Eberhard Karls Universität Tübingen
Seminar für Sprachwissenschaft

Master Thesis

Approaches to the automatic detection of machine-generated text

Winkler Aron

October 2024

Reviewers

Prof. Dr. Çağrı Çöltekin
Seminar für Sprachwissenschaft
Universität Tübingen

Prof. Dr. Gerhard Jäger
Seminar für Sprachwissenschaft
Universität Tübingen

Winkler Aron:
Approaches to the automatic detection of machine-generated text
Master Thesis
Eberhard Karls Universität Tübingen
Thesis period: July - November 2024

Abstract

Recent developments in Natural Language Processing (NLP) have resulted in the development and popularization of highly effective Large Language Models (LLMs), capable of generating convincing and seemingly creative linguistic material. LLMs have garnered much attention, both from researchers and the general public, and continue to be increasingly applied to a variety of fields. However, the breakneck speed at which these systems are adopted leaves unattended some of the security concerns regarding their use. Since the language produced by the models is of such high quality, it is not always feasible to distinguish authentically human contributions from machine-generated text, which can enable a multitude of nefarious applications of LLMs. This work explores the history and inner workings of LLMs, how they can be misused, and possible antidotes to the problem of machine-generated text detection, with a careful eye toward a good balance of computational cost and performance of detection strategies.

Acknowledgements

Contents

1 Introduction

Recent advances in the field of language generation have unlocked the door to the widespread use of language models, across almost all fields of human life – academics, marketing, arts, programming, and so forth. For anyone engaging with digital media, talk of generative artificial intelligence (AI) has become a daily occurrence, and many companies are integrating language technology in some form or another. As it stands, chatbots like ChatGPT and image generators like DALL-E are used by millions to make art, tell exciting stories, improve customer relationships, create tailor-made learning environments for students, and much more. In this panorama of exciting new technologies, however, there is no lack of worrying signs, and potential for the misuse of these innovative tools.

Language generation has been around since last century, so its massive popularization in 2020 wasn't necessarily as groundbreaking for the discipline as it was for the general public. What represented the biggest leap was the quality of the output that the language models were capable of. The generation of material largely indistinguishable from human-written text was available for everyone to use through popular interfaces like ChatGPT. Given the scale at which AI-generated content has been flooding into all manners of digital phenomena, it has become increasingly crucial to develop up-to-date technologies to detect when a piece of media (for the purposes of this work, only text is considered) is authentically human, or machine-generated.

Language generation has a huge number of positive applications, and it stands to improve people's lives in many ways. Still, the potentially nefarious uses are equally as many. In fact, even without bad intent, it is easy for model operators to generate and use harmful language in some form.

One example of language models empowering malicious actors is phishing and scamming. These attacks involve attempting to make their victims reveal sensitive information, or perform certain actions (for example, send money to the attacker or reveal their login information to some service). Phishing thus far has been (fortunately) plagued by scaling issues – to increase the number of attacks, one had to hire new human attackers, which is expensive and time-consuming. With language generation, it could become possible to bypass this limitation by assigning some (or even most) of the work to automated systems. As models grow more sophisticated, they might become even better suited to gain people's trust for an attacker to exploit, or trick them out of sensitive information outright.

The information landscape, already shaky in the digital age of social media, could also suffer from the malicious employment of language generation. Disinformation campaigns in the political, social, and commercial spheres have been commonplace throughout human history, but, similar to phishing, have been notoriously expensive to scale. The ability to generate highly convincing content to further disinforming claims is a great boon to those who seek to spread fabrications and widen divisions. Worryingly, AI is already being shown to suit the intent to *dis*inform particularly well, to the point that it may even work better as a tool for deception rather than truth.

One need not even intend to harm when employing NLG systems to inadvertently cause damage. Plagiarism, as an example, is a very real danger with language models, since at any point they may generate one-to-one reproductions of their training data. On the same note, if the model's training data contained biases and misinformation – either naturally or as a result of adversarial manipulation – the generated text could further these without obvious signs or explicit intent. Chapter 3 contains further elaboration on these issues, which were only superficially mentioned in this introduction.

In the past, automatic generation of text material relied on relatively simple technologies, and was thus comparatively easier to detect. The ability to accurately flag generated text could be relied upon with pre-LLM systems, and the quality was often insufficient for the more sophisticated tasks. Being alerted to the use of AI lessens its danger in most cases – an inflammatory article is far less effective, a scam email far less convincing, and fake homework much less troublesome when the AI authorship can be reliably exposed. Unfortunately, starting with GPT-3, not only has the quality of model output grown considerably, but even human evaluators have a tough time separating LM generations from authentic human productions. Needless to say, automatic detection systems have also suffered a decrease in effectiveness and have had to upgrade their arsenal to keep up.

This thesis aims to delve into the field of machine-generated text detection, exploring the current state of the art, as well as limitations and future possibilities. Outside of analyzing the various approaches to the problem, this work is meant as a contribution to the development of systems that do not forgo all size concerns in favor of performance. Modern detection systems rely on models whose running cost is comparable to the massive language models doing the generation – but the performance gains of this strategy do not necessarily justify the tradeoffs.

When deploying computationally heavy solutions, it is often assumed that they will not run on the end user's machine – it is after all unreasonable to expect the average user to have hundreds of Gigabytes of RAM on hand. Instead, the client software is only a relay to the centrally hosted service, with which it communicates over the network. This may be acceptable in some cases, but it is not desirable in others – for example, users may and should be reluctant to dispatch their private communications to some remote service in the name of detecting generation.

Fortunately, as with many things, system design is a game of tradeoffs with performance on the one end and complexity on the other. Highly complex systems, relying on huge models to make predictions, will likely be the best detectors of machine-generated text. This fact has been true in the field as well as in other areas of computational linguistics: state of the art models often take several GPUs to train and run, with high associated cost. Still, it is often the case that 80% of performance can be salvaged at only 20% of the cost. This work is written with the somewhat substantiated belief that a version of this proves true in designing solutions for machine-generated text detection as well.

There are many strategies that can be employed to detect machine generation at a reasonable computational costs. Some examples include pretrained embeddings, which are a valid substitute of contextual embeddings provided by LLMs, and linguistically motivated features, an classical representation of text information that can be computed cheaply. Task-specific models and even language models are still useful tools. Not all models are huge, and there is fruitful research of transformer-based models like DistilBERT, which maintain performance close to the original with far fewer parameters. These methods are discussed in higher detail in Chapter 4.

Target selection is of course important, but customer-facing system designers should aim for standalone solutions that would be able to run on mid-to-high-end machines, when this is feasible. In this work, a series of options to achieve this are presented, with some of them directly experimented with in the context of Task 8 at SemEval 2023. The proposed models to detect LM generation employ, among other strategies, GPT-2-based perplexity metrics, linguistically motivated features, pretrained embeddings, contextual embeddings with small models, and task-specific character-level models. A full discussion of the methods is presented in Chapter 5.

Note to professor: I plan to write 2-3 more paragraphs here to conclude the introduction more elegantly, but I haven't been able to come up with a version I like yet.

2 Background

In 1951, Gnome Press published Isaac Asimov's *Foundation* (Asimov, 1951), the first title of a trilogy that would go on to become one of the cornerstones of modern science fiction. In the novel, set in the distant future, scientist Hari Seldon predicts the fall of the Galactic Empire, an event that would pave the way to an era of barbarism in the story's fantastical universe.

