the Doc2Vec model will be trained on the training data using the test and training split discussed in Section 3.4.

3.5.2 Text informativeness

The boilerplate of a document is calculated by providing the lowercased, lemmatised and tokenised text corpus. The abstracts are tokenised using the "mts.sent_tok" function provided by Jiang and Srinivasan [Jiang and Srinivasan, 2023]. Additionally, the n-grams, with the default value of 4, as chosen by this research, should be included. Furthermore, the min_doc parameter should be specified to exclude n-grams with a document frequency lower than the provided value. It is advised to set this parameter to 30% of the number of documents [Jiang, 2022]. In this research, the frequency of n-grams is not of interest, therefore get_ngram is False, which causes the function to not produce a dataframe containing the n-grams and their respective frequencies.

To compute the redundancy of the abstracts, the abstracts are also lowercased, lemmatised and tokenised using "mts.sent_tok". Using these cleaned abstracts and the number of n-grams to use in the calculation, the redundancy can be calculated. The default number of n-grams is 10, which is also used in this research. To calculate the specificity, the abstracts only need to be lowercased and lemmatised before being passed to the function.

3.5.3 Text readability

To assess the readability of the abstracts, this research used the humanly-authored and ChatGPT-generated abstracts as the complete uncleaned texts needs to be used to asses the readability. Using the Textstat library [Bansal and Aggarwal, n.d.], the Flesch Reading Ease Test, Gunning Fog Index, ARI Dale Chall and the reading time in seconds can be calculated.

To assess the grammatical and syntactical errors, the Python Language Tool library [Morris, n.d.] is used. This library is a wrapper for the grammar check developed by D. Naber [Naber et al., 2003]. This research uses the English model of the Great Britain and US to determine the number of errors within a text. The errors in a text is determined by the matching errors found by the Great Britain and US version. Thus, if the tool spots ten errors using the US version and eight using the Great Britain version or the other way around, this research considers the number of errors within the text to be eight.

3.5.4 Text characteristics

To calculate the lexical diversity using TRUNAJOD [Palma et al., 2021], the abstracts are first lowercased and lemmatised before being provided to spaCy [Honnibal et al., 2020]. Since named entity recognition and text classification are not of interest, these are disabled. The spaCy load function is employed to generate a list of processed documents with corresponding tokens and Part-Of-Speech tags. The resulting list of processed documents is subsequently analysed using the TRUNAJOD library. This analysis provides the lexical diversity score for each abstract.

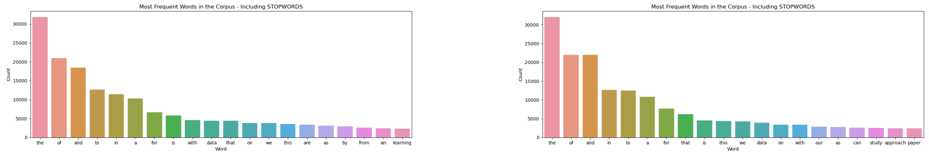
4 Results

The first subsection first describes the characteristics of both the humanly-authored and ChatGPT-generated text corpus using Zipf's and Heaps' Law. Subsequently, this research will compare the performance of these corpora on the features. The second subsection will present the classification results and show which futures are found to be more important.

4.1 Data Exploration

4.1.1 Visualising the corpus

The figure below, Figure 1, shows the frequency distribution of the 20 most frequent words in both the humanly-authored and ChatGPT-generated text corpus. It can be observed in Figure 1a that although the first word may not exhibit a frequency that is twice that of the second most frequent word and three times that of the third most frequent word, the frequency of the words does decline. Conversely, the frequency distribution for the ChatGPT-generated text corpus, as illustrated in Figure 1b, depicts a less steep decline in frequencies. The figure shows that the frequencies of the second and third most frequent word are almost equal.

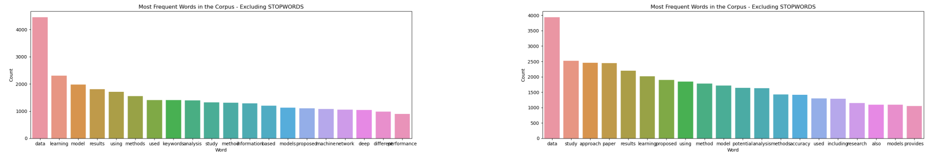


(a) Humanly-authored text corpus.

(b) ChatGPT-generated text corpus.

Figure 1: Comparison of the frequency distribution of the 20 most frequent words of the humanly-authored and ChatGPT-generated text corpus - Including stop words.

If the stop words are removed from the text corpus, the graphs show a more equal frequency distribution. This figure can be seen in Figure 2.

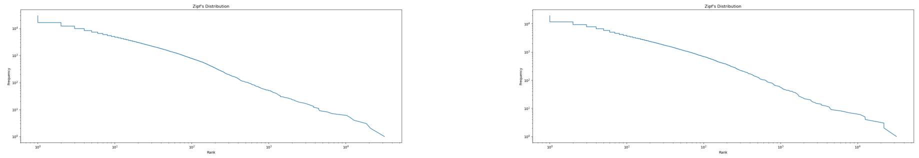


(a) Humanly-authored text corpus.

(b) ChatGPT-generated text corpus.

Figure 2: Comparison of the frequency distribution of the 20 most frequent words of the humanly-authored and ChatGPT-generated text corpus - Excluding stop words.

If the frequency distribution of the whole corpus is visualised, the graphs are similar. This can be seen seen in Figure 3.

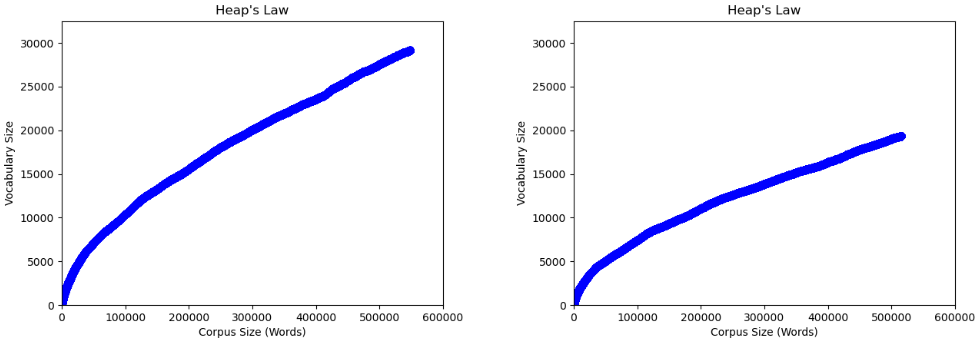


(a) Humanly-authored text corpus.

