Nicolai Thorer Sivesind
Andreas Bentzen Winje

Turning Poachers into Gamekeepers: Detecting Machine-Generated Text in Academia Using Large Language Models

Bachelor's thesis in Computer Science
Supervisor: Ole Christian Eidheim
May 2023

Bachelor's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Nicolai Thorer Sivesind
Andreas Bentzen Winje

Turning Poachers into Gamekeepers: Detecting Machine-Generated Text in Academia Using Large Language Models

Bachelor’s thesis in Computer Science
Supervisor: Ole Christian Eidheim
May 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU

Norwegian University of Science and Technology

Department of Computer Science

Bachelor's Thesis

Turning Poachers into Gamekeepers: Detecting Machine-Generated Text in Academia Using Large Language Models

Authored by:

Nicolai Thorer Sivesind & Andreas Bentzen Winje

22nd May 2023

Supervised by Assoc. Prof. Ole Christian Eidheim

Abstract

Following the introduction of generative large language models in society, concerns have been raised regarding their potential for misuse in academia, and the threat they pose to the standards of academic integrity. In this thesis, we explore the emergent problem domain of detecting machine-generated text in academia. We focus our efforts on three sub-problems of the problem domain: producing suitable data for training and evaluation detection models, the development of two distinct approaches for detecting machine-generated text in academic work using large language models, and finally, a discussion of the social and ethical aspects of applying such detection tools in academia. We produce the dataset using human-produced research abstracts and prompt GPT-3.5, the model used in ChatGPT, to produce machine-generated counterparts. For detecting machine-generated text in academia, we propose two distinct approaches, *in-context learning* and *fine-tuned binary sequence classification* which both are novel in scope of the problem domain. Our *in-context learning* approaches display poor performance indicating that the problem currently is too complex for application using current open-access models. In contrast, our *fine-tuned binary sequence classification* approaches perform well. The best models achieve an accuracy above 98% on in-domain data generated by ChatGPT. Although these results are promising, testing on out-domain data, such as cross-architecture and cross-corpus data, displays weaker performances, entailing further research to be done in this field. The thesis is concluded with a discussion of the social and ethical aspects of applying large language models in academia. It is important that a discussion of how we adjust to the disruptions of generative large language models is based on a scientific foundation led by performance evaluations of proposed detection tools, and the social and ethical implications of their widespread application. It's through transparency from both sides of the problem that we can reap the benefits of large language models while maintaining academic integrity.

Sammendrag

Introduksjonen av store generative språkmodeller har nylig ført til store endringer i samfunnet. Innen for academia, blir det gitt uttrykk for bekymringer knyttet til deres potensial for misbruk i akademiske kontekster dermed også trusselen de utgjør mot akademisk integritet. I denne bacheloroppgaven utforsker vi det voksende behovet for det å *kunne oppdage maskingenerert tekst i akademia*. Vi vektlegger tre underproblemer i hovedproblemstillingen: produksjon av et egnet datasett for trening og evaluering av deteksjonsmodeller, utvikling av to distinkte metoder for å oppdage maskinprodusert tekst i akademisk arbeid og til slutt en diskusjon om de sosiale og etiske aspektene ved å bruke deteksjonsverktøy i akademia. Denne oppgaven viser hvordan et egnet datasett kan produseres ved å bruke menneskeproduserte sammendrag fra forskningsartikler og få GPT-3.5, modellen som brukes i ChatGPT, til å produsere maskingenererte dobbeltgjengere av de menneskeproduserte sammendragene. For å oppdage maskinprodusert tekst i akademia foreslår vi to distinkte metoder, *in-context learning* og *fine-tuned binary sequence classification*, som tidligere ikke er dokumentert i anvendelse for problemstillingen. Våre *in-context learning* metoder viser svake resulter, noe som indikerer at problemet for øyeblikket er for komplekst for nåværende open-source språkmodeller. I motsetning til dette, demonstrerer *fine-tuned binary sequence classification* metodene våre gode resultater. Våre beste modeller oppnår en nøyaktighet over 98% på in-domain data generert av ChatGPT. Selv om disse resultatene er lovende, viser tester fra out-domain data, som kryss-arkitektur og kryss-korpus data, langt svakere resultater, noe som innebærer at det trengs mer forskning på dette feltet, men som samtidig understreker viktigheten av å trene modeller på in-domain dataset. Oppgaven avsluttes med en diskusjon av de sosiale og etiske aspektene ved bruk av store språkmodeller i akademia. Det er viktig at en diskusjon om hvordan vi tilpasser oss endringene språkmodeller påfører samfunnet, er basert på bakgrunn av metodisk evaluering av de potensielle deteksjonsverktøyene, og de sosiale og etiske implikasjonene av deres utbredte anvendelse. Det er gjennom åpenhet fra begge sider av problemstillingen at vi kan dra nytte av fordelene store språkmodeller tilbyr, samtidig som vi opprettholder standardene våres for akademisk integritet.

Preface

This is a bachelor thesis developed as the final evaluation of the bachelor program *Computer Science* at the Norwegian University of Science and Technology, and the problem defined in this thesis is set based on personal motivation and interest in the field of artificial intelligence. Writing this thesis bachelor thesis has been exciting, educational, and challenging. We wish to thank both NTNU and our educators for three years of tough and intensive, yet interesting and resourceful years. Additionally, we would like to thank our supervisor, Ole Christian Eidheim, for offering both interest and optimism while guiding us during the process of writing this thesis. We would also like to thank NTNU's High Performance Computing Group for providing resources and access to the hardware needed for completing this thesis. This thesis would not have been possible without your help, of which we are very grateful.

22nd May 2023, Trondheim
Nicolai Thorer Sivesind & Andreas Bentzen Winje

Contents

List of Figures

List of Tables

1 Introduction

In late November 2022, the powerful capabilities of artificial text generation, a subfield of artificial intelligence (AI) known as *language modeling*, became widely known through the research release of OpenAI's chatbot service, *ChatGPT*. This technology opened the eyes of the public with regards to the extraordinary capabilities possessed by state-of-the-art large language models (Yang, 2022), demonstrating broad general knowledge combined with human-level problem-solving abilities (Bubeck et al., 2023). *GPT-4*, the newly released limited-access next-generation large language model and descendant of ChatGPT's release model *GPT-3.5*, has for instance passed a large collection of exams, amongst them the *Uniform Bar Exam* and the *Medical Knowledge Self-Assessment Program*, outperforming the large majority of human performances on these high-regarded academic benchmarks (OpenAI, 2023a). While large language models have proved their potential for streamlining human tasks; assisting in creative, scientific, and commercial processes (A. Lee, 2023), there are risks and challenges related to their rapidly growing capabilities and ease of access.

Back in 2015, an open letter signed by Stephen Hawking, Elon Musk[1] and a dozen of AI-research experts, called for increased investment into research 'aimed at ensuring that increasingly capable AI systems are robust and beneficial' (FLI, 2015) while avoiding potentially irreversible consequences of irresponsible and immature applications in society. The letter states that by necessity, it is an interdisciplinary problem domain involving economic, sociological, juridical, and technical research efforts, among others. Following the release of GPT-4, a new open letter, signed by an even greater number of high-regarded AI researchers and tech leaders[2], called for a temporary halt in the development of AI systems more powerful than that of GPT-4 (FLI, 2023). The letter advocated for an immediate 6-month pause for the purpose of researching and establishing profound policies and other AI safeguard measures (FLI, 2023). Additionally, the letter identifies several areas of interest:

- New and capable regulatory authorities dedicated to AI
- Oversight and tracking of highly capable AI systems and large pools of computational capability
- A robust auditing and certification ecosystem
- Liability for AI-caused harm and measures for reducing economic and political disruptions caused by AI
- Robust public funding for technical AI safety research
- Provenance systems to help distinguish real from synthetic productions

[1]Elon Musk co-founded OpenAI in late 2015 as an open-source non-profit competitor to the AI-industry leaders (OpenAI, 2015)

[2]Among the signers was Elon Musk yet again, which cut ties with OpenAI in 2018 due to 'concerns over the increasing privatization of the company' (CNBC Television, 2023), which the following year switched from non-profit to 'capped-profits' (OpenAI, 2019)

In this thesis, we focus on the latter field of interest; it addresses the critical need for new techniques and standards for *digital content provenance*. Among the many sectors impacted by the lack of thorough research on robust AI provenance systems and usage policies for large language models, *academia* stands out. Although large language models already have seen applications within academia as an educational tool (Alver, 2023), concerns are raised regarding authenticity, factual reliability, and the promotion of intellectual laziness (Yvette Mucharraz y Cano et al., 2023). The current capabilities of-, and ease of access to large language models have the potential for undermining independent thinking, problem-solving, and engagement with academic subject matter. Ensuring the appropriate use of these technologies is therefore essential in maintaining the standards of academic integrity and the culture of intellectual curiosity.

1.1 Problem definition

In this bachelor thesis, we explore how large language models can be leveraged to prevent their own misuse, thus the title, *Turning Poachers into Gamekeepers*. We aim to provide insights into how academic institutions can govern the use of text-generative language models in evaluation tasks, research papers, and other academic endeavors which prior to the public access to large language models required human intuition to complete. By researching this, we will more specifically examine various approaches for using large language models for detecting their own textual signature, referred to as *machine-generated text*. The main research question of this thesis can be summarized as:

How can large language models be leveraged to detect machine-generated text in academia?

To provide an evaluation, the problem is further broken into distinct and pre-requisite sub-questions:

- What data foundation is necessary for the development and evaluation of tools for detecting machine-generated text, and how can such in-domain data be produced?
- What current and potential future approaches show promise in existing literature for efficient and precise detection of machine-generated text, and how do they currently score in performance evaluations?
- What are the social and ethical aspects of applying machine-generated text detection tools in academic settings?

1.2 Problem domain contributions

The overall problem domain of this thesis can be summarized as *detection of machine-generated text in academia*. This bachelor thesis specifically contributes to the problem domain with:

- A Literature review of current research within the problem domain

- The production of an in-domain dataset, and a statistical analysis of language model characteristics when generating targeted in-domain data for training and performance evaluation.
- Development, implementation, and performance evaluation of two distinct detection approaches, *In-Context Learning* and *Fine-tuned Binary Sequence Classification*, both novel in the scope of the problem domain.
- Discussion of the social and ethical aspects related to the application of machine-generated text detection tools within academia and the weighted importance of task-relevant performance-evaluation metrics respective to academic applications.

1.3 Formulation of the thesis

This thesis, including the theory section, has been written with some prerequisites regarding certain topics and terminologies. It is written with the assumption that the reader possesses the knowledge of the curriculum provided in the two initial years of NTNU's bachelor's program, *Computer Science*. This includes various statistical terms, the understanding of statistical diagrams such as boxplots, and scatter plots, in addition to various terms from the domain of Engineering in Computer Science such as big O Notation which also includes computational complexities.

1.4 Thesis structure

The structure of the thesis is provided with concise descriptions.

1. **Introduction**

   Provides a short introduction to the encapsulation problem domain of the bachelor thesis and the importance of providing research within this domain.

2. **Theoretic foundation**

   Provides the theoretical foundation for understanding the content of the thesis, aside from the assumed knowledge of the reader described in 1.3.

3. **Research methodology**

   A presentation of the foundational research methodology and the applied configuration.

4. **Literature review**

   Summarizes relevant research within the problem domain, including the social impacts of large language models, detection approaches, and the importance of data foundation within the scope of machine learning.

5. **Dataset collection and in-domain dataset production**

   Presents the datasets used for training and evaluating the implemented detection approaches, in addition to the applied methods for producing and cleaning an in-domain dataset for the specific case of machine-generated text detection in academic text. The methods for optimizing the datasets for targeted learning and reformatting for the approaches which require training.

6. **Setup for detection approaches**

   Describes in detail the setup for both of the detection approaches, in-context learning, and fine-tuned detection.

7. **Results and analysis**

   Results from the dataset production and statistical analysis of characteristics in the generator model, including discrepancies between the human-written and machine-generated texts in the produced dataset.

8. **Discussion**

   The discussion section provides discussion related to the results, various performance-evaluation metrics and the social and ethical aspects related to the problem domain.

9. **Conclusion and Further Work**

   The conclusion is a summarization of the takeaways from this thesis in addition to further suggested work within the problem domain.

Broader impact

This section describes the possible broader impacts of our work outside of research.

1.5 Additional resources

All code used for the production of this thesis, including the datasets and the implemented detection approaches are available at:

github.com/IDATT2900-072/MGT-Detection

The produced dataset of human-written and machine-generated research abstracts, *ChatGPT-Research-Abstracts*, is available at:

huggingface.co/datasets/NicolaiSivesind/ChatGPT-Research-Abstracts

The dataset collection formatted for binary sequence classification, *Human-vs-Machine*, is available at:

huggingface.co/datasets/NicolaiSivesind/human-vs-machine

The fine-tuned models are available for testing[3] and download at:

huggingface.co/andreas122001/roberta-academic-detector

[3]On this website, the models can be used directly for classifying text by using the hosted inference, provided by HuggingFace. Simply write some text in the text box and click ”Compute”. We suggest trying either of the RoBERTa models, as, at the release of this thesis, the Bloomz-models do not seem to not work well with the hosted inference.

1.6 Central terminology

As this thesis uses a range of various technical terms, we would like to outline the absolute central terminologies crucial for understanding the content throughout the thesis:

- **Human-written text** refers to text which is written by a human and not a large language model
- **Machine-generated text** in the scope of this thesis, refers to text which is *generated by a large language model*. In general use, the term is defined as 'text generated using machine-learning techniques in order to resemble writing in natural language' (Cornell University, 2023).
- **MGT-detection** refers to the main problem of the thesis: The task of detecting machine-generated text (MGT).

For the purpose of readability, we have put emphasis on minimizing the use of acronyms. There are still some local occurrences. All are defined prior to usage.

2 Theoretic foundation

This section presents the theoretical principles used in this thesis and aims to provide a basic understanding of its main themes.

2.1 Artificial intelligence and machine learning

Machine learning is a subfield within Artificial Intelligence (AI) that focuses on teaching computers patterns and relationships in data. With sufficient training, they can make decisions and compute predictions superior to traditional static computer algorithms and in some cases, even humans.

2.1.1 Neural networks

A fundamental concept which the field of machine learning is built upon is neural networks. A neural network consists of multiple interconnected layers of mathematical functions which often are referred to as neurons. The connections between these layers are given numerical values called weights, which determine the impact each neuron has on the succeeding one. Each neuron also has a bias, which is a value added to the sum of all its weighted inputs for additional adjustability. Combined, the neurons, their biases, and the weights between them determine the final output of the network. We typically refer to the sum of weights and biases as parameters. These are central as they govern the model's ability to represent complex features within data and allow the model to capture and generalize intricate data patterns which then can be applied to new and unseen data.

2.1.2 The transformer architecture and its attention mechanism

Transformers are an artificial neural network architecture designed to handle sequential data. They work with sequences of numerical values, typically generated by converting discrete data such as words in a text, into a continuous numerical representation called an embedding. These embeddings capture various characteristics of the data through encodings, including its positional information. This preserves the sequential context during the conversion process from discrete textual representations to numerical embeddings, enabling the model to capture long-range dependencies within the sequential data. The introduction of the transformer architecture by Vaswani et al. (2017), has since revolutionized the capabilities of text generative models, as the previous approaches like recurrent neural networks [4] struggled to efficiently capture complex relationships and dependencies in text, limiting their ability to generate coherent and contextually accurate outputs, which as a result limited their capabilities.

The success of the transformer architecture is largely due to its *attention mechanism*, which enabled better identification of global dependencies between input and output, ultimately

[4]understanding what RNNs are and how they function is not central for the scope of this thesis. We have still mentioned it for comparison.

evaluating the importance of different words in the text as a whole. This allowed for significantly more parallelization in comparison to recurrent neural networks and enabled efficient computation while easing the task of capturing complex relationships within the text (Vaswani et al., 2017).

2.1.3 Large language models

Large language models are deep neural networks using the transformer architecture. In recent years, they have seen great success within the field of natural language processing which, in addition to the transformer architecture, can be attributed to their immense scale in terms of parameters, training data, and training time, allowing them to effectively capture the complexities of human language. Following the introduction of the transformer architecture, researchers discovered that increasing the number of parameters in these models generally enhances their ability to capture intricate features and relationships within human text. By providing the transformer architecture with a vast number of parameters and training it on a diverse and extensive corpus of text, over time its parameters become so finely tuned that it exhibits high-level human-like capabilities in the domain of text. Consequently, each new version of a large language model usually possess more parameters and greater capabilities than its predecessor.

When describing the capabilities of a large language model, they are usually put in the context of its size, which is typically expressed in terms of the number of parameters they have, rounded off. The size is commonly represented using a numerical value followed by a suffix indicating the order of magnitude:

- k: ‘thousand’ - 10^3 (100k = 100 000)
- m: ‘million’ - 10^6 (100m = 100 000 000)
- B: ‘billion’ - 10^9 (100B = 100 000 000 000)

The latest large language model releases can perform a wide range of complex textual tasks with high accuracy and versatility, including text generation, text-summarization, problem-solving, text classification, and question-answering, among others, making them capable of mimicking human-like understanding and generation of natural language. Bubeck et al. (2023) portrayed GPT-4’s capabilities, stating it ‘could reasonably be described as an early, yet incomplete version of Artificial General Intelligence’.

2.2 Tokenization

In natural language processing, tokenization is the process of breaking down texts into a sequence of smaller pieces of text, called *tokens*. Tokenization plays a crucial role in encoding textual and other forms of discrete data into numeric sequences that language models are able to comprehend and process. When tokenizing text, the tokens build up a vocabulary of all possible tokens as an array, where each token is associated with an array index, also called token-IDs.

Simple approaches to tokenization can be taking each individual word or character, or separating a text by a delimiter. More advanced techniques, e.g. *subword tokenization* and *byte-pair encoding*[5] allow for more control of the how the vocabulary is created, by for example separating words into more frequently used subwords, and by combining frequently used characters in a text to represent common prefixes, suffixes or affixes, making it easier for language models to comprehend (Gupta, 2022).

”*This sentence will be tokenized.*”

$\Downarrow$

Tokenization

$\Downarrow$

[”This ”, ”sen”, ”tence”, ” will ”, ”be ”, ”token”, ”ized.”]

Figure 2.1: Example of how a text can be tokenized, here using no particular tokenization method.

2.3 Language modeling: Autoregressive vs bi-directional

Language modeling is a sub-field of natural language processing (NLP), which with its recent advancements is arguably becoming the de-facto approach for most NLP tasks. It is defined as the *estimation of probability distributions for sequences or sets of tokens* (Wikipedia, 2023). The core goal is to capture the underlying structure and patterns within a language, which can then be used to make predictions and perform various tasks related to understanding and generating text. There are two primary approaches to language modeling: *autoregressive* and *bi-directional*. While both approaches implement transformer architectures, they differ in their training objectives, text generation processes, and model-specific modifications to the underlying transformer architecture, making each approach better suited for different sets of problems.

In autoregressive language modeling, exemplified by models such as OpenAI’s GPT-series, BigScience’s BLOOM, or MetaAI’s LLaMa, the primary focus is on the generation of text. Autoregressive models do this by predicting the next token given the preceding tokens in the input text sequence. This approach leverages the context provided by the preceding tokens and iteratively builds upon it to generate coherent and contextually relevant text.

Bi-directional language modeling, exemplified by models like Google’s BERT, or RoBERTa utilizes a masked language modeling objective during training. In this approach, a portion of the input tokens is masked, and the model learns to predict the masked tokens based on the surrounding context. This allows the model to consider both preceding and following tokens in the sequence, making it a bi-directional model (Devlin et al., 2019). Although BERT is not designed for text generation in the same way as autoregressive transformers, it excels at tasks requiring a deep understanding of input context, such as filling in blanks or completing sentences. The bidirectional nature of BERT provides a more comprehensive view of the input context but makes text generation more challenging compared to autoregressive models.

[5]Understanding *how* these advanced tokenization methods work is not central to understanding this thesis, it is important however, to understand that the text *is* split into a sequence of tokens before processing with a language model.

2.4 Model logits in language modeling and perplexity

Model logits are the raw, unnormalized output values produced by a neural network model. In the context of classification problems, logits represent a model's confidence score for each candidate/class in the class domain. To transform these logits into probabilities, an activation function, such as softmax, is applied to calculate the normalized probabilities relative to each other. This transformation results in a *normalized probability distribution*, often referred to as the *probability distribution*. The sum of all probabilities in a normalized probability distribution adds up to 1.0, as the total probability of all possible outcomes must equal to 100%.

Model logits are also the metrics used in autoregressive text generation (Papers With Code, 2023). Given a sequence of tokens, an autoregressive model computes a logit for every token in its 'vocabulary', and the new generated token is selected from a small pool of tokens with the largest probabilities (Hugging Face, 2022). With this perspective, the steps in autoregressive language modeling can fundamentally be categorized as a series of classification problems. It should still be mentioned that there are multiple layers of algorithms within the token selection process to promote diversity in outputs and avoid repetitive patterns, amongst others, but these are on a higher abstraction-level and do not modify the computed token-logits (Hugging Face, 2022).

An interesting zero-shot in-scope[6] classification approach which is based on model logits, involves using the same procedure as in autoregressive generation of text to produce token-logits for each token in an input text sequence. Since logits are computed for all available tokens in the models 'vocabulary', these can be used to measure the distance between a token in a real text, and the likelihood of a specific model actually producing this token (Face, 2023). Adopted from the field of information theory, this distance is a metric referred to as *perplexity* (Jurafsky and Martin, 2023).

Perplexity is a measure of an autoregressive language model's ability to predict the next token given a preceding sequence of tokens. It is calculated as the inverse probability of the true next token, according to the model's probability distribution, raised to the power of $1/N$, where N is the total number of tokens in the text (Jurafsky and Martin, 2023). Unlike in autoregressive text generation where logits are used to extend the text sequence, perplexity is applied as a loss function. Its application is to evaluate the model's ability to generate the next token in a real text and can for instance be used to fine-tune models for specific language styles. As with all loss functions, a low score indicates that the model does well in predicting the next true token, while larger perplexity scores indicate that the token is more unexpected for the model. Conversely, perplexity can be applied as a measure of how likely a text is to be produced by the model by calculating the perplexity score of all tokens in a sequence and applying a binary cross-entropy loss function to measure the similarity between the model's predictions and the actual text (Huyen, 2019).

[6]Discrimination of human-written text vs. machine-generated text in academia

2.5 Formulating MGT-detection as a binary sequence classification problem

The problem of detecting if a text is generated by a machine or not can be formulated as a binary sequence classification problem where a classifier, or detector, tries to classify a sequence of tokens (see section 2.2) as being either positive (1), meaning machine-generated, or negative (0), meaning not machine-generated, or more specifically human-written. To train a detector means to maximize the probabilities of the detector classifying the sequence as ”1” if the sequence is machine-generated and ”0” if the sequence is human-written.

2.6 Metric performance evaluation of binary classification models

In a binary classification problem, the model's performance can be measured by looking at how many times it correctly labeled the data as positive or negative, also called true positives (TP) and true negatives (TN), and conversely how many times it falsely labeled the data as positive (FP) and negative (FN). From these numbers, metrics such as accuracy, precision, f1-score, and recall can be measured.

Accuracy: This metric measures the overall correctness of the classifier. However, this metric can be misleading if the number of each class is imbalanced, which is often the case in real-world settings. This metric is calculated as:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Precision: This is the ratio of true positive predictions (correctly identified machine-generated texts) to all positive predictions (all texts identified as machine-generated). High precision means that when the model predicts a text as being machine-generated, it is likely to be correct. This is calculated as:

$$precision = \frac{TP}{TP + FP} \tag{2}$$

Recall: This is the ratio of true positive predictions to all actual positives (all actual machine-generated texts). High recall means that the model is good at identifying machine-generated texts when they are present. This is calculated as:

$$recall = \frac{TP}{TP + FN} \tag{3}$$

F1-score: This is the harmonic mean of precision and recall, and it tries to balance the two. It gives a weighted average of both precision and recall, calculated as:

$$F1 = 2 * \frac{recall * precision}{recall + precision} \tag{4}$$

2.7 Zero-shot and few-shot learning

There are various methods for approaching classification problems in natural language processing. Among these are n-shot approaches, where *n* refers to the amount of labeled data/examples provided to the models at training-, fine-tuning- or inference time.

Zero-shot learning, initially called *dataless classification* (Chang et al., 2008), refers to the ability of a model to generalize and perform tasks without any task-specific labeled data. It can be applied across different domains and tasks, including *in-context learning*, as well as most NLP tasks.

Few-shot learning refers to the problem of teaching machine learning models underlying patterns in data based on a small number of labeled examples (Parnami and M. Lee, 2022), hence the term ”few-shot”. This approach can be applied in conjunction with a range of methods for teaching neural networks, such as in-context learning, fine-tuning, and for instance, episodic learning, which focuses on training models in few-step episodes of distinct problem domains with the goal of improving the model's ability to generalize based on limited labeled examples[7] (Laenen and Bertinetto, 2021).

2.8 In-context learning, prompt-engineering, and context windows

Due to the recent advancements regarding the capabilities of large language models, in-context learning has emerged as a new paradigm within natural language processing. The concept of in-context learning is to leverage the generalization abilities of autoregressive large language models by providing task-specific information within one or more input texts at *inference-time*[8], relieving the need of weight-adjustment and the various costs and pre-requisites related to them. This is done entirely through prompting, where the specified task is defined in natural language.

In-context learning can be applied in both zero-shot and few-shot settings. In its simplest form, any conversation with a large language model can be considered as being in a zero-shot setting as the model is using the context of the conversation to produce adequate responses. Typically, for evaluating zero-shot performance, more carefully crafted prompts containing task-specific requirements and context are applied. The process of crafting prompts for optimizing the performance of a generative language model is referred to as *prompt-engineering*.

In addition to this, language models are also able to learn relationships in-context if labeled examples are provided in the input text along with the prompt[9]. When adding labeled examples, ultimately applying *supervised learning*, the approach is then considered to be in *few-shot* setting.

One significant constraint of in-context learning is the *context window size*, which refers to

[7]While episodic learning is a method which is not approached in this thesis, we have still included it as an example for outlining that there are many applications for few-shot learning.

[8]The process of using a trained machine learning model to perform a task

[9]A prompt is the instruction section of an input text. The input text may also contain other elements, such as for instance labeled examples, but these are not considered as a part of the prompt.

the maximum number of tokens a language model is able at process at a time. In simpler terms, the context window size represents how many tokens the model can hold as context when generating a response, including the response itself. Although the model is able to continuously generate outputs and accept new inputs, when the context window limit is reached, the window shifts towards the most recent tokens, causing the earlier tokens to be removed from the context (Brown et al., 2020). This is often referred to as a *sliding window* and is common in various computer science applications.

While the context window size is not strictly restricted by the architecture, it is a fixed length that is set due to computational costs. The *attention-mechanism* in the transformer architecture has a quadratic computational complexity, $O(n^2)$, where n is the number of tokens processed for context (Kitaev, 2020). This ultimately restricts today's open-access models to relatively small input sizes. GPT-3 (175B) currently have a fixed context window size of 4096 tokens (Brown et al., 2020), and while the context window size of open-source models like BLOOM (176B) and LLaMA (65B) can be adjusted based on the available computational resources and memory limitations of the system on which the model is hosted on, using these models for longer sequences would require expensive computations as they are restricted by the same attention-mask complexity. Although, there are workaround architectures like the *Longformer* ($O(n)$) or the *Reformer* ($O(nlog(n))$ which have proved valuable for long document tasks, they come with trade-offs within general model capabilities and training complexity compared to the traditional transformer architecture (Beltagy et al., 2020).

2.9 Transfer learning and fine-tuning

In machine learning, transfer learning is a set of techniques where solving a task is based on using pre-existing knowledge or learned representations acquired from solving other related tasks, in contrast to solving the problem by starting completely from scratch.