To preserve humanity's knowledge and technical skills, Hari Seldon establishes the Foundation on an uninhabited planet on the periphery of the Empire, a sort of outpost dedicated to being the home to the archival effort. The novel follows the political and technological adventures of the Foundation and its leaders, with one of the first plot points being the first conflict between the Foundation and a major local power in the periphery, Anacreon, which declared its independence as the Empire's influence in the periphery weakened. Seeking protection from the Empire against Anacreon's expansionary stance, the Foundation hosts a diplomatic emissary from the Empire, a Lord Dorwin, finally obtaining a convoluted treaty between the Empire and Anacreon over their respective spheres of influence.

> "Before you now you see a copy of the treaty between the Empire and Anacreon – a treaty, incidentally, which is signed on the Emperor's behalf by the same Lord Dorwin who was here last week – and with it a symbolic analysis."
>
> The treaty ran through five pages of fine print and the analysis was scrawled out in just under half a page.
>
> "As you see, gentlemen, something like ninety percent of the treaty boiled right out of the analysis as being meaningless, and what we end up with can be described in the following interesting manner:
>
> "Obligations of Anacreon to the Empire: None!"
>
> "Powers of the Empire over Anacreon: None!"
>
> *Isaac Asimov, Foundation,*
> *Part II: The Encyclopedists*

At this point the Foundation's scientists, through a technique they call *"symbolic analysis"*, condense several pages of treaty into a few lines, revealing the hidden meaning behind the layers of legal dissimulation. By doing so, they expose the inability of the dying empire to exert its influence over its own periphery, and they realize that moving forward, they can only rely on themselves, marking perhaps the true starting point of the story in Asimov's *Foundation*.

Despite first reading this passage when I was a teenager, perhaps over a decade ago, these fictional twists stayed with me through the years. They were, after all, my first indirect exposure to the field of computational linguistics and natural language processing (NLP). I remember being mesmerized by the potential of machine computation applied to natural language, in what I would later learn to better define as a mixture of information retrieval and automatic text summarization.

While Asimov's pen definitely hit the mark in predicting some of the most intriguing and successful applications of NLP in the years ahead, what granted computational linguistics perhaps its brightest moment in the limelight was one of its other, albeit related, subfields: language modelling and generation.

2.1 Language modelling

Teaching a machine to understand and produce natural language is intuitively a difficult task. Even if one could reliably collect all ingredients that make up human language, creating a system that emulates it even just well-enough is a very tall order, since there would likely be millions if not billions of cases to consider. Linguists have documented hundreds of languages, each with their own grammar, peculiarities, exceptions, all of which have yet to be described under one common ruleset. Manually building a program from the ground up for even just one language is beyond what current technology is capable of.

The very first chatbot, ELIZA (Weizenbaum, 1966), simulated conversation through pattern matching and substitution, essentially repeating and paraphrasing their interlocutor's statements. While it successfully bypasses the necessity of programming a machine with *intelligence*, such an approach does not result in a system that can be described as creative in any sense. In other words, ELIZA will never write a poem, or surprise their conversation partner with a witty turn of phrase. It would never be able to tell whether May has 30 or 31 days because it has no notion of what *May* and *days* are. Teaching language is, after all, not only an issue of grammar, but one of world knowledge as well.

If *teaching* language to machines as one would to humans is not possible, and rule-based approaches such as ELIZA inevitably reach a bottleneck, then it becomes necessary to adopt a new strategy, rooted in statistics. This new approach consists in the realization that the sentence "he's wearing a circumference jacket" is much less likely to be uttered than "he's wearing a yellow jacket". Extrapolating the pattern, the set of words that can fill the gap in "he's wearing a _____ jacket" is varied, but "yellow" will have a much higher *probability* of showing up than "circumference". Language models are the tools that are employed to estimate these probabilities.

Due to recent innovations in NLP, the phrase "language model" evokes big and expensive systems, trained on huge amounts of data and costing enormous amounts of money to develop. While this is certainly understandable, the label in itself has no presupposition of size or cost. In essence, language models, and NLG (Natural Language Generation) systems in general, break down the massively complex problem of "teaching language to machines", into the more manageable task of "statistically learning what words are likely to follow others". In other words, language models produce next-word (or, more generally, next-token) probabilities based on an input sequence (Gao and Lin, 2004). After this is achieved, the resulting model can be prompted over and over, one word at a time, in order to generate long pieces of text, by a process called autoregressive generation (Lin et al., 2021).

For example, for the completion "fifteen minutes of _____", one would expect a (good) language model to offer words such as "fame" or "overtime". One idea to achieve this is to collect some linguistic data and observe what words follow "fifteen minutes of" and extrapolate a probability distribution from the observed frequencies. So-called *n-gram* language models (Chen and Goodman, 1999) are built in this fashion, with the n in *n-gram* specifying the amount of left context taken into consideration.

The simplest of these models, the bigram (2-gram) language model, records co-occurring word pairs in the sample dataset. Consequently, for this model, only the last word of a sequence determines the prediction over the following word. This results in a model that can reliably generate short collocations, such as "Marie *Curie*", but cannot generate coherent sentences, and would likely even fail to offer "fame" as a completion to "fifteen minutes of _____", since the only available context for the prediction is the word "of". To correctly predict "fame", one would need at least a 4-gram language model, which would finally allow for such a "long" context requirement. However, while taking more context tokens into

consideration increases the performance of n-gram language models, it does so at a steep (especially memory) cost: for 4-gram LMs with a vocabulary size (i.e., how many words the model knows) of 1000, for example, implementations without optimizations would require the frequency counts for 10^4 n-grams to be accessible for predictions.

Another issue that presents itself with frequency-based models is the handling of unseen n-grams. For example, the 3-gram "duck goose pony" may never come up in the model training data, in which case some near-0 probability is assigned to the sequence (due to smoothing, see for example Chen and Goodman (1999)). While some n-grams will fail to appear due to being ungrammatical or nonsensical like in "duck goose pony", others may be perfectly well-formed but either rare or just absent from the training data due to chance: if "trees need hydration" was never observed, then the model would fail to recognize this n-gram to be more likely than, for example, "trees need circumference" (assuming the latter was also not observed, which is quite likely in organic text). Even the n-gram "roundly faucet knowledge" would be equally as likely as "trees need hydration" if both were never observed in training. While the latter example can be solved by including lower-order n-grams in the probability calculation ("trees need" may have been observed even if "trees need hydration" wasn't, but "roundly faucet" is unlikely to have been observed; see Katz (1987)), the former case requires more subtle knowledge. In order to correctly assess "trees need hydration" as a quite likely n-gram, the model would need to identify the similarity between the words "water" and "hydration", and infer that "trees need hydration" should have higher probability than, say, "trees need virtual", due to "water" and "hydration" being similar.

Due to such limitations, n-gram language models are not the piece of technology that propelled language modelling to the heights that we associate with it today. The missing piece of the puzzle are neural networks (Anderson, 1995), which when applied to language generation give rise to *neural language models*.

Following the example above, both n-gram and neural language models solve the fundamental problem of estimating the probability that the word "fate" follows "fifteen minutes of". However, in order to do so, n-gram language models draw upon explicit frequency observations when generating its output, an approach that often fails to consider an adequate amount of context, or to take into account the fact that similar words appear in similar contexts. In contrast, neural language models draw upon their internal parameters - the weights and biases (Anderson, 1995) associated to the various layers of the neural network that makes up the model.

2.2 Neural language modelling

Neural networks are an extremely powerful for many applications across several disciplines. Providing an effective summary of all neural networks is a difficult task, since they manifest themselves in different variations for different tasks. Still, they are generally understood as interconnected layers of *neurons* that map an input vector of numbers to an output vector. Each neuron in the network is made up of a vector of weights, a bias, and an activation function, which are used to transform an input vector to an output number. For the purposes of this work, it is more important to understand the overall network rather than its individual parts: neural models are a way to apply complex transformations to vectors. The *parameters* of the model, i.e. the combined weights and biases of the individual neurons, can be used to encode knowledge in a way that simpler statistical models struggle to achieve.