(b) ChatGPT-generated text corpus.

Figure 3: Comparison of the frequency distribution of the humanly-authored and ChatGPT-generated text corpus.

Figure 4 shows the vocabulary growth as the size of the corpus of both the humanly-authored and ChatGPT-generated text corpus increases. By comparing Figure 4a and Figure 4b, it shows that the vocabulary of the humanly-authored text corpus exceeds that of the ChatGPT-generated text corpus. The humanly-authored text corpus has a vocabulary of approximately 30,000 words while the ChatGPT-generated text corpus has a vocabulary of around 20,000 words.



(a) Humanly-authored text corpus.

(b) ChatGPT-generated text corpus.

Figure 4: Comparison of the growth of the vocabulary size with text corpus size of the humanly-authored and ChatGPT-generated text corpus.

4.1.2 Vector embeddings

By using the Doc2Vec algorithm to create vector representations of the abstracts, this research is able to compare the similarity of the humanly-authored and ChatGPT-generated abstracts on the same title. Using the parameters defined in Section 3.5.1, in 12.7% of the cases, the vector representations of the humanly-authored and ChatGPT-generated abstract on the same title are most similar.

Table 4 shows two abstracts on the same title which are considered to be most similar. In Table 5, the opposite can be seen where the humanly-authored and ChatGPT-generated abstract are not the most similar.

Written	**Generated**
Continuous intracranial pressure (ICP) monitoring is a cornerstone of neurocritical care after severe brain injuries such as traumatic brain injury and acts as a biomarker of secondary brain injury. With the rapid development of artificial intelligent (AI) approaches to data analysis, the acquisition, storage, real-time analysis, and interpretation of physiological signal data can bring insights to the field of neurocritical care bioinformatics. We review the existing literature on the quantification and analysis of the ICP waveform and present an integrated framework to incorporate signal processing tools, advanced statistical methods, and machine learning techniques in order to comprehensively understand the ICP signal and its clinical importance. Our goals were to identify the strengths and pitfalls of existing methods for data cleaning, information extraction, and application. In particular, we describe the use of ICP signal analytics to detect intracranial hypertension and to predict both short-term intracranial hypertension and long-term clinical outcome. We provide a well-organized roadmap for future researchers based on existing literature and a computational approach to clinically-relevant biomedical signal data. Keywords: data science; intracranial pressure; machine learning; prognostics and health maintenance; traumatic brain injury.	Traumatic brain injury (TBI) is a devastating condition that can lead to intracranial hypertension (ICH) and subsequent brain damage. Monitoring intracranial pressure (ICP) is an important tool for managing ICH and reducing the risk of secondary brain injury. ICP monitoring generates a complex signal that contains valuable information for clinical decision-making, yet the signal's interpretation remains challenging. In this narrative overview, we provide a comprehensive review of the current state of research on ICP monitoring signals after TBI. We focus on the advances in the understanding of signal characteristics related to TBI severity and outcome prediction. We discuss various signal processing and machine learning techniques that have been used to analyze ICP monitoring signals over time, including traditional methods and recent algorithms based on Convolutional neural networks (CNN) and Deep Learning (DL). Finally, we propose a conceptual data science framework for the analysis of ICP monitoring signals after TBI, combining signal processing and machine learning techniques. We argue that this framework can contribute to significant improvements in the management of TBI patients, and ultimately reduce the morbidity and mortality associated with this devastating condition.

Table 4: Abstracts where the humanly-authored and ChatGPT-generated abstracts on the same title are the most similar - Title: Intracranial Pressure Monitoring Signals After Traumatic Brain Injury: A Narrative Overview and Conceptual Data Science Framework

Written	**Generated**
The digitalization of the health-care system has resulted in a deluge of clinical big data and has prompted the rapid growth of data science in medicine. Data science, which is the field of study dedicated to the principled extraction of knowledge from complex data, is particularly relevant in the critical care setting. The availability of large amounts of data in the ICU, the need for better evidence-based care, and the complexity of critical illness makes the use of data science techniques and data-driven research particularly appealing to intensivists. Despite the increasing number of studies and publications in the field, thus far there have been few examples of data science projects that have resulted in successful implementations of data-driven systems in the ICU. However, given the expected growth in the field, intensivists should be familiar with the opportunities and challenges of big data and data science. The present article reviews the definitions, types of algorithms, applications, challenges, and future of big data and data science in critical care. Keywords: big data; critical care; data science; machine learning; prediction models.	The use of big data and data science in critical care has the potential to revolutionize healthcare by providing unprecedented insights into patient care and outcomes. This paper explores the potential applications of big data and data science in critical care and its impact on patient outcomes.The application of big data in critical care provides clinicians with real-time access to patient data, including clinical notes, genomic data, and imaging studies. This wealth of data can be analyzed using machine learning algorithms to identify patterns and predict patient outcomes.Data science can also be applied in critical care to develop personalized treatment plans for patients based on their unique characteristics. The availability of big data provides an opportunity to develop predictive models that can guide clinical decision-making. The use of big data and data science in critical care has not been without challenges. Privacy concerns and data governance issues are significant barriers to the implementation of this technology. Despite these obstacles, the potential benefits of big data and data science in critical care are significant, and the healthcare industry must work towards finding solutions to overcome these barriers to ensure that patients receive the best possible care.