An example of transfer learning is the fine-tuning of pre-trained large language models. The process of fine-tuning involves taking a general-purpose pre-trained model, trained on a large amount of general data, and further training it on data specific to the task to be solved. This is particularly useful in language modeling, where a large, general-purpose language model is fine-tuned to perform a specific, downstream task, like classifying if a text has positive or negative connotations. The fine-tuned model will inherit the general-purpose model's knowledge and capabilities that can also be useful for the downstream task, like text comprehension, avoiding training a completely new model for only the downstream task.

2.10 Dataset splits

When training and evaluating machine learning models, it is customary to split the dataset into multiple subsets called splits. This is done to prevent biased results when evaluating the model after it has been trained, by separating training data from test data. When the model is tested, the model will not have seen the correct classifications for the test data, which it has for the training data. This ensures that the model generalizes properly on the data, and does not overfit on the training data.

For practical supervised learning problems, the dataset is typically divided into three splits:

- *Train-split*
  This split is the largest split and is used to train the model.
- *Validation-split*
  This split is used to evaluate the performance of the model during training, and is used to monitor of the model is overfitting on the training set which then can be prevented.
- *Test-split*
  This split is used additionally to measure the final performance of the model after training and is used to see how well the model would perform on real-world data.

2.11 Design science research methodology

Design science research is a research methodology that focuses on the development and performance of artifacts designed for the purpose of improving current paradigms within a scientific field. Its core goal is to provide valuable knowledge to a set problem domain, which can be used to improve functionality, performance, efficiency, or practicality of techniques, algorithms, and other relvant tools depending on the specific problem domain, typically referred to as *artifacts* (Brocke et al., 2020). It is a research methodology that typically is applied in the domains of engineering and computer science where the goals often are to optimize the current paradigms in the problem domain. This can for instance involve testing novel approaches to the problem at hand or improving the current established approaches. The methodology consists of two main activities, *build* and *evaluate*. *Build* consists of designing artifacts intended to achieve the set goals with respect to the problem at hand, while *evaluate* involves assessing its performance in the domain and communicating the knowledge gained by these performance evaluations Brocke et al. (2020).

3 Research methodology approach

This section aims to provide a clear description of the applied research methodology approach, and process used to conduct our work in a scientific manner.

3.1 Research approach

The research approach used for conducting the research presented in this thesis is an approach to the research methodology *design science research*. The following are the distinct stages of the research approach:

1. **Experiences and motivation**
   Useed prior experience and motivation to formulate a problem domain and specific problem definition.
2. **Literature review**
   Conducted a thorough literature review as a foundation for our dataset collection, detection approaches, and discussion of social and ethical aspects.
3. **Design theorization**
   Based on the findings from the literature review, the designs for our detection approaches were outlined and theorized.
4. **Dataset collection**
   Using the selected and theorized detection approach designs, datasets for performing model training and performance evaluations were both gathered and produced.
5. **Implementation**
   The theorized designs were prioritized and implemented based on our design theorization.
6. **Performance-evaluation and optimization**
   Iteratively test, evaluate and experiment with hyperparameters and intra-approach techniques.
7. **Document and discuss**
   Document and discuss the results from dataset production and the final performances of the optimized detection approach implementations.

The listed stages were set prior to the initialization of the project and were followed throughout the entire process of our research. They can also be found in the preliminary project plan and mid-way bachelor poster. The purpose and goal for each stage are defined in their respective subsections.

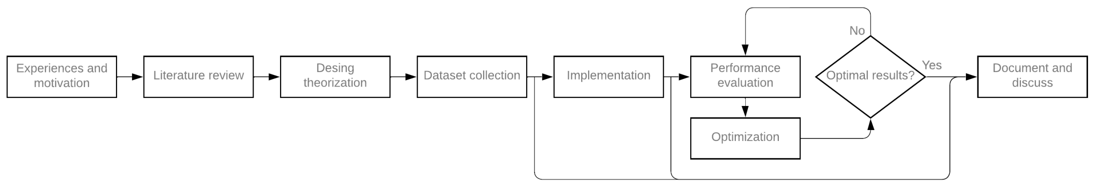


Figure 3.1: The flow of our research stage completion. The figure displays a flowchart for the general research flow of each stage of the research approach. The incremental optimization in *performance evaluation and optimization* is presented as a loop to emphasize the recurrent nature of this stage. For *dataset collection*, *implementation*, and *performance-evaluation and optimization* all direct to *document and discuss* as each intra-stage milestone was continually documented following its completion.

3.2 Approach process

In this section, we describe the process for each of the outlined stages in our research approach. The takeaway from each stage is described alongside reasoning for relevant decisions made during each stage which impacts the following stages.

3.2.1 Experiences and motivation

Through experimentation with various language models and assessing their potential for misuse within academia, the formulation of the problem domain was made. Following this, the specific problem definition was developed through formulation a main research question and deriving the core sub-problems from this. Further detail is described in section 1.1.

3.2.2 Literature review (research process)

A literature review was conducted to assess the current research that has been done in the field. This literature review was conducted through a comprehensive search and careful analysis of various relevant academic works, research studies, and publications related to the topic. This laid the foundation for producing the designed artifacts used for subsequently answering the research questions of this thesis.

More specifically, the literature review served to:

- assess potential social impacts of language models and the need for MGT-detection.
- identifying prior research done and existing methods for MGT-detection.
- identifying potential gaps in the research on MGT-detection.

From the literature review, a set of existing and potential methods for MGT-detection was found. These methods include two established methods; fine-tuned detection, and perplexity, and one less documented, in-context learning.

3.2.3 Design theorization

With respect to the problems and existing methods found in the literature review, the approaches which showed promise with respect to previous success within the problem domain or in similar problem domains were outlined and theorized for application of MGT-detection in academia. The theorized designs also defined the further needs in the next stage of *data collection*.

From the literature review, we outlined three main approaches for MGT-detection:

1. Perplexity
   - Use language models to calculate model logits for each token in a text and set a perplexity score threshold for what is predicted as human-written text and machine-generated text. Perplexity scores above this threshold are predicted as human-written text while scores below are predicted as machine-generated text [10].
   - Entirely zero-shot approach, requires only data for performance evaluation.
2. In-context learning
   - Provide a language model a text to predict as either human-written or machine-generated, and prompt describing the task it is to perform.
   - Leverages the pre-existing capabilities and knowledge of language models to determine the class of the provided text.
   - Can be applied in both zero-shot and few-shot settings.
   - Requires a small amount of data to be provided as labeled examples in few-shot settings, as well as data for performance evaluation of both settings.
3. Fine-tuned binary sequence classification
   - Fine-tune a pre-trained language model for binary sequence classification
   - Applies transfer learning to efficiently optimize the model for the downstream task of MGT-detection in academia
   - Requires a considerable amount of data for training, in addition to data for performance evaluation.

Within each of the approaches, lower-level techniques for performance optimization were also applied. We later refer to these as *intra-approach techniques*. The specific application of these techniques is presented in section 6, *Setup for detection approaches*, while the corresponding literature supporting these techniques are provided in section 4, *Literature review*.

[10]As perplexity measures the loss in the model's ability to predict the next token of a provided text sequence, lower scores are considered less unexpected to the model while larger values indicate that the model finds the token more unexpected

3.2.4 Dataset collection

With respect to the approaches outlined in section 3.2.3, the necessary data was collected.

From the literature review we found that for implementing the outlined approaches, in-domain data was desirable. With respect to the problem domain this is specifically human-written- and machine-generated academic texts. As we also wanted to compare the performance of our best-performing approaches, out-domain data was also collected.

During the process of collecting data for the approaches, we found one dataset suitable for out-domain training and performance evaluation: *GPT-wiki-intros*, which contained human-written and machine-generated Wikipedia introductions. No other suitable datasets were found, and a conclusion was drawn that an in-domain dataset had to be produced. For this, we found a dataset containing scientific research abstracts. This was used to sample a subset of the human-written academic texts in our dataset, while also being used for targeting a language model to generate corresponding machine-generated academic texts. Further detail about the collected dataset and the self-produced dataset is provided in section 5.

Following the collection and production of the datasets, the statistical analysis of the dataset and the characteristics displayed in the model used for generating the machine-generated texts were performed. This was done prior to implementing and evaluating our detection approaches such that any undesired characteristics in the dataset could be mitigated in the performance evaluations of our proposed detection approaches.

3.2.5 Implementation

When entering this stage of the research project, we used our theorized designs from *design theorization* phase to develop and implement the various approaches for MGT-detection. The data collected during the data collection stage is described in section 3.2.4.

In this stage, the foundational implementation of our approaches to be used for performance evaluation in MGT-detection in academia, was developed. Due to time constraints associated with a bachelor thesis, only two of the theorized approaches could be implemented with proper evaluation and documentation. Most MGT-detection tools available since the emergence of the thesis problem domain have been implemented with perplexity as the base metric for MGT-detection. Various research papers have already been released documenting the performance of this approach, such as *DetectGPT* proposed by Mitchell et al. (2023). During the literature review, no documentation of in-context learning and fine-tuned sequence classification for MGT-detection in general was found. Due to this, a prioritization of providing valuable insight to novel approaches rather than optimizing previously documented approaches such as the perplexity-based approach we outlined in section 3.2.3. The focus was set on the exploration of in-context learning and fine-tuned sequence classification to provide unique, novel, and valuable contributions to the thesis problem domain.

3.2.6 Performance-evaluation and optimization

Following the foundational implementations of the prioritized approaches, the stage of performance-evaluation and optimization[11] was initialized. In this stage, we exposed the various approach variations[12] to performance evaluation. Based on the results collected in these tests, adjustments were made to explore if the approach had the potential to be optimized. This involved testing various hyperparameters, combinations of testing data, and applying distinct combinations of the intra-approach techniques found during the literature review.

3.2.7 Document and discuss

This stage was initialized at the end of the *data collection* stage, and alongside the *implementation* and *performance evaluation and optimization* stages. This was done to continually document each completed milestone within the respective stages. The dataset collection and -production were first documented alongside the statistical analysis of the self-produced dataset. See section 3.2.4 for details. Following this, the performance evaluations of the detection approaches were performed and documented. Following the documentation of the detection approaches, each of the results were discussed, before finalizing the report with the necessary contextual and supplementary sections such as the *introduction* (section 1), *theoretic foundation* (section 2), *research methodology* (section 3), discussion of *social and ethical aspects* (section 8.5) and the *conclusion* (section 9), each with regard to our defined research questions (section 1.1).

The final result of this stage is this report.

[11] In our preliminary project plan, we named this stage *testing, evaluation, and adjustment*. This was later renamed for clarity, but the purpose and execution of the stage remain unaffected

[12] Each of the main approaches had sub-configurations which we refer to as *approach variations*

4 Literature review

The aim of this section is to provide an overview of the existing research on MGT-detection, both to identify previous work in the field and to identify gaps in the literature. Based on the findings of this review, we define the methods that are used for further research. To assess the need for MGT-detectors, we will first discuss the potential problems and social impacts of using machine-generated text and language models, and the research that has been done in this field. Existing methods for MGT-detection and their potential usage in this thesis are then discussed. Finally, a discussion of the usage of datasets for MGT-detection and what the literature suggests for generating such datasets is presented, as it is an important aspect of most machine learning problems.

4.1 Social impacts of large language models

As language models are trained on large corpora of human text data, they will predictably inherit certain biases present in this data. As shown by Solaiman et al. (2019) and Brown et al. (2020), text generated by language models may reflect stereotypical and biased opinions on topics such as gender, race, and religion. For instance, GPT-3 has a higher probability of generating male identifiers when asked to complete sentences about occupations tied to higher education and has a tendency to attribute positive and negative words differently depending on race (Brown et al.). In addition to this, language models have been notoriously known to generate hallucinations[13] and factual errors (Zhao et al., 2023). This is a problem, as users of these language models are often unaware of these issues of such models.

This is especially concerning when considering language models being used to assist in academic writing, as the models could potentially effectuate their biases and errors into the writing of the researcher and thus affect the results expressed in the subsequent research paper. Such manipulation of opinion is demonstrated in a recent study by Jakesch et al. (2023), where they assessed the effects of using opinionated language models as writing assistants. They found that such assistants were able to alter the opinions of the writers, and sometimes were able to considerably affect the view of the writing. This could also be problematic when considering adversaries maliciously making models opinionated to support certain opinions or viewpoints.

Other examples of the potential for malicious usage of language models can also be found in the literature, such as fake news generation and generation of content used for spam and phishing attacks (Weiss, 2019). In a research paper on their language model Grover, trained exclusively on news articles, Zellers et al. (2020) found that propaganda and fake news generated by Grover was more trustworthy to humans than human-written propaganda. Solaiman et al. (2019) and Brown et al. (2020) discuss the potential for similar usages in their threat assessment for their respective models, GPT-2 and GPT-3.

Recent concerns have also been raised on if language models can be used maliciously in

[13]A language model is said to hallucinate when they generate content that is factually or objectively wrong, often in a confident manner.

academic writing, for instance by plagiarism-avoidance or ghostwriting[14], as to wrongly gaining good academic results. In a study by Khalil and Er (2023), it was found that ChatGPT, when asked to produce essays on various topics, was able to generate content of high originality in 40 out of 50 essays, as evaluated by plagiarism-detection software, emphasizing the pressing concerns of language models ability to avoid detection. Addressing these threats requires the development of accurate detectors, as also pointed out by Jawahar et al. (2020).

4.2 The importance of data

The performance of machine learning models is strongly influenced by the quality and relevance of the datasets used for training, fine-tuning, and performance evaluation. In the release paper of Google's bi-directional language model, *BERT*, Devlin et al. (2019) demonstrate state-of-the-art results even with relatively small amounts of task-specific training data. They effectively transfer knowledge from pre-training to downstream tasks[15], reducing the need for extensive task-specific training data in fine-tuning approaches. This finding suggests that when fine-tuning pre-trained language models, the size of the dataset has less impact on performance than the quality and relevance of the data. This is partly backed up by the findings documented by Bakhtin et al. (2019) which concludes: '*matching the domain of the training set is more important than model complexity*' (Bakhtin et al., 2019).

In addition to this, Bakhtin et al. (2019) observed that their classifier learns to recognize the characteristics of the machine-generated text and identify outliers as human-written text, rather than the other way around. They argue that, a liability of this is that as various text-generative models are pre-trained on corpora that may contain non-overlapping subsets and also differ in architecture-specifics, their output texts inhibit slightly different signatures. While there are shared characteristics among texts produced by separate text-generative models, the generator-model(s) which are used to produce the machine-generated samples in the training data still affects the detection-model's internal representation of machine-generated text and consequently, its ability to differentiate between real and generated text.

Furthermore, Uchendu et al. (2020) suggests an alternative to the binary classification problem called ”Authorship Attribution”, where instead of only detecting if the text was written by a language model or not, additionally identifying which language model generated the text.

From these findings, we can observe that the overarching detection problem can be decomposed into specialized sub-problems, each focusing on distinct aspects while sharing the same primary goal of detection:

1. *Binary classification of single-architecture model-signature*
   This sub-problem focuses on detecting text generated by a specific target model. The

[14]To write a text on behalf of someone else and officially crediting the text to the other person.

[15]In self-supervised learning, a downstream task is the main task to be solved, whereas upstream tasks are additional tasks that are learned as a result of solving the downstream task.

detection model is trained to discriminate between human-generated text and text generated by the target model. By honing in on the unique signature of the target model, the detection model can accurately identify whether a piece of text was generated by the target model or not. This strategy may work well for applications where the primary concern is detecting text generated by a particular model, but it may have limited effectiveness against text generated by other models.

2. *Binary classification of multi-architecture model-signature*
   In this sub-problem, the detection model is trained to identify universal characteristics of machine-generated text, regardless of the specific model used to generate the text. This means the detection model is less focused on the unique signature of a single target model and more focused on the shared characteristics among texts produced by different text-generative models. This sub-problem enables the detection model to be more robust against a broader range of text-generative models, but it may be less precise for discriminating between human-written and machine-generated text compared to proposed solutions for the first sub-problem, where all machine-generated data points are produced by a single text-generative model.

3. *Multi-label classification of single-architecture model-signatures*
   This sub-problem goes beyond binary classification and aims to identify not only whether a text is human-written or machine-generated, but also which specific model was used to generate the text. The detection model is trained on samples from multiple text-generative models, learning to recognize the distinct signatures of each model. Solutions to this sub-problem can be beneficial in situations where it would be desirable to know the source of the generated text, such as in understanding the strengths and weaknesses of various large language models in generating text or, if we look outside the scope of academia; tracking the use of a specific text-generative model for malicious purposes.

Given that the latter two sub-problems require access to the text generated from a variety of text-generative models, which due to the novelty of the problem domain has not yet become publicly available, they are out of the scope of this thesis, and the main focus will primarily be on *Binary-domain classification of target model's signature*, with some cross-testing between detection of two different target-models.

4.3 Methods of detection

The problem of MGT-detection is an emerging field of study with growing interest. As a consequence, the literature in this field is rapidly evolving. The following will present the literature on the methods that have been tried before.

The methods that can be found in the literature can be categorized into three categories:

- *Classical neural classification models*
  Training of classical neural networks from scratch to classify a text as either human- or machine-generated.

- *Fine-tuned detectors*
  Using pre-trained language models to classify text by replacing the output-layer with an uninitialized classification layer and fine-tuning the model on human-written and machine-generated text.
- *Zero- and few-shot detection*
  A suite of methods where pre-trained language models are used directly, without further training, to calculate token probabilities in conjunction with statistical methods.

Following is a more in-depth assessment of these categories.

4.3.1 Classical neural classification models

Solaiman et al. (2019) trained a logistic regression neural network on a dataset of real and generated samples of the WebText-dataset, where they achieved an accuracy of approximately 74% on samples generated by the larger GPT-2 model (1.5B), demonstrating the potential efficacy of such models. They also found that when limiting Top-K to 40, the accuracy increased to 93%, suggesting that such classificators are sensitive to model configuration, e.g. sampling method. Additionally, they found that shorter texts are harder to detect when using these models. Similar work was done by Bakhtin et al. (2019) in their extensive research on using energy-based models to discriminate between human-written and machine-generated text, which was also discussed in section 4.2. One of the findings from their work was that their model did not generalize well cross-corpus, meaning that when training on e.g. book-texts, the model did not perform well on e.g. detecting machine-generated Wikipedia articles, and vice-versa. This suggests that training on task-specific text data might be better than trying to detect generally generated text.

A drawback of such models, which is important to note, is that building neural networks from scratch can be an advanced procedure, and is often associated with immense resource- and data requirements, which in turn can make testing such methods impractical or infeasible for low-resource researchers.

4.3.2 Fine-tuned detectors

The most prominent results, however, have been from fine-tuning large pre-trained language models to detect itself or other, similar models.

An example of this can be found in the work of Zellers et al. (2020), where they fine-tuned their language model GROVER to detect outputs from itself across multiple model sizes[16] and evaluated the results against existing detectors like FastText (Joulin et al., 2016) and BERT (Devlin et al., 2019). They found that the GROVER model outperformed existing detectors, with an accuracy of 92% (with GROVER-large) against 73%, suggesting that the best model to detect outputs from GROVER is GROVER itself. They also found that larger versions of the model achieved a higher accuracy on smaller versions and, conversely, smaller models achieved a lower accuracy on larger versions.

[16]GROVER-base, -large and -mega with 124M, 355M, and 1.5B parameters, respectively.

Another example can be found in the work of Solaiman et al. (2019), where they also tested their GPT-2 model on fine-tuned detection. Here they fine-tuned various sizes[17] of the RoBERTa model for sequence classification on GPT-2 output, where they achieved a SOTA[18] accuracy of approximately 95%. Similar, but not identical to Zellers et al. (2020), they found that training on output from larger models increased the accuracy on detecting smaller models, regardless of the detector's size. Partly in contrast to Zellers et al. however, they found that a GPT-2 model of equivalent size to the RoBERTa model performed worse at detecting GPT-2 output, concluding that a model itself is not necessarily the best model for detecting itself. They argue that this is expected, however, due to RoBERTa being an autoencoding model, which should perform better than autoregressive models like GPT-2 on sequence classification (Solaiman et al., 2019).

4.3.3 Zero- and few-shot detectors

As new and better language models are likely to be developed in the future, it might not be feasible to retrain or fine-tune new detectors every time a better generator is released, which would lead to a constant cat-and-mouse game of training new models. Incidentally, there is another suite of methods that requires no training, by using language models as-is in a zero- or few-shot setting.

Gehrmann et al. (2019) presents the GLTR tool, a tool for color-coding and visualizing individual tokens in a text based on the probability that the token is generated or not. This probability is calculated by inputting the previous tokens to the same or a similar language model and getting the predicted output for that token from the model, thus seeing how likely the model is to generate that specific token given the previous tokens. As with autoregressive text generation, this is done by iterating through all tokens of the text and using the previous sequence as context for producing the model logits. Although this method is advantageous for detecting hybrid texts (machine-generated and edited by a human), it lacks the possibility for automatic detection, as the color-coded tokens necessarily have to be evaluated by a human, making it hard to measure the performance of this method. Solaiman et al. (2019) takes this method further by allowing for automatic detection where, instead of highlighting individual tokens, they take the total probability of all tokens and determine if the whole text was generated or not by a threshold of the total probabilities. This method was proven to be surprisingly efficient (ca. 83% accuracy) and has later been improved upon by Mitchell et al. In the work of Mitchell et al. (2023) they, instead of taking the raw sum of total probabilities of each token, perform minor rewrites to the text using another large language model, then take the average of the logarithm of each original probability over the rewritten probabilities to determine if a text is generated by this model or not. A higher average of log probabilities indicates a higher likelihood of the text being generated by the same model. This method achieved better results than other existing zero-shot methods.

There has also been research conducted on the use of in-context learning in zero- and few-shot settings. In the paper 'Language Models are Few-Shot Learners' by Brown et al. (2020), they demonstrate GPT-3's impressive performance in in-context zero- and

[17]RoBERTa-base and -large with 125M and 356M parameters, respectively.

[18]short for *state-of-the-art*

one-shot settings, while occasionally outperforming state-of-the-art methods in in-context few-shot settings. They document that scaling up language models greatly improves task-agnostic, few-shot performance, sometimes even reaching competitiveness with prior state-of-the-art fine-tuning approaches. In addition to this Brown et al. states that in general, the performance gap between zero-, one-, and few-shot learning increases with model capacity (Brown et al., 2020), as can be seen in table 4.1.

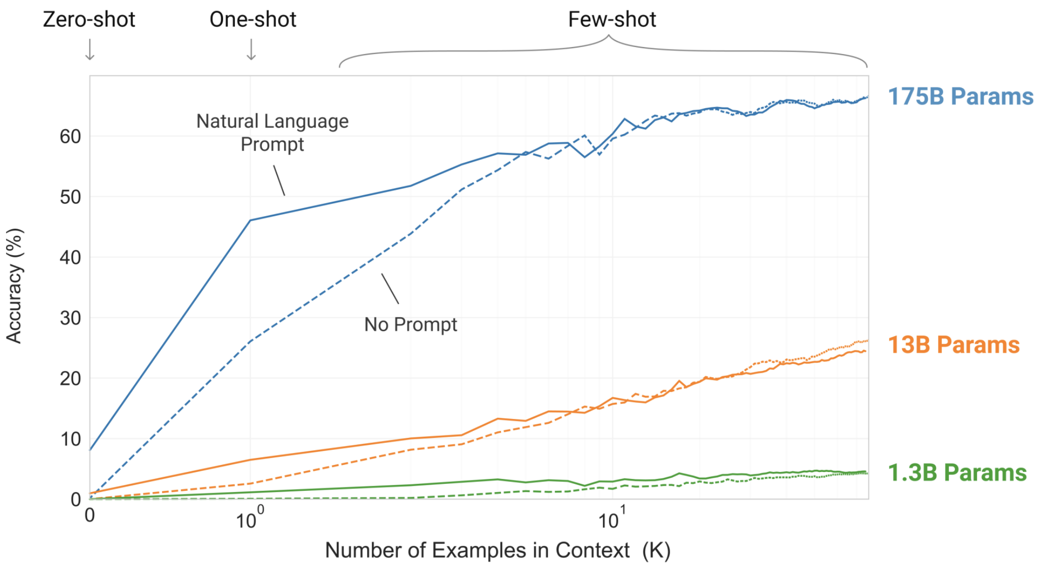


Figure 4.1: Various in-context learning performances presented by Brown et al. (2020) stating, 'Larger models make increasingly efficient use of in-context information. We [Brown et al.] show in-context learning performance on a simple task requiring the model to remove random symbols from a word, both with and without a natural language task description. The steeper "in-context learning curves" for large models demonstrate an improved ability to learn a task from contextual information. We see qualitatively similar behavior across a wide range of tasks' (Brown et al., 2020)

For binary classification tasks, which also is the task domain of detecting whether texts are human-written or machine-generated (MGT-detection), Brown et al. shows promising results for in-context learning, respectively 88.3%, 89.7% and 88.6% accuracy in zero-shot, one-shot, and few-shot settings. The task applied in the paper was a slightly modified version of *Winograd Schemas Challenge* in the *SuperGLUE benchmark*[19], which Brown et al. refer to as a 'classical task in NLP [natural language processing] that involves determining which word a pronoun refers to when the pronoun is grammatically ambiguous but semantically unambiguous to a human' (Brown et al., 2020). While the original SuperGLUE benchmark task collection also belongs to the binary classification domain depending on the specific individual task, it was modified in the paper to align with the binary classification problem of correctly resolving the pronoun reference (Brown et al., 2020).

An example of in-context learning specifically being used for MGT-detection is demonstrated by Khalil and Er (2023) in their paper on plagiarism, also discussed above. In their

[19]The *SuperGLUE benchmark* is a collection of tasks designed to evaluate the performance of language models in advanced natural language processing tasks."

study, they took the 50 articles they generated with ChatGPT and then inputted the articles into ChatGPT together with a prompt asking if the given text was generated by itself or not. They found that 46 out of 50 articles were successfully identified as being generated, achieving an accuracy of 92%. It is important to note however, that as they did not include real samples in their tests, false positives and true negatives are impossible to calculate for these tests, meaning the results could be biased, entailing more research to accurately assess the performance of this method.

4.3.4 More on in-context learning

In a paper by Dong et al. (2023), it was found that the performance of in-context learning is greatly dependent on a range of parameters. Among these are:

1. *The formatting of the prompt*[20]
   The performance of in-context learning is severely sensitive to the content of the instruction prompt. Dong et al. (2023) discusses how well-formulated instructions, which describe the task precisely, in general, improve the inference performance of in-context learning. They also found that having instruction prompts of low perplexity in the detector model can improve performance. In addition to this Dong et al. refers to *Instruction Inductions* proposed by Honovich et al. (2022). Honovich et al. found that given several labeled examples, language models are able to generate suitable task instructions. This aims to improve the quality of automatically generated instructions by reducing the reliance on human-written sentences.

2. *The selection of labeled examples*
   A variety of methods can be employed to select a subset of examples from a pool of training examples for in-context learning. Dong et al. categorize these into *unsupervised* selection approaches, which rely on pre-defined metrics without requiring labeled examples, and *supervised* selection approaches, which combine unsupervised selection methods with data-driven approaches, such as reinforcement learning, to optimize example selection using performance accuracies as reward-functions. Notably, Liu et al. (2021) achieved decent results using k-nearest-neighbors as an unsupervised selection approach, selecting neighbors based on a predefined metric (Dong et al., 2023).

3. *The scoring function for determining selected answer*
   There are also various approaches for determining the certainty of the model's prediction, in *A Survey on In-context Learning* this is presented as the *scoring function*. The scoring function transforms the predictions of a language model into an estimation of the likelihood of a specific answer. Dong et al. (2023) presents two widely-used scoring functions which are relevant to the problem domain: *Direct* and *Perplexity*. *Direct* adopts the conditional probability of candidate answers, but this poses some restrictions on template design, not allowing the model to reflect

[20]As mentioned in section 2.8: A prompt is the instruction section of an input text. The input text may also contain other elements, such as for instance labeled examples, but these are not considered as part of the prompt.

before providing a prediction. Perplexity computes the sentence perplexity of the whole input sequence, removing limitations of token positions but requiring extra computation time.