Above, the example of "trees need water" and "trees need hydration" was briefly discussed. While n-gram models have no structural way to note the similarity between "water" and "hydration", and thus fail to recognize that the admissible contexts for the two words have

some overlap, neural networks have been employed to solve this problem exactly because of their capacity to progressively store knowledge. Word embeddings (Selva Birunda and Kanniga Devi, 2021) are a way of representing words as numerical vectors, such words with similar semantics will be close to each other in terms of vector distance. The embeddings for "water" and "hydration" would therefore be closer in vector space than "water" and "dog". Several ways have been developed to derive embeddings from text data — for example, the CBOW (Mikolov et al., 2013) algorithm uses neural networks to predict the missing word given the surrounding context through the use of a neural network. For many iterations, the model is presented with a piece of text with a gap, and the objective of guessing the missing item. With each example, the model parameters are updated, or *nudged* in the correct direction (for information on gradient descent, see Zhang (2019)), i.e. towards a state that are more conducive to the correct prediction. Importantly, among the parameters of the neural network are the embeddings for every word, which are used by the model to compute the final prediction across subsequent layers — these are random at the beginning, but progressively more and more refined. At the end of training, most model parameters are discarded, but the embeddings are kept, and hopefully the result will have captured semantic similarities between the entries.

While the CBOW algorithm discards all model parameters aside from the embeddings, it is naturally possible to train the neural model with the objective of keeping all of them, still taking advantage of the architecture's capacity to learn and store information. This is the basis for modern, highly sophisticated language models, with billions or even trillions of parameters.

2.3 Large Language Models

Language models have a long and intricate history, having been iterated upon from different perspectives and with different architectural approaches. The introduction of the transformer (Vaswani et al., 2023), a model type that for more efficient training over extremely large text data, propelled language model quality forward considerably, to the point that all language models commonly known today follow this architecture. The ability to train models with relatively little expenses allowed models to grow further and further in perplexity, eventually resulting in what we identify today as Large Language Models (LLMs). In terms of research attention, BERT (Devlin et al., 2019) and GPT-2 (Radford et al., 2019b) have perhaps been the most resonant examples earlier on.

Bidirectional Encoder Representations from Transformers (BERT) is slightly different from the language models discussed so far, in that its primary purpose is not generation. The word embeddings described above are a powerful tool for obtaining meaningful representations, but have the significant flaw of not being context-aware. For example, it's clear that the word "honey" should have different embeddings between "bees make honey" and "honey, wake up!".

BERT is a solution to this problem: instead of training a model to then only keep the embedding layer (in other words, computing all embeddings *offline*), BERT is a full-fledged language model that evaluates pieces of text as a whole to return their representations *online*. As such, when evaluating "bees make honey" and "honey, wake up!" with BERT, "honey" will have vastly different embeddings to account for the different context.

BERT was developed, in other words, to compute context-aware word representations, hence the name. It cannot be used for language generation, mostly because it's nature as a *bidirectional* model. For the example language model architectures described above, there was an underlying assumption that only the *left* context is visible to the model. This makes

intuitive sense: when generating language, only what came before influences the probability distribution of the next word. This is not the case for BERT. Since the objective here is to evaluate finished productions, BERT may use all previous and subsequent tokens at each timestep.

Despite being a fairly recent introduction to the scene, being made available in just 2018, BERT has made big waves in nearly all fields where processing natural language is even tangentially relevant. It even resulted in the birth of the somewhat jokingly named discipline of BERTology, which is meant to convey the detailed analysis of how BERT and its different variations exactly arrive at the high-quality embeddings that they are known for.

In the same year as BERT was published, another historical language model also made its debut: OpenAI's GPT-2. GPT-2 (Radford et al., 2019b) lines up closer to the popular idea of what constitutes a language model. It is large in size with 1.5 billion parameters, and was primarily conceived for language generation. In the original paper, the authors highlighted the ability of the model to approach several different problems without explicit training, such as question answering, test summarization, machine translation, and so forth. In this sense, GPT-2 is one of the first successful examples of language models displaying generalized problem-solving skills, that require both articulation and minute world knowledge.

GPT-2 has become more of a baseline than a challenger in the years following its introduction, such was its impact in research applications. It has garnered much attention and analysis, similarly to BERT, a process that also highlighted some of its flaws (see, for example, the GPT-2 unicorn story[1]). While talk about GPT-2 could not at all be considered undertone, it was perhaps cut short, in that nowadays it is more rarely employed compared to BERT. This is in large part due to its new iteration, GPT-3 (Brown et al., 2020).

2.4 Fears and reactions to LLMs

In 2020, GPT-3 was announced by the company OpenAI, and was made available to the general public through an interface called ChatGPT. With its introduction, the gates were open to the generation of high-quality text through AI. This was perhaps the first example of language model garnering tremendous general attention, not only in academic circles but from the public at large. ChatGPT even experienced service outages due to its servers not being able to handle the astounding traffic they were receiving. The introduction of GPT-3 provided an unprecedented boost to language models as a consumer technology, leading to a sort of arms race both in integrating AI into customer-facing products, as well as in model development itself. In the first wave of AI competition, Facebook's Llama (Touvron et al., 2023) and the open source model Mistral (Jiang et al., 2023) also entered the scene, alongside Google's now discontinued Bard (see for example Fowler (2023)). In a subsequent wave, further developments have reached even higher generation quality, with models such as GPT-4 (OpenAI et al., 2024) and Gemini (Team et al., 2024).

Such developments marked the beginning of language models being commonplace technology. The arts, education, technology, sales, and dozens of other fields have seen major changes to the way they operate, either in alongside or in opposition to artificial intelligence. Education, in particular, was one of the first areas to experience a problematic application of language generation, with a prevailing initial fear that students would opt to have a language model solve their homework for them. Surprisingly, programming also saw the rapid birth of assistive technology, such as Github Copilot (Chen et al., 2021), leading to fears that many

[1] `https://pbs.twimg.com/media/DzYpsJOU0AA1PO9.png:large`

software professionals would be made obsolete by the new technologies.

People employed in artistic fields, particularly writers, also took a deeply cautious stance from the beginning with respect to AI. In a landmark development, the 5-month-long strike of the Writer's Guild of America[2] resulted in an agreement which included safeguards for writers against the use of artificial intelligence[3]. Such a stance turned out to be far from unfounded, with how AI-generated content has flooded several Internet platforms. As recently as March 2024, even the Google search engine has had to address the issue of unoriginal content, which is in large part meant to target results which contain generated text[4]. At the same time, the search engine itself adopted a new feature called AI Overviews, an integrated AI-generated response to the user's query[5].

The use of large language models thus has been observed to spark mixed reactions, both emotionally and legislatively. However, what was considered so far in this examination is the *lawful* and (perhaps arguably) moral use of language technology. This does not mean that such technologies cannot be employed with harmful intent — quite the contrary. the discussion about malicious use of language generation has been left on the sidelines in the early years of LM adoption, but worries are progressively making their way to the forefront of the debate, and countermeasures are becoming increasingly pursued research objectives.