Table 5: Abstracts where the humanly-authored and ChatGPT-generated abstracts on the same title are not the most similar - Title: Big Data and Data Science in Critical Care

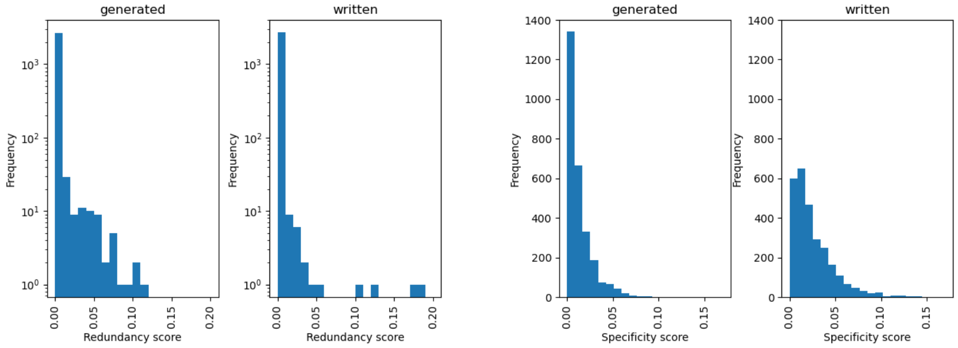
4.1.3 Text informativeness

In Table 6, the mean and standard deviation scores on the informativeness measures can be seen for both the humanly-authored and ChatGPT-generated text corpus. Here it shows that the mean boilerplate score for both is exactly 0 with a standard deviation of 0 as well. However, the mean scores on redundancy and specificity do differ. The mean redundancy score for both the humanly-authored and ChatGPT-generated abstracts is quite small. This indicates that the measured variable generally has low values. The mean redundancy score for ChatGPT-generated abstracts is slightly higher than the mean score for humanly-authored abstracts. The standard deviation of the ChatGPT-generated abstracts is also slightly larger than that of humanly-authored abstracts, indicating a greater variability. Regarding the specificity scores, the mean for humanly-authored abstracts is higher than that of the ChatGPT-generated abstracts. This indicates that humanly-authored abstracts tend to have a higher value compared to the ChatGPT-generated abstracts. The standard deviation of the humanly-authored abstracts is also bigger, indicating a wider spread.

	Generated text corpus	*Written text corpus*
Informativeness metric		
Boilerplate	0 (± 0)	0 (± 0)
Redundancy	0.001 (± 0.008)	0.000 (± 0.006)
Specificity	0.0126 (± 0.014)	0.025 (± 0.023)

Table 6: Mean and standard deviation of the informativeness metrics for the ChatGPT-generated and humanly-authored text corpus.

The analysis of the measures on informativeness for the humanly-authored and ChatGPT-generated abstracts can be extended by examining the histograms in Figure 5. Both corpora have a comparable histogram on the redundancy score while the histogram on specificity differs. Notably, the frequency of abstracts with a specificity score of 0 is considerably higher for ChatGPT-generated abstracts than for humanly-authored abstracts. As both histograms do not show that the data is normally distributed, the Mann Whitney U test is used to test for significance. As the p-value for redundancy and specificity is below 0.05, the comparisons are statistically significant. As the mean for boilerplate is exactly equal for both the humanly-authored and ChatGPT-generated text corpus there is no evidence to reject the null hypothesis. Therefore, it is concluded that there is no significant difference between the boilerplate score for the humanly-authored and ChatGPT-generated corpus.



(a) Redundancy

(b) Specificity

Figure 5: The performance of the humanly-authored and ChatGPT-generated text corpora on the informativeness measures.

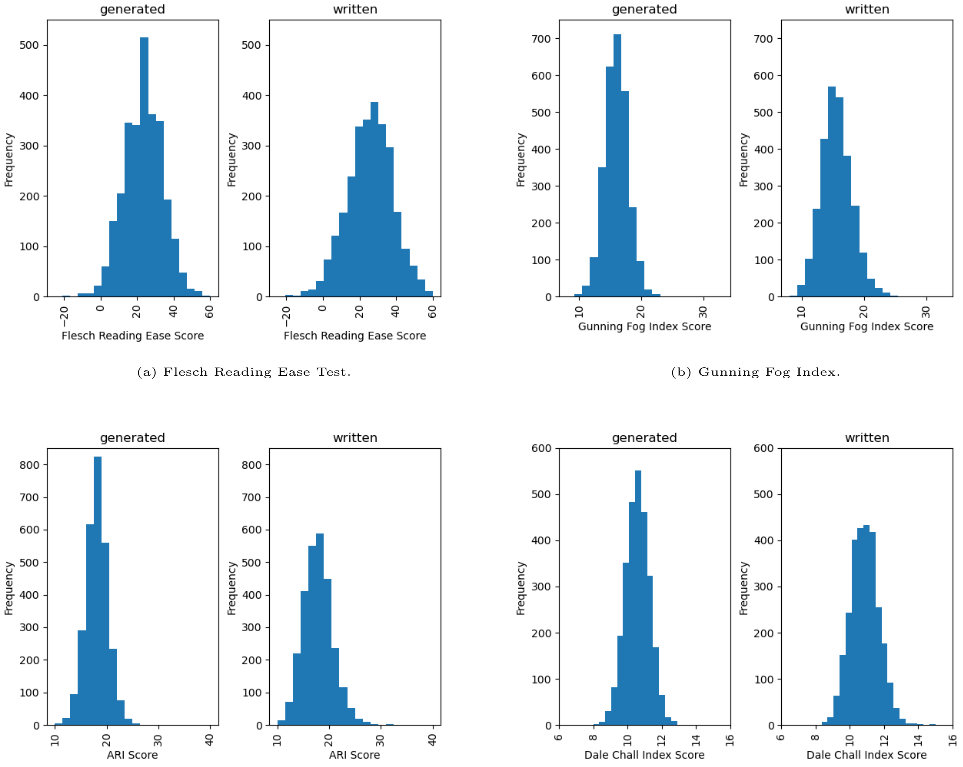
4.1.4 Text readability

In Table 7, the mean and standard deviations on readability metrics for both the humanly-authored and the ChatGPT-generated text corpus are shown. Here it can be seen that the mean scores for most of the readability test are similar, with small differences in some cases. This suggests that both the humanly-authored abstracts and ChatGPT-generated abstracts have a comparable performance on the readability tests. The standard deviations for the ChatGPT-generated texts however are lower than those of the humanly-authored abstracts. This indicates that the scores for ChatGPT-generated abstracts are more concentrated around the mean. Regarding the mean of the grammatical and syntactical errors within humanly-authored and ChatGPT-generated abstracts, it is interesting to note that on average the humanly-authored abstracts contain one grammatical error more as opposed to the abstracts generated using ChatGPT. The spread of the grammatical errors for humanly-authored abstracts is also higher compared to that of ChatGPT-generated texts, indicating a wider spread.