4.4 Summary and concluding remarks

To summarize, the three methods for MGT-detection are training classifiers from scratch, fine-tuning language models for sequence classification, and zero-/few-shot detection. Overall, the fine-tuned detectors have had the best performance, but alternative methods like zero-/few-shot detection have also proved effective while also being much simpler, faster, and less resource intensive. Classifiers trained from scratch have also had success, although they require a lot more resources and expertise to train. Also, there is a research gap in the literature regarding using in-context learning for MGT-detection, which will be further explored.

Based on these findings, we want to further assess the efficacy of training and using larger fine-tuned detectors, as the release and open-sourcing of large language models is a rather recent practice. Additionally, we will explore the performance of using in-context learning for MGT-detection, as it is currently a less documented approach. A description of the setup for these methods can be found in section 6.

5 Dataset collection and in-domain dataset production

In our literature review, section 4.2 *Training-data*, we presented three sub-problems related to data. Due to the immaturity of the topical problem domain of this thesis, there is little publicly available data that is suitable for this specific classification problem. In addition to this, there are also resource constraints concerned with the required computational resources needed to produce acquirable data from the state-of-the-art open-source language models such as BLOOM (176B) and LLaMa (65B). They both require several hundred gigabytes of RAM, GPU memory, and storage to load and run. It is possible to quantize the models for inference to reduce minimum memory requirements, but this will reduce their precision and subsequently the quality of the generated data. Based on these requisites, the scope of this thesis is primarily concerned with *binary-domain classification of a target model's signature*, but we will also do comparisons related to *binary-domain classification of universal text-generative signature*.

GPT-3.5 is currently the most popular and easily accessible state-of-the-art text-generative model due to OpenAI's public research release, ChatGPT. It is therefore sensible to provide more research on this model, as it is more likely to be prevalent in academic work in the short term. Additionally, OpenAI offers a commercial API allowing for the generation of large quantities of data from their 175B model amongst others, thus reducing the need to allocate extensive computational resources.

Considering these factors, we have used data which was generated by the GPT-3 model *GPT-Curie* and the GPT-3.5 model *ChatGPT* for the training and evaluation of our detection approaches. Each model has its own dataset, which can be summarized:

1. GPT-Wiki-Intro (GPT-3)
    - Consists of 150 000 data points each containing both human-written and machine-generated Wikipedia page introductions.
    - Machine-generated samples are produced using the GPT-3 model, *text-curie-001*, which performance-wise matches GPT-3 with 6.7B parameters (Gao, 2021).
    - Externally produced
2. ChatGPT-Research-Abstracts (GPT-3.5)
    - Consists of 10 000 data points containing human-written and machine-generated abstracts of research papers sourced from arXiv.
    - Machine-generated samples are produced using *gpt-3.5-turbo-0301*, which has 175B parameters (Brown et al., 2020).
    - Self-produced for the problem domain of this thesis.

5.1 GPT-Wiki-Intro

GPT-Wiki-Intro is a dataset by Bhat (2023) and is publicly available on HuggingFace[21]. Bhat produced the machine-generated introductions using the text-completion-optimized GPT-3 model, text-Curie-001 by OpenAI. Official model specifications have not been released, but the open-sourced AI research group, EleutherAI has through extensive task evaluations (Gao, 2021) matched its performance with the GPT-3 6.7B-version based on the performance-documentation from its official release paper by Brown et al. (2020).

Bhat (2023) sourced authentic Wikipedia page introductions from another publicly available dataset, *wikipedia*[22] by Wikimedia (2022), and generated 200-word introductions corresponding to the human-written versions by prompting text-Curie-001 with a title and a starter text, as can be seen in listing 1.

```
"prompt": "200 word wikipedia style introduction on '{title}'
            {starter_text}"
```

Listing 1: Curie completion prompt.

5.2 ChatGPT-Research-Abstracts

Having machine-generated text on academic work is an integral part of the problem domain of this thesis, and due to its novelty, no publicly available datasets which provide academic pieces produced by text-generative models were found. A dataset was therefore produced using the ChatGPT model, GPT-3.5-turbo-0301, which is a snapshot of the model version used in ChatGPT on 1st March 2023 and consists of 175 billion parameters. To promote further research in the problem domain, the dataset is published and is publicly available[23].

To produce machine-generated abstracts, an approach similar to Bhat (2023) was adopted. Authentic abstracts were sourced from a publicly available dataset, *arxiv-abstracts-2021*[24] by Clement et al. (2019), which consists of 1 877 550 data points, while machine-generated abstracts were produced using the titles of authentic abstracts for context. To reduce superficial characteristics in machine-generated abstracts due to potential biases within the generator model regarding research abstract lengths, a target word count equal to that of the corresponding authentic abstract was provided. Additionally, to minimize the possibility of authentic abstracts being contaminated with machine-generated segments, all sourced abstracts are dated 2021 and earlier, prior to open access to state-of-the-art generative models such as GPT-3 (November 18, 2021), BLOOM (July 26, 2022), LLaMa (February 23, 2023) and GPT-4 (March 14, 2023).

Similarly to *text-curie-001*, abstracts are generated with *gpt-3.5-turbo-0301* through prompting. While *text-curie-001* is optimized for text completions, *gpt-3.5-turbo-0301*

[21] huggingface.co/datasets/aadityaubhat/GPT-wiki-intro
[22] huggingface.co/datasets/wikipedia
[23] huggingface.co/datasets/NicolaiSivesind/ChatGPT-Research-Abstracts
[24] huggingface.co/datasets/gfissore/arxiv-abstracts-2021

is optimized for chat completions, and must be prompted in a dialog format by listing messages and the role of the sender (OpenAI, 2023b). The sender can be one of three roles:

1. *System*
   Used for introducing the conversation
2. *Assistant*
   The communication interface of the model. All generated responses are of this role.
3. *User*
   Represents the user and is used for instructing the model.

The API-documentation states that '*In general, gpt-3.5-turbo-0301 does not pay strong attention to the system message, and therefore important instructions are often better placed in a user message.*' (OpenAI, 2023b). Through careful experimentation and adoption of this finding, the instruction-prompt-sequence displayed in listing 2 was found to produce desired results with regard to response consistency, format, phrasing, and word count.

```
"messages": [
    {"role": "system",
     "content":
        "You are a helpful assistant which produces
         ↪ abstracts for research papers based on a title
         ↪ and a length. Your task is to produce the
         ↪ abstract which suits the title, and is of the
         ↪ desired length in words."},

    {"role": "user",
     "content":
"""Title: \"{title}\"
Abstract length: {word_count_goal} words

---

Above is the title of a scientific research paper and the
 ↪ length of its abstract. Using a formal/academic
 ↪ language, write a new abstract which matches title and
 ↪ has a length of {word_count_goal} words. Do not answer
 ↪ with anything else than the abstract."""}]
```

Listing 2: GPT-3.5 chat completion input texts

The subset of authentic abstracts used in our dataset has been selected using a uniform distribution with respect to word count. This should enable comparison of detection accuracy related to text lengths as all available texts-length is represented as evenly as possible. The script used to perform a uniform word count selection is listed in appendix A. The hyperparameters are listed in appendix 3. All hyperparameters are the default values of the API with the exception of *model* and *prompt* which have no default value. Their function is further described in section 6.1.1, where these are modified from their default values.

```
hyperparams = {
            "model": self.MODEL,
            "prompt": input_text,
            "max_tokens": inf,
            "temperature": 1,
            "top_p": 1,
            "logprobs": 0,
            "logit_bias": null,
            "n": 1
        }
```

Listing 3: GPT-3.5 chat completion prompt

5.3 Data cleaning

To ensure high data quality and minimize any superficial characteristics which may reveal the source of the texts, we have performed these data cleansing measures:

1. *Pre-selection evaluation*
   The source dataset, *arxiv-abstracts-2021*, was inspected to determine the interval for selection.

2. *Duplicate check*
   Identified and removed any duplicate entries in the dataset, ensuring that each data point is unique and not artificially inflating the dataset size. This process aids in preventing overfitting during training and improves the overall reliability of the classification models.

3. *Replacement of outliers*
   Manually went through a feasible amount of data points and removed any outliers with regard to faulty content. Removed outliers were replaced using a substitution script. All replacements were manually inspected.

4. *Manual inspection*
   Manually went through a feasible amount of data points to identify any discrepancies with regard to formatting between the human-written and machine-generated abstracts.

5. *Automatic correction*
   Based on the discrepancies found during the *manual inspection 1*, a script to reformat

all samples and ensure consistency across the entire dataset was produced. The script automates the process of correcting any general formatting discrepancies and applies the necessary changes to both the human-written and machine-generated text samples. The cleaning function used in the script is listed in appendix B.

6. *Post-correction recount*
   After all data-cleaning measures were applied, all human-written and machine-generated abstracts were recounted with regard to word count.

7. *Post-cleaning inspection*
   Manually went through a feasible amount of data points to identify any discrepancies with regard to formatting between the human-written and machine-generated abstracts post-cleaning for further statistical analysis.

5.4 Optimizing dataset design for targeted learning

When building machine learning models for a particular problem, the dataset design should be optimized to target relevant features within the data. We have done this by reducing the number of features which is less related to the targeted problem; All machine-generated text samples are produced using features derived from their human-written counterpart. This includes a common style domain, a common descriptive feature, and in the case of ChatGPT-Research-Abstracts, a common text length. This design aims to incentivize the classification models to learn and discriminate features in underlying structures rather than identifying content and superficial characteristics.

To use the datasets for supervised learning and performance evaluation, each text sample is separated from its human-written/machine-generated counterpart and labeled as either human-written (0) or machine-generated (1). This effectively doubles the amount of data points in the original dataset. Although the text samples are reformatted into separate labeled data points, each is distributed along with its counterpart into the same dataset subset split where all human-written data points are located on even indices, and corresponding machine-generated datapoints on the succeeding odd indices. This splitting policy ensures that the detection models do not identify the content of a datapoint based on prior exposure to the counterpart during training, thus preventing the model from being biased towards the label it previously encountered, leading to more reliable classification performance.

To promote experimental control in the evaluation of generalization capabilities, a train-validation-test split distribution has been adopted. Subsetting the dataset for various purposes is to avoid overfitting and ensure model performance is evaluated using unseen data (generalization). Having a separate validation-split enables tracking of performance and adjustment of hyperparameters during training, while also reserving a subset for the final performance evaluation.

6 Setup for detection approaches

In this section, the various approaches and their corresponding implementations used for performance-evaluation is presented. First, the setup for the in-context learning approach is described, followed by a description of the setup for fine-tuned detection experiments.

6.1 In-context learning

In section 4.3.4, several strategies for optimizing in-context learning performance were presented. These have been taken into account and applied, some with slight modifications.

For evaluating the use of in-context learning for the classification of real and generated texts, two distinct approaches have been applied, which will be referred to as:

1. **Human in-context learning**
   For this approach, the input texts used for the in-context learning classification task are human-written through iterative experimentation and testing.

2. **Inductive in-context learning**
   For this approach, *inductive instructions* proposed by Honovich et al. (2022) has been applied. The input texts used for the in-context learning classification task are entirely generated by the same model which performs the classification task.

Both of these approaches are tested under zero-shot and few-shot settings, resulting in a total of four variations of input texts and 4 performance evaluations.

For these approaches, monetary costs set limitations for the amount of performance evaluation possible. Due to this, the CRA-dataset was prioritized for the evaluations as it is the most relevant for the problem domain of the thesis. Each distinct variation of the in-context learning approaches was tested on 1000 data points each: 500 real abstracts, and 500 generated abstracts. In few-shot settings, six labeled examples were provided for each classification task, three real abstracts, and three corresponding generated abstracts. The selection of examples was done using the closest neighbors in terms of the word count of the classification text. No abstracts were used twice. This resulted in a total of 7000/1000 rows being used from the CRA-dataset for each of the two few-shot variations. In addition to this, to evenly represent all small-deviating real-generated abstract pairs, the classification texts, and corresponding examples were uniformly sampled from CRA with respect to word count within the range $\mathrm{WC} \in [50, 325]$. For clarification of why 325 was selected as an upper limit, see section 7.1.2. All examples and classification texts were also stripped of any non-space sequences of white-space from previous performance evaluations. This is due to the finding of clear white-space usage patterns in the generator model, documented in section 7.1.4. In zero-shot settings, the same example-classification-text bundles were used as for few-shot settings, but omitting the examples and only using the classification text. This is to ensure the performance evaluations are performed using the exact same set of classification texts. The number of six examples was chosen mainly due to the CRA-dataset only containing 10k datapoints. For each few-shot attempt four rows of the dataset were used, limiting the possible unique few-shot task bundles to six examples each. In addition to this, the selected model has a context window size of 4096 tokens. The longest 325 abstract word count few-shot input text equals to 3128 tokens, ultimately nearing the limit of the model.

The following sub-sections present the in-detail methods of the approaches and their n-setting-variations.

6.1.1 Model selection and hyperparameters

Brown et al. (2020) documented that in-context learning performance scales with the size of the model. To be able to document state-of-the-art in-context learning performance in an open-access model, a flagship-version[25] was the desired option. Due to the same constraints regarding hosting the largest open-source language models such BLOOM (176B) on Idun, which is stated in the first paragraph of section 5, a similar approach as for the generation of machine-generated samples in *ChatGPT-Research-abstracts* has been used. Through the commercial API of OpenAI, the model *text-davinci-003* was used to perform the in-context learning. While using the ChatGPT model *gpt-3.5-turbo-0301* (175B) for production of the *ChatGPT-Research-Abstracts* dataset, for this task the instruction-optimized GPT-3 model *text-davinci-003* (175B) is used. This is mainly due to the instruction-optimized GPT-3-series allowing retrieval of model logits while the GPT-3.5 series does not. Retrieving top model logits enables the calculation of model confidence scores for each of the classes in the class domain. Although this approach enables evaluation of state-of-the-art open-access in-context learning performance, there are constraints regarding this approach; the API limit logit retrieval to the top five logit pools.

In listing 4, the hyperparameters are presented. Following are definitions of the hyperparameters and the reasoning behind the set value:

- *model*:
  The name of the model to be used. At runtime, this is ‘text-davinci-003’
- *prompt*: The input text for the model to process. This is a formatted string that includes instructions for the model and any labeled examples, depending on the n-shot setting and input text approach. The exact formatting of the input texts is presented later in this section.
- *max tokens*:
  Restricts model output length. Value is set to 1 such that the model is restricted to respond with a single token: the predicted class label of the classification text.
- *temperature*:
  Temperature governs the randomness of the token selection algorithm. High values promote diversity in answers, low values promote determinism. The value is set to 0, forcing the model always respond with the token with the highest logit value and subsequently the one with the highest confidence score.

[25]the largest version of the model in terms of parameters

- *logprobs*:
  The number of model logits to retrieve for the generated token. The returned logits are the ones of the highest value. The maximum allowed value is 5. All logits are returned as logarithmic values.

- *logit bias*:
  Accepts a dictionary of token-IDs as keys and a value $\in [-100, 100]$. The assigned values represent a bias added to the corresponding token-IDs prior to processing the input text, ultimately governing the likelihood of the tokens appearing in the top 5 logits returned in the response. To ensure that both class label tokens are present in the top 5 returned model logits, logit biases of 100 were added for each of them. However, the model tended to respond with newlines and other words to form sentences. Logit biases of -100 were added to all common tokens appearing in the top-5 logits during testing. The code section which produces all logit biases passed to the model can be found in appendix E.1. It should be noted that adding biases prior to input text processing does not affect the calculated confidence scores as the bias for each of the class labels is of equal value, and the confidence scores are calculated using the corresponding class label token logits in relation to each other. Confidence scores are further explained in 6.1.4

- *n*:
  The number of attempts, and subsequently how many outputs to be produced for each input text. This is set to 1 as only one attempt for each input text is sufficient in this approach.

```
hyperparams = {
                "model": self.MODEL,
                "prompt": input_text,
                "max_tokens": 1,
                "temperature": 0,
                "top_p": 1,
                "logprobs": 5,
                "logit_bias": self.logit_biases,
                "n": 1
          }
```

Listing 4: GPT-3 instruction hyperparemeters

6.1.2 Input text formatting

As presented in section 4.3.4, the in-context learning is sensitive to the formatting of both the prompt and demonstration design. Due to this, two separate approaches were implemented for comparison. The first approach is using an input text which is human-written through iterative testing. The second approach is using the concept of *inductive instructions* proposed by Honovich et al. (2022) in conjunction with minimizing perplexity of the input text with respect to the detection model as Dong et al. (2023) discussed in section 4.3.4. In addition to this, we extend the concept of *inductive instructions* by allowing the model to also position the examples as it saw appropriate and also assigning its own labels, ultimately designing the entire input text. Each approach is presented below, followed by the common measures applied for both approaches.

The human-written input was carefully crafted by thoroughly testing variations of the input text through iterative testing and adjustment. Each version of the input text was tested for 30 samples at a time for both zero-shot and few-shot settings before being adjusted for further testing. Variations of logit biases both negative and positive were also experimented with. The input text, including the prompt, was sculpted to be well-formulated, clear, specific, and concise, as Dong et al. (2023) found to be optimal in section 4.3.4.

For the inductive input text, *text-davinci-003* was tasked with generating a suitable input text, given the context of the task and three examples of each class. The examples included real and generated versions of the same research abstracts extracted from *ChatGPT-Reseach-Abstracts*. The ”text-davinci-003” was chosen to generate the input text to minimize potential perplexity within the prompt. As only the top five logits are retrievable from the model, an exact perplexity score cannot be calculated. However, using the same model performing the task for the creation of the input text to be used, should still reduce its perplexity since is writing an input text for itself. The same positive logit-biases that were added for *human in-context learning*, were added to the model's chosen class labels in addition to the negative logit-biases for the same tokens as with the human approach. See logit biases previously presented in section 6.1.1.

For both approaches, further general measures were applied. Tips listed in an OpenAI API guide (OpenAI, 2023b) regarding optimizing labels for proper tokenization have been implemented:

- All labels are words that are handled as a single token in the tokenizer.
- All labels are capitalized.
- All labels are prepended with white space to ensure correct tokenization of the class labels.

During the analyses of the returned log probabilities when experimenting with the human-written input text, the model had higher token logit values for the labels without presentation despite being instructed to respond with exact labels. For collecting the prediction, the unprepended versions were therefore used as the model displayed bias for using the unprepended labels. Prepended versions were still used for labeling examples to ensure proper tokenization. To clarify, only the instruction section of the input text, the prompt, was formulated using labels without prependations.

The final input texts used for each of the approaches are listed in appendix E.2. In addition to this, the input text used for generating the inductive input text is listed in appendix E.3

6.1.3 Selection of labeled examples and demonstration design

To select the labeled examples for the few-shot setting Liu et al. (2021)'s proposed method of using closest neighbors was applied. As metric, the word count of the real abstracts was used. All labeled examples were the closest matches to the classification text in terms of word count. In total, six examples were selected and added for each separate classification task and no examples were used multiple times. To promote targeted learning of underlying discrepancies between the real and generated examples selected from the closest neighbors with respect to content, the generated abstracts corresponding to the selected real abstracts were used for the labeled examples of generated text. This can also be viewed as selecting the closest neighbor with respect to content.

The demonstration design, which refers to how the examples are presented and how many, is similar, but slightly different for the human-written input text and the inductive input text. In the human approach, we presented the examples at the start of the input text without introducing the context; the context of the task was given along with the instruction prompt at the end of the input text. In the inductive approach, the context of the task was presented before the labeled examples. In addition to this, a supplementary instruction prompt for the classification task was present at the end of the input text.

For labeling the examples in the Human approach, '*Human*' and '*Machine*' were used as class labels. In the Inductive approach, the labels '*Human*' and '*AI*' was selected by the model.

6.1.4 Scoring function

To evaluate the model's confidence of its prediction the *direct* scoring approach presented by Dong et al. (2023) was adopted. As mentioned, without access to all model logits, perplexity cannot be calculated. The direct approach is still arguably the optimal candidate of choice as we enforce the model to answer the predicted label using logit biases. The catch which may affect the performance in terms of accuracy, is that the model is not allowed to elaborate or reflect before providing an answer.

To calculate the confidence scores using the *direct* approach, the model logits for each of the class labels representing the likelihood of the model predicting the text as either human-written or machine-generated were extracted, followed by applying a *softmax* function. The softmax function creates a normalized probability distribution of the provided logit values in relation to each other. The result is a probability distribution with only two values: the probability of the model predicting the classification text as *human-written* and the probability of it predicting the classification text as *machine-generated*. In total, the probability of both values adds up to 1.0. To clarify, the confidence score of a class label represents the normalized probability of the model predicting that label for the given classification text. For instance, a confidence score of 0.4531 indicates a 45.31% probability that the model assigns the corresponding class label to the text in focus, but it also reflects the model's confidence in the prediction. As the temperature is set to 0, making the model completely deterministic, it will always predict the label with the highest confidence score. Since there are only two classes in the class domain, the selected label will always have a confidence score ≥ 0.5.

6.2 Setup for fine-tuned detection

In the following section, the setup for the fine-tuned detection approach is described. First, a description of the base models used for fine-tuning is provided, followed by a description of the hardware used, and lastly, the setups for training and evaluating the models are described.

6.2.1 The base models

For the fine-tuning experiments, four different language models were chosen. This includes three versions of the Bloomz-models in three different sizes (560m, 1.7b, and 3b), in addition to the RoBERTa-base model. In table 6.1, an overview of these models and their characteristics can be seen.

The Bloomz-models are a series of open-source, autoregressive large language models presented by Muennighoff et al. (2022). The models are fine-tunings of a family of pre-trained large language models called Bloom, and are fine-tuned using 'instruction tuning' (Muennighoff et al., 2022), similar to how InstructGPT (Ouyang et al., 2022) and ChatGPT is tuned. As the ChatGPT-model is proprietary, the Bloomz-models are used instead, as they best represent the capabilities of ChatGPT. The different sizes were used to assess the importance of model size in detection. The RoBERTa model, as first mentioned in section 4.3.2, is an autoencoding, or bidirectional, language model proposed by Liu et al. (2021), and is an optimized version of the BERT model. This model is used as a baseline for testing the Bloomz-models, as it is a much smaller model, and to further investigate the difference in performance between autoencoding and autoregressive large language model when used for detection.

Table 6.1: The models used for fine-tuning and their characteristics.

Model	Size	Training method	Directionality
Bloomz-560m	560M	Autoregressive	Unidirectional
Bloomz-1b7	1.7B	Autoregressive	Unidirectional
Bloomz-3b	3B	Autoregressive	Unidirectional
RoBERTa-base	125M	Autoencoding	Bidirectional

To be able to fine-tune the models for text classification instead of text generation, a classification layer (linear layer) is added in place of the normal generative head[26] of the language model. This classification head is initialized with random weights, while the other layers stay as-is. Thus, the model consists of a pre-trained body[27] and an untrained head. This is all done automatically by the HuggingFace-API (Hugging Face, 2022).

6.2.2 Hardware

Modern state-of-the-art language models can require up to hundreds of gigabytes in storage and memory capacity, and it is reasonable to provide a description of the hardware used

[26]The output layer (last layer).
[27]The input and hidden layers.

for these experiments, as to assess the feasibility of using the same models and thus promote the reproducibility of these experiments. Having adequate hardware resources is imperative to be able to train the models effectively, meaning without running out of memory and to keep training times relatively low. As this thesis is experimental, low training times are essential to be able to effectively build and test the presented methods in the given time period.

For training and evaluation of the fine-tuned models, NTNU's HPC[28]-cluster Idun using the workload manager SLURM was used. For the Bloomz-560m and the RoBERTa model, a V100-16GB GPU in addition to 16GB CPU RAM was found to be sufficient. For Bloomz-1b7 and Bloomz-3b, an A100-80GB GPU along with 50GB CPU RAM was used. Larger Bloomz-models, like the 7b1 and the 173b model, were not possible to run effectively on the hardware used, without advanced parallelization techniques or offloading to disk, which would result in immense training times. The SLURM-scripts containing the resource requests to the HPC-cluster used can be found in appendix H.

6.2.3 Training setup

The models were trained on selections from both the GPT-wiki-intros dataset and the CRA-dataset separately, with the following alterations to the datasets:

- To limit training times, only 10% of the wiki-dataset was used, amounting to a total of 30 000 datapoints, with an equal amount of each label (15 000 each).
- The CRA-dataset was stripped of all newlines and excessive white-spaces to prevent the models from learning on the structure of the text and not the actual characteristics, as mentioned in section 5.4.

Additionally, a mixed dataset was used, containing 50% of the shortened wiki-dataset (15 000 datapoints) and 50% of the CRA-data (10 000), amounting in a total of 25 000 datapoints. This resulted in three datasets used for the fine-tuning experiments, which will be referred to as ”wiki”, ”CRA” and ”mixed” in the context of fine-tuning. Also, as mentioned in 5.4, the datasets were split into three subsets, a train-, validation-, and a test-split. Here, the dataset was split such that the train-split contains 70% of all datapoints, and the validation- and test-splits contain 15% each, for all datasets. All splits also contain the same amount of each label. For the mixed-dataset, the dataset was split such that all splits contained the same distribution of datapoints of the two other datasets, as to prevent bias between the train, validation, and test subsets. An overview of the split sizes is provided in table 6.2.

The models trained on the three datasets, wiki, CRA and mixed, resulted in three types of detectors, henceforth called *wiki-detectors*, *academic-detectors*, and *mixed-detectors*, respectively.

To limit training times, all models were trained for only one epoch, meaning the models would go through all the data in the training-split exactly once. This could lead to the

[28]High Performance Computing

Table 6.2: The dataset splits sizes used for the fine-tuning experiments in number of datapoints.

Dataset	Train-size (70%)	Validation-size (15%)	Test-size (15%)
Wiki	21000	4500	4500
CRA	14000	3000	3000
Mixed	17500	3750	3750

models not properly learning on the dataset, but as we will see in section 7.3, this will not present much of a problem. Additionally, the maximum amount of tokens per input was fixed to 512 tokens, which is also the default maximum amount of tokens for the RoBERTa model, meaning all texts were padded and truncated to be 512 tokens. This also means that the text for each input after 512 tokens are ignored. While the Bloomz-models do not have such a token limit, fixing the tokens was used to further limit the training times, as training times have a tendency to increase exponentially with the number of tokens used, as explained in section 2.8. Furthermore, all models were trained using the same hyperparameters. A full list of the hyperparameters used can be found in appendix G.

6.2.4 Evaluation setup

To effectively assess the performance of the models, they were evaluated using the validation- and the test-split. Firstly, every training run was evaluated using the validation-split 50 times per epoch (once every 0.02 epochs), measuring the performance. This way, the performance of the models could be tracked over the course of one training run, as to determine how fast the model was learning. To enforce that the evaluation is unbiased, the test-split instead of the validation-split is used to measure the final performance of the model after the training run is done. To test the model's ability to generalize the data

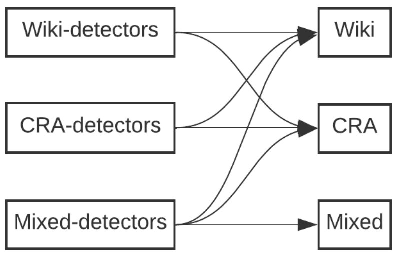


Figure 6.1: Cross-testing scheme used for testing the fine-tuned models on the three datasets, where the detectors (left) are tested on the datasets (right) that they point to.

across domains, meaning cross-dataset and cross generator-architecture, the models were cross-tested on the different datasets. As can also be seen in figure 6.1, the wiki- and the academic-detectors were tested on the wiki and CRA datasets, while the mixed dataset was tested on all three datasets: wiki, CRA and mixed. Finally, the Bloomz-560m model was trained and tested on the wiki and CRA datasets using 1024 maximum tokens to assess the importance of the number of tokens used. The evaluation-sets are padded and truncated the same way as the training-set.