[2] An article summarising the strike can be viewed at https://www.vox.com/culture/2023/9/24/23888673/wga-strike-end-sag-aftra-contract

[3] See for example https://www.nbclosangeles.com/news/local/hollywood-writers-safeguards-against-ai-wga-agreement/3233064/ for an account of the agreement

[4] Google published a blog post explaining the changes at https://blog.google/products/search/google-search-update-march-2024/

[5] Google published a blog post explaining its new AI integration at https://blog.google/products/search/generative-ai-google-search-may-2024/

3 Threats posed by NLG systems

NLG systems, and language models in particular, have emerged as an extraordinarily useful tool in a number of creative and technical fields, but it stands to reason that they would lend themselves to nefarious applications just as well as ethical ones. Following Crothers et al. (2023), several threat models (Shostack, 2014) have been proposed to tackle the potential dangers of language generation, some of which have already had real-life realizations as well.

Threat modelling provides researchers and professionals with the tools to predict potential dangers related to resources or technologies, even (and especially) in absence of historical expertise related to the relevant threats. For example, when designing a messaging application, a possible exploitation attempt could involve a malicious actor eavesdropping on other users' conversations. A threat model designed around this scenario would try to define the characteristics of the attacker, the likely strategies used, and how to counter them, preemptively or reactively. The methodical evaluation of potential threats and the implementation of relevant protection mechanisms is of particular concern in the field of information technology, since this area is so synonymous with scalability. Indeed, when something goes wrong in systems that operate on a huge scale, as do many if not most digital services people interact with daily, the impact is often proportionally disastrous.

NLG systems integrate themselves into this picture better than one might initially assume. Importantly, language models scale far more easily than a human force. Humans need to be taught, few at a time and for a period of time, to perform a task. Automated systems, however, have no such limitation: once trained, one must only increase the compute to increase coverage, an operation that takes far less time and resources than training people.

As such, language generation makes the known universe of digital threats that much more severe. On the one hand, it boosts the scale at which already existing attacks were operating, such as scams and phishing attempts - both of which had to be performed by human actors in some way thus far. On the other hand, it creates entirely new threats which were previously unfeasible, such as generating convincing documents for homework-evasion.

3.1 Scam and phishing attempts

Attacks and exploitation attempts targeted at individuals are particularly well-suited for NLG usage, since they usually involve using natural language to convince the victim of performing some action, such as revealing sensitive information or granting access to a restricted resource.

Phishing is a cyberattack method where attackers impersonate legitimate entities to trick individuals into revealing private information, such as passwords, credit card numbers, or personal details. This is typically done through deceptive emails, messages, or websites that appear trustworthy but are actually fraudulent. Victims are often lured into clicking malicious links or downloading harmful attachments, leading to the theft of their data or the compromise of their devices. Phishing is a widespread and dangerous tactic used to gain unauthorized access to systems and conduct identity theft, financial fraud, or other malicious activities.

An example of a phishing attack could involve a fake email that appears to come from a well-known bank. The email might use the bank's logo, official-sounding language, and a convincing sender address to create a sense of urgency, such as warning the recipient that their account has been compromised. The message instructs the recipient to click on a link to verify their account information immediately. When the user clicks the link, they are directed to a counterfeit website that looks almost identical to the bank's real site. Once on

this fake site, the victim is prompted to enter their login credentials, which are then captured by the attackers. With this information, the cybercriminals can access the victim's real bank account, potentially leading to financial loss and identity theft.

NLG has been proven to be effective across several varieties of phishing attacks. Depending on the scope of the attack, one can distinguish phishing attacks targeting indiscriminate groups (massive phishing), specific communities (community phishing) or individuals in particular (spear phishing).

Regarding the former, the exploitation attempt targets a specific group or community, such as members of a social club, employees of a particular company, or even residents of a neighborhood. The attacker leverages the shared characteristics, interests, or affiliations of the community to create a more convincing and believable scam. For instance, a cybercriminal might send an email that appears to come from the community's leader, such as a club president or company CEO, announcing an upcoming event or an urgent matter requiring action. The message may ask recipients to click on a link to RSVP, pay dues, or access important information. This link, however, leads to a fraudulent website designed to capture login credentials, financial details, or other sensitive information. By exploiting the trust and familiarity within the community, these phishing attacks can be particularly effective, as victims are more likely to lower their guard and engage with the malicious content.

E-mail masquerade attacks are among the most common and most well-studied (Khonji et al., 2013) digital vectors for phishing attacks. Language generation has been studied in correlation to them even before GPT-3 came to the forefront, and has been shown to be effective at creating a variety of attacking strategies with only few hours of effort (Baki et al., 2017). Aside from providing an important method to scale up email-based attacks, language generation also renders traditional spam and phishing filters less effective.

One of the most widely used method for distinguishing legitimate emails from spam is Bayesian filtering. This approach works by analyzing each word in an email and comparing it against a database of words commonly found in spam messages. The filter calculates the probability of the email being spam based on the presence and frequency of these words. If the likelihood exceeds a certain threshold, the email is flagged as spam. Despite some shortcomings, such as being vulnerable to adversarial manipulation of the underlying frequency tables, Bayesian filtering remains an effective tool for detecting and blocking unwanted emails.

NLG systems could throw a wrench in this common approach, since large language models are capable of higher and less predictable lexical variety than traditional NLG systems. Community Targeted Phishing, as described by Giaretta and Dragoni (2019) aims explicitly to analyze the language of specific groups to craft complex emails, which with the help of today's LLMs can more easily evade traditional defenses. For example, a conversational LM could be trained on the style and publications of a well-known researcher, to then be used as a phishing actor against colleagues in the same area, who may not be directly familiar with the renowned figure. This clone may much more convincingly get victims to surrender personal information and click on malicious links, while evading.

The more dangerous variety of phishing attacks, however, is *spear phishing*. Spear phishing focuses on a single individual, using personalized information gathered from research to craft a highly convincing and tailored attack. This makes spear phishing more difficult to detect, as the attacker often impersonates someone the target knows or trusts, increasing the likelihood of success. The personalized nature of spear phishing often leads to greater harm, as the attacker aims to extract highly sensitive information or gain unauthorized access to critical systems.

The bigger limitation of these attacks – from the perpetrator's perspective – is that much time investment and human time is needed for each target. NLG systems could provide a tool for scaling such attacks, for example by being employed in early stages for automated conversations, thus gaining the victim's trust before the attack proper.

The language generation features provided by modern LLMs thus pose a substantial threat when it comes to their ability to boost the effectiveness of phishing attacks. Their ability is well suited to tricking large groups of people indiscriminately, as well as to strategies that prove insidious to threaten specific communities. They also enable the scaling up of phishing attacks targeted at individuals, since they can engage in conversation, preparing the field for the malicious extraction of sensitive information. The industry's tools for protection against traditional attacks are rendered somewhat obsolete when LLMs enter the picture, thus effectively detecting when a language model is being deployed is crucial.

3.2 Information poisoning and influence campaigns

Another area where NLG can cause a negative impact is information and communication. Disinformation campaigns have been extremely common across all of human history. One coon look as back as the Donation of Constantine, a false document used in the 13th century to justify the territorial claims of the papal state, or as recently as the election meddling by foreign states in the US elections of the 21st century.

Disinformation campaigns in particular have traditionally suffered from being dependent on human editors to compose the message and select the diffusion vector – be it emails, social media, blogs, and so on - therefore performing them at scale comes at a steep cost. Language models have the potential of solving alleviating this limitation. By relying on NLG to generate most of the content of the campaign messages, it is possible for attackers to forgo – at least in part – this resource and time-consuming factor.

Buchanan et al. (2021) explore the role of GPT-3 in a human-machine setting, studying therefore the effectiveness of GPT-3 in generating disinformation campaigns when paired with human-crafted prompts along the process. Their findings suggest that while GPT-3 is unlikely to completely replace human operators in disinformation efforts, it significantly enhances their capabilities by enabling the creation of moderate- to high-quality content at an unprecedented scale. This amplification of output could make disinformation campaigns more pervasive and harder to counter, highlighting the need for robust strategies to mitigate such threats.