Readability metric	*Generated text corpus*	*Written text corpus*
Flesch Reading Ease Test	23.5 (± 10.4)	25.8 (± 12.3)
Gunning Fog Index	16.0 (± 1.9)	15.6 (± 2.5)
ARI	18.1 (± 2.1)	17.8 (± 2.9)
Dale Chall Index	10.6 (± 0.7)	10.9 (± 0.8)
Reading time (in seconds)	16.7 (± 1.5)	17.7 (± 1.9)
Grammatical errors	1.8 (± 2.7)	2.9 (± 3.3)

Table 7: Mean scores on the readability metrics for the ChatGPT-generated and humanly-authored texts.

The distribution of the readability scores on the different readability tests can be seen in Figure 6. Here it can be seen that both corpora have a comparable performance on the readability tests as the shapes of the histograms show similarities. However, the histograms for the humanly-authored corpus is more evenly distributed while that of the ChatGPT-generated corpus consists of higher peaks. As the histograms show the scores on the readability metrics are normally distributed, the t-test is used. As the p-values are all below 0.05 the comparisons are statistically significant for the readability tests.



(a) Flesch Reading Ease Test.

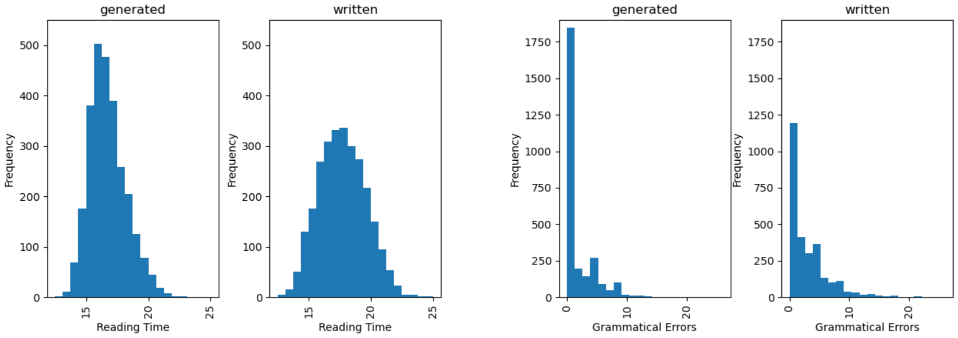
(b) Gunning Fog Index.

(c) ARI.

(d) Dale Chall Index.

Figure 6: The performance of the humanly-authored and ChatGPT-generated text corpora on the readability measures.

In Figure 7 the performance regarding the reading time and grammatical errors can be seen. The histograms in Figure 7a show that the reading time is more evenly distributed for humanly-authored abstracts. The peak of the histogram regarding the reading time of ChatGPT-generated abstracts is a bit more to the left, thus indicating that the abstracts take a little less long to read. In Figure 7b, the distribution of the grammatical errors among abstracts can be seen. Notably, several ChatGPT-generated abstracts are free of grammatical or syntactical errors. The frequency of humanly-authored abstracts without grammatical and syntactical errors is lower than that of ChatGPT-generated abstracts. Considering the t-test for the significance of reading time and the Mann Whitney U test for the number of grammatical errors it shows that the comparisons are statistically significant as the p-values on both measures are below 0.05.



(a) Reading time.

(b) Grammatical errors.

Figure 7: The performance of the humanly-authored and ChatGPT-generated text corpora on the reading time and grammatical errors.

4.1.5 Text characteristics

In Figure 4, it could be seen that the vocabulary size of the text corpus of the ChatGPT-generated abstracts is smaller than the vocabulary size of the text corpus of the humanly-authored abstracts. However, upon examining the lexical diversity of individual abstracts, no discernible differences emerge between the humanly-authored and ChatGPT-generated abstracts. Both the humanly-authored and ChatGPT-generated text corpora exhibit a mean lexical diversity score of exactly 1.0 with a standard deviation of 0, indicating that all abstracts consist of a diverse range of vocabulary. As the mean for the lexical diversity score is exactly equal for both the humanly-authored and ChatGPT-generated text corpus, it is concluded that there is no significant difference in lexical diversity between both corpora on the individual abstract level.

4.2 Classification and feature importance

In this research three different XGBoost models have been constructed. The first model is build on the Doc2Vec vector embeddings, the second on the text-extracted features and the third model combines both. The first model is able to correctly classify 83.9% of the abstracts. In the confusion matrix in Figure 8 the overall performance of the Doc2Vec model can be seen. Here it can be seen that the model classifies a humanly-authored abstract as generated in 8.3% of the cases. A ChatGPT-generated abstract is more often wrongly predicted, in 24.2% of the cases a ChatGPT-generated abstracts is wrongly classified as humanly-authored.

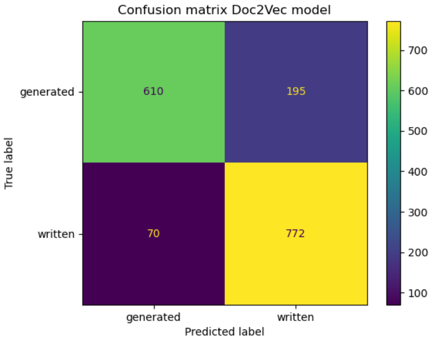


Figure 8: The performance of the Doc2Vec model on classifying humanly-authored and ChatGPT-generated abstracts.

The second model, build on text-extracted features is able to correctly classify 72.4% of the abstracts. The overall performance of this model is presented in the confusion matrix in Figure 9. The model classifies humanly-authored abstracts as ChatGPT-generated in 29.9% of the cases. This is close to the percentage of ChatGPT-generated abstracts that are inaccurately labeled as humanly-authored in 25.2% of the cases.

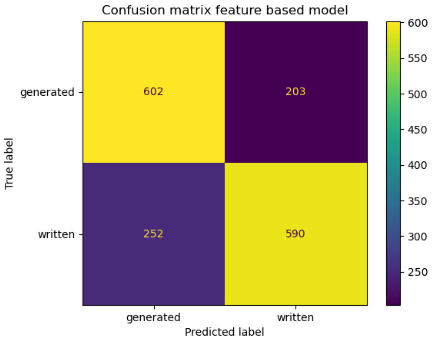


Figure 9: The performance of the text-extracted feature model on classifying humanly-authored and ChatGPT-generated abstracts.