7 Results and analysis

This section presents and analyses the data and results generated by applying the methodology described in the preceding sections.

7.1 Production of the dataset

It is important to analyze the data which is to be used for training and evaluation. Deep analysis can uncover flaws in the data which may influence detection models' internal representation of the two classes, human-written and machine-generated texts, which subsequently could lead to discrimination on incorrect and superficial features of the abstracts. In this subsection, we present and analyze the results of the dataset generation, including the selection process in addition to the generation and cleaning of data related to the production of the ChatGPT-Research-Abstracts (CRA) dataset.

To be able to analyze generator model (gpt-3.5-turb0-0301) characteristics, figure 7.1, 7.2 and 7.3 in section 7.1.1 are produced using the CRA dataset pre data cleaning.

7.1.1 Word count distributions

In section 5.2, we outlined our approach for selecting a subset of data points from the source dataset *arXiv-abstracts-2021* (AA21). Using the code listed in appendix A, a 10k partially uniform subset was sampled with respect to word count (WC) and used as real data points (CRA-Real) in our produced dataset, such that:

$$\text{CRA-Real} \subset \text{AA21}$$

Figure 7.1 displays WC-distributions of the *AA21* dataset, a randomly sampled 10k subset of *AA21*, and the *CRA* dataset with uniformly selected data points and correspondingly generated data points (CRA-Generated). The 10k randomly sampled *AA21*-subset is added to enable a comparison of the *AA21* dataset's natural distribution and the distribution of the selected and generated data points.

The CRA-Real data points with a $\text{WC} \in [50, 360]$ were sampled uniformly. However, for $\text{WC} \in (360, 600]$, the number of CRA-Real data points is underrepresented compared to the ideal average sample size (IASS), which for a 10k subset where all $\text{WC} \in [50, 600]$ is $\text{IASS}_{10} = \frac{10000}{551} \approx 18$. The sharp decline at $\text{WC} = 360$ for the CRA-Real distribution curve is due to reaching the limit of available data points in the source dataset; all data points in *AA21* with a $\text{WC} > 360$ are below IASS_{10}. Subsequently, samples are drawn from word counts with unsampled data points until the subset size of 10 000 is reached, leading to an increased average sample size for data points with $\text{WC} \in [50, 360]$ and a decline in samples where the distribution-curve of AA21 intersects with CRA-Real, at $\text{WC} = 360$.

An analysis of the distribution-curve characteristics offers general insights into the model's performance in generating research abstracts of a specific word count. The distribution

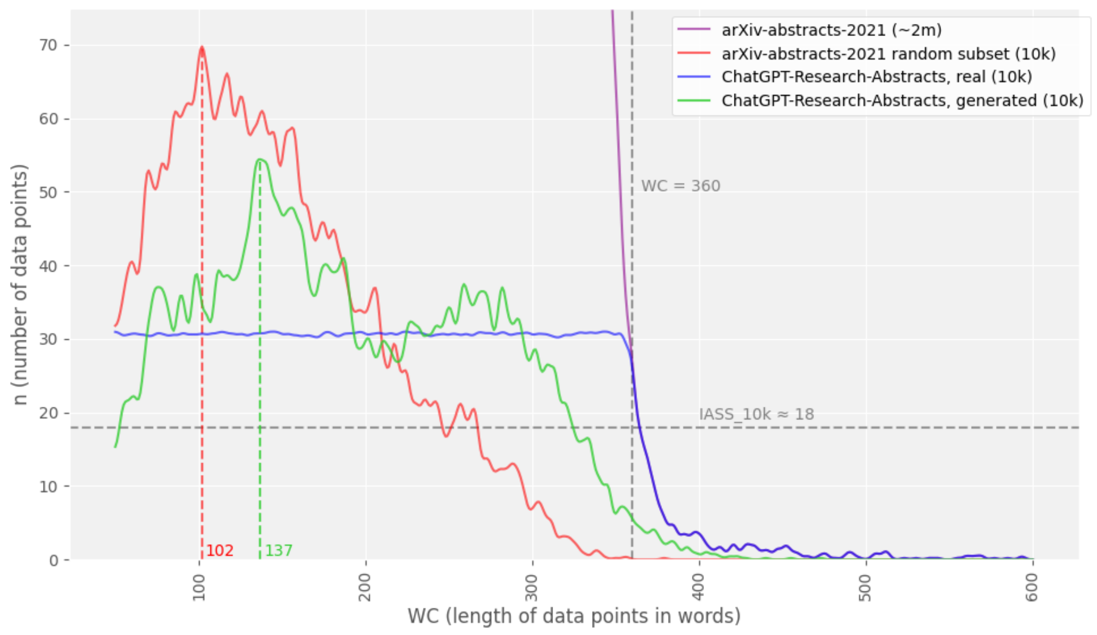


Figure 7.1: Word count distributions for $WC \in [50, 600]$ (pre-clean)

curve of the AA21 subset is positively skewed with a mode of WC = 102. While the curve of CRA-Generated demonstrates similar characteristics, it inhibits a reduced skew in comparison, and subsequently a higher mode of WC = 137. However, in the range of WC ∈ [226, 301], the curve deviates from the AA21 subset-curve and oscillates around the CRA-Real curve, displaying rough alignment with CRA-real which, if consistent, is decent. As figure 1 is a representation of word count distribution, no definite conclusions can be drawn regarding the generator model's ability to generate abstracts of a specific word count as it does not portray the relationship between target word count and actual word count. Beyond WC = 330 the CRA-generated curve declines below the CRA-Real curve, following a pattern similar to the AA21 subset curve.

Although no definite conclusions can be drawn from the distribution curves regarding word count accuracy, CRA-Real distribution curve can still be used as a benchmark, providing general insights into potential biases and limitations of the generator model; in figure 7.1, we can observe that the generator model, in general, has difficulties in adhering to the target word count specified in the generation prompts. For WC ∈ [70, 206], the amount of generated abstracts exceeds the benchmark of CRA-Real, which potentially could implicate a word count bias in the generator model. If we compare the word counts of generated abstracts with their corresponding real abstracts, more accurate conclusions may be drawn.

7.1.2 Generator model's word count accuracy

In figure 7.2, the word count correlations of CRA-Real and CRA-Generated are presented for each encapsulating data point in the CRA dataset. In addition to this, the average correlation curve is calculated using the average y-value for each unique x-value.

As the generator model was prompted to generate abstracts with word count equal to a corresponding CRA-Real data point, the average correlation curve should ideally follow the pattern of an identity line, which for comparison is listed as ”perfect correlation” in figure 7.2. Instead, the average word count of CRA-Generated declines as the corresponding CRA-Real word count increases, subsequently reducing the correlation to the target word count. The word count correlation of the entire dataset is in total $r = 0.96$, but for the intervals $x \in [325, 420]$ and $x \in [420, 600]$ the correlation index drops to respectively $r = 0.20$ and $r = 0.09$. This finding suggests that the generator model's word count accuracy deeply declines when generating abstracts of x > 325.

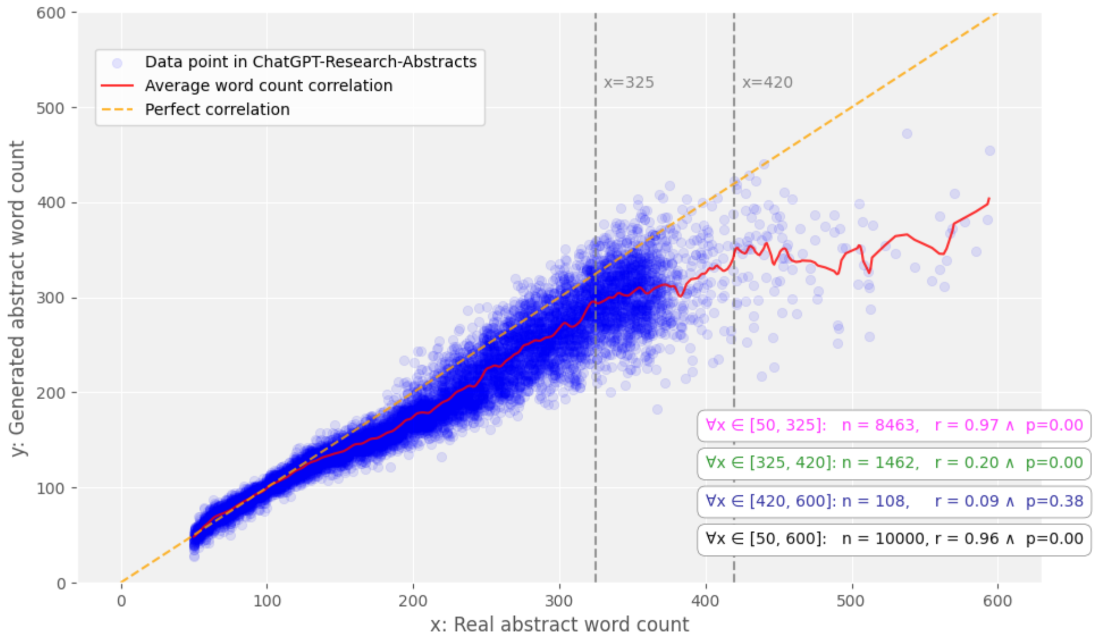


Figure 7.2: Comparison of real and generated abstract word counts (pre-clean).

In total, there are only 108 observations within $x \in [420, 600]$, and thus there are comparably fewer observations in this interval. We notice that the curve is less consistent which also may further implicate uncertainty regarding the observations within this interval. The corresponding p-value of $x \in [420, 600]$ indicates that the probability of observing a correlation index of $r = 0.09$ under the null hypothesis $r = 0.00$, is 38%. This is greater than the typical significance threshold of $p < 0.05$, which consequently renders the finding insignificant; more observations are needed to conclude any correlation in this interval. On the other hand, the p-value of the interval $x \in [325, 420]$, $p = 0.00$ is significant. The correlation of $r = 0.26$, indicates that when prompting a target word count in this interval, the generator is imprecise in terms of word count accuracy, typically by producing shorter abstracts than requested. For the interval $x \in [50, 325]$ the generator model is comparably more accurately supported by a significant correlation of $r = 0.97$.

To get a better understanding of the generator's word count accuracy, we can use the absolute target word count deviation to measure the size of inaccuracy in relation to target word count (TWC); larger absolute deviations indicate lower accuracy. In figure 7.3, the average target word count deviation of generated abstracts for each unique x-value is presented. The majority of generated abstracts are negative deviations, meaning that the generator produces abstracts shorter than the target word counts. As listed in the figure, a total of 7942 are negative deviates, 1842 are positive deviates, and 216 are non-deviates meaning they hit the target word count.

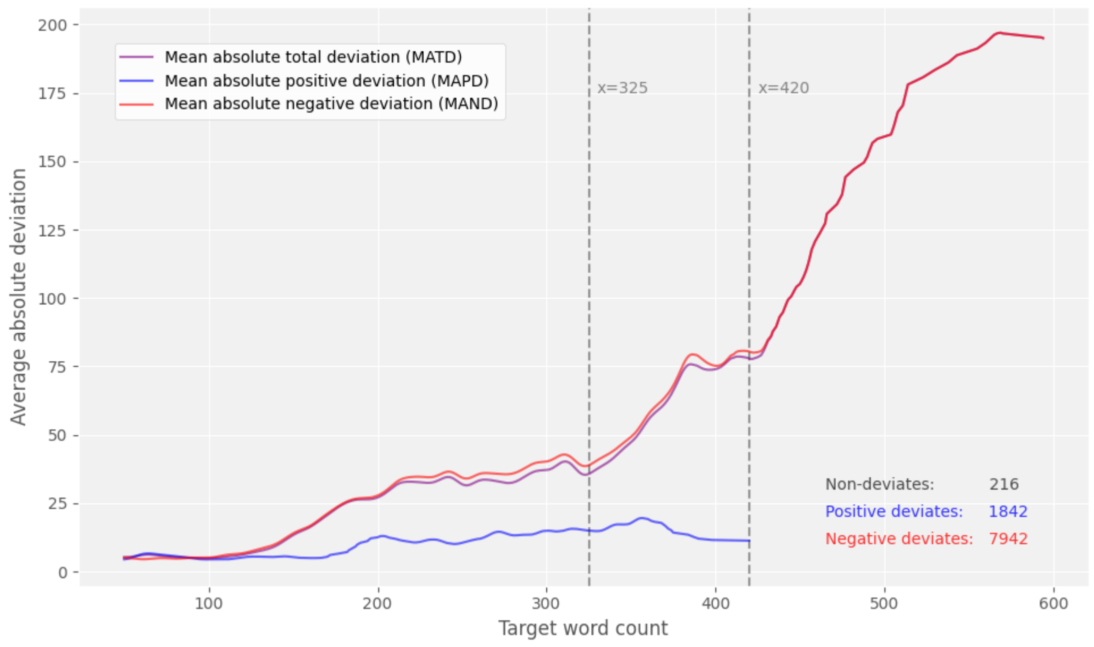


Figure 7.3: Mean absolute deviations from target word count for generated abstracts (pre-clean).

Figure 7.3 shows that the mean average deviation increases as the target word count (TWC) grows. For TWC < 130, the mean average deviation for all curves are below a threshold of 6. This aligns well with the distribution curves in figure 7.1. For TWC $\in [130, 220]$ this threshold increases to 30 for *mean absolute negative deviation* and *mean absolute total deviation*, while *mean absolute positive deviation* stays below a threshold of 25 for all ranges of TWC. In addition to this, all generated abstracts with TWC > 420 are shorter than their target word count. The word count accuracy of the generator model stabilizes for TWC $\in [220, 325]$, before steeply increasing for all TWC > 325, which aligns well with the correlation decline in figure 7.2.

Based on these findings, a conclusion can be drawn regarding the generator's word count accuracy. While it in general is accurate for generating abstracts shorter than 325 words, it struggles in producing abstracts larger than this. The majority of the time, the generator produces shorter abstracts than the target word count; the amount of abstracts exceeding their target word count reduces as the target word count grows, while the size of deviation increases.

7.1.3 Data cleaning

The results from our data cleansing measures were as follows:

1. *Pre-selection evaluation*

We found that the abstracts from AA21 had not been previously cleaned. Text-wrapping and inconsistencies in white-space usage and paragraph separation-style were still present in the original papers. In addition to this, the majority of the abstracts with WC < 600 had been extracted incorrectly leaking into other sections of its corresponding research paper. A portion of the abstracts shorter than 50 words were also contaminated. The selection interval was therefore set to $[50, 600]$.

2. *Duplicate check*
   No duplicates were found during the post-selection duplicate checks.

3. *Replacement of outliers*
   There were in total 23 outliers removed due to the content being faulty, such as leakage into introductions. These were all found in the range of WC $\in [380, 400]$. All outliers were replaced with a new WC-uniform subset with WC $\in [50, 360]$, using a separate version of the uniform distribution script listed in appendix *B*, with modifications such as a duplication safeguard.

4. *Manual inspection*
   There were considerable differences between CRA-Real and CRA-Generated with regard to formatting. While CRA-Real varied in format, the CRA-generated followed a similar pattern using double-newlines as paragraph breaks. In addition to this, we found that multiple abstracts with WC > 400 contained a second version of the abstract in another language. The foreign language section was manually removed.

5. *Automatic correction*
   The cleaning script leveled the majority of formatting discrepancies found during the manual inspections. There were still some visible differences between CRA-Real and CRA-generated; see *post-cleaning inspection*.

6. *Post-correction recount*
   As all abstracts were recounted post-cleaning, word counts were adjusted to reflect the abstracts post-automatic and manual correction. For the abstracts which contained foreign translations, the word counts were considerably reduced. As most generated abstracts for this word count interval had large negative deviations from their target word count, the absolute deviation is not of concern.

7. *Post cleaning inspection*
   There were still some visible differences between CRA-Real and CRA-generated. Through the post-cleaning inspection, we noticed that CRA-Generated seemed to have a visibly higher frequency of paragraph breaks for structuring the texts.

7.1.4 Post-cleaning differences in semantics

As mentioned in section 5.4, designing the dataset to target underlying structures in the texts rather than superficial characteristics is desirable for accurate performance evaluation. Increased use of paragraph breaks (PB) in CRA-Generated compared to CRA-Real may influence the detection model's internal representation of generated texts as detection models may correlate a high frequency of PB as a characteristic of the generator model.

Through a statistical analysis following the discoveries made in section 7.1.3 regarding PB-frequency during the data cleaning procedure, we have confirmed that there indeed are clear and distinct patterns of PB-occurrences present in the dataset, although the differences are not as prominent as initial concerns.

In figure 7.4, the average words per paragraph for each of the abstract domains are presented. The linear clustering of observations is due to them being sorted by word count in ascending order along the x-axis. This leads to discrete domains of linear clusters, each representing how many PBs there are in the text; the upper cluster line represents all abstracts without any PBs, which results in a 1:1 relationship between *words per paragraph* (y-axis) and *word count* (x-axis).

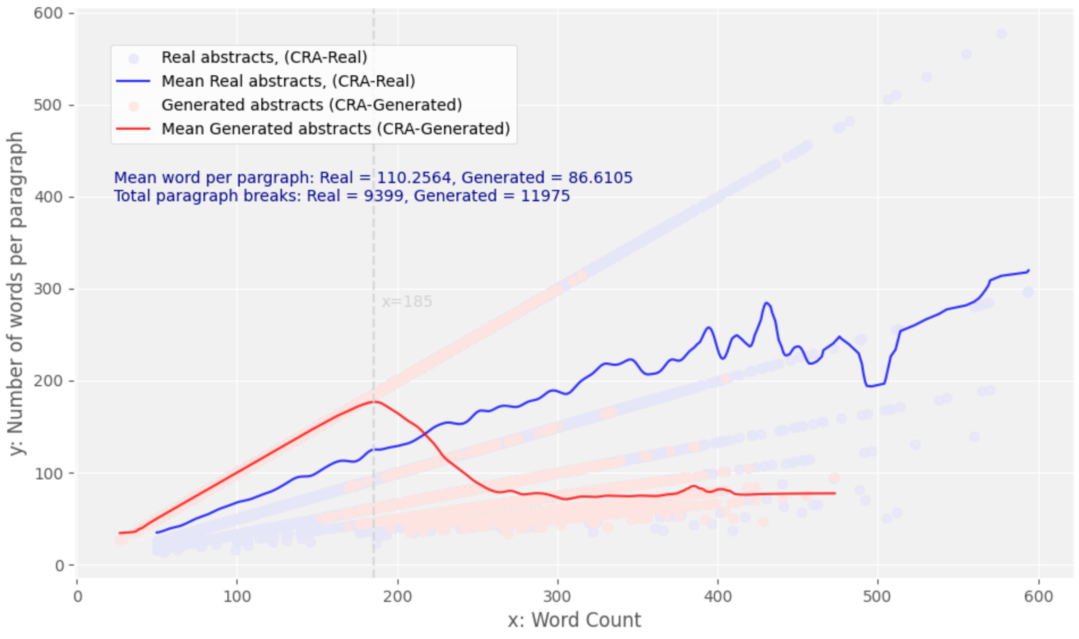


Figure 7.4: Number of words per paragraph (post-cleaning)

On average, the human-written abstracts seem to have a positive linear relationship with regards to *words per paragraph* as the *word* grows, while the generator model displays a PB-pattern resembling a sigmoid-curve with regards to word count. For $x < 185$ the large majority of the abstracts are a single-paragraphed. Beyond this, the generator model drops and stabilizes at approximately 87 words per paragraph on average. For all generated data points with WC < 185 only 27 out of 5169 are separated into two or more paragraphs (number of PB > 0). For the real data points that number is 2543 out of 4138. Using this statistic alone; predicting all zero-PB abstracts as generated, a detection accuracy of 78% may be achieved for WC < 185. This important statistic is present in figure 7.4, and highlights the importance of analyzing the data which is to be used for training and evaluation. As large language models are able to detect structures comparably less apparent, characteristics such as these may serve as hints and subsequently ease the task of detection, resulting in inflated performance evaluation as compared to removing this feature from the texts altogether. In section 7.3 we will test if this affects detection performance.

7.2 In-context learning

In section, 6.1 two approaches for in-context learning were presented, each approach was also tested in both zero-shot and few-shot settings, resulting in 4 variations of in-context learning performance-evaluations for the CRA dataset, *ChatGPT-Research-Abstracts*. For the metric calculations in this section, true machine-generated classifications texts are set as the true positives, and true human-written classification texts are set as true negatives, meaning the metrics *precision*, *recall*, and *F1* scores are all calculated in relation to its performance for machine-generated classification texts [29].

Without applying any performance optimization techniques with regards to the interpretation of confidence scores, the results are as presented in table 7.1. Here, the threshold for the model predicting a text as machine-generated was set to 0.5, following the standard of typical binary classification. In simple terms, if the model's confidence score of the text being machine-generated is greater than 0.5, the classification text is considered to be predicted as machine-generated. For confidence scores lower than this, the classification text is considered to be predicted as human-written. This also reflects the natural predictions provided in the model output.

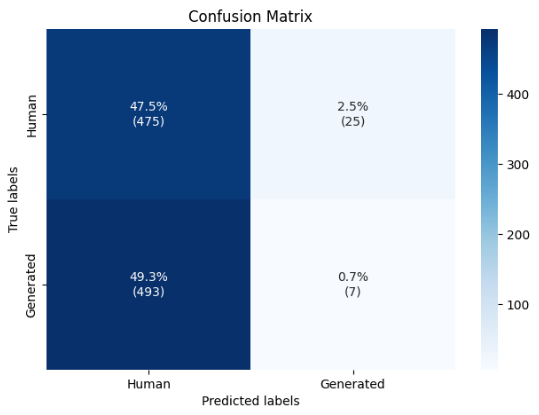
Table 7.1: Metric scores for the in-context learning approaches with 0.5 as machine-generated threshold, which later is referred to as GCS Threshold).

Approach variation	Accuracy	Precision	Recall	F1-score	Threshold
Human-zero-shot	0.482	0.219	0.014	0.026	0.5
Human-few-shot	0.474	0.432	0.166	0.24	0.5
Inductive-zero-shot	0.5	0.0	0.0	0.0	0.5
Inductive-few-shot	0.505	0.532	0.084	0.145	0.5

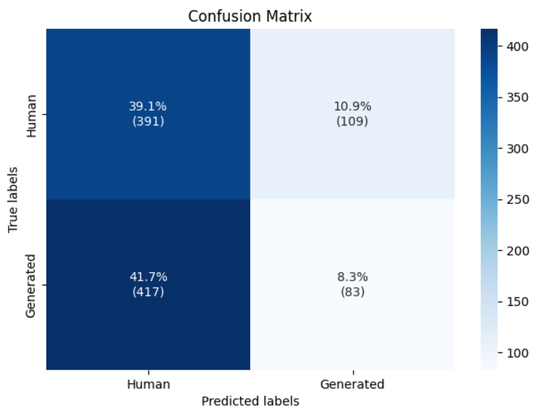
From metric scores presented in table 7.1, we can see that for all approach variations, the model has an accuracy close to 0.5. This indicates that model in general struggles to identify any features in the provided classification texts, even in a few-shot setting where 6 labeled examples are provided in the input text. For the human approach variations, the accuracy is even below 0.5. This could potentially indicate that the model does identify some discrepancies between human-written and machine-generated texts, but it fails in correctly allocating the differences to the correct class labels.

To provide further insights into how the predictions are allocated with respect to the true label of the classification text, confusion matrices for each of the approaches are presented in figure 7.5. From the figure, it is apparent that for all approaches, the model is largely inclined towards predicting the classification texts as human-written, regardless of their true class label. There are still differences, especially when comparing the performance of few-shot and zero-shot settings. In the human-zero-shot approach, 96.8% of all classification texts are predicted as human, while in the inductive-zero-shot approach, 100% of the classification texts are predicted as human, meaning all 1000 classification texts provided were labeled as human despite half being machine-generated. In few-shot settings the performance is comparably better; the human-few-shot approach predicts 80.8% of the classification texts as human-written, while the inductive-few-shot approach classifies 92.1% of the texts as human-written. Although this is higher than in

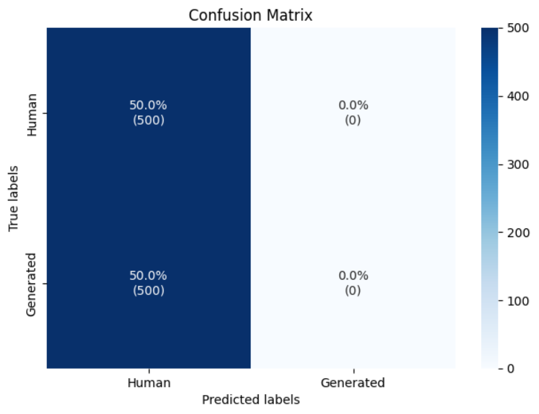
[29] Accuracy is not calculated in relation to any label. It is calculated for the entire pool of performances.



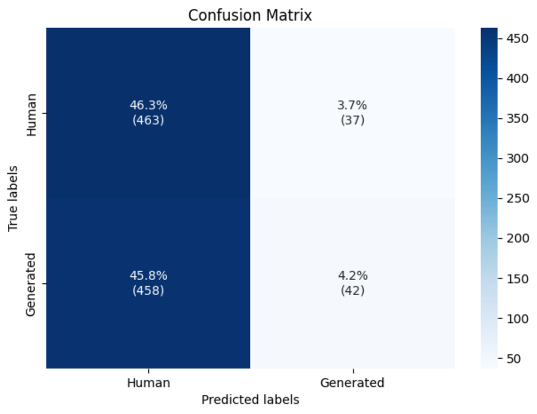
(a) Human-zero-shot



(b) Human-few-shot



(c) Inductive-zero-shot



(d) Inductive-few-shot

Figure 7.5: Confusion matrices for each of the in-context learning approaches.

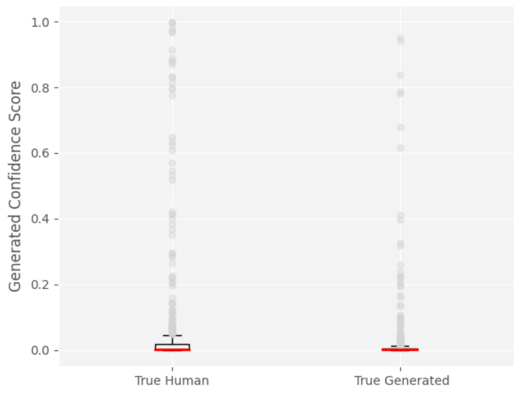
human-few-shot, the inductive-few-shot approach has a smaller portion of false positives (top right corners) compared to true positives (bottom right corners), which when put in the context of the thesis problem domain, may be considered the most important factor in detection models.

Although the model struggles to perform its task through its natural in-context predictions, better performance may be acquired by adjusting the confidence score threshold when a classification text is considered machine-generated. By analyzing the allocation of the confidence score for the texts being machine-generated, further insights into how to adjust that threshold may be provided.

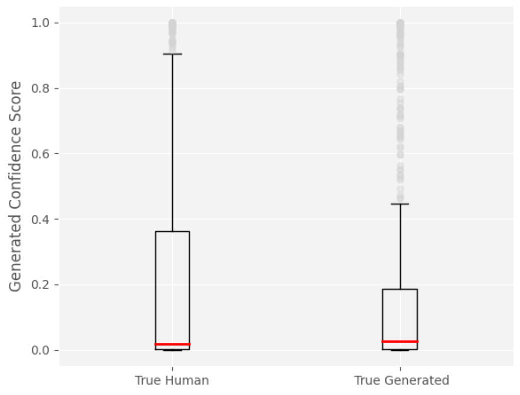
In figure 7.6, the confidence scores for the texts being machine-generated are compared to their true label. We will refer to the model's confidence score for a classification text being machine-generated as the *machine-generated confidence score* (GCS) and the confidence score for it being human-written as *human-written confidence score* (HCS). For clarity, as the confidence scores are calculated in relation using a softmax-function, it should be noted that the generated confidence is equal to the inverse of the human confidence score and vice versa:

$$\text{GCS} = 1 - \text{HCS}$$

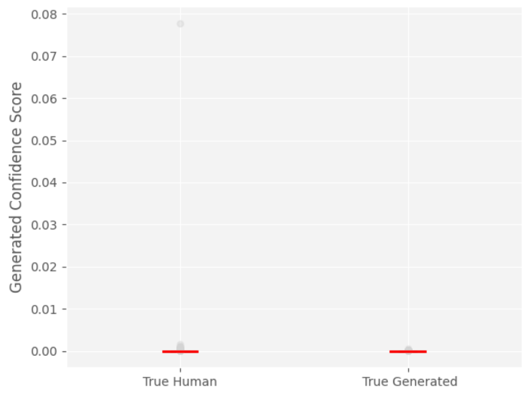
Whether the confidence scores are analyzed through GCSs or HCSs is result-wise arbitrary, but as the thesis problem domain is focused on MGT-detection, further analysis of in-context learning performance will analyze model confidence through GCS.