Among the various activities that have been identified in disinformation campaigns, GPT-3 was observed to perform exceptionally in some, while needing stricter human supervision in others. It excelled in *narrative reiteration*, a simple task that involves generating multiple variations of a short message that repeat a particular theme. For example, when providing the model with a number of successful climate-change denier tweets, researchers were able to obtain high-quality approximations on the first try and without particular refinery. With how low the effort to mass-produce this type messaging seems to be compared to the effectiveness of the output, it stands to reason that LLMs could enable a previously unseen proliferation of deceitful content.

If reiteration focused on generating tweet-length documents, *narrative elaboration* expands this to medium to long length content, such as articles and blog posts. For this task, GPT-3 was tested for its ability to sound reputable in its output, i.e. in its ability to replicate the tone of a renowned news source – unfortunately the believability of the generated articles was not verified in this setting. By the same token, this task certainly requires more human supervision, whose effects and variability are hard to design an experiment around, than

simple reiteration, since getting the prompt right will influence the tone and message of the generation immensely (for example, the headline or sample text to emulate must provide enough information to the model to recognize the style and worldview). Nevertheless, GPT-3 was able to fool classifiers only partially: the evaluation model, which classified genuine articles with over 90% accuracy over three news networks, had only around 65% accuracy. This indicates that GPT-3 is good at the elaboration task, but not perfect. It should however be said that fine-tuning the model could make it more potent in this regard – a fact that was observed with GPT-2 but could not be done on GPT-3 due to technical limitations.

Narrative seeding sets itself apart from the previous tasks, in that it requires that the model come up with original conspiratorial material. The objective in this case would be akin to the rise of QAnon between 2017 and 2020, a Twitter account that would post short messages containing conspiracy theories. While the effectiveness of most disinformation campaigns is, at least in part, a function of the scale of the operation, in this case a single human can be an effective catalyst, as long as the content is captivating. While GPT-3 appears to have the capability to come up with such messages – and its proneness to hallucinations might even help it do so – it is not obvious how one would go about measuring its potential in this endeavor.

Exploiting existing stories and divisions that exist in the target societies and groups is equally as important in disinformation campaigns as fabricating importance. *Narrative wedging* involves locating a source of conflict and widening it with disinformation. In the original experiment, messages were generated to influence US religious groups (Christian, Arabic, Jewish) to vote a certain way (Democrat, Republican, Abstain). The process by which the messages were obtained relied very heavily on human-machine cooperation, where a team of humans would identify the best arguments for a target group to vote a certain way through repeated prompts, and the best arguments were then used for message generation. The resulting 110 messages were reviewed by four experts, with 95% of messages found credible by at least one, and 65% by all four. Surprisingly, some of the messages were quite insidious, with the model soliciting Arabic voters (a traditionally left-leaning block) to vote Republican by calling for them to vote based on individual interests rather than group affiliation.

A similar task, *narrative manipulation*, involved identifying existing news stories and rewriting them to fit a particular narrative. Researchers used GPT-3 to rewrite a series of news articles concerning US political events around 2020, such the Black Lives Matter protests and the incipit of Covid-19. Again, this involved thick interplay between a human team and GPT-3, as the rewriting process was found to be most effective when it was executed in a series of steps (in short: summarize the article, alter the summary, expand again into longer text). Articles were modified to reflect both a left and a right-leaning position, and were then presented to human evaluators to test the effectiveness of the alteration, along with the corresponding authentic reference articles for comparison. This was proven to be mostly the case, i.e. GPT-3 was shown to be consistently able to not only give articles the correct spin, but to even output content that was found to be mostly authentic by the reviewers (the original articles received an average authenticity score of 3.8 on average, whereas generated articles only 2.4, but the best generations were believably authentic).

Finally, *narrative persuasion* tested GPT-3's performance in creating tailored messages for specific targets in an attempt to alter their convictions. Messages were generated for two topics (US involvement in Afghanistan and China), both for and against each issue, with the target's party affiliation in mind. Of 20 generations in each pairing, the 10 best were selected by a human team, and presented to over one thousand human respondents. The participants

generally found GPT-3's statements to be convincing, with 63 percent rating them as at least somewhat persuasive, even when the statements were targeted at an audience with opposing political views. While only about 12 percent found the statements "extremely convincing," the majority still found them somewhat persuasive. Additionally, the survey indicated that GPT-3's statements were effective in shifting respondents' opinions on various topics, with a notable and sometimes stark increase in support for the positions advocated by GPT-3 after being exposed to its arguments (in regards to action against China, GPT-3 managed to overturn a situation of absolute majority in favor of sanctions by reducing the pro-sanction faction from 51% to 33% of respondents).

In summary, GPT-3 performed particularly well with *narrative reiteration* (e.g. generating a series of tweets based on an original) and *narrative elaboration* (e.g. generating a blog post or article based on a title or paragraph) – which consist in the generation of short or medium length messages given a particular world view and a prompt. When malicious actors have a well-defined message that they seek to advance, NLG systems seem therefore to be a dangerous tool at their disposal, as they performs exceptionally well in this setting, without requiring particularly skilled operators. GPT-3 was observed to perform worse – or to require more careful prompting from the operator – for tasks requiring higher levels of creativity and originality. *Narrative wedging*, for example, aims to identify a source of division in the target society or group, and amplify its effects by means of carefully crafted disinformation messages. Similarly, *narrative manipulation* is a technique that involves framing existing emergent stories to align with a pre-defined world view. Neither task requires coming up with outright lies, and for both a human-machine team was observed to be necessary to obtain credible and convincing messages.

Buchanan et al. (2021) go on to highlight that GPT-3's writing characteristics – which can likely be extended to several other language models – suits the goal of disinforming more than that of informing. Indeed, GPT-3 is known to frequently go on tangents and make information up even in neutral to benign generation contexts. These traits might unfortunately make GPT-3 more effective in creating misleading or false narratives than in producing reliable, accurate content, underscoring its potential risks in the hands of those seeking to deceive.

Generic and task-specific language models can also be employed to exploit online review systems to influence consumer behavior. LLMs can be used to generate fake reviews with a target sentiment based on a prompt or a base review to emulate. This is an evolution of the *crowdturfing* method, which involves paying large numbers of people to write fake reviews. Due to the associated economic cost, these attacks were limited to large-scale attacks, and therefore carried a built-in limiting element.

Adelani et al. (2019) devise a strategy that employs language models to compose fake reviews based on an example displaying the desired sentiment towards the thing being reviewed. The generated fakes do not undergo manual review later, but employ a BERT-based sentiment classifier to filter out counterproductive items. To take it a step further, they use a fine-tuned version GPT-2 and BERT on Amazon and Yelp review databases, an operation that perhaps wouldn't necessarily be typical of attacks based on GPT-3 and other modern LLM variants. Nonetheless, using a task-specific version of GPT-2 might make it more comparable to task-agnostic, modern LLMs such as GPT-3 and GPT-4, making the findings more generalizable to modern systems.

To verify the fooling power of the fake reviews, sets of 3 fake reviews and 1 real review were presented to human evaluators, with the objective of correctly finding the "real" review. The results demonstrate that participants often struggled to discern the authentic review,

as their success rates were close to the random chance rate of 25%. This suggests that the evaluators were essentially guessing, indicating that the fake reviews were convincing enough to be mistaken for real ones. Such findings indicate that even in the premodern era of language modeling (i.e. before GPT-3), NLG already proved a valid substitute for the more expensive crowdturfing attacks.