The third model was able to achieve the best performance, the model constructed using Doc2Vec vector embeddings as well as text-extracted features was able to correctly classify 86.5% of the abstracts. The overall performance of this model can be observed in the confusion matrix presented in Figure 10. The combination of vector embeddings and text-extracted features has decreased the number of times that a humanly-authored abstracts is classified as generated, using this model this is only happens 6.7% of the time. Additionally, the number of times a ChatGPT-generated abstract is wrongly classified as humanly-authored is also decreased, using this model this happens in 20.7% of the cases.

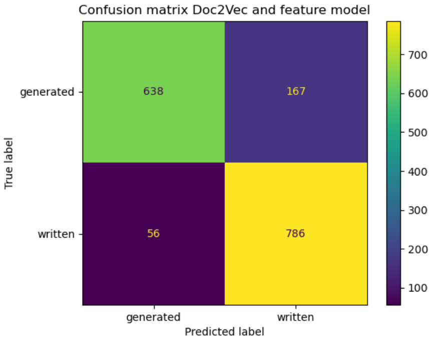
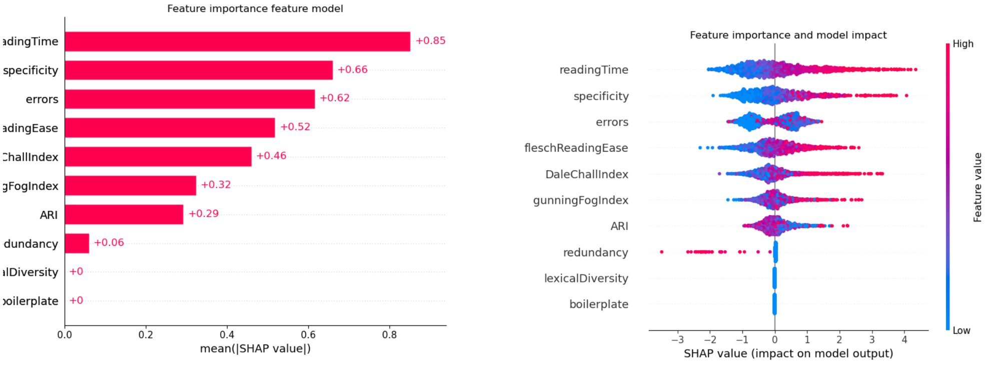


Figure 10: The performance of the Doc2Vec and text-extracted feature model on classifying humanly-authored and ChatGPT-generated abstracts.

In Figure 11, the importance of the features in the text-extracted feature model can be seen. Figure 11a shows the magnitude of feature attributions, features with a large SHAP value thus are most important. Here it can be seen that the reading time of an abstract thus is the most important predictor for assessing whether a text is humanly-authored or ChatGPT-generated. It can also be seen that the lexical diversity and boilerplate score do not contribute to the prediction while the redundancy score only contributes a little.

Figure 11b combines the feature importance with the feature effects. Each point in the plot corresponds to a specific feature's SHAP value for a particular instance. The colour of each point indicates whether the feature value for that instance is relatively low (blue), mediocre (purple) or high (red). The x-axis represents the influence of the SHAP value, indicating whether it has a positive or negative impact on the model output. In this research, a positive influence of a SHAP value suggest a higher likelihood of an abstract being humanly-authored. Conversely, a negative SHAP value indicates a higher probability of an abstract being ChatGPT-generated. Abstracts with a relatively high reading time are thus more likely to be classified as humanly-authored while abstracts with a relatively low reading time have a higher probability of being classified as ChatGPT-generated.

The plot shown in Figure 11b indicates that attaining high values across most features, with the exception for redundancy, is associated with an increased probability of being classified as humanly-authored. The redundancy feature shows to mostly have low values which do not impact the model's output. However, if the redundancy feature is observed to have a high value, the abstract is more likely to be classified as ChatGPT-generated. Mediocre values on the features do not substantially alter the probabilities of being classified as ChatGPT-generated or humanly-authored. The plot highlights that high or low values on features can influence the probability of an abstract being classified as either ChatGPT-generated or humanly-authored.



(a) The feature importance.

(b) The feature importance feature effects.

Figure 11: The feature importance of the features used in the text-extracted feature model.

By summing the mean absolute SHAP values of vector embeddings and text-extracted features enables a direct comparison between the two feature groups, revealing which group is of greater overall importance. The sum of the mean absolute SHAP values for the vector embeddings is equal to 15.9, whereas the summation for the text-extracted features amounts to 2.8. This indicates that the vector embeddings are of greater importance, being 5.7 times bigger than the summation of text-extracted features.

In Figure 12, the top 10 of the most important features in the combined feature model can be seen. Additionally, the figure provides insight into the cumulative distribution of the features that are not explicitly depicted in the plot. The analysis reveals that reading time is the most influential feature in the combined model.

Among the top 10 features, six are vectors derived from the vector embeddings, with vector 295 identified as the most important vector. Similar to the model relying solely on text-extracted features, boilerplate and lexical diversity do not contribute to the prediction task. Moreover, redundancy, which demonstrated predictive value in the text-extracted feature based model, does not contribute to the prediction task. The ARI score, Flesch Reading Ease score and Gunning Fog Index contribute minimally to the prediction task with respective contributions of 0.05, 0.05 and 0.06.

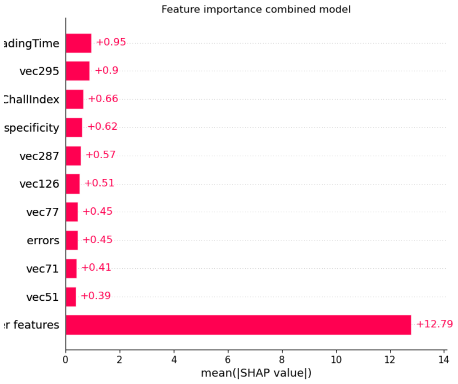


Figure 12: top 10 most important features in the combined model.

To give more insight into the feature importance, Table 8 shows the impact on accuracy when each feature is individually dropped from the combined model. The results reveal that omitting the reading time feature leads to the most significant decrease in accuracy. Conversely, dropping the Flesch Reading Ease score, redundancy score and Gunning Fog Index actually results in an increase in accuracy.