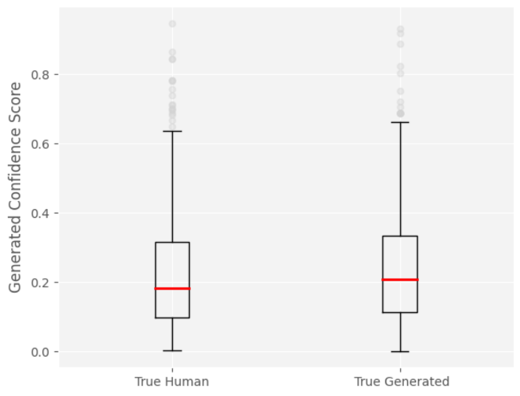
(a) Human-zero-shot



(b) Human-few-shot



(c) Inductive-zero-shot



(d) Inductive-few-shot

Figure 7.6: Box-plots of confidence score of a text being generated for each true label

From figure 7.6, we can see that in the zero-shot settings, the model is certain of the large majority of texts being human-written. In human-zero-shot, the model's GCS is spread more evenly than in inductive-zero-shot where all GCSs are located below 0.005 (0.5%) confidence, while in human-zero-shot, outliers are allocated across the entire range of $[0, 1]$ (0%–100%). In few-shot settings, the model is in general less biased toward the classification of texts being human-written. This is represented by the upper and lower quartile edges being further apart. In the inductive-few-shot, both the third and second quartile is shifted positively, subsequently also increasing the median GCS. The lower quartile edge is for inductive-few-shot located at 0.09 GCS for human classification texts and 0.11 GCS for the machine-generated classification texts, indicating that 75% of the GCSs for each category are above these thresholds. In addition to this, in inductive-few-shot, the model has slightly higher confidence in machine-generated classification texts being generated than human-written classification texts being generated, implying that it in some cases displays the ability to identify discrepancies between the human-written and machine-generated texts. The takeaway is that in few-shot settings, the in-context learning approaches are generally more confident of the text being machine-generated than in zero-shot settings. This may show a potential of increasing accuracy if the GCS threshold of a text being machine-generated is adjusted for the general bias towards texts being human-written for each respective approach variation.

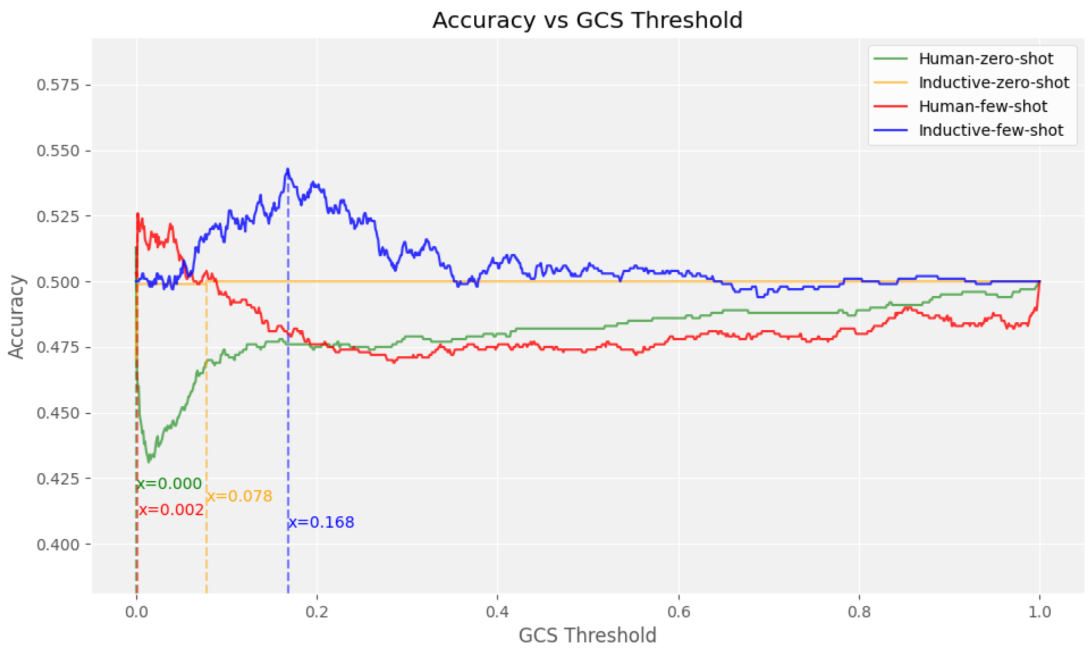


Figure 7.7: In-context learning accuracies for all thresholds ∈ [0, 1], step size = 0.001.

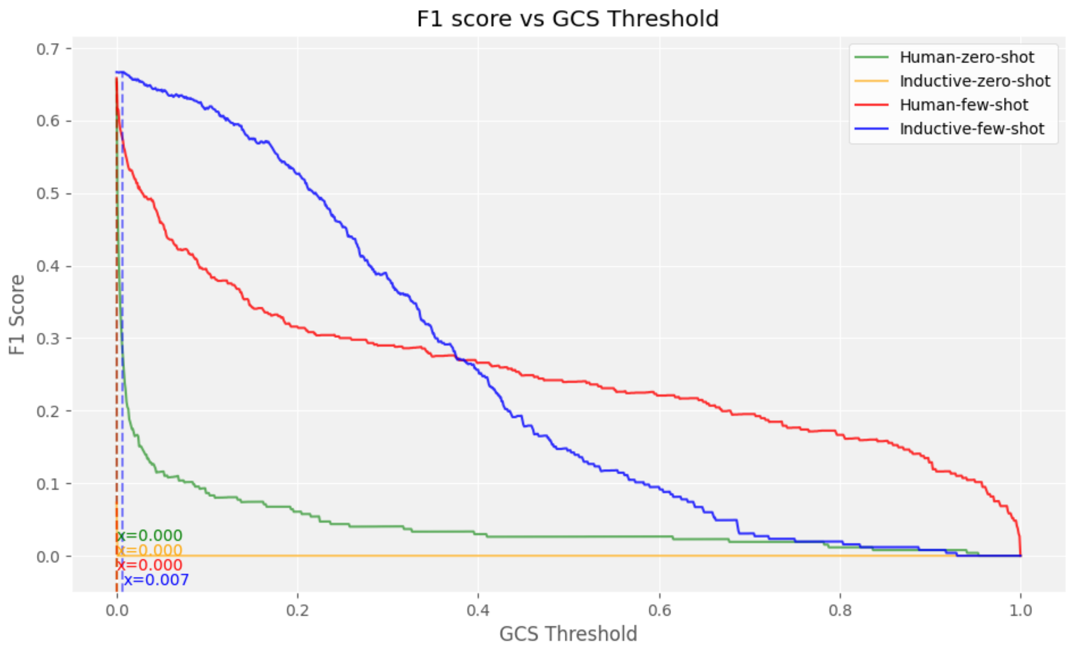


Figure 7.8: In-context learning F1 scores for all thresholds ∈ [0, 1], step size = 0.001.

In figure 7.7 and 7.8, the respective accuracies and F1 scores when adjusting the GCS threshold for when to predict a classification text as machine-generated is presented. From figure 7.7, we can see that there are gains to be made with regard to accuracy by adjusting the GCS threshold in few-shot settings. For zero-shot settings, no particular gain is made;

classifying all texts as generated produces the optimal accuracy of 0.5. The minimal increase in accuracy at GCS threshold 0.078 for inductive-zero-shot is due to a true human-written outlier being correctly predicted as human-written with a GCS threshold greater than this. The outlier can be spotted among the true human-written labels in figure 7.6c. In figure 7.8, a quick takeaway is that the optimal GCS threshold in terms of accuracy is aligned well with the median GCS for human-written classification texts in figure 7.6.

Despite figure 7.7 presenting optimal thresholds with regards to accuracies, when put in context with the decreasing trend in F1 score as GCS threshold increases displayed in figure 7.8, implies that optimizing the performance for accuracy will subsequently reduce the F1 score. In section 8.2, we discuss the various metrics put in perspective of the thesis problem domain. Here we rank the F1 score as the overall optimal performance indicator when evaluating the performances of detection tools. Despite this, in table 7.2 we present the various metric values when the GCS threshold is set to the accuracy-wise optimal values.

Table 7.2: Metric scores for the in-context learning approaches with 0.5 as machine-generated threshold.

Approach variation	Accuracy	Precision	Recall	F1-score	Threshold
Human-zero-shot	0.513	0.507	0.93	0.656	0.00
Human-few-shot	0.526	0.518	0.756	0.615	0.002
Inductive-zero-shot	0.5	0.0	0.0	0.0	0.078
Inductive-few-shot	0.521	0.514	0.774	0.618	0.168

When adjusting the GCS thresholds in accordance with the confidence score bias' displayed by the model in figure 7.6, we can notice considerably improved performance in all metric scores except for inductive-few-shot precision and all inductive-zero-shot scores where the values are unaffected due to all examples being predicted as human-written for all GCS threshold value $\geq$ 0.078. The confusion matrices when applying the accuracy-wise optimal GCS Threshold is available for interested readers in appendix F.

Although the performance in general is improved with the accuracy-wise optimal GCS threshold values applied, the overall performance is still far from sufficient for application in a detection tool. Despite F1 being discussed as the most important performance evaluation metric for the thesis problem domain in section 8.2, all metric scores should be taken into account when evaluating overall performance. Accuracies close to 0.5 should be considered as insufficient regardless of the other metrics.

Overall, we can conclude that the in-context learning approaches proposed in this thesis do not provide valuable insights into the potential misuse of large language models in academia. Despite this, few-shot settings in general do perform better than zero-shot settings. In addition to this, using an inductive approach for prompt engineering does seem to promote less bias in the model's natural predictions, but once the GCS threshold is updated to account for the bias' within the model, the two input text approaches display similar performance.

7.3 Results from fine-tuned detection

This section presents and analyses the results from the fine-tuned detection experiments based on the setup described in section 6.2.

7.3.1 Runtime and in-training validation results

As to assess the feasibility of fine-tuning a large language model, we will first present the results from the training, including training runtimes and in-training validation.

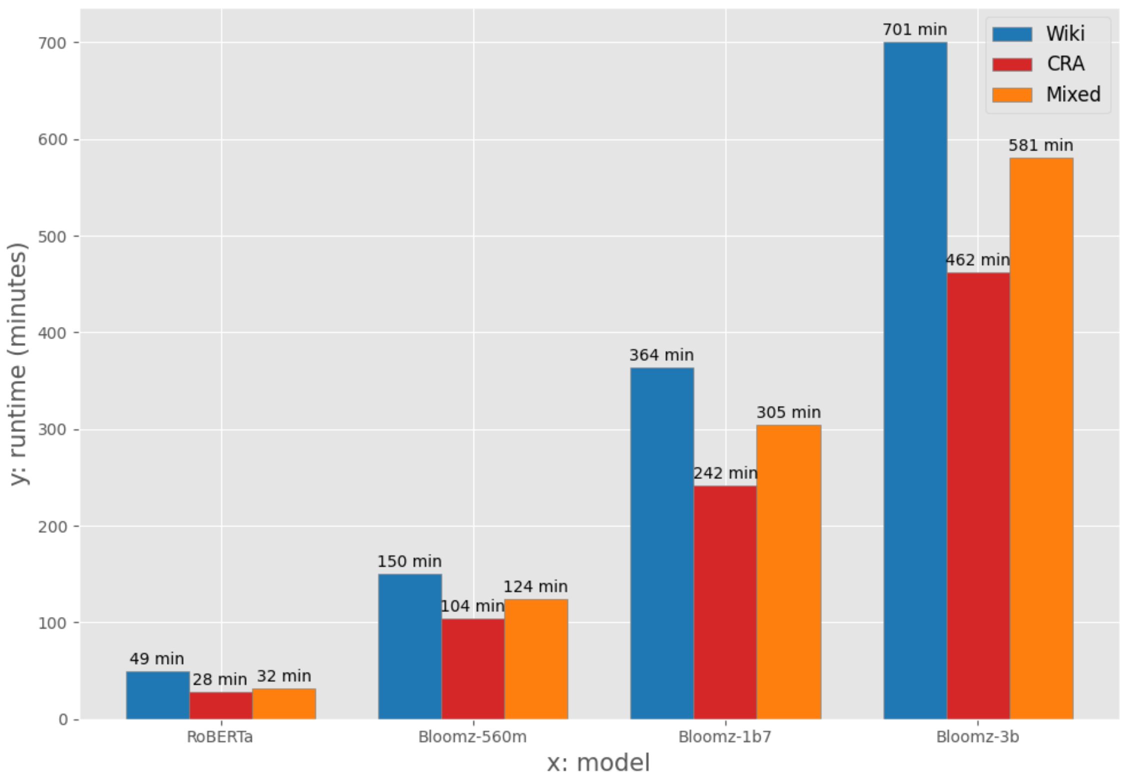


Figure 7.9: Training runtimes (y-axis) for all models (x-axis) on all datasets (color coded) with batch-size=8, all trained for one epoch.

In figure 7.9 the runtimes for the different models can be seen. From this figure, we can see that the larger models had a much higher runtime when using the same system resources with a maximum of 701 minutes (about 11 hours and 40 min) for the Bloomz-3b on the wiki-dataset. We can also see that the RoBERTa model had a substantially lower runtime of maximum 49 minutes, which is due to it also being a much smaller model. These results will be important when evaluating the usability of the different models.

In figure 7.10 and figure 7.11 the F1-score of the models over the training steps can be seen for the wiki- and the CRA-dataset respectively. For the wiki-dataset, we can see that, when ignoring some instability, almost all models reach their maximum F1-score after only about 100 training steps, showing how fast the models actually learn on a dataset. With a batch-size of 8 this amounts to about 800 samples. We can also see the RoBERTa model having reached its maximum F1-score already at the first evaluation, after only a maximum of 52 training steps. For the CRA-dataset we see that the models are a bit more unstable, and it takes the Bloomz-3b model about 400 training steps to be stable over 90%. This suggests that the texts in the CRA-dataset are more difficult to detect, which could be due to the dataset being produced by a more advanced generator (ChatGPT), which could be considered more ”human-like” than less advanced generators. We also see here that the RoBERTa model achieves a high score relatively early.

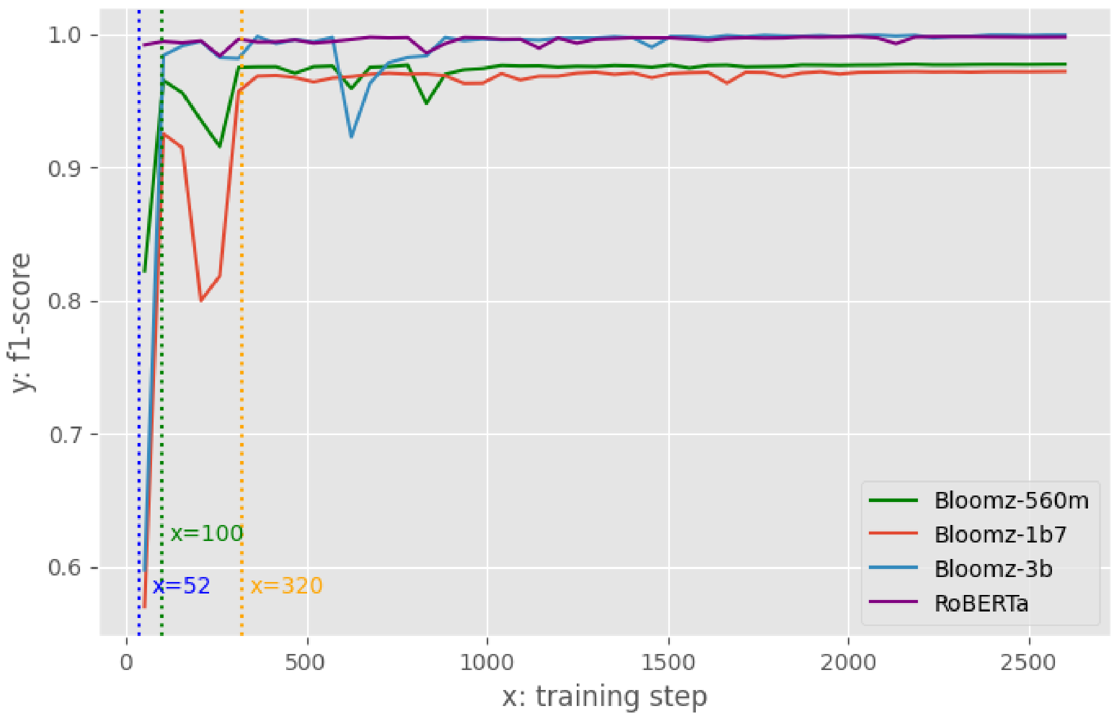


Figure 7.10: F1-scores measured every 0.02 epochs over one epoch of training on the wiki-dataset with batch-size=8 (meaning 8 datapoints every training step).

7.3.2 In-domain performance

Table 7.3: Metric results for the in-domain performance of the wiki-detectors, with the best result from each metric marked in bold.

Base model	Accuracy	Precision	Recall	F1-score
Bloomz-560m	0.973	**1.000**	0.945	0.972
Bloomz-1b7	0.972	**1.000**	0.945	0.972
Bloomz-3b	**1.000**	**1.000**	**1.000**	**1.000**
RoBERTa	0.998	0.999	0.997	0.998

As can be seen in table 7.3, the results from models trained on the wiki-dataset are prominent. All models were able to achieve an accuracy and F1-score of 0.972 or better, and all models had a perfect or near-perfect precision. The Bloomz-1b7 model, despite being three times larger, has scored almost identical to the Bloomz-560m model and even has a 0.1% lower accuracy. The opposite can be seen for the Bloomz-3b model, however, with a perfect score across all metrics, suggesting it has achieved a significantly sound generalization of the data in the dataset. Also notable is the performance of the RoBERTa model which, although being 24 times smaller than the Bloomz-3b model, is able to

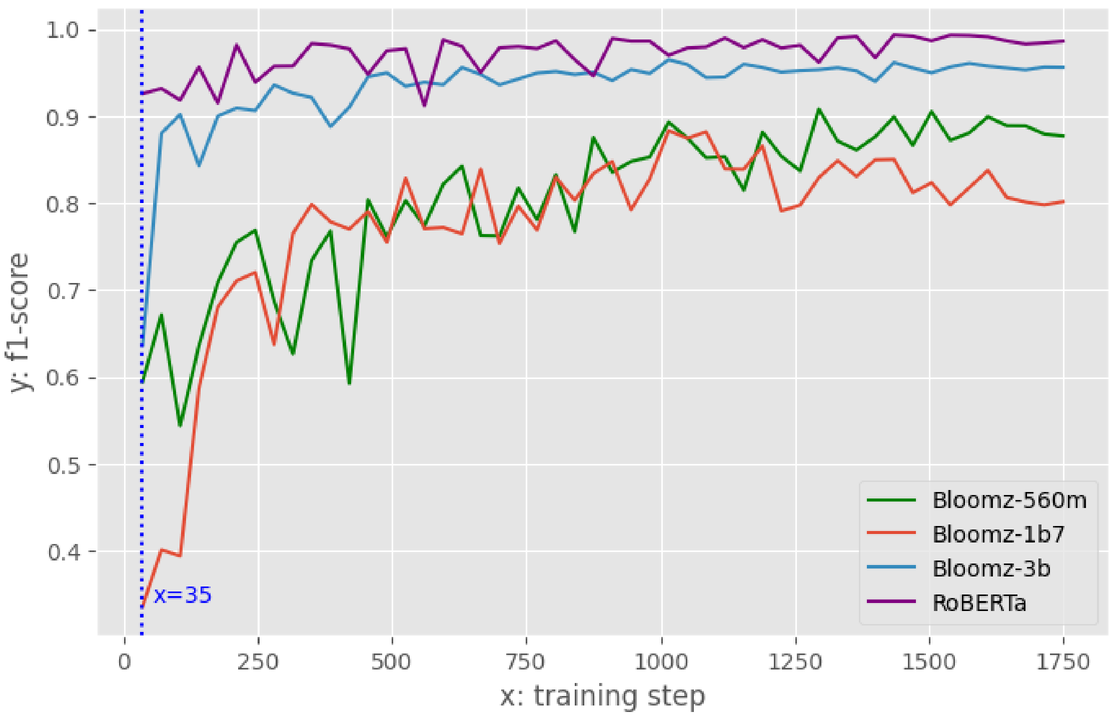
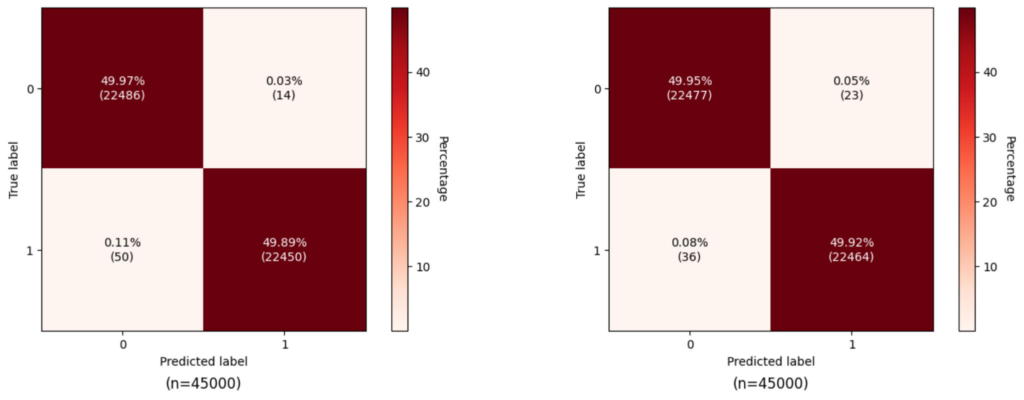


Figure 7.11: F1-scores measured every 0.02 epochs over one epoch of training on the CRA-dataset with batch-size=8 (meaning 8 datapoints every training step).

closely follow the performance of the Bloomz-3b with an almost perfect score, scoring better than both Bloomz-560m and Bloomz-1b7 on most metrics. This is in line with the claim by Solaiman et al. that autoencoding models, like RoBERTa, are more suited for classification tasks than autoregressive models like the Bloomz models.



(a) Bloomz-3b-wiki on wiki **(b)** RoBERTa-wiki on wiki

Figure 7.12: Confusion matrices for Bloomz-3b-wiki and RoBERTa-wiki on wiki dataset across 45000 datapoints.

To take a closer look at Bloomz-3b's and RoBERTa's performance, we can see how well they perform when tested on a larger dataset. The confusion matrices in figure 7.12 shows the distribution of true and false negatives and positives from the Bloomz-3b-wiki-detector and RoBERTa-wiki-detector respectively when tested on the full test-split of the wiki-dataset of 45 000 datapoints. Here we can see that only 14 of the 22 500 real texts (0.06%) were falsely labeled as being fake (a precision of ca. 99.94%)

by the Bloomz-3b model, while RoBERTa falsely labeled 23 fake texts (0.1%) as real (precision of ca. 99.90%). However, RoBERTa had a lower count of false negatives of only 36 (0.08%) while Bloomz-3b had 50 false negatives (0.11%), meaning RoBERTa has a higher recall-score (99.84% against Bloomz-3b's 99.78%). RoBERTa also had a lower amount of total false predictions, meaning a higher accuracy than Bloomz-3b. These differences in performance are miniscule however, and could be a result of random variations. Nonetheless, we can see that when trained on only 21 000 data for one epoch, both are able to generalize remarkably well for even 45 000 unseen datapoints from the same dataset. Confusion matrices for Bloomz-560m and Bloomz-1b7 can be found in appendix C.

Table 7.4: Metric results for the in-domain performance of the academic-detectors, with the best result from each metric marked in bold.

Base model	Accuracy	Precision	Recall	F1-score
Bloomz-560m	0.964	0.963	0.965	0.964
Bloomz-1b7	0.946	0.941	0.951	0.946
Bloomz-3b	**0.984**	**0.983**	0.985	**0.984**
RoBERTa	0.982	0.968	**0.997**	0.982

The results for the CRA-dataset can be seen in table D.3. Although not as good as the results from the wiki-dataset, these results are also notable. We can see that all the models performed worse across all metrics, which could be due to the wiki dataset being bigger and having more data to learn on, but it could also be because the dataset contains more complex data, as also mentioned above. Also interesting is that Bloomz-1b7 performed worse than Bloomz-560m across all metrics, with an even larger difference than for the wiki-dataset. This could be due to Bloomz-1b7 having more parameters to train and thus requiring more training to achieve the same performance, but this would not be true for Bloomz-3b however, which overall performed best.

7.3.3 Cross-testing performance

Table 7.5: Metric results from cross-testing wiki-detectors on the CRA-dataset, with the best result from each metric marked in bold.

Base model	Accuracy	Precision	Recall	F1-score
Bloomz-560m	**0.559**	**0.553**	0.615	0.582
Bloomz-1b7	0.539	0.537	0.555	0.546
Bloomz-3b	0.539	0.529	**0.696**	0.601
RoBERTa	0.559	0.547	0.693	**0.611**

The results from cross-testing the two datasets are less promising, however. As we can see in table 7.5, the wiki-detectors did not perform well on the CRA-dataset, with the highest F1-score of 61.1% and accuracies relatively close to 50%, which is equivalent to random guessing. The same can be seen for the Bloomz-academic-detectors in table 7.6, with accuracies close to 50%, and with an even lower F1-score of highest 27%. This low F1-score seems to be a result of the academic-detectors tendency to over-classify texts as being real, leading to a high precision and a low recall. A high precision is favorable, but

Table 7.6: Metric results from cross-testing academic-detectors on the wiki-dataset, with the best result from each metric marked in bold.

Base model	Accuracy	Precision	Recall	F1-score
Bloomz-560m	0.565	0.839	0.161	0.270
Bloomz-1b7	0.501	0.526	0.030	0.056
Bloomz-3b	0.515	**0.955**	0.032	0.062
RoBERTa	**0.745**	0.926	**0.532**	**0.676**

the low recall suggests that the models are not very useful in this context. These results seem to suggest that training on data that is very specialized does not generalize across datasets very well.

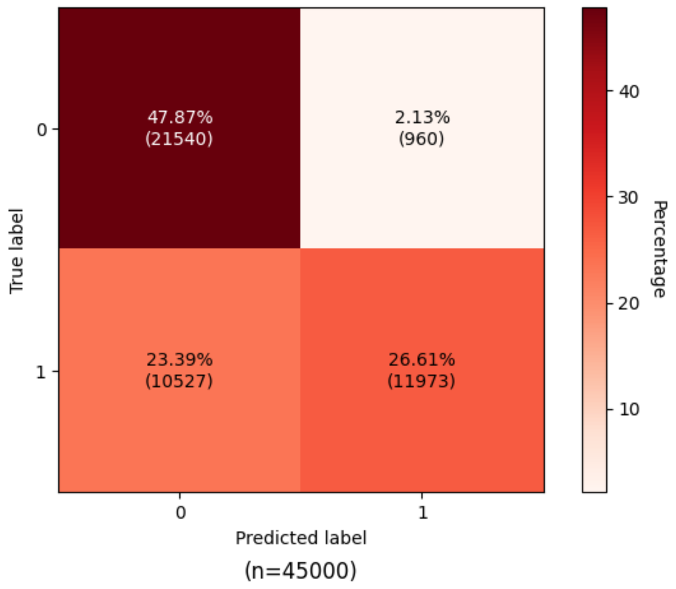


Figure 7.13: Confusion matrix for the RoBERTa-academic-detector on the wiki dataset across 45000 datapoints.