3.3 Faking human authorship

Other malicious uses of LLMs rely on its ability to appear human, and not so much on its ability to specifically compose high-quality text. One such application is the generation of pseudo-scientific papers to contribute to one's research numbers. Examples of generation of such papers are common even before language modelling became popular. SCIgen (Hargrave, 2005) is a tool introduced in 2005 that allowed for the generation of nonsensical scientific papers by using a context-free grammar based generator. Such papers even pass the peer-review process from time to time and appear in respected publications. As recently as 2021, Cabanac and Labbé (2021) identified 197 SCIgen articles for which there had been no withdrawal notice, and were at times even being sold.

While SCIgen can be identified with relatively low computational effort and high accuracy, detecting content produced by more advanced generation tools, such as LLMs, presents a far greater challenge. With their ability to produce highly convincing and coherent text, these models could significantly worsen the issue, particularly in non-peer-reviewed technical documents. Beyond merely disrupting the scientific process, the proliferation of AI-generated technical content could also be exploited to influence public opinion and create confusion around important scientific issues, making it a growing threat in both academic and public discourse.

(Un)fortunately, there don't appear to be studies investigating the concrete prevalence of LLM-generated scientific papers across the recent academic landscape – perhaps both due to the recency of the phenomenon, as well as the difficulty in detecting the employment of modern NLG systems. Rodriguez et al. (2022) investigated possible detection strategies with self-made document tampering, but used the older GPT-2 for document contamination as opposed to more modern versions, therefore their results are hard to generalize (admittedly, this example investigate domain-specific fine-tuned generation, something that is harder and more expensive to do with future versions of GPT). Nevertheless, this study provides valuable insight into the future of LLM-generated papers. Further discussion of their detection strategies can be found in Chapter 4.

Among the overtly malicious uses of NLG, there are other that do not seek to harm directly. Students using language models to, either partially or completely, complete their coursework for them is one such case. Overreliance on systems like ChatGPT can discourage students from developing analytical and thinking skills. Passively incorporating the generated answers is a real danger when interfacing with such systems without an adequate evaluation framework – checking for factual correctness is one element among many that need checking before accepting a generation.

Another indirectly harmful application of NLG is content generation for social media. AI-generated text and images risk diluting the true human experience present on the platforms, to such an extent that they may become inhospitable places to organic users.

Furthermore, anyone using LLMs, however well-intentioned, is exposing themselves to the risk of plagiarism. The importance of checking the model output before incorporating it – especially in formal and academic writing – needs hardly be restated. Gao et al. (2022) investigated abstract generation for papers in the biomedical domain, and found no plagiarism

in the output, and the machine-generated nature of the resulting abstract was discovered to be predictable with high accuracy. This finding alleviates concerns somewhat, though the risk of inadvertent plagiarism is never fully zero, especially in longer contexts.

4 Previous approaches

If the above foray into the threat models concerning language generation that have been discussed in research has revealed anything, it is that expecting model operators to disclose the nature of texts submitted to their recipients is a naive and potentially dangerous habit. Thankfully, as language models have grown larger and more sophisticated, so have strategies and technologies detecting them matured alongside them. Though it has certainly received and influx of manpower interest since then, the field of automatic detection of machine generated text has been an active area of research even from before 2020 – after all, as was previously discussed, there was no shortage of attempts to exploit NLG technologies well before the recent GPT craze.

In Chapter 2, this thesis presented an overview of different ways in which text could be represented, on a spectrum ranging from surface-level metrics to neural contextual embeddings. Many parallels can be drawn between these strategies and the methodologies employed to detect machine generation. In the following sections, several well researched and empirically tested detection strategies will be explored, some taking advantage of linguistically motivated features or even frequency-based metrics, while others lean into the times and use LLMs themselves to discriminate between authentic and generated productions. It should however be underlined that the movement toward more sophisticated vectorization processes does not necessarily entail a one-sided improvement across all possible aspects involved in the detection pipeline.

On the contrary, moving up the ladder of complexity comes at the cost of requiring increasing amounts of computational power. Such approaches are extremely valid explorations of what is the best that can be achieved, but applicability in the real world is often limited since the end users cannot run the required software on commonplace commercial machines. Separating the application into client and server, which is the common approach taken in other AI fields, is only acceptable in the narrower subset of cases which have no privacy requirement regarding the data being verified. Chapter 5 will dive deeper into the more "cheap" approaches, meaning those that forgo heavy systems, aiming instead to strike a balance between performance and (compute) accessibility.

4.1 Frequency and feature-based methods

A simple way to think of documents is as so-called bags of words (BOW) (Murphy et al., 2006). BOW representation strategies only consider the words that appear in a text, without maintaining any information about the order. One application is the simple count vectorization strategy (Wendland et al., 2021), in which a text is represented by the counts of each word that occurs in it.

A closely related, but more refined and widely applied method is the extraction of TF-IDF matrices over sets of documents (Ramos et al., 2003). If one only looks at word counts, then prepositions and other common elements will likely account for much of the information retrieved from the text. TF-IDF offers a fix by implementing the intuition that a text containing the words "geothermal" and "power-plant", even if only with a few occurrences, is more informative about its nature than it having hundreds of examples "the" and "of". As such, term frequencies in TF-IDF are weighted by how rare they are (i.e. in how many documents of the sample they appear), with rarer words gaining a proportional score boost.

TF-IDF has a rich history of successful usage in almost all fields of NLP, including information retrieval (Ramos et al., 2003), sentiment analysis (Cahyani and Patasik, 2021), and text-classification (Zhang et al., 2011) among many others. Machine text detection has

also tapped into this tradition. Recently, Fröhling and Zubiaga (2021) employed higher-dimensional TF-IDF both as a classification accuracy baseline and as an input to meta-learners, i.e. ensemble models that take predictions from multiple classifiers to produce a final label. While falling short of state-of-the-art solutions that will be discussed later in this chapter, this methodical experiment still managed to build competitive systems by combining (computationally) simple approaches, in part by drawing upon the information captured by TF-IDF.

Alongside n-gram frequency features, like the ones discussed so far, another common approach to text vectorization is represented by linguistic feature sets. Instead of relying on simple token counts to derive a way to represent human productions with numbers, this strategy utilizes the quantifiable properties of the language being used. Some trivial examples may be the global length of a text, the average word length of the sentences contained within it, the type-token ratio, and so forth. Naturally, one can devise much more refined metrics to extract from texts, for example regarding text fluency, readability, grammatical properties, the richness of the vocabulary, cohesion, or even its purpose. The inquiry mentioned above, conducted by Fröhling and Zubiaga (2021), combined TF-IDF representations with feature-based classifiers across a variety of settings and datasets – in fact, these linguistic features drive most of the best models' performance.

The first targetable linguistic characteristic that language models have been observed to exhibit is a lack of cohesion (Holtzman et al., 2019). A study on GPT-2, for example, found that decoding strategies that maximize overall probability are likely to run into repetitive language (See et al., 2019), and even topic-drift, a phenomenon in which language models fail to stay within the confines of a particular topic (Badaskar et al., 2008). Traditional readability features, such as the Gunning-Fog Index and the Flesch-Kincaid Index, have been used successfully to identify generated text (Crothers et al., 2022a). Other methods involve modeling relationships between entities across documents through auxiliary models (Barzilay and Lapata, 2008), which seek to automatically determine the primary elements of the text, and track how they are presented throughout a text – with the idea being that human authors will refer back to the main points more often than machines.