Feature	**Effect on accuracy**
Reading time	-0.017%
Dale Chall Index	-0.011%
Specificity	-0.006%
Grammatical errors	-0.006%
ARI	-0.001%
Boilerplate	0%
Lexical diversity	0%
Flesch Reading Ease	+0.001%
Redundancy	+0.004%
Gunning Fog Index	+0.004%

Table 8: The effect of dropping out a feature in the combined model on the accuracy

It is worth mentioning that the overall accuracy is only marginally affected, with small fluctuations observed when features are removed. The impact on accuracy is not substantial, indicating that the model's performance remains relatively stable in case a single text-extracted feature is excluded.

5 Discussion

The objective of this study was to identify the key features that can be used to differentiate between humanly-authored and ChatGPT-generated texts. This research first constructed two text corpora, the first comprising of humanly-authored abstracts and the second of ChatGPT-generated abstracts. To determine the most effective features, this research first explored the data. The text corpora have been visualised using Zipf's and Heaps' Law after which the vector embeddings and the performance on the text-extracted features were compared.

Visualising the humanly-authored and ChatGPT-generated corpus according to Zipfs' Law, the frequency distributions exhibit striking similarity. This implies that the frequency distributions alone do not offer a distinguishing factor between humanly-authored and ChatGPT-generated texts. The visualisation of both corpora following Heaps' Law does provide a distinguishing factor. Figure 4 shows that the vocabulary of the humanly-authored corpus is larger compared to the vocabulary of the ChatGPT-generated corpus. This indicates that the ChatGPT-generated texts are generated using a subset of the language.

The visualisations have been able to show the similarities regarding the frequency distribution and the dissimilarities in the vocabulary size on a corpus level. Using the Doc2Vec vector embeddings the abstracts on the same title can be compared on an individual level. It was shown that in 12.7% of the cases the vector representations of the humanly-authored and ChatGPT-generated abstracts on the same title were the most similar. As the abstracts are generated on paper titles within the data science domain, the outcome suggests that in most cases ChatGPT falls short in incorporating sufficient distinctive information into an abstract to make them closely resemble the content of the authentic, humanly-authored abstracts.

Regarding the performance on the text-extracted features, the humanly-authored and ChatGPT-generated corpus show similar performances on the boilerplate, redundancy, Flesch Reading Ease Test, Gunning Fog Index, ARI, Dale Chall Index and lexical diversity. The mean, standard deviations and histograms on the measures are comparable for both text corpora indicating that the features will not have discriminative power. Although Heaps' Law showed a difference in vocabulary size on a corpus level, the lexical diversity of a single abstract thus does not seem to provide discriminative power.

However, the performance on the specificity, reading time and grammatical errors features is different for both corpora. The mean, standard deviations and histograms for these features do differ and therefore possibly do seem to provide discriminative power. When testing for significance it is found that the comparisons are statistically significant for all features except for boilerplate and lexical diversity. This indicates that the statistically significant features possibly can be used to distinguish humanly-authored and ChatGPT-generated text.

Second, this research constructed three different XGBoost classifiers to assess feature importance. The first classifier was built using Doc2Vec vector embeddings, the second using text-extracted features and the third combining both. The classifiers were able to correctly predict whether a text was humanly-authored or ChatGPT-generated with an accuracy of 83.9%, 72.4% and 86.5% respectively. These results show that a model based on Doc2Vec vector embeddings outperforms one based solely on text-extracted features. Furthermore, the combined model exhibits an accuracy improvement of 2.6% compared to the Doc2Vec based classifier and 14.1% compared to the text-extracted feature model.

Individual feature importance analysis in the combined model reveals that vector embeddings are considered more important than the text-extracted features. The summation of the mean SHAP values indicate that the vector embeddings are 5.7 times more important than text-extracted features in this model. This observation aligns with the superior performance of the model constructed using Doc2Vec vector embeddings compared to the text-extracted feature model. Within the top 10 most important features in the combined model, six are vector embeddings, while only four are text-extracted features.

Upon inspecting the feature importance on an individual level, it shows that for both the text-extracted feature model and the combined model, based on the SHAP values, reading time is the most important feature. In the text-extracted feature model, specificity, grammatical errors, Flesch Reading Ease, Dale Chall Index, Gunning Fog Index and ARI follow reading time in importance. Additionally, redundancy slightly contributes to the prediction task in this model, but at a lesser extent.

Conversely to the text-extracted model, in the combined model, the Dale Chall Index emerges as the second most important text-extracted feature, followed by specificity and grammatical errors. In this case, the ARI, Flesch Reading Ease and Gunning Fog Index have a minimal influence on the prediction task. Notably, while redundancy contributes to the prediction task in the text-extracted feature model, it does not contribute in the combined model.

It is interesting to note that although the data exploration showed a comparable performance for both text corpora on the Dale Chall Index, the feature provides discriminative power in both the text-extracted feature and combined model. The Flesch Reading Ease Test, Gunning Fog Index and ARI do provide discriminative power in the text-extracted feature model but only contribute slightly in the combined model. Redundancy only contributed slightly to the prediction task in the text-extracted feature model while not contributing to the prediction in the combined model. As the mean and standard deviation for the lexical diversity and boilerplate was identical for both corpora it was anticipated that both features would not at all contribute to the prediction task in both models.

The leave one feature out approach confirms the order of importance of text-extracted features in the combined model. Using this approach, it shows that reading time causes the biggest decrease in accuracy. As lexical diversity and boilerplate do not contribute to the prediction task, omitting these features from the model does not affect the accuracy. Moreover it is interesting to see that omitting the Flesch Reading Ease, redundancy and Gunning Fog Index actually increases the accuracy. Furthermore, it is worth noting that dropping out one text-extracted feature only effects the accuracy by a small margin, substantiating the importance of the Doc2Vec vector embeddings in the combined model.

Notably, the model using both Doc2Vec vector embeddings and text-extracted features has an improved accuracy as compared to the GLTR [Gehrmann et al., 2019], 86.5% and 72% respectively. The model is also better in correctly classifying AI-generated texts as compared to the OpenAI classifier. The model built in this research correctly classifies 79.3% of the cases as opposed to the 26% achieved by OpenAI [Kirchner et al., 2023]. Furthermore, it is less likely to classify humanly-authored texts as ChatGPT-generated, only classifying 6.7% as generated in contrast to the 9% of the OpenAI classifier. GPTZero however, is able to outperform the model built in this research [Tian, 2022].