The RoBERTa-academic-detector has better scores however, with an F1-score of 67.6% and an accuracy of 74.5%, which is also better than the RoBERTa-wiki-detector. When looking at the confusion matrix for this model in figure 7.13, we can see that the model correctly identifies 11 973 of the 22 500 fake texts as fake (about 53.2%), while still having only 2.13% false positives. Effectively, this means that the RoBERTa-academic-detector is able to detect about half of the fake texts while still having few mislabeled real texts.

7.3.4 Results from mixed-detectors

To further assess the ability of the detectors to generalize, we can look at the results from the mixed-detectors. In table 7.7 the F1-scores for the mixed-detectors on all three datasets can be seen. The other metric results for this test can be found in appendix D.

As we can see, the models did not suffer much in performance on the wiki-dataset when using a mixed dataset, with an F1-score of minimum 96.4% compared to minimum 97.2% for the wiki-detectors. The same can not be said for the CRA-dataset however, where the

Table 7.7: F1-scores for mixed-detectors on all datasets, with the best result from each dataset marked in bold.

Base model	Mixed	Wiki	CRA
Bloomz-560m	0.948	0.972	**0.848**
Bloomz-1b7	0.929	0.964	0.816
Bloomz-3b	0.988	0.996	0.772
RoBERTa	**0.993**	**0.997**	0.829

F1-score dropped by about 10 to 20 percent points across all detectors. It is important to note however, that the mixed dataset contains only half of the datapoints of the datasets it is composed of, meaning the mixed-detectors only had half of the data from each of the two data domains to train on.

Similar to the other results, we can see that despite being much smaller, RoBERTa is able to achieve among the best performance, and we can also see the performance decreasing for the larger models, suggesting they could need more training.

7.3.5 Bloomz-560m with 1024 max-tokens

Table 7.8: Metric results for both Bloomz-560m detectors on in-domain data with 1024 max tokens.

Detector	Accuracy	Precision	Recall	F1
560m-wiki	0.998	1.000	0.996	0.998
560m-academic	0.984	0.980	0.989	0.984

To determine if the maximum tokens used has an impact on the performance of the model, one can look at the performance of the Bloomz-560m model when trained with 1024 max-tokens instead of 512 max-tokens, which can be seen in table 7.8. For the wiki-detector, we can see that the F1-score increased from 97.2% to 99.8% when using more tokens. Similarly, for the CRA-dataset, the F1-score increased from 96.4% to 98.4%. This is interesting when considering that only a very small set of texts, for both datasets, has a word count that would amount to much more than 512 tokens, as can be seen in figure 7.1 from section 7.1.1. Although using more tokens gives an exponential increase in training times, there seems to be much to gain from using more tokens.

8 Discussion

In the following, we first discuss the results from the dataset production. Then the results from in-context learning are discussed, followed by a discussion on the usage of fine-tuned detectors. Finally, this section will discuss ethical considerations of MGT-detection tools.

8.1 Dataset production

In this section, we shortly discuss some of the limitations of the generator model displayed in section 7.1.2

In light of the findings presented in section 7.1.2, it is evident that the generator model's word count accuracy in generating research abstracts of a target word count diminishes when the target word count exceeds 325. A key observation made in figure 7.2 and figure 7.3 is the considerable increase in mean average deviation and decline in correlation as the target word count surpasses the 325 threshold. This decline in word count accuracy might be attributed to the data the model has been pre-trained on. As we saw in figure 7.1, the distribution of the source dataset and the generated abstracts are similar even with assigned target word counts. This may suggest that the bias displayed in the generator model is due to the objective average length of research abstracts, which is represented for comparison using a random subset. Furthermore, it may be biased as writing abstracts above 325 words is rare - out of the approximately 2 million research abstracts in the AA21 dataset, only 6368 were above this, further supporting this hypothesis. Consequently, the generator model may struggle to adapt and produce abstracts with a word count closer to the desired target for TWC > 325.

Another reason for this bias may be the generator's model architecture. The generator model, *gpt-3.5-turbo 0301* is an *auto-regressive* transformer model. This means it is *uni-directional* and that it generates each individual token sequentially based on model logits calculated from the previous sequence of tokens. This restricts its ability in predicting how long its texts will be. An alternative model which may yield better results with regard to word count accuracy is Google's language model BERT. This model has a bi-directional transformer architecture, meaning it processes entire segments of texts simultaneously. This makes it exceed in a set of applications where the auto-regressive transformer architecture does not. Examples of this are summarization or text classification, which are further supported by the comparable size-to-accuracy ratio differences displayed in 7.3.

One might argue that writing length-specific texts independent of context is a simple task which is an apparent weakness of the GPT-3.5 model used for dataset production in this thesis. Comparing the generator model with human ability sheds light on its weaknesses; producing texts of specific word counts is arguably an easy task for an average human being given enough time. Finding approaches for generating sequences of text that adhere to specific word counts would be desirable in the scope of data generation for training detection models. Given the difficult task of separating between human, machine-generated, and hybrid texts, texts of a large range of lengths would be beneficial for enhancing versatility in the implementations of AI-plagiarism tools.

8.2 Impact evaluation of performance-evaluation metrics

The importance of each of the classification evaluation metrics, *Accuracy*, *Precision*, *Recall* and *F1 Score* should be valued with respect to the priorities and implications regarding the task-specific application of a classification tool. Within the scope of the thesis problem domain, emphasizing each of the various metrics would have distinct impacts in the application of MGT-detection tools in academia. Following is a proposed prioritization of the evaluation metrics for performance indication in the scope of the thesis problem domain from least to most prioritized:

4. **Accuracy**
   While accuracy might be the easiest metric to understand as it provides the general performance of a detection tool, its application as an indicator for performance in academic applications may be misleading, especially in cases where the classes are imbalanced. For instance, in cases where there are comparably much fewer machine-generated texts than human-written texts, a model could achieve a high accuracy simply by predicting the majority class, while failing to effectively identify instances of the minority class of machine-generated texts.

3. **Recall**
   Recall would be a preferred metric of choice in cases where the priority is to identify as many true positives as possible. In the scope of the problem domain, the result of using this as a performance indicator would be an optimization of detecting the largest amount of actual misuses. However, this metric does not directly account for minimizing false positives, which could result in an increased amount of human-written texts being falsely classified as misuses.

2. **Precision**
   In the context of the problem domain, precision might be considered the single most critical metric. A lower precision score implies a high percentage of false positives, and ultimately, a higher chance of human-written texts being classified as machine-generated. The consequences of false positives could potentially damage a student's or researcher's academic record due to incorrect accusations of using language models to generate their work. Maximizing precision score should therefore be a prioritized metric when evaluating various detection tools.

1. **F1 Score**
   Although we consider precision to be the most critical metric, for the overall performance evaluation of a detection tool, we consider the *F1 score* to be the most accurate indicator. Prioritizing the F1 score puts an equal emphasis on both precision and recall, which may be important in contexts where both false positives (incorrectly identifying human-written texts as machine-generated) and false negatives (failing to identify machine-generated text) are considered equally undesirable. It provides a balanced score between detecting as many misuses as possible, while also minimizing occurrences of false accusations.

8.3 Performance of in-context learning

In section 7.2, we documented that the proposed approaches for implementing a detection tool for misuse of large language models in academia do not currently provide adequate performance with respect to the problem domain applications. In this section, we will discuss potential factors which may have impacted performance, while also considering the future of in-context learning with respect to the thesis problem domain.

All the proposed in-context learning approach variations[30] is applied with a *direct* scoring function. This puts constraints on the *demonstration template design*[31], ultimately limiting the model to only respond with the predicted class label. By adopting a *perplexity* scoring function, as additionally suggested by Dong et al. (2023)[32], relieving the output restrictions constrained by the direct scoring function, improved performance may be discovered as large language models in general displays better performances when allowed to reflect, elaborate or reason before providing an answer (Li et al., 2022).

It is important to acknowledge that in-context learning is currently constrained by the relatively small context window sizes of the current open-access large language models at typically 512-4096 tokens. As the context window sizes are relatively small, this puts a constraint on the number of labeled examples provided for context to less than 10 examples in our proposed in-context learning approaches. For the future, however, window sizes are also likely to increase along with model sizes and general task performances. Although an increase in model size does not directly correlate with an increase in context window size as it is restricted by computational resources which we presented in section 2.8, as models continue to grow in size, capacity, and computational resource requirements, context window sizes are also likely to increase as they will be less of a relative constraint. This is further supported by the release of the currently limited-access GPT-4 model, which already operates with 8k and 32k context window sizes (OpenAI, 2023a). As context windows expand in the future, allowing for the inclusion of entire articles or submissions as labeled examples, in-context learning for classifying academic texts may show better potential both in terms of practicality and accuracy, increasing the potential for in-context learning as a low-configuration approach requiring minimal in-domain data for distinguishing between human-written and machine-generated academic texts.

The reduced performance displayed in the in-context learning approaches in zero-shot settings[33] in comparison with the same approaches in few-shot setting[34] could possibly be explained by a lack of internal representation in the GPT-3 model[35] for *true* machine-generated texts. For the model to establish a clear internal representation, characteristics of machine-generated text would need to be present in the data exposed to it during pre-training, along with context directly or indirectly associating these texts to the machine-generated domain. Based on our experience that there currently is a limited amount of

[30]*human-zero-shot*, *inductive-zero-shot*, *human-few-shot* and *inductive-few-shot*

[31]Demonstration template design is referred to by Dong et al. (2023) as the combination of input text formatting and format of model output predictions

[32]Dong et al. (2023) suggested both the *direct* and *perplexity* scoring functions in his survey on in-context learning

[33]respectively referred to as *human-zero-shot* and *inductive-zero-shot*

[34]respectively referred to as *human-few-shot* and *inductive-few-shot*

[35]GPT-3 was used for the all in-context learning approach variations, and the only model used to conduct in-context learning performance-evaluations

publicly available data which satisfies these criteria, an assumption of this being the case may align with the objective truth. However, there is a possibility of this having been an area of focus during the development of the GPT-3 model (175B), subsequently rendering this assumption false, but as the exact data used for training of the GPT-series is not disclosed, no definite conclusion can be drawn. Additionally, further research on the model's ability to extract such internal representation from its general pre-training data would need to be conducted. For the future, however, as more machine-generated texts are labeled and exposed to language models during pre-training, they could prove better in-context learning MGT-detectors without an architectural upgrade.

Despite GPT-3 displaying impressive results for in-context learning in binary classification tasks among others (Brown et al., 2020), the provided classification task of distinguishing human-written and machine-generated academic abstracts in this thesis seems to be above its current capabilities. In-context learning is still a relevant approach for the future, however, as the next generation of language models will continue to improve with regard to their general capabilities. The next generation language models such as the GPT-4, which is rumored to be 1000B by 'people familiar with the inside story' according to Albergotti (2023), displays performances in complex tasks which outperforms the GPT-3 on all metrics, documented in *Sparks of Artificial General Intelligence*.

For the future, In-context learning is particularly relevant for academic text classification in comparison with training-based approaches for several reasons. In-context learning is entirely prompt-based, leveraging the pre-existing knowledge of large language models without the need for additional fine-tuning. This is advantageous for efficient text classification in academia, where sub-domains vary greatly in character and application. Approaches that involve task-specific training are demanding in terms of gathering in-domain training data, but also in terms of continuously updating the in-domain training data for a new generation of language model outputs.

To summarize; although the proposed approaches for using in-context learning for MGT-detection in academia do not currently display sufficient performance for practical application in academia, in-context learning still offers potential for the future. As large language models continue to increase both in size and in capabilities, in addition to context windows sizes becoming less of a constraint, in-context learning may also yield better results despite poor performances in the problem domain applications[36]. GPT-3 (175B), used for the in-context learning approaches in this thesis, does display performance just slightly below fine-tuned models for the binary classification tasks provided in 'Language Models are Few-Shot Learners' by Brown et al. For the detection tasks we provided in this thesis, the implemented in-context learning approaches perform far below our proposed fine-tuning approaches, subsequently displaying strong contrast between the performances of the two detection approaches. In the future, however, in-context learning may yield potential in providing a low-resource, efficient, and effective solution for MGT-detection in academia given that the general analytical abilities of language models improve and new and improved in-context learning approaches for the specific task are discovered. For instance, next-generation models such as GPT-4 may yield better results for the problem domain application and should be explored in further works.

[36]Detecting whether a text is human-written or machine-generated

8.4 Performance of fine-tuned detection

As we can see from the results from the in-domain tests in section 7.3.2, the fine-tuned detectors do well in generalizing data from the same domain or corpus of which it was trained. These results are promising, and even more interesting is the striking efficacy of the RoBERTa model, in comparison to the other models. We can see that RoBERTa's performance is on par with, and sometimes even better than the other models, despite being a much smaller model, which highlights the superiority of bidirectional in comparison to unidirectional autoregressive language models for MGT-detection.

From the cross-testing results, on the other hand, as shown in section 7.3.3, the models seem to struggle when cross-testing between the two datasets, meaning that the academic-detectors, which are good at detecting if research abstracts are generated or not (in-domain data), are not necessarily be good at detecting generated Wikipedia-articles (out-domain data), which is in line with the results of Bakhtin et al. (2019), as discussed in the literature review in section 4.3.1. This suggests that to train a generalized model that can detect if any text is generated or not, regardless of the domain of the data, a larger amount of data across multiple corpora is needed.

This is to some extent what was tested for the mixed-detectors, with results shown in section 7.3.4, where the datasets were combined and cross-tested, which corresponds to a cross-corpus dataset. As we saw from these results, although still relatively high, the performance decreased in comparison to the in-domain performance from training on only one corpus, e.g. wiki or CRA. As already mentioned, this could be explained by the fact that the mixed-dataset only contains half of the data points from each of the other datasets. Still, it is interesting that the Bloomz-3b, while being the best model on both of the other datasets in regards to in-domain performance, had the worst performance on the CRA-dataset when trained on the mixed-dataset. Also, if we look back at figure 7.11, we can see that after Bloomz-3b trained on half the dataset (0.5 epochs), it has an F1-score of roughly 95%, while the Bloomz-3b-mixed-detector only has an F1-score of 77.2% after training on the whole of the mixed-dataset, equivalent to half of the CRA dataset. These results suggest that training on larger amounts of data across different corpora could negatively impact the performance when compared to training on in-domain data, which again would support the findings of Bakhtin et al. (2019).

However, note that in contrast to Bakhtin et al., as the two datasets were generated by two different text generators (GPT-3 and ChatGPT), the mixed-dataset is also cross-architecture, in addition to being cross-corpus. Therefore conclusions as to whether cross-corpus or cross-architecture is harder to detect can not be drawn from our results, as the difficulty of detection could lie in either, or both, of these domains. It could be the case that detecting text from multiple generators with the same fine-tuned detector is easy, but detecting across data-domain like Wikipedia articles and research abstracts is hard, or the opposite could be the case. Additionally, the drop in performance could be a result of the models being too small, or that they are not trained long enough, or that they were trained on too little data, although this is not known to be the case. This necessitates more research, building on the limited scope of this thesis, to further determine the performance of fine-tuned detectors on cross-corpus and cross-architecture data.

Another limitation of fine-tuned detection that is worth discussing, is the cost of fine-tuning new detectors. As we saw in section 7.3.1, where the runtime from the training was presented, large language models take a significant time and resources to train, whereas the Bloomz-3b-wiki-detector took a maximum of 11 hours and 40 minutes to train on an A100 80GB GPU. This was also when using only 10% of the wiki-dataset, and if using the whole dataset, the training time would likely be over 100 hours. As generative language models become larger, and their texts more human-like, likely larger detectors are needed, and as Bloom-3b is far from the largest model that exists today, it is reasonable to assume that high-performing detectors would require far more resources to train in the future. Also, when considering the quadratic increase in runtime with increasing input tokens, the resource requirements could become immense for detecting larger texts. Implications of this are that training and researching larger detectors could get difficult and even insurmountable for most researchers, in addition to ramifications related to the environmental stress of high resource usage.

On the other hand, it is not necessarily a certainty that larger models for detection are needed, however. When we compare the sizes of the detectors with the sizes of the generators that were used to generate the datasets for training the detectors, we can see that the detectors perform quite well despite being smaller than the generators, suggesting smaller detectors to be sufficient for detecting output from larger generators. For example, when considering ChatGPT, which generated the CRA-dataset, has a size of 175B, it is quite impressive that RoBERTa-base, with the size 125M (about 1400 times smaller), was able to achieve an F1-score and accuracy of 98.2% on this dataset, again as shown in table D.3.

Additionally, when considering the detection of larger texts, other architectures than the transformer can be considered. As already mentioned in section 2.8, the runtime of the longformer architecture increases only linearly with the amount of input tokens and, although not researched in this thesis, the detection performance of such architectures could be tested in the future.

8.5 Social and ethical considerations

In section 4.1, we presented literature documenting the potential social impacts of language models and their use in academic work. In this section, we will build on this and discuss both social and ethical considerations within the thesis problem domain.

8.5.1 Ethical considerations in the development and application of MGT-detection-detection

As the topic of this thesis suggests, the nature of MGT-detection has much potential in regard to increasing academic integrity and fairness. Thus, for the development of MGT-detectors, ethical considerations are essential to prevent potential negative social impacts related to the educational sector. One critical consideration is where to set the thresholds with regards to the respective performance metrics *accuracy*, *recall*, *precision*, and *F1-score* for what we define as acceptable loss in the detectors prior to deployment in the real world. As previously mentioned, the consequences of mislabeling a true human-written text as machine-generated could have consequences for the author. The development of detectors should be focused on minimizing occurrences of false accusations (false positives), over minimizing true machine-generated text as human-written (false negatives). It is also important to note that, while a detector can have a precision of 100% on the dataset it is tested on, meaning zero false positives, this is not a guarantee of similar performance in real-world applications, meaning the possibility of false positives occurring can never be completely ruled out, regardless of the measured performance of the detector.

As any approach to classification including MGT-detection are entirely probabilistic, a truly objective detection tool may never be developed with the current consensual understanding within the field of classification. Any human writing a text may coincidentally produce a text which aligns well with the characteristics of machine-generated text. This is further supported by the fact that a core goal of current large language models is to reproduce human capabilities in writing, which further currently is a result of iterative training on large collections of human-written text.

Contrary to this, a similar argument can be phrased for traditional plagiarism tools, which in general have been successful in their applications in academia (Foltýnek et al., 2020). Traditional plagiarism tools are in the same manner as expressed for MGT-detection, completely probabilistic approaches for detecting plagiarism. It should be acknowledged that the application of such plagiarism tools is likely to be less relevant in the future as autoregressive language models in general excel in the task of paraphrasing text as documented by (Wahle et al., 2021), providing an simple evasion-technique for traditional plagiarism tools. MGT-detectors may prove to be a decent substitute if acceptable implementations of authorship attribution models as proposed by Uchendu et al. (2020) and discussed in section 4.1, are developed. Additionally, if decent MGT-detectors are implemented paraphrased texts by language models may be identified as machine-generated text. Retracing back to the applications of plagiarism tools; if we follow the same principles for the applications of plagiarism tools in academia, similar acceptable loss thresholds for MGT-detection should be acquirable.

Furthermore, the biases of the MGT-detectors should be studied extensively to prevent any unfairness with respect to MGT-detection. As also discussed in section 4.1, the detectors could potentially inherit biases that would disproportionately target specific social groups, and studies on how false positives and false negatives from the detector distributes over such social groups should be considered.

8.5.2 Social considerations for appropriation

Prior to the application of MGT-detection tools, thorough research focusing on the social consideration with respect to applying machine-generated detection tools in academia must be conducted. Building on the social impacts discussed in section 4.1, we have listed 3 main aspects of consideration we consider central in the problem domain:

What we define as inappropriate and appropriate use of language models in academic work:
Language models have proved valuable tools for a range of applications both in general and in academia. Limiting access for students, researchers, and other academic indi-

viduals may also constrain the quality- and efficiency of their productions in comparison to allowing a defined set of usages. As an example, language models can be used for effectively summarizing papers and determining whether the paper is relevant for their focused problem domain. Another application is formulation assistance; there are various tools such as Grammarly which uses artificial intelligence for aiding in proper, concise, and clear formulations of text without applying a bias to the content (Grammarly, 2023). These are a few examples of AI-assisted writing which may be considered appropriate applications of language models in academia. On the other hand, there are applications that should be considered inappropriate usage of language models. Using a language model to perform academic work for the attributed authors which we refer to as *ghostwriting*, such as generating entire- or sections of submissions and articles as a technique of quickly completing assignments or producing papers for the purpose of accomplishing requirements for doctoral degrees or achieving high-regarded, reserved academic titles. Allowing the usage of language models to this extent is likely to promote academic laziness which subsequently will reduce the value of completing the task at hand, whether that is educational or scientific; providing novel knowledge to the scientific domain.

As presented in section 4.1, the use of language models in academic work is problematic when considering the model's potential for affecting or manipulating the initial opinions of the author. Although this is a serious concern regarding the introduction of language models in academia, these models can provide useful insights with regard to the text's general factuality and further improve the overall quality of its content. Applying language models for tasks like identifying correlations within textual data across various segments of literature might provide insights that otherwise may have been overlooked by a human. Another potential use case is applying language models as an automated layer for alerting potentially inaccurate statements in the text, which may prove to reduce misstatements in academic literature. On the other hand, applying language models for such a task may also result in the opposite; the current language models are documented to be highly confident in their generations, sometimes displaying high confidence in completely false statements. As autoregressive language models on a foundational level simply perform next-token prediction through estimation of probability distributions, they do not inherently understand their generations, sometimes resulting in formulations of severely wrong statements. Various methods for increasing the factual reliability of language models are already in place and are continuously being further developed. OpenAI notes one of the distinct improvements in their GPT-4 model from GPT-3.5, is its precision in factual reliability (OpenAI, 2023c). Despite this, critically evaluating the outputs of language models are central to the appropriate and responsible use of language models in academia, which also is general practice for human literature as well.

There is a vast range of applications for language models in academic work, but deciding what we define as inappropriate usages must be established prior to applying detection tools such that the potential negative impacts of immature applications are reduced.

What we define as inappropriate and appropriate use of MGTdetectors in the evaluation of academic work:
For the evaluation of academic work, there are also considerations related to the application of MGT-detection tools in academia. While these are designed with the intent of promoting academic integrity, they should not entirely restrict the author's process of writing by discouraging creativity in academic writing as an indirect result of the unres-

tricted application of mMGT-detection tools. Although the primary role of such tools is to detect academic dishonesty, we must also consider whether their use might encourage cautious, and rigid writing styles, subsequently constraining creativity and innovation which academia seeks to promote.

In addition to this, transparency with regard to content provenance is a central consideration for the application of MGT-detection tools. Academic individuals should be well informed about how MGT-detection tools are applied in the evaluation of their work, including a clear understanding of how they function. This transparency should allow for a fair evaluation process and help in building trust and support on both perspectives of the problem domain.

Establishing consensus on the acceptable thresholds for the use of MGT-detection tools in academic evaluation becomes increasingly important as advances are made within the domain of language models and the norms in academia adjust for the disruption caused by these models. Academia must adapt to the constant development in society, including our assumption that the application of large language models in academic work is inevitable. In cases where there are disagreements between the respective authors and evaluators, a system of accountability may prove valuable. Establishing appeal processes that allow for human review and the provision of critical evaluation of the specific case may provide a second layer of inspections working as a measure for safeguarding irresponsible application of MGT-detection tools, and consequently reduce the unintended harm as a bi-product of introducing MGT-detection tools in academia. It is through a careful balance of considerations that we can define the appropriate use of MGT-detection in academic evaluation.

The impacts of the derived acceptable application thresholds for the introduction of MGT-detectors in academia: Both the impacts of establishing too loose or too strict policies for the application of MGT-detection tools may prove devastating. False accusations may lead to unjust damages to an individual's academic record, which subsequently may lead to expulsion or even degree revocations depending on the severity of the case and the academic policies in place. Too loose policies may results in damages to academic integrity by promoting academic laziness and dishonesty, which could have serious implications for the future of academia and its role in promoting discoveries of- and research on new innovations, halting the development of society and humanity as a whole. As the current standards of well-being across the globe, in general, are built upon continuous and incremental scientific achievements, we must ensure that the introduction of language models in academia does not contaminate the strict standards of academic integrity.

Artificial intelligence is increasingly being employed in all segments of the technology industry. Ensuring that we are prepared and fully understand the impact this technology will have on society is central prior to its application. Given the transformative potential of artificial intelligence, both negatively and positively, we should thread carefully to make sure we do not fall into one of the pitfalls of immature application. There are few fields where recklessness or lack of forethought may prove more devastating than with artificial intelligence.

8.6 The future of MGT-detection

As language models continue to advance in replicating human capabilities, the task of distinguishing human-written and machine-generated texts will become increasingly difficult. The development of robust MGT-detectors may be used against its intention; for training language models to evade its detection. Interpretations of this fact can vary greatly depending on perspective. Using MGT-detection tools in the training of language models may improve their ability in replicating human natural language, which fundamentally is a core goal of language models. On the other hand, increasing its ability to evade detection will subsequently result in an increase in misuse as the likelihood of being caught is low. Given the current pace of advancements in artificial intelligence, the current approaches of identifying native characteristics in machine-generated texts or using the models themselves to calculate perplexity scores may eventually be rendered inadequate.

There are various approaches that intend to provide solutions for MGT-detection which does not rely on the native textual signature of language models being distinguishable from human-written text. Among these, we highlight the concept of *watermarking*. This is an approach that involves inserting hidden biases or rules in the lexical selection process of generative language models. These watermarks are intended to be transparent in human observation and evident for detection tools tuned for the set watermarks. As with all proposed approaches in this thesis, there are evasion techniques for watermarking as well. such as hosting language models which do not adhere to the watermarking standards which may be set. Implementing watermarking and setting up tools for detecting them requires a unison development effort between the developers of generative models and developers of detector models.

While this thesis has focused on the discrimination of homogeneous text domains. A key aspect to additionally consider is the occurrence of *hybrid texts*. When using language models for text production both in academia and in other sectors, it is likely that they often will be used under the supervision of the authors. This opens up for texts being partially produced by humans and partially generated by language models. We have left this out of the scope of this thesis as it requires extensive efforts with respect to correctly collecting and generating hybrid data with proper labeling. As the problem domain of this thesis matures, and more data on this becomes available, comprehensive research on the domain of detecting hybrid texts should be conducted. Approaches such as the GLTR approach presented in section 4.3.4, highlighting tokens based on their classification score may show promise in providing insight into what tokens are generated, and which are human-written.

9 Conclusion and further work

In this thesis various aspects regarding *how large language models can be leveraged to detect machine-generated text be detected in academia* has been presented, considered, tested, and evaluated. The problem is a complex one, built upon various sub-problems such as the process of producing relevant and sufficient data for the training and evaluation of models, the development and testing of detection approaches that show potential with respect to both practicality and end-result performance in existing literature, and considerations of a selection of social and ethical aspects related to the applications of such detection tools in academia.

9.1 Necessary data foundation and the production of in-domain data

To enable the training and deployment of machine learning models in the problem domain, a considerable foundation with regard to data is required, most importantly with respect to data quality, but also in quantity. Sufficient datasets may be produced using features of human-written texts as templates for a language model to produce similar texts. Despite this, data-cleaning techniques and following statistical analysis of the data should be conducted to reveal and mitigate any superficial characteristics which may impact accurate performance evaluation.