Another useful set of features involve the use of varied vocabulary, and one that avoids repetition as much as possible. Texts authored by humans typically exhibit creative strategies to maintain narrative flow, such as building deep coreference chains instead of bogging down the text with frequent repetitions (Feng et al., 2010) – LMs tend instead to err toward toward the opposite side (Gehrmann et al., 2019). Generated text display frequent use of repetition, leading to reduced lexical diversity within the text (Zellers et al., 2020a). Lexical richness is, fortunately, one of the language aspects that can be more readily measured through features such as type-token ration, content or stop-word ratio, POS-tag distribution, and frequency of rare words, among many (Van Hout and Vermeer, 2007). See et al. (2019) identify a concrete trend in word-type usage in generated texts, where LMs favor verbs and pronouns more than humans, who make richer use of nouns and adjectives. Moreover, the reduced usage of varied ways to refer to entities can be measured by extracting the length of coreference chains (Feng et al., 2010), with the expectation that machines will produce shorter chains and more explicit repetitions.

The repetitive nature of generated text can also be attributed to the underlying sampling strategies. Ippolito et al. (2019) observe that a huge proportion of the probability mass is concentrated in the first few hundred, most common tokens in the case of top-k sampling. To make use of this property, another set of features measuring frequency of common texts can be employed, and it can lead to good detection performance when the model is sampled

with top-k (other sampling methods do not necessarily exhibit the same tendency).

Typically, generated text detection is a supervised task, which means that model training takes place with a labelled set of data. One interesting study explores the issue at hand with an unsupervised strategy: Gallé et al. (2021) use repeated high-order n-grams (re-occurring multi-word expressions) to distinguish machine text from human productions. They find that *supermaximal* substrings (i.e. the longest substrings that do not occur in any other substring of the collection) can be efficiently computed for a collection, and their presence in texts can be used as input for binary prediction. Specifically, they randomly select a subset of supermaximal repeats, and pre-label every document containing them as machine-generated. Then, these are used as positive examples to train a classifier, with an equal number of human texts as negative examples (they experiment with separate true human texts, as well as presumably human, picked at random from the same unannotated collection). This process is repeated a number of times with different selections of repeated substrings, and the results from each classifier is combined into a final ranking. The result is a list, where the top-scoring items are highly suspected to be machine-generated. Precision is then reported regarding the top m documents in the ranking, since the authors' intent is to design a system that pushes the final decision onto human judges, with the intent of only identifying suspicious documents. As summarized in Figure 1, this approach was tested against GPT-2 and it was able to reach an impressive 80% precision (in top-5000) even with nucleus sampling (Holtzman et al., 2019), which makes generation much harder to detect than top-k strategies (which this strategy detects at near-perfect precision) and is therefore an extremely interesting strategy when only few in-domain examples are available.

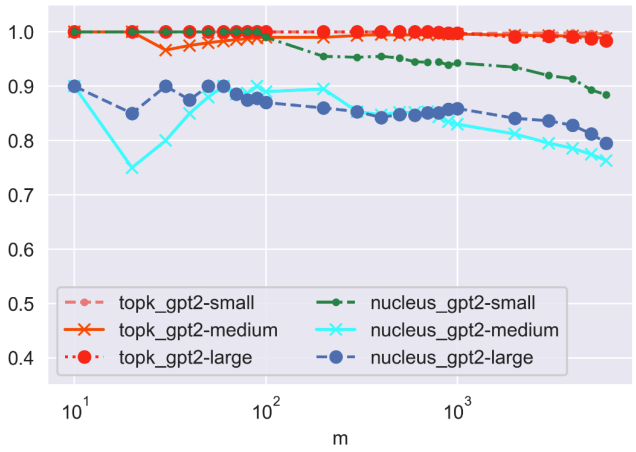

Figure 1 Precision reported by Gallé et al. (2021) for most suspicious documents, with 100 separate classifiers informing the decision. m indicates the precision at the first m elements of the ranking. It is noteworthy that precision for GPT2-Large with nucleus sampling stays relatively high even at m in the thousands.

4.2 Neural approaches

The strategies discussed so far have been traditionally paired been used with statistical models (chiefly logistic regression and support vector machines). However, the field of generated text detection has naturally also adopted the huge technological shift which of all NLP experienced, represented by neural networks and neural language models (Papers With Code,

2024). Since statistical features predictably underperform neural features (Crothers et al., 2022b), most detection approaches in this area forgo linguistic and frequency-based features altogether. However, recent results confirm their validity even when employed alongside LLMs: although they reach lower peaks than transformer-based counterparts, they are much more robust against adversarial attacks. Adversarial attacks describes malicious attempts of various types made by third parties to induce a misclassification in the model. Some examples may be illicitly accessing the training data of the detector model and insert or corrupt its records, or intercept the input at inference time and introduce changes in order to evade detection. In this context, an attacker might introduce small perturbations (like word or character edits) into the generated text to make the detector model fail to notice the generation. Statistical features were observed to have very high resistance to such attacks, and a hybrid model combining statistical features with Transformer-based attributes was reported to significantly enhance adversarial robustness, while maintaining solid performance in standard, non-adversarial scenarios (Crothers et al., 2022b).

Of course, no survey into the various methods to detect automatic generation can ignore the two classes of state-of-the-art approaches, both rooted in leveraging pretrained neural language models. On one side, there are so-called zero-shot approaches (Zellers et al., 2020b), which take advantage of the logits outputted by the models and attempt to make a prediction based on those, with various degrees of sophistication. On the other, approaches based on fine-tuning large bi-directional transformers (Solaiman et al., 2019) boast the contribution of the absolute best models to the field at present. The BERT variant RoBERTa (Liu et al., 2019b) has emerged as a winner in this space, both because due to its strong performance, as well as the relative ease with which fine-tuning approaches can be taken off the ground (Alec Radford and Jeff Wu and Jong Wook Kim, 2019).

A comprehensive research suite conducted by Zellers et al. (2020b) has evaluated GPT-2 and GROVER (a transformer LM geared towards article generation which they themselves introduced) in an effort to use them as discriminators for machine-generated text. Following (Radford, 2018), this study runs the input texts through the models with a special *[CLS]* token in the final position. The output embeddings corresponding to this token are then passed onto a linear layer performing binary classification. Evaluation then proceeds for different model sizes both for the generator (GROVER is also used to generate the articles to be predicted) and the discriminator. Perhaps unsurprisingly, the GROVER-based discriminators are best at detecting the GROVER-generated AI texts, and detection accuracy decreases as the generator is allowed more parameters (but classifiers benefit from positive examples generated by smaller models even against the largest model's outputs). Unfortunately, as is often the case in even very recent research, newer GPT variants are prohibitively expensive (Gallé et al., 2021) to employ on large benchmark datasets, which leads to few front-runner studies reporting results with them.

To date, DetectGPT (Mitchell et al., 2023b) is the best zero-shot approach to detect generated text, and the best detection tool for out-of-domain classification in general, at least in the white box setting (Gehrmann et al., 2019). This strategy leverages the observation that machine-generated text is much more susceptible to log probability perturbation (according to the source model) when edits are introduced to it, i.e. altering words or rewriting short passages of the text will cause the log probability to fall measurably. On the contrary, human-written texts do not exhibit the same property, and their log probability may change in a variety of ways in response to perturbation, but edits that preserve the semantic meaning of the text will tend to cause no perturbation at all. Armed with this knowledge, DetectGPT introduces minor, sense-preserving edits (such as swapping words for synonyms or rewriting

pieces with mask-filling) to a target text through an auxiliary model – in this study, T5 (Raffel et al., 2023) – end evaluates the difference in log probability between the original and altered versions. If the difference is above a certain threshold, the input is flagged as machine-generated.