GPTZero uses perplexity, burstiness and other variables to determine whether a text is AI-generated or humanly-authored [Tian, 2023]. The features used in their model thus appear to have more predictive power as opposed to the features used in this research. It is however worth noting that the performance of GPTZero improves when the size of the input increases, as is the case for the OpenAI classifier. As the size of the text for the test data in their results is unknown, it is thus difficult to compare the results.

As the model built in this research is trained on text consisting of in between 170 and 230 words, it is ill-advised to use this model in the educational setting as of now, as the lengths of texts might differ. Moreover, as the GPTZero model is able to outperform the model built in this research, it would still be advised to use this model. However, further research could improve the model built in this research. This research has shown that solely using Doc2Vec vector embeddings already results in achieving a high accuracy in classifying humanly-authored and ChatGPT-generated text. Furthermore, adding features as the reading time, Dale Chall Index, specificity and grammatical errors in the combined model adds predictive power to the model. For those features it is found in the text-extracted feature model that whenever the reading time is longer, the text is more specific or more difficult to comprehend, it is more likely to be humanly authored.

For further research and expanding upon the model built in this research it would be interesting to train the algorithm on different text lengths and add normalisation for the length of a text. According to GPTZero [Tian, 2022] and the OpenAI classifier [Kirchner et al., 2023], the accuracy of detecting AI-generated texts increases if the input is longer. Therefore adjusting the model to account for different lengths of texts, and therefore adding the need to normalise the features by the length, could improve the accuracy of the classifier. It is also mentioned above that this would be necessary before the model could be incorporated within the educational setting.

Next to the limitation of being trained on texts in between 170 and 230 words, a different limitation that is also found by GPTZero [Tian, 2023] and the OpenAI classifier [Kirchner et al., 2023], is that AI is ever evolving. The model built in this research is trained on detecting ChatGPT-generated text based on the GPT-turbo 3.5 model while the next model is already available in beta use. Therefore, to be up to date, the model needs to be retrained every time a new version of ChatGPT is released.

For further research it would also be interesting to experiment with different Doc2Vec parameters. In this research, the importance of the feature groups, the vector embeddings and the text-extract features, in the combined model is directly compared. However, as this research uses 300 vectors in comparison to ten text-extracted features this is a skewed comparison. Therefore, it would be interesting to do this comparison once again when the number of vectors and features is equal.

To add to this, it would be interesting to see, that while this research has elaborated on the use of Doc2Vec vector embeddings, whether different vector embeddings would be able to achieve a better performance. Furthermore, a different set of features can be incorporated into the model. It would be interesting to see whether the other features mentioned, or newly found features, would be able to add more predictive power to the model.

In summary, this research successfully identified key features for differentiating humanly-authored and ChatGPT-generated texts. The findings underscored the superiority of models incorporating Doc2Vec vector embeddings. While the comparisons on all features, except for boilerplate and lexical diversity, were statistically significant reading time was seen as the most influential future by performing SHAP analysis. The analysis also highlighted the limited contribution of certain text-extracted features, as the lexical diversity and boilerplate did not contribute to the prediction task. As the model build in this research is outperformed by the GPTZero algorithm, educational institutions are not advised to use the model build in this research to assess whether a text is humanly-authored or ChatGPT-generated. However, further research on the model, including training on newer versions of ChatGPT, different feature sets, larger texts and normalisation for text length, could improve the model to make it more suitable to be used within the educational environment.

References

Avinash, M., & Sivasankar, E. (2019). A study of feature extraction techniques for sentiment analysis. *Emerging Technologies in Data Mining and Information Security: Proceedings of IEMIS 2018, Volume 3*, 475–486.

Bansal, S., & Aggarwal, C. (n.d.). *Documentation textstat* [Accessed on 23-04-2023]. https://pypi.org/project/textstat/

Basarkar, A. (2017). Document classification using machine learning.

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with python: Analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, *3*(Jan), 993–1022.

Brockman, G., Murati, M., Welinder, P., & OpenAI. (2020). *Openai api* [Accessed on 15.04.2023]. https://openai.com/blog/openai-api

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785–794.

Coleman, M., & Liau, T. L. (1975). A computer readability formula designed for machine scoring. *Journal of Applied Psychology*, *60*(2), 283.

Dale, E., & Chall, J. S. (1948). A formula for predicting readability. *Educational Research Bulletin*, *27*(1), 11–28. Retrieved April 10, 2023, from http://www.jstor.org/stable/1473169

Damashek, M. (1995). Gauging similarity with n-grams: Language-independent categorization of text. *Science*, *267*(5199), 843–848.

Danilak, M. M. (2021). *Langdetect* [Accessed on 16.04.2023]. https://github.com/Mimino666/langdetect

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, *41*(6), 391–407.

Demberg, V., & Keller, F. (2008). Data from eye-tracking corpora as evidence for theories of syntactic processing complexity. *Cognition*, *109*(2), 193–210.

Else, H. (2023). Abstracts written by chatgpt fool scientists. *Nature*, *613*(7944), 423–423.

Flesch, R. (1979). How to write plain english. *University of Canterbury. Available at http://www. mang. canterbury. ac. nz/writing_guide/writing/flesch. shtml.[Retrieved 5 February 2016]*.

Gehrmann, S., Strobelt, H., & Rush, A. (2019). GLTR: Statistical detection and visualization of generated text. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 111–116. https://doi.org/10.18653/v1/P19-3019

Gharehchopogh, F. S., & Khalifelu, Z. A. (2011). Analysis and evaluation of unstructured data: Text mining versus natural language processing. *2011 5th International Conference on Application of Information and Communication Technologies (AICT)*, 1–4.

Grant, N. (2023). *Google calls in help from larry page and sergey brin for a.i. fight* [Accessed on 22.02.2023]. https://www.nytimes.com/2023/01/20/technology/google-chatgpt-artificial-intelligence.html

Harris, Z. S. (1954). Distributional structure. *Word*, *10*(2-3), 146–162.

Heaps, H. S. (1978). *Information retrieval, computational and theoretical aspects*. Academic Press.

Hickman, L., Thapa, S., Tay, L., Cao, M., & Srinivasan, P. (2022). Text preprocessing for text mining in organizational research: Review and recommendations. *Organizational Research Methods*, *25*(1), 114–146.

Honnibal, M., Montani, I., Van Landeghem, S., & Boyd, A. (2020). Spacy: Industrial-strength natural language processing in python.