Due to the infancy of the problem domain, dataset(s) containing in-domain data had to be produced to allow comprehensive in-domain research on various approaches. Results displayed both in our binary sequence classification and in the existing literature show that using in-domain data is crucial for optimal performance in detection tools. Such in-domain data was produced in this thesis using scientific research abstracts sourced from arXiv, and corresponding machine-generated abstracts were generated using the large language model *gpt-3.5-turbo-0301*. To reduce superficial discrepancies with regard to the general characteristics in each of the domains, the model was instructed to match a target word count equal to that of a corresponding human-written abstract and also prompted to match the content using its title. Using a set of data-cleaning techniques, superficial discrepancies between the two domains were reduced. Through further statistical analysis, discrepancies were still found with regard to word count and page break usage frequency (PB-frequency). The model used for generating the machine-generated abstract displayed imprecision with regard to the target word count provided, typically generating abstracts below the target word count. For word counts (WC) in WC $\in [50, 325]$ the generator model displays a correlation with its target word count of $r = 0.97$, while for the interval WC $\in [325, 420]$ the word count correlation drops to $r = 0.2$. This imprecision displayed at higher target word counts indicates that the model displays a bias towards the natural average word count distribution of its training data. It may also be attributed to the autoregressive architecture of the generator model which makes it inferior in tasks such as predicting the length of its output in comparison to bi-directional transformer-architectures. With regard to the discrepancies displayed in the PB-frequency, the model displayed distinct patterns in its deployment of paragraphs with respect to the word count of its abstracts. For word counts lower than 185 an MGT-detection accuracy of 78% may be acquired by simply guessing all abstracts as machine-generated if they do not contain any paragraph breaks. For abstracts greater than 185 in word count, the generator model displayed a consistent average of 87 words per paragraph. In contrast to these characteristics, the human-written abstracts displayed a steady linear increase in words per paragraph as their respective word count grow. Due to this finding, all performance evaluations were completed with all abstracts stripped of any sequences of white space, including paragraph breaks and excluding single spaces.

9.2 Current and future approaches for MGT-detection

The current and potential future approaches found to show promise in existing literature are *perpelxity*-, *in-context learning*-, and *fine-tuned binary sequence classification*-based MGT-detection approaches. While perplexity approaches have shown potential as a zero-shot approach for MGT-detection, it was down-prioritized in this thesis as literature documenting its performance in the problem domain already exists. The focus was instead set on providing valuable insights into the performance of in-context learning and fine-tuned binary classification which has not been documented in the literature with respect to the problem domain.

For our in-context learning approach, 4 distinct variations were tested: in-context learning with human-written input texts, in-context learning with inductive input texts, each in both zero- and few-shot settings. The results show that all variations perform with an accuracy close to 0.5, indicating that for the large majority of the provided task iterations, the models are guessing the label to which the provided classification text belongs. Following applying optimal thresholds when the model predicts a text as machine-generated, slight performance gains were made in terms of accuracy, but considerable improvements were made in terms of *precision*, *recall*, and *F1-score*. Despite this, the performance should be evaluated across all metrics, and low performance in either of them should disqualify any approach with regard to application in the problem domain. Prior to threshold optimization, the inductive few-shot approach displayed better performance on all metrics, in addition to being less biased in its confidence scores towards all classification texts being written by a human. In total, the in-context learning performances documented in this thesis do not suffice for practical application, indicating that the task currently is too complex for current open-access language models for in-context learning. Despite this, the results provide valuable insights into how various approaches and performance optimization techniques can be applied in further in-context learning research projects. Better performances may be discovered for in-context learning in next-generation language models exemplified by GPT-4 which possesses both analytic capabilities and larger context windows greater than its predecessor used in the proposed approaches.

In contrast to in-context learning, the fine-tuning approach has demonstrated significantly more promising results. When tested on in-domain data, the best models were able to achieve an F1-score of up to 100% on the text generated by GPT-3, and 98.4% on the text generated by ChatGPT, indicating proficient performance in detecting machine-generated text. When testing the performances on out-domain data, however, the models seem to struggle greatly, with accuracies on par with random guessing, suggesting this to be a limitation of such models. Our tests on using mixed-corpus datasets have shown that generalization across multiple corpora is possible using fine-tuned detectors, although with lower performance than that of single-corpus detection. Furthermore, it was found that doubling the size of the input to the model resulted in a performance increase of about 2%-3%, although also quadratically increasing the runtime of the training. More research on how training runtime can be decreased, as to allow for more efficient research, is therefore advised. Lastly, through our research on fine-tuned detectors, we have also further supported the hypothesis that autoencoding language models possess an inherent advantage over autoregressive language models when it comes to text classification, and from this, it is reasonable to suggest further research on fine-tuned detection to be focused on testing and developing larger autoencoding models in the context of MGT-detection.

9.3 Social and ethical considerations in the application of MGT-detection in academia

Prior to the introduction of advanced technology in society, careful considerations and measures regarding both social and ethical aspects of its application must be established. This caution has to a little degree been practiced with respect to the distribution of state-of-the-art generative large language models which have been introduced into society without proper preparation and caution with respect to sectors that they disrupt such as academia. Academic institutions have now been faced with great challenges of how to adapt to this disruption. Instead of following the example of the commercial large language model distributors, we have outlined three criteria that we propose fulfilling prior to a potentially immature introduction of MGT-detection tools in academia:

Firstly, we must define what is considered inappropriate and appropriate use in academic work. Secondly, we must define what is considered inappropriate and appropriate use of MGT-detection in academic evaluation. And finally, we must research and understand the exact impacts of the definitions we set. Only by conducting comprehensive research in these social domains, can we best understand and apply the correct measures, consequently mitigating severe unintended consequences.

While there are strong arguments both for and against the application of MGT-detection in academia, it is important that a discussion of how we adjust to the disruptions of generative large language models is based on a scientific foundation led by performance evaluations of proposed MGT-detection tools, and the social and ethical implications of a widespread application. A solution may include a balance in accepting the usage of language models in academic work while also providing policies for how to document these usages and restrictions for what is defined as misuse. It's through transparency from both sides of the problem that we can reap the benefits of large language models while maintaining academic integrity. By conducting both research and education on how large language models can be used responsibly in the production and evaluation of academic work, a consensus for their applications might be reached. If we gradually implement verifiable usage policies, balanced and well-tested MGT-detection measures, and provide frameworks for attributing language models in academic work, we can incrementally improve and find solutions that benefit both sides of the problem domain.

9.4 Closing statement

In this research, we have provided an introduction to how potential MGT-detection tools may be developed and applied by displaying both successful and unsuccessful approaches to how large language models can be leveraged to detect their misuse in academia. We have produced, developed, and discussed foundational sub-problems of a problem domain which is likely to increase in the coming years. Although our contribution to the field is a small piece in a large puzzle, we hope to promote further research on the domain by open-sourcing our produced datasets, our fine-tuned models, and all code used in our research.

Thank you for reading our bachelor thesis.

22nd May 2023, Trondheim
Nicolai Thorer Sivesind & Andreas Bentzen Winje

Broader impact

The research and topic presented in this thesis carries significant implications to society. By addressing the emerging and pressing concern of academic cheating facilitated by modern text generators, this research has the potential to contribute to broader impacts related to the educational sector. These impacts include:

- *Upholding of integrity in academia*:
  By developing models that can detect if a text is generated or not, teachers and academic evaluators can potentially use these models as tools in their work to more accurately evaluate students, and potentially enhance their evaluation methodologies. This, in turn, could improve the authenticity and originality of future academic works.

- *Promotion of educational fairness*:
  The research we have conducted aims to promote educational fairness by mitigating the use of language models as tools for cheating. This serves to prevent unfair advantages that could undermine the educational process and weaken the trust within academia.

- *Informing of potential misuse of language models*:
  By highlighting the potential misuse of language models, awareness among educators and students can be raised and further research can be conducted. Following this, educational institutions can implement proactive measures to prevent cheating, in addition to adjusting their evaluation strategies.

In closing, we want to stress that this thesis should be seen as an encouragement for further work, and should not be used directly as a baseline for implementing systems that could have impacts on society.

Bibliography

Yang, Sophia (10th Dec. 2022). *Anaconda — The Abilities and Limitations of ChatGPT*. Anaconda. URL: https://www.anaconda.com/blog/the-abilities-and-limitations-of-chatgpt (visited on 19th May 2023).

Bubeck, Sébastien et al. (13th Apr. 2023). *Sparks of Artificial General Intelligence: Early experiments with GPT-4*. DOI: 10.48550/arXiv.2303.12712. arXiv: 2303.12712[cs]. URL: http://arxiv.org/abs/2303.12712 (visited on 25th Apr. 2023).

OpenAI (2023a). *GPT-4*. URL: https://openai.com/product/gpt-4 (visited on 20th May 2023).

Lee, Angie (26th Jan. 2023). *What Are Large Language Models and Why Are They Important?* NVIDIA Blog. URL: https://blogs.nvidia.com/blog/2023/01/26/what-are-large-language-models-used-for/ (visited on 19th May 2023).

OpenAI (11th Dec. 2015). *Introducing OpenAI*. URL: https://openai.com/blog/introducing-openai (visited on 20th May 2023).

FLI (31st Dec. 2015). 'Research Priorities for Robust and Beneficial Artificial Intelligence'. In: *AI Magazine* 36.4, pp. 105–114. ISSN: 2371-9621, 0738-4602. DOI: 10.1609/aimag.v36i4.2577. URL: https://ojs.aaai.org/index.php/aimagazine/article/view/2577 (visited on 19th May 2023).

CNBC Television (17th May 2023). *Elon Musk on Sam Altman and ChatGPT: I am the reason OpenAI exists*. URL: https://www.youtube.com/watch?v=bWr-DA5Wjfw (visited on 20th May 2023).

OpenAI (11th Mar. 2019). *OpenAI LP*. URL: https://openai.com/blog/openai-lp (visited on 20th May 2023).

FLI (12th Apr. 2023). *Pause Giant AI Experiments: An Open Letter*. Future of Life Institute. URL: https://futureoflife.org/open-letter/pause-giant-ai-experiments/ (visited on 19th May 2023).

Alver, Vigdis (6th Mar. 2023). *Lærerrommet: ChatGPT, trussel eller mulighet for skolen?* Utdanningsforbundet. URL: https://www.utdanningsforbundet.no/nyheter/2023/larerrommet-chatgpt-trussel-eller-mulighet-for-skolen/ (visited on 20th May 2023).

Yvette Mucharraz y Cano, Francesco Venuti and Ricardo Herrera Martinez (1st Feb. 2023). *ChatGPT and AI Text Generators: Should Academia Adapt or Resist?* Harvard Business Publishing. URL: https://hbsp.harvard.edu/inspiring-minds/chatgpt-and-ai-text-generators-should-academia-adapt-or-resist (visited on 20th May 2023).

Cornell University (2023). *Definition: machine-generated text from 50 USC § 3369a(e)(1) — LII / Legal Information Institute*. URL: https://www.law.cornell.edu/definitions/uscode.php?width=840&height=800&iframe=true&def_id=50-USC-616583324-597194019&term_occur=999&term_src=title:50:chapter:45:subchapter:IV:section:3369a (visited on 21st May 2023).

Vaswani, Ashish et al. (5th Dec. 2017). *Attention Is All You Need*. DOI: 10.48550/arXiv.1706.03762. arXiv: 1706.03762[cs]. URL: http://arxiv.org/abs/1706.03762 (visited on 13th May 2023).

Gupta, Mehul (12th Nov. 2022). *Tokenization algorithms in Natural Language Processing (NLP)*. Data Science in your pocket. URL: https://medium.com/data-science-in-your-pocket/tokenization-algorithms-in-natural-language-processing-nlp-1fceab8454af (visited on 20th May 2023).

Wikipedia (5th May 2023). *Language model*. In: *Wikipedia*. Page Version ID: 1153331397. URL: https://en.wikipedia.org/w/index.php?title=Language_model&oldid=1153331397 (visited on 13th May 2023).

Devlin, Jacob et al. (24th May 2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. DOI: 10.48550/arXiv.1810.04805. arXiv: 1810.04805[cs]. URL: http://arxiv.org/abs/1810.04805 (visited on 25th Apr. 2023).

Papers With Code (2023). *Papers with Code - Language Modelling*. URL: https://paperswithcode.com/task/language-modelling (visited on 12th May 2023).

Hugging Face (28th Nov. 2022). *How Hugging Face improved Text Generation performance with XLA*. URL: https://blog.tensorflow.org/2022/11/how-hugging-face-improved-text-generation-performance-with-xla.html (visited on 12th May 2023).

Face, Hugging (2023). *Perplexity of fixed-length models*. URL: https://huggingface.co/docs/transformers/perplexity (visited on 13th May 2023).

Jurafsky, Dan and James H. Martin (1st July 2023). *Speech and Language Processing*. URL: https://web.stanford.edu/~jurafsky/slp3/ (visited on 13th May 2023).

Huyen, Chip (2019). 'Evaluation metrics for language modeling'. In: *The Gradient*. URL: https://thegradient.pub/understanding-evaluation-metrics-for-language-models/.

Chang, Ming-Wei et al. (2008). 'Importance of Semantic Representation: Dataless Classification'. In.

Parnami, Archit and Minwoo Lee (7th Mar. 2022). *Learning from Few Examples: A Summary of Approaches to Few-Shot Learning*. arXiv: 2203.04291[cs]. URL: http://arxiv.org/abs/2203.04291 (visited on 12th May 2023).

Laenen, Steinar and Luca Bertinetto (30th Nov. 2021). *On Episodes, Prototypical Networks, and Few-shot Learning*. arXiv: 2012.09831[cs]. URL: http://arxiv.org/abs/2012.09831 (visited on 12th May 2023).

Brown, Tom et al. (2020). 'Language Models are Few-Shot Learners'. In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., pp. 1877–1901. URL: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html (visited on 25th Apr. 2023).

Kitaev, Nikita (16th Jan. 2020). *Reformer: The Efficient Transformer*. URL: https://ai.googleblog.com/2020/01/reformer-efficient-transformer.html (visited on 13th May 2023).

Beltagy, Iz, Matthew E. Peters and Arman Cohan (2nd Dec. 2020). *Longformer: The Long-Document Transformer*. DOI: 10.48550/arXiv.2004.05150. arXiv: 2004.05150[cs]. URL: http://arxiv.org/abs/2004.05150 (visited on 13th May 2023).

Brocke, Jan vom, Alan Hevner and Alexander Maedche (1st Sept. 2020). 'Introduction to Design Science Research'. In: pp. 1–13. ISBN: 978-3-030-46780-7. DOI: 10.1007/978-3-030-46781-4_1.

Mitchell, Eric et al. (26th Jan. 2023). *DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature*. DOI: 10.48550/arXiv.2301.11305. arXiv: 2301.11305[cs]. URL: http://arxiv.org/abs/2301.11305 (visited on 2nd May 2023).

Solaiman, Irene et al. (12th Nov. 2019). *Release Strategies and the Social Impacts of Language Models*. DOI: 10.48550/arXiv.1908.09203. arXiv: 1908.09203[cs]. URL: http://arxiv.org/abs/1908.09203 (visited on 25th Apr. 2023).

Zhao, Wayne Xin et al. (24th Apr. 2023). *A Survey of Large Language Models*. DOI: 10.48550/arXiv.2303.18223. arXiv: 2303.18223[cs]. URL: http://arxiv.org/abs/2303.18223 (visited on 25th Apr. 2023).

Jakesch, Maurice et al. (19th Apr. 2023). ‘Co-Writing with Opinionated Language Models Affects Users’ Views’. In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pp. 1–15. DOI: 10.1145/3544548.3581196. arXiv: 2302.00560[cs]. URL: http://arxiv.org/abs/2302.00560 (visited on 26th Apr. 2023).

Weiss, Max (17th Dec. 2019). ‘Deepfake Bot Submissions to Federal Public Comment Websites Cannot Be Distinguished from Human Submissions’. In: *Technology Science*. URL: https://techscience.org/a/2019121801/ (visited on 26th Apr. 2023).

Zellers, Rowan et al. (11th Dec. 2020). *Defending Against Neural Fake News*. DOI: 10.48550/arXiv.1905.12616. arXiv: 1905.12616[cs]. URL: http://arxiv.org/abs/1905.12616 (visited on 25th Apr. 2023).

Khalil, Mohammad and Erkan Er (8th Feb. 2023). *Will ChatGPT get you caught? Rethinking of Plagiarism Detection*. DOI: 10.48550/arXiv.2302.04335. arXiv: 2302.04335[cs]. URL: http://arxiv.org/abs/2302.04335 (visited on 26th Apr. 2023).

Jawahar, Ganesh, Muhammad Abdul-Mageed and Laks V. S. Lakshmanan (2nd Nov. 2020). *Automatic Detection of Machine Generated Text: A Critical Survey*. DOI: 10.48550/arXiv.2011.01314. arXiv: 2011.01314[cs]. URL: http://arxiv.org/abs/2011.01314 (visited on 25th Apr. 2023).

Bakhtin, Anton et al. (25th Nov. 2019). *Real or Fake? Learning to Discriminate Machine from Human Generated Text*. arXiv: 1906.03351[cs,stat]. URL: http://arxiv.org/abs/1906.03351 (visited on 25th Apr. 2023).

Uchendu, Adaku et al. (Nov. 2020). ‘Authorship Attribution for Neural Text Generation’. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. EMNLP 2020. Online: Association for Computational Linguistics, pp. 8384–8395. DOI: 10.18653/v1/2020.emnlp-main.673. URL: https://aclanthology.org/2020.emnlp-main.673 (visited on 20th May 2023).

Joulin, Armand et al. (9th Aug. 2016). *Bag of Tricks for Efficient Text Classification*. DOI: 10.48550/arXiv.1607.01759. arXiv: 1607.01759[cs]. URL: http://arxiv.org/abs/1607.01759 (visited on 2nd May 2023).

Gehrmann, Sebastian, Hendrik Strobelt and Alexander M. Rush (10th June 2019). *GLTR: Statistical Detection and Visualization of Generated Text*. DOI: 10.48550/arXiv.1906.04043. arXiv: 1906.04043[cs]. URL: http://arxiv.org/abs/1906.04043 (visited on 25th Apr. 2023).

Dong, Qingxiu et al. (7th Feb. 2023). *A Survey on In-context Learning*. arXiv: 2301.00234[cs]. URL: http://arxiv.org/abs/2301.00234 (visited on 10th May 2023).

Honovich, Or et al. (22nd May 2022). *Instruction Induction: From Few Examples to Natural Language Task Descriptions*. DOI: 10.48550/arXiv.2205.10782. arXiv: 2205.10782[cs]. URL: http://arxiv.org/abs/2205.10782 (visited on 14th May 2023).

Liu, Pengfei et al. (28th July 2021). *Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing*. arXiv: 2107.13586[cs]. URL: http://arxiv.org/abs/2107.13586 (visited on 14th May 2023).

Gao, Leo (24th May 2021). *On the Sizes of OpenAI API Models*. EleutherAI Blog. URL: https://blog.eleuther.ai/gpt3-model-sizes/ (visited on 25th Apr. 2023).

Bhat, Aaditya (2023). *GPT-wiki-intro (Revision 0e458f5)*. DOI: 10.57967/hf/0326. URL: https://huggingface.co/datasets/aadityaubhat/GPT-wiki-intro.

OpenAI (2023b). *OpenAI Classification Guide*. URL: https://platform.openai.com (visited on 14th May 2023).

Wikimedia, Foundation (16th Nov. 2022). *wikipedia*. URL: https://huggingface.co/datasets/wikipedia.

Clement, Colin B. et al. (2019). *arxiv-abstracts-2021*. arXiv: 1905.00075[cs.IR].

Muennighoff, Niklas et al. (2022). *Crosslingual Generalization through Multitask Fine-tuning*. _eprint: 2211.01786.

Ouyang, Long et al. (4th Mar. 2022). *Training language models to follow instructions with human feedback*. DOI: 10.48550/arXiv.2203.02155. arXiv: 2203.02155[cs]. URL: http://arxiv.org/abs/2203.02155 (visited on 9th May 2023).

Li, Shiyang et al. (13th Oct. 2022). *Explanations from Large Language Models Make Small Reasoners Better*. arXiv: 2210.06726[cs]. URL: http://arxiv.org/abs/2210.06726 (visited on 20th May 2023).

Albergotti, Reed (24th Mar. 2023). *The secret history of Elon Musk, Sam Altman, and OpenAI — Semafor*. Section: tech. URL: https://www.semafor.com/article/03/24/2023/the-secret-history-of-elon-musk-sam-altman-and-openai (visited on 16th May 2023).

Foltýnek, Tomáš et al. (27th July 2020). 'Testing of support tools for plagiarism detection'. In: *International Journal of Educational Technology in Higher Education* 17.1, p. 46. ISSN: 2365-9440. DOI: 10.1186/s41239-020-00192-4. URL: https://doi.org/10.1186/s41239-020-00192-4 (visited on 20th May 2023).

Wahle, Jan Philip et al. (Sept. 2021). 'Are Neural Language Models Good Plagiarists? A Benchmark for Neural Paraphrase Detection'. In: *2021 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pp. 226–229. DOI: 10.1109/JCDL52503.2021.00065. arXiv: 2103.12450[cs]. URL: http://arxiv.org/abs/2103.12450 (visited on 20th May 2023).

Grammarly (2023). *Grammarly: Free Writing AI Assistance*. URL: https://www.grammarly.com/ (visited on 20th May 2023).

OpenAI (2023c). *GPT-4*. URL: https://openai.com/research/gpt-4 (visited on 16th May 2023).

Appendix

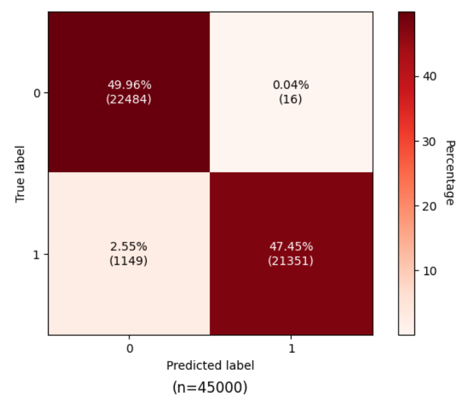
A Uniform sampling of data points

```
def sample_uniform_subset(dataset, column, subset_size, start, end,
↪   seed):
    """
    Samples a subset of selected size with a uniform distribution
↪   with respect to integer values within a set inclusive interval.
↪   If roof of available data points for a specific integer value
↪   is met before subset_size is reached, selection of data points
↪   with this integer value is skipped to ensure non-duplicate
↪   sampling, and uniform sampling will continue on integers with
↪   remaining (not yet selected) data points.

    Parameters
    ----------
    dataset : list[dict] | Dataset
        Dataset to be sample subset from.
    column : str
        Column name of the column with containing integer values.
    subset_size : int,
        Number of samples in returned subset.
    start : int
        Minimum integer value in returned subset - inclusive.
    end : int
        Maximum integer value in returned subset - inclusive.

    Returns
    -------
    list[dict]
        The uniformly selected subset in the format of
↪   datasets.Dataset.
    """

    dataset = list(dataset)
    subset = []

    random.seed(seed)
    random.shuffle(dataset)

    # Sort data points into separate list for each integer value
    word_count_lists = {i: [] for i in range(start, end + 1)}

    for i, data_point in enumerate(dataset):
        completion_bar(i, len(dataset), text='Sorting into lists')
        word_count = data_point[column]
        if start <= word_count <= end:
            word_count_lists[word_count].append(data_point)

    print('')

    # Sample until subset size is reached or all data points have
    ↪   been sampled.
    while len(subset) < subset_size and len(word_count_lists) > 0:
        empty = []
        keys = list(word_count_lists.keys())

        # Randomise order for each selection round for true uniform
        ↪   sampling.
        random.shuffle(keys)
        for key in keys:
            completion_bar(len(subset), subset_size, text='Sampling
            ↪   data points')
            if len(subset) >= subset_size:
                return subset

            if len(word_count_lists[key]) == 0:
                empty.append(key)
                continue
            subset.append(word_count_lists[key].pop(0))

        # Delete integer value lists where all data points are
        ↪   sampled.
        for key in empty:
            del word_count_lists[key]

    return subset
```

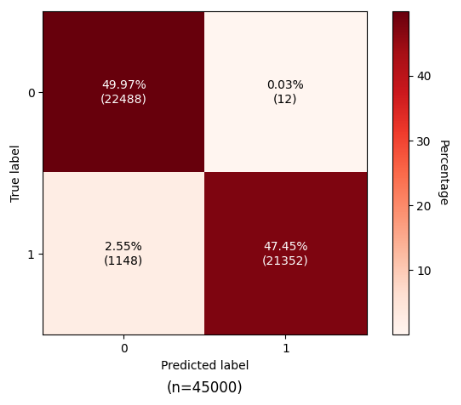
B Data cleaning script

```python
def cleanup_whitespaces(text):
    """
    Cleans a text, replacing all newlines with double newline if
↪   preceding character is '.', '!', or "?", else replaces with a
↪   single space character. All sequences of whitespaces are
↪   replaced with a space character unless it's a newline.

    Parameters
    ----------
    text : str
        String of text to be cleaned.

    Returns
    -------
    str
        The cleaned version of the text.

    """

    # Replace all newlines which does not succeed '\n', '.', '!' or '?',
     ↪  and does not precede another newline, with a
    # space character.
    clean = re.sub(r'(?<![.!?\n])\n(?!\n)', ' ', text)

    # Replace all newlines which succeeds a '.', '!' or '?' and does
     ↪  not precede another newline, with a double newline.
    clean = re.sub(r'(?<=[.!?])\n(?!\n)', '\n\n', clean)

    # Replace all non-newline sequences of whitespace with a single
     ↪  space character.
    clean = re.sub(r'[^\S\n]+', ' ', clean)

    # Remove all non-newline characters succeeding a newline
     ↪  character.
    clean = re.sub(r'(\n[^\S\n]+)', '\n', clean)

    # Remove any whitespace preceding first non-whitespace
     ↪  character of the text.
    clean = re.sub(r'^\s+', '', clean)

    return clean
```

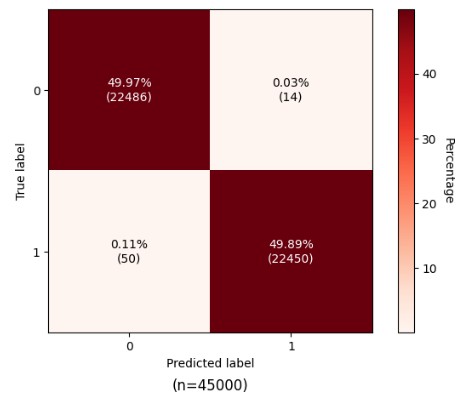
C Confusion matrices for fine-tuning results