Jiang, J. (2022). *Morethansentiments: Text analysis package* [Accessed on 1.05.2023]. https://towardsdatascience.com/morethansentiments-a-python-library-for-text-quantification-e57ff9d51cd5

Jiang, J., & Srinivasan, K. (2023). Morethansentiments: A text analysis package. *Software Impacts*, *15*, 100456.

Jurafsky, D., & Martin, J. H. (2023). *Speech and language processing* (3. edition draft, [Pearson International Edition]) [Accessed on 15.04.2023]. https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf

Kaushik, H. M., Eika, E., & Sandnes, F. E. (2020). Towards universal accessibility on the web: Do grammar checking tools improve text readability? *Universal Access in Human-Computer Interaction. Design Approaches and Supporting Technologies: 14th International Conference, UAHCI 2020, Held as Part of the 22nd HCI International Conference, HCII 2020, Copenhagen, Denmark, July 19–24, 2020, Proceedings, Part I 22*, 272–288.

Kenter, T., Borisov, A., & De Rijke, M. (2016). Siamese cbow: Optimizing word embeddings for sentence representations. *arXiv preprint arXiv:1606.04640*.

Kim, D., Seo, D., Cho, S., & Kang, P. (2019). Multi-co-training for document classification using various document representations: Tf–idf, lda, and doc2vec. *Information Sciences*, *477*, 15–29.

Kincaid, J. P., Fishburne Jr, R. P., Rogers, R. L., & Chissom, B. S. (1975). *Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel* (tech. rep.). Naval Technical Training Command Millington TN Research Branch.

Kirchner, J. H., Ahmad, L., Aaronson, S., & Leike, J. (2023). *New ai classifier for indicating ai-written text* [Accessed on 22.03.2023]. https://openai.com/blog/new-ai-classifier-for-indicating-ai-written-text

Le, Q., & Mikolov, T. (n.d.). *Models.doc2vec - doc2vec paragraph embeddings* [Accessed on 2.05.2023]. https://radimrehurek.com/gensim/models/doc2vec.html

Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. *International conference on machine learning*, 1188–1196.

Lippi, M., Montemurro, M. A., Degli Esposti, M., & Cristadoro, G. (2019). Natural language statistical features of lstm-generated texts. *IEEE Transactions on Neural Networks and Learning Systems*, *30*(11), 3326–3337.

Lloret, E., & Palomar, M. (2013). Tackling redundancy in text summarization through different levels of language analysis. *Computer Standards & Interfaces*, *35*(5), 507–518.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems 30* (pp. 4765–4774). Curran Associates, Inc. http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

Mc Laughlin, G. H. (1969). Smog grading-a new readability formula. *Journal of reading*, *12*(8), 639–646.

McAlpine, R. (2012). From plain english to global english.

McKnight, P. E., & Najab, J. (2010). Mann-whitney u test. *The Corsini encyclopedia of psychology*, 1–1.

Mhlanga, D. (2023). Open ai in education, the responsible and ethical use of chatgpt towards lifelong learning. *Education, the Responsible and Ethical Use of ChatGPT Towards Lifelong Learning (February 11, 2023)*.

Morris, J. (n.d.). *Documentation language tool pyton* [Accessed on 23-04-2023]. https://pypi.org/project/language-tool-python/

Naber, D., et al. (2003). A rule-based style and grammar checker.

Nagda, S. (2013). How to write a scientific abstract. *The Journal of Indian Prosthodontic Society*, *13*, 382–383.

O'hayre, J. (1966). *Gobbledygook has gotta go*. US Department of the Interior, Bureau of Land Management.

OpenAI. (n.d.). *Models - openai api* [Accessed on 16.04.2023]. https://platform.openai.com/docs/models

OpenAI. (2022a). *Dall-e 2: Extending creativity* [Accessed on 17.03.2023]. https://openai.com/blog/dall-e-2-extending-creativity

OpenAI. (2022b). *Introducing chatgpt* [Accessed on 17.03.2023]. https://openai.com/blog/chatgpt

Palma, D. A., Soto, C., Veliz, M., Karelovic, B., & Riffo, B. (2021). Trunajod: A text complexity library to enhance natural language processing. *Journal of Open Source Software*, *6*(60), 3153. https://doi.org/10.21105/joss.03153

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, *14*(3), 130–137.

Pubmed. (n.d.). https://pubmed.ncbi.nlm.nih.gov/

Rawassizadeh, R., Sen, T., Kim, S. J., Meurisch, C., Keshavarz, H., Mühlhäuser, M., & Pazzani, M. (2019). Manifestation of virtual assistants and robots into daily life: Vision and challenges. *CCF Transactions on Pervasive Computing and Interaction*, *1*, 163–174.

Reitz, K. (n.d.). *Requests: Http for humans.* https://requests.readthedocs.io/en/latest/

Richardson, L. (2007). Beautiful soup documentation. *April.*

Robert, G. (1952). The technique of clear writing. *New York.*

Schütze, H., Manning, C. D., & Raghavan, P. (2008). *Introduction to information retrieval* (Vol. 39). Cambridge University Press Cambridge.

Senter, R., & Smith, E. A. (1967). *Automated readability index* (tech. rep.). Cincinnati Univ OH.

Smith, E. A., & Kincaid, J. P. (1970). Derivation and validation of the automated readability index for use with technical materials. *Human factors*, *12*(5), 457–564.

Spache, G. (1953). A new readability formula for primary-grade reading materials. *The Elementary School Journal*, *53*(7), 410–413.

Student. (1908). The probable error of a mean. *Biometrika*, *6*(1), 1–25.

Tian, E. (2022). *Gptzero faq* [Accessed on 20.03.2023]. https://gptzero.me/faq

Tian, E. (2023). *Gptzero update v1* [Accessed on 19.03.2023]. https://gptzero.substack.com/p/gptzero-update-v1

Van Rossum, G. (2020). *The python library reference, release 3.8.2.* Python Software Foundation.

Zamanian, M., & Heydari, P. (2012). Readability of texts: State of the art. *Theory & Practice in Language Studies*, *2*(1).

Zhuang, H., & Zhang, W. (2019). Generating semantically similar and human-readable summaries with generative adversarial networks. *IEEE Access*, *7*, 169426–169433.

Zipf, G. K. (1999). *The psycho-biology of language: An introduction to dynamic philology* (Vol. 21). Psychology Press.

A Code

GitHub repository: https://github.com/rijndersmike/ThesisProject