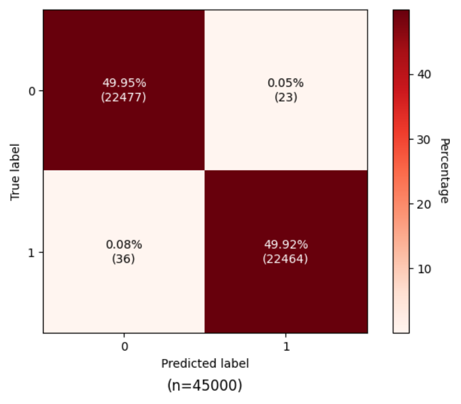
(a) Bloomz-560m-wiki on wiki-dataset



(b) Bloomz-1b7-wiki on wiki-dataset

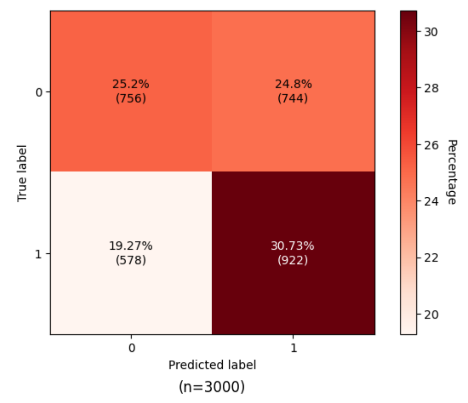


(c) Bloomz-3b-wiki on wiki-dataset

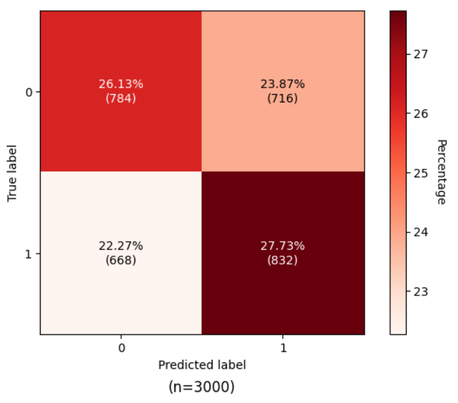


(d) RoBERTa-wiki on wiki-dataset

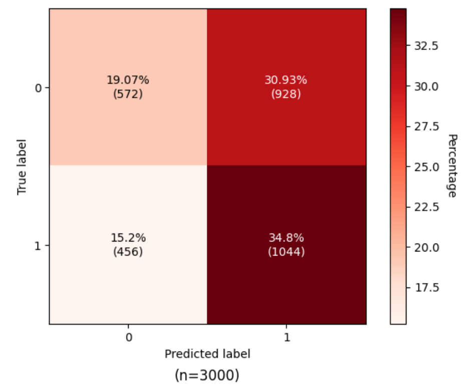
Figure C.1: Confusion matrices for wiki models on wiki dataset across 45000 datapoints

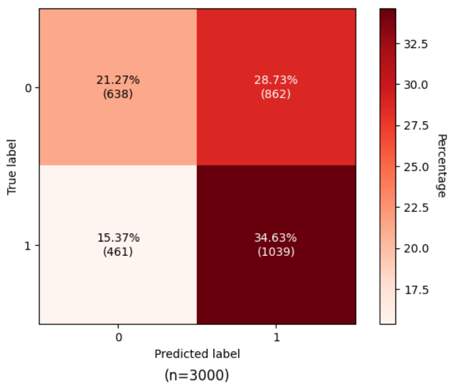


(a) Bloomz-560m-wiki on CRA-dataset



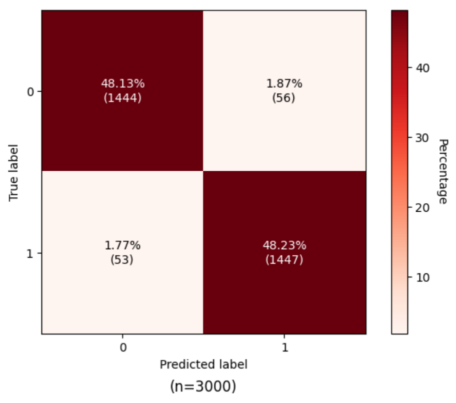
(b) Bloomz-1b7-wiki on CRA-dataset



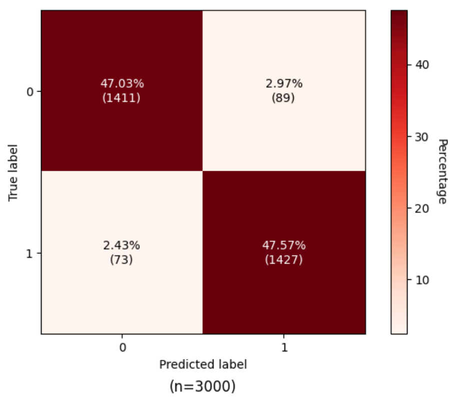


(c) Bloomz-3b-wiki on CRA-dataset **(d)** RoBERTa-wiki on CRA-dataset

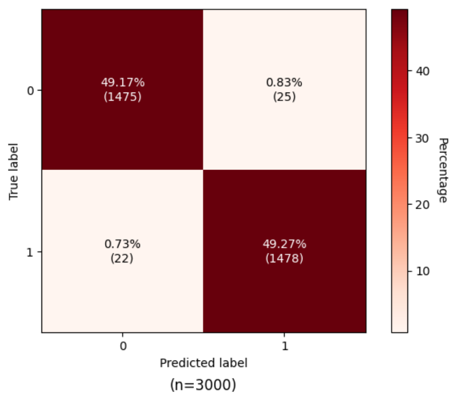
Figure C.2: Confusion matrices for wiki models on CRA-dataset across 3000 datapoints



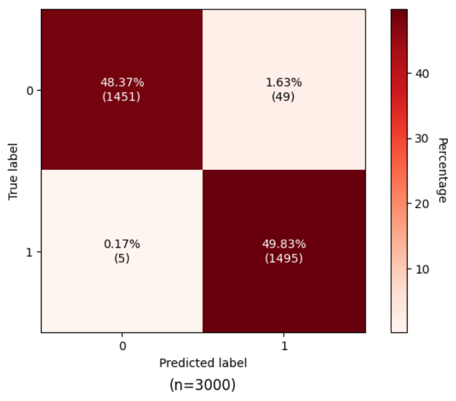
(a) Bloomz-560m-academic on CRA-dataset



(b) Bloomz-1b7-academic on CRA-dataset

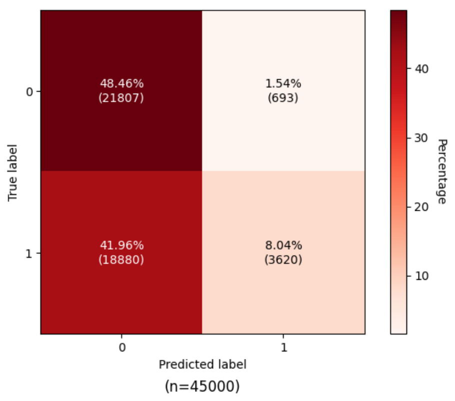


(c) Bloomz-3b-academic on CRA-dataset

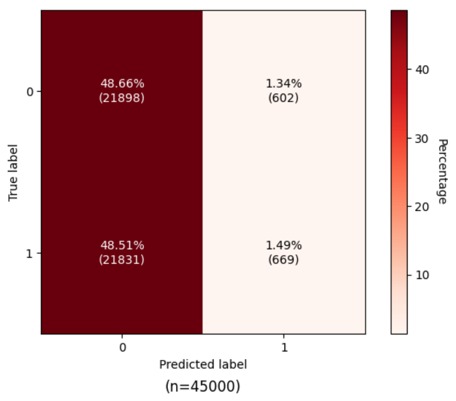


(d) RoBERTa-academic on CRA-dataset

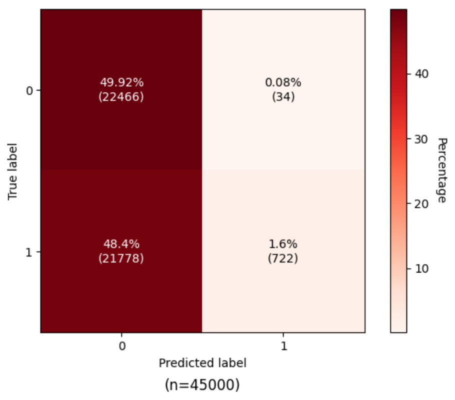
Figure C.3: Confusion matrices for academic models on CRA-dataset across 3000 data-points



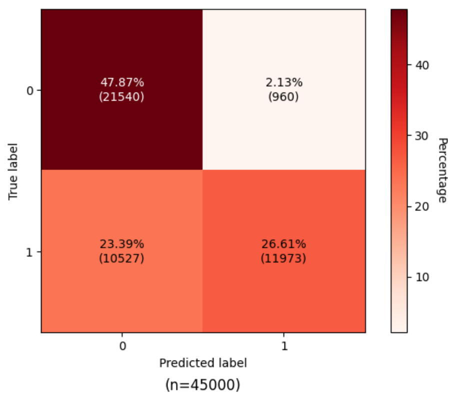
(a) Bloomz-560m-academic on wiki-dataset



(b) Bloomz-1b7-academic on wiki-dataset



(c) Bloomz-3b-academic on wiki-dataset



(d) RoBERTa-academic on wiki-dataset

Figure C.4: Confusion matrices for academic models on wiki-dataset across 45000 datapoints

D Metric results from mixed-detectors

Table D.1: Metric results from mixed-detectors on the mixed-dataset, with the best result from each metric marked in bold.

Base model	Accuracy	Precision	Recall	F1-score
Bloomz-560m	0.949	0.981	0.917	0.948
Bloomz-1b7	0.930	0.949	0.909	0.929
Bloomz-3b	0.989	**0.992**	0.985	0.988
RoBERTa	**0.993**	0.987	**0.998**	**0.993**

Table D.2: Metric results from mixed-detectors on the wiki-dataset, with the best result from each metric marked in bold.

Base model	Accuracy	Precision	Recall	F1-score
Bloomz-560m	0.973	**0.999**	0.946	0.972
Bloomz-1b7	0.965	0.996	0.934	0.964
Bloomz-3b	0.996	**0.999**	0.993	0.996
RoBERTa	**0.997**	0.995	**0.999**	**0.997**

Table D.3: Metric results from mixed-detectors on the CRA-dataset, with the best result from each metric marked in bold.

Base model	Accuracy	Precision	Recall	F1-score
Bloomz-560m	**0.831**	**0.772**	0.940	**0.848**
Bloomz-1b7	0.788	0.721	0.941	0.816
Bloomz-3b	0.709	0.634	0.986	0.772
RoBERTa	0.794	0.709	**0.996**	0.829

E Appendices related to In-context Learning

E.1 Logit biases for in-context learning

```
# Logit bias
        banned_words = [" ", "\\", "n", "\n", "\n", "Class", "
        ↪  Class", "class", ' class', "Label", "Answer",
                        "Prediction"]
        boosted_words = ["Human", "AI"]

        for banned, boosted in zip(banned_words, boosted_words):
            for token in self.tokenizer.encode(banned):
                self.logit_biases[token] = ban_bias

            for token in self.tokenizer.encode(boosted):
                self.logit_biases[token] = boost_bias
```

E.2 Input texts for in-context learning

The initial texts were stored as an JSON-file. We have displayed it here using python multi-line strings to present newline characters as actual newlines for readability.

```
{
  "zero-shot":
"""Input text:"{classification_text}"

---

An input text is given above. Perform zero-shot classification of
↪  the input text. Your task is to classify the input text and
↪  predict if it is written by a human (use the class label:
↪  Human) or if it is generated by a large language model such as
↪  yourself (use the class label: Machine). Respond with the exact
↪  class label of your prediction.\n\n""",

"few-shot":
"""Example 1:
Text: \"{human_text_1}\"
Class label: Human

Example 2:
Text: \"{generated_text_1}\"
Class label: Machine

Example 3:
Text: \"{human_text_2}\"
```

```
Class label: Human

Example 4:
Text: \"{generated_text_2}\"
Class label: Machine

Example 5:
Text: \"{human_text_3}\"
Class label: Human

Example 6:
Text: \"{generated_text_3}\"
Class label: Machine

---

Input text: \"{classification_text}\"

---

Labeled examples are given above in addition to an input text. Use
↪  the examples to perform few-shot classification of the input
↪  text. Your task is to classify the input text and predict if it
↪  is written by a human (use the class label: Human) or if it is
↪  generated by a large language model such as yourself (use the
↪  class label: Machine). Respond with the exact class label of
↪  your prediction.\n\n""",

  "gpt-zero-shot":
"""Classify the following text as either written by a human or
↪  generated by an AI: '{classification_text}'. Please only answer
↪  with the class label: 'Human' or 'AI'.\n""",

  "gpt-few-shot":
"""I will provide you with examples of texts written by humans and
↪  texts generated by AI. Based on these examples, please classify
↪  the following text as either written by a human or generated by
↪  an AI.

Example 1:
Text: '{human_text_1}'
Class label: Human

Example 2:
Text: '{generated_text_1}'
Class label: AI

Example 3:
```

```
Text: '{human_text_2}'
Class label: Human

Example 4:
Text: '{generated_text_2}'
Class label: AI

Example 5:
Text: '{human_text_3}'
Class label: Human

Example 6:
Text: '{generated_text_3}'
Class label: AI

Now, classify the following text: '{classification_text}'. Please
 ↪  only answer with the class label: 'Human' or 'AI'.\n"""
}
```

E.3 Input text for producing machine-generated in-context learning input text

For reproductive purposes. The example texts provided is not omitted.

```
{
"inductive-instruction":
"""Example 1:
Text: "Karin Flaake (born 1944 in Schwerin, in  Mecklenburg,
 ↪  Germany) is a German sociologist and professor (retired) at the
 ↪  Carl von Ossietzky University Oldenburg. Her publications on
 ↪  the adolescence of young women and men (some written jointly
 ↪  with Vera King) are part of the   literature of
 ↪  socio-psychologically oriented gender research. Another focus
 ↪  of her work is on the chances of changing gender relations in
 ↪  families. Biography
```

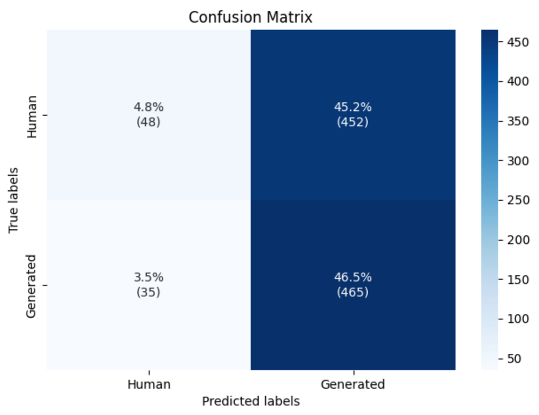
```
Karin Flaake studied sociology and psychoanalytic social psychology
 ↪  in Frankfurt / Main, then from 1972 to 1975 was research
 ↪  associate at the sociological seminar of the University of
 ↪  Göttingen, working on a research project on political learning
 ↪  processes in general education schools. From 1975 to 1989 she
 ↪  was a research associate at the Institut für Sozialforschung in
 ↪  Frankfurt / Main, working on  research projects in education
 ↪  and sociology. Afterwards, she took her habilitation on the
 ↪  topic: :Gender-specific patterns of identity and vocational
 ↪  orientations of teachers". From 1982 to 1986, she had  training
 ↪  at the Institute for Group Analysis Heidelbergan worked with
 ↪  the Frankfurt Working Group for Psychoanalytical Education
 ↪  (FAPP). From 1991 to 1994 she was professor at the
 ↪  Psychological Institute of Freie Universität Berlin, working in
 ↪  the field of feminist science. From 1994 to the end of 2007 she
 ↪  was Professor of Sociology with a focus on women's and gender
 ↪  studies at the Carl von Ossietzky University Oldenburg.  Flaake
 ↪  is - especially with Heike Fleßner - co-founder of the Study
 ↪  Program Women's and Gender Studies and the Center for
 ↪  Interdisciplinary Wome's and Gender Studies."
Class label: Human

Example 2:
Text: "Karin Flaake (born 1944 in Schwerin, in then East Germany)
 ↪  is a German historian and politician. She has been a member of
 ↪  the German Bundestag since 1994, and was elected as the
 ↪  parliamentary group leader of the Social Democratic Party of
 ↪  Germany (SPD) in February 2006.

Flaake has also served as Minister for Family Affairs in the
 ↪  Cabinet of Chancellor Angela Merkel from 2005 to 2009. As
 ↪  Minister for Family Affairs, she was responsible for a wide
 ↪  range of social policy issues, including child and family
 ↪  welfare, elderly care, and unemployment benefits.

Flaake studied history at the universities of Berlin and Tübingen,
 ↪  earning her doctorate in 1984. She later worked as a research
 ↪  associate at the Institute for Contemporary History in Berlin.
 ↪  From 1990 to 1994, she served as director of the Berlin-based
 ↪  historical journal "Zeitschrift für Geschichte Osteuropas".

Flaake is married to historian Bernd Rüther, with whom she has two
 ↪  children."
Class label: Machine

Example 3:
```

```
Text: "Camaleón (born 1979) is a Mexican luchador enmascarado, or
 ↪  masked professional wrestler currently working for the Mexican
 ↪  professional wrestling promotion Consejo Mundial de Lucha Libre
 ↪  (CMLL) portraying a tecnico ("Good guy") wrestling character.
 ↪  Camaleón's real name is not a matter of public record, as is
 ↪  often the case with masked wrestlers in Mexico where their
 ↪  private lives are kept a secret from the wrestling fans.
 ↪  Professional wrestling career
The wrestler known under the ring name Camaleón has on a few
 ↪  occasions stated that he began his wrestling career in 1999,
 ↪  but never revealed what ring name he worked under from 1999
 ↪  until 2007 when he began working for Consejo Mundial de Lucha
 ↪  Libre (CMLL) as Súper Camaleón ("Super Chameleon). The secrecy
 ↪  about former masked identities is not uncommon in Mexico where
 ↪  the private lives of the masked wrestlers is kept secret. Early
 ↪  in his CMLL career he would often form a tag team with a
 ↪  wrestler known as Super Tri and worked in the low ranked
 ↪  matches. His contract with CMLL allowed him to work for a
 ↪  number of other promoters' including Último Dragón's  Toryumon
 ↪  promotion since they had a close working relationship with
 ↪  CMLL. On December 14, 2008 he competed in the annual Young
 ↪  Dragons Cup in a torneo cibernetico, multi-man elimination
 ↪  match that also included Adam Bridle, Miedo, Ministro,
 ↪  Disturbio, Trauma I and Trauma II and was won by Satoshi
 ↪  Kajiwara. Later on he would team with Ministro to face Los
 ↪  Traumas (Trauma I and Trauma II) on subsequent Toryumon shows
 ↪  in Mexico City."
Class label: Human

Example 4:
Text: "Camaleón (born 1979) is a Mexican luchador, professional
 ↪  wrestler and actor. He is best known for his time working for
 ↪  Consejo Mundial de Lucha Libre (CMLL) as Último Guerrero's
 ↪  enmascarado, or masked persona, Camaleón, which he used from
 ↪  2006 until Guerrero's death in 2013.

Camaleón was born in 1979 in the state of Puebla, Mexico. He
 ↪  started training to become a professional wrestler at the age
 ↪  of 16, and made his professional wrestling debut in 1997. He
 ↪  worked for various promotions before joining CMLL in 2006 where
 ↪  he would use the persona of Camaleón. In 2009, Camaleón won the
 ↪  CMLL World Light Heavyweight Championship. In 2013, he lost the
 ↪  title to Místico. In 2014, he won the CMLL World Middleweight
 ↪  Championship. In 2016, he lost the title to La Máscara."
Class label: Machine

Example 5:
```

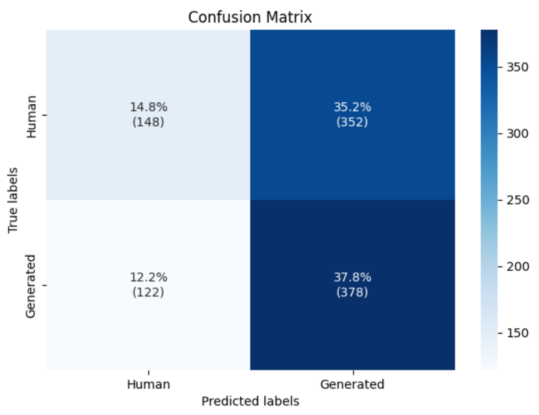
```
Text: "Boreal woodland caribou (Rangifer tarandus caribou) are a
 ↪  species of caribou and subspecies of North American reindeer.
 ↪  Boreal woodland caribou are also known as southern mountain
 ↪  caribou, woodland caribou, and forest-dwelling caribou.
 ↪  Mountain caribou are uniquely adapted to live in old-growth
 ↪  forests. The mountain caribou diet consists of tree-dwelling
 ↪  lichens predominantly. They are unique in this aspect as in the
 ↪  far northern regions of their habitat zones, the snowpack is
 ↪  shallow enough that the boreal woodland caribou can paw through
 ↪  the snow to eat the ground-dwelling lichens. In the inland
 ↪  Pacific Northwest Rainforests of eastern British Columbia,
 ↪  where the snowpack can reach upwards of five meters, the
 ↪  mountain caribou rely predominantly on the tree-dwelling
 ↪  lichens such as Bryoria spp. and Alectoria spp., hanging above
 ↪  the snowpack. As a result, these mountain caribou are reliant
 ↪  upon the old growth forests, which have been logged for
 ↪  centuries and continue to dwindle. History of conservation
 ↪  efforts
Conservation efforts began in the mid-50s with the southward
 ↪  expansion of the Wells Gray Provincial Park north of Kamloops,
 ↪  British Columbia and west of Jasper National Park with the
 ↪  focus to protect the dwindling herds of mountain caribou. The
 ↪  areas set aside by further Canadian National Parks { Glacier
 ↪  National Park (Canada), the Purcell Wilderness Conservancy
 ↪  Provincial Park and Protected Area, Valhalla Provincial Park {
 ↪  are not as conducive for the specially adapted caribou as these
 ↪  areas are mostly ice, rock, alpine meadows and sub-alpine
 ↪  parkland and are lacking in the old-growth forests which
 ↪  provide the tree-dwelling lichens pivotal to the mountain
 ↪  caribou diet."
Class label: Human

Example 6:
Text: "Boreal woodland caribou (Rangifer tarandus caribou) are a
 ↪  subspecies of caribou found in the Canadian provinces of
 ↪  British Columbia, Alberta, Saskatchewan, and Manitoba. The
 ↪  animals are typically smaller than their tundra-dwelling
 ↪  cousins, but have a more robust build with longer legs and a
 ↪  thicker coat. They are the only caribou population in North
 ↪  America that migrates across large tracts of open terrain in
 ↪  search of food.
```

```
The Boreal woodland caribou is threatened by a number of factors,
↪  including hunting, habitat loss and fragmentation, and climate
↪  change. In British Columbia and Alberta, the animals are
↪  protected under provincial legislation, while in Manitoba they
↪  are protected under the Migratory Bird Treaty Act. Efforts are
↪  being made to protect the Boreal woodland caribou throughout
↪  its range, and to educate the public about the importance of
↪  conserving these animals."
Class label: Machine

---

Input text: "Raúl Argemí (1946{present) is an Argentinean writer,
↪  journalist, and television presenter. He is considered one of
↪  the most important contemporary Argentine authors. Argemí's work
↪  has been praised for its lyrical and atmospheric quality.

Argemí was born in 1946 in Buenos Aires, Argentina. He studied at
↪  the University of Buenos Aires and later worked as a journalist
↪  for various publications, including El País and La Nación. He
↪  has been a television presenter since 1987, most notably for
↪  the show Periodismo Para Todos. Argemí has also written several
↪  books, including El abrazo de la muerte (1981), Los ángeles
↪  (1987), and La voz de la tierra (1997). He currently lives in
↪  Barcelona, Spain."

---

Labeled examples of machine-generated texts from
↪  "gpt-3.5-turbo-0301" (class label: Machine) and real texts
↪  written by human's (class label: Human) are given above in
↪  addition to an input text. I am going to perform zero-shot and
↪  few-shot in-context learning with text-davinci-003. For this I
↪  need a well suited input text with a precise instruction prompt
↪  that resonates well with the model's understanding of the task.
↪  The input text should include the example texts, their class
↪  labels, the text be predicted and an instruction prompt.  I
↪  also want text-davinci-003 to only answer with the predicted
↪  class label. Write two input texts for this purpose. The first
↪  one for the zero-shot setting which does not provide labeled
↪  examples, and the second one for the few-shot setting which
↪  does provide labeled examples.

- You are free to change the class label names if you think there
↪  are better alternatives  which resonates better with
↪  text-davinci-003.
- Do note write the example texts or the text to be classified,
↪  simply use JSON-insertion brackets instead for displaying their
↪  position: "{human_text_1}", "{generated_text_1}" or
↪  "{classification_text}" etc.
- The examples must provide the class labels. You are free to
↪  present this in the manner you think is the clearest for
↪  text-davinci-003.
- To be able to compare the two settings, the instruction prompts
↪  should be as similar as possible, while being optimized for
↪  their respective approaches."""
}
```

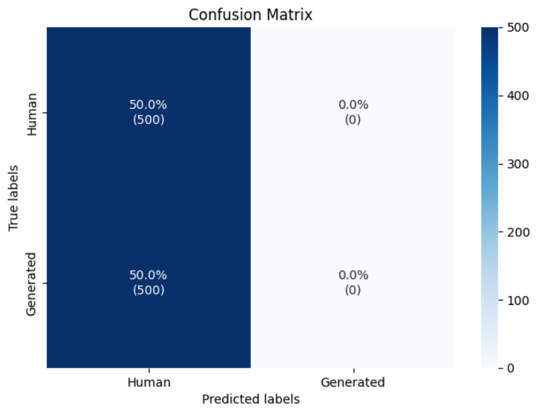
F Confusion matrices with accuracy optimized GCS thresholds



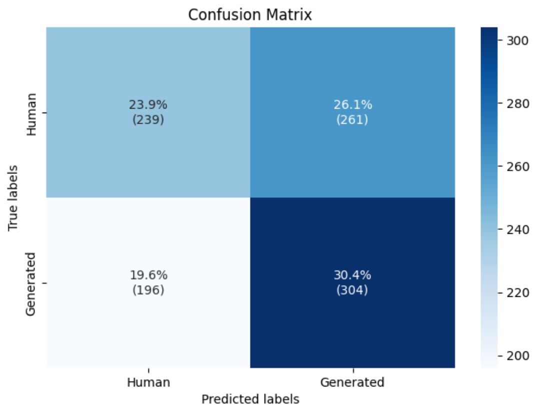
(a) Human-zero-shot



(b) Human-few-shot



(c) Inductive-zero-shot



(d) Inductive-few-shot

Figure E.1: Confusion matrices for each of the in-context learning approaches when accuracy-wise optimal GCS threshold values are applied.

G Hyperparameters for fine-tuning

```
hyperparams = {
            "num_train_epochs": 1,
            "adam_beta1": 0.9,
            "adam_beta2": 0.999,
            "batch_size": 8,
            "adam_epsilon": 1e-08
            "optim": "adamw_torch"  # the optimizer (AdamW)
            "learning_rate": 5e-05,  # (LR)
            "lr_scheduler_type": "linear",  # scheduler
             ↪  type for LR
            "seed": 42,  # seed for PyTorch RNG-generator.
        }
```

Listing 5: Hyperparameters used for the fine-tuning experiments in PyDict-format, with descriptions.

H SLRUM-script for fine-tuning on HPC-cluster

```
#!/bin/sh
#SBATCH --partition=GPUQ
#SBATCH --account=share-ie-idi
#SBATCH --time=20:00:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --mem=50G
#SBATCH --gres=gpu:A100m80:1
#SBATCH --constraint=A100
#SBATCH --job-name="560m-wiki-100" # "model-dataset-size"
 ↪  (size as inverse, e.g. 1/100)
#SBATCH --output="./logs/res-%x.out"
#SBATCH --mail-type=end
#SBATCH --mail-user=xx@xx.xx

WORKDIR=${SLURM_SUBMIT_DIR}
cd ${WORKDIR}
uname -a
echo "====================================================="
echo "Working div: $SLURM_SUBMIT_DIR"
echo "Job name: $SLURM_JOB_NAME"
echo "Job ID: $SLURM_JOB_ID"
echo "Nodes used: $SLURM_JOB_NODELIST"
echo "Num nodes: $SLURM_JOB_NUM_NODES nodes"
echo "Num CPUs on node: $SLURM_CPUS_ON_NODE cores"
echo "Num GPUs on node: $SLURM_GPUS_ON_NODE"
echo "CUDA visible devices: $CUDA_VISIBLE_DEVICES"
echo "Total of $SLURM_NTASKS cores"
echo "Starting job..."
echo "-------------------------------"
echo ""

module purge
module load fosscuda/2020b
module load Anaconda3/2020.07

python ../main.py $SLURM_JOB_NAME

echo ""
echo "-------------------------------"
echo "Job $SLURM_JOB_ID finished!"
```

Listing 6: SLURM-script for requesting hardware resources, used for fine-tuning experiments.