DetectGPT outperforms even RoBERTa-based models when the target text is outside of the domain of the fine-tuned transformers' training sets, while still maintaining near-top performance when in-distributed texts are classifier. The simpler task definition of the white box setting, i.e. that the generator model is known accessible, is however a limitation of the experiment. Real-world scenarios can often include situations in which the generator model is unknown, therefore a proxy scorer needs to be used instead (Mireshghallah et al., 2023). Figure 2 offers an overview of DetectGPT performance across different scoring and generator models. Performance is predictably best when the scoring model is the same as the generator, but good-enough alternative models can be found – perhaps smaller, efficient models can be found for detecting the prominent generators, contributing to a less resource-intensive solution landscape. Ensemble solutions may even cover against many potential generators, instead of relying on a single scoring model in the black box setting (i.e. when the generator model is not known).

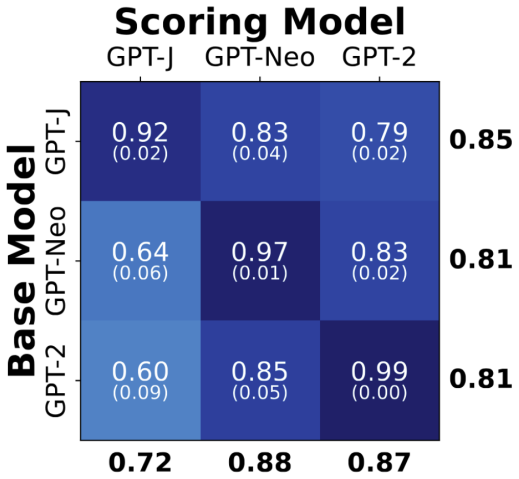

Figure 2 Performance overview of DetectGPT (Mitchell et al., 2023b), reporting AUROC (Bradley, 1997) over the XSum, SQuAD, and WritingPrompts datasets. Same scorer-generator pairs show the best performance, but results show that some models may act as better proxies than others, inviting further research.

The state-of-the art approach in machine-generated text detection has been, for a couple years at the time of writing, indisputably the fine-tuning approach (Solaiman et al., 2019). This involves fine-tuning a pre-trained bi-directional language model. The best resulting model is derived from providing task-specific training to RoBERTa (Liu et al., 2019a), a masked general-purpose language model based on BERT (Devlin et al., 2019) with a focus on classification. The primary weakness of RoBERTa is that it doesn't necessarily handle cross-domain examples out of the box, an aspect that the aforementioned DetectGPT (Mitchell et al., 2023a) improved on greatly, despite not outgunning RoBERTa large from in-distribution datasets. Rodriguez et al. (2022) also offer an improvement on this front by evaluating model performance on a new domain with a few hundred attacker examples selected by experts. While DetectGPT might be very useful in situations where the attacker

characteristics are entirely unknown, this latter approach might still offer more in the more standard case of a back-and-forth effort between attackers and defenders.

4.3 Domain differentiation

Aside from general solutions mentioned above to the problem of generated text detection, there is a rich inventory of domain-specific tools and strategies. For example, Rodriguez et al. (2022) experiment with detection tasks in the technical domain, specific targeting physics and chemistry. The study successfully applies a RoBERTa-based classifier fine-tuned on the biomedical domain on the physics domain, with relatively few new examples obtained from human experts. While only relating to GPT-2, the results reported are encouraging in the sense that knowledge may be more easily transferable across similar domains, even if they are technical in nature.

In the real of social media, Twitter has received consistently high research attention. Fagni et al. (2021) report on a number of different strategies that can help detect generated tweets, including some that are less commonly seen in research nowadays, such as TF-IDF and character-level models. Interestingly, character-encodings outperformed BERT-based approaches, and only lagged 0.05 percentage points behind RoBERTa, further underlining the validity of imperfect but computationally realistic solutions. Tourille et al. (2022) expand on this by introducing more training examples from GPT-2, which proved significantly harder to correctly classify in the original by Fagni et al. (2021), and show that prediction accuracy can be boosted in this way, though globally this is hard to notice, since RoBERTa already had greater than 90% accuracy. In general, it is hard not to raise the criticism anew that all of the most recent research, including the aforementioned twitter studies, do not move beyond GPT-2, so it remains a question how well they would perform if the attacker models were the much more sophisticated contemporary LLMs.

Another interesting domain of detection is chatbots and direct messages. Bot detection is already a well-researched field (Latah, 2020) where NLP can make a contribution when it comes to detecting AI agents faking human personhood. Bhatt and Rios (2021) expand the methodologies applied here by analyzing not only the messages suspected of having been generating, but also how the human interlocutors reacted to them. They verify the results with a number of different models across different representational strategies, such as BOW models, feature-based ones, and BERT-based classifiers. Not only does the inclusion of the human responses into the input improve cross-dataset generalization of the models, but non-BERT models fall short of simpler BOW models for some (substantially different) cross-dataset runs. Unfortunately, this experiments was again completed on the Conv2AI (Dinan et al., 2019) dataset, which does not make use of the most recent LLM technology in its generated examples.

Lastly, product reviews are also an area of interest when it comes to detecting automated text. Recently, Salminen et al. (2022) have used ULMFiT (Howard and Ruder, 2018) and GPT-2 to generate fake reviews based on the Amazon (Ni et al., 2019) dataset. They then use this newly generated dataset to fine-tune a RoBERTa-based classifier, which they compare against an SVM model and an OpenAI off-the-shelf fake review detector, also a RoBERTa classifier. Results show that the newly created model outperforms both baselines, reaching over 90% accuracy. These findings are validated against the Deceptive Reviews Dataset (Ott et al., 2011), where the newly introduced classifier again obtains the best result of the three models, with an AUC of 0.696, followed by OpenAI's off-the-shelf model with 0.595. Automated detection strategies seem especially necessary in this area, since the study also finds that human judges do not perform better than chance at identifying generations.

The adversarial nature of machine-generated text detection is especially at the forefront in this area, since it is easy to imagine a future in which attackers and detectors play a "cat-and-mouse" game, with attackers trying to evade detectors, and the latter developing new models and strategies in turn.

5 Task 8 at SemEval 2024

The 18th International Workshop on Semantic Evaluation (SemEval-2024) was a large event in NLP, offering various challenges that teams across the world could undertake. Task 8 at SemEval-2024 (Wang et al., 2024a) centered around machine-generated text detection in a black-box setting (i.e., the generator models for the test set were not known while the competition was ongoing), in both monolingual and multilingual components.

This shared task spanned 3 subtasks: subtask A a binary classification task between generated and human texts, subtask B consisted in multi-class classification between multiple LLM generators, as well human texts, while subtask C was a boundary detection problem, where participants had to correctly pinpoint the boundary between a human and a machine-generated segment in each test. During the active phase of SemEval-2024, which ended in February 2024, I undertook Task 8 jointly with Daniel Stuhlinger, a fellow student at the University of Tübingen. The full report of our participation, carried out under the team name "TueCICL", can be viewed in Stuhlinger and Winkler (2024).

As team TueCICL, we submitted results for two of the subtasks to the leaderboards, namely subtask A and subtask C, which were the two subtasks involving classification between solely human and machine-generated texts, whereas subtask B focused more on inter-generator differentiation. For subtask A, consisting in binary classification over the whole texts, there were two phases development, one for shared task proper, and subsequent post-deadline experimenting in the context of this Master Thesis. Due to the more pronounced research interest in subtask A, as well as to better present the higher volume of material associated with it, this section shall first discuss subtask C, then dive deeper into subtask A later in this Chapter.

5.1 Change point detection

Subtask C of the shared task addresses detection environments that are characterized by active adversarial agents performing the generation. Specifically, detection technologies have been observed to be weak to techniques also found in authorship obfuscation (Macko et al., 2024), such as paraphrasing (Krishna et al., 2024) and noise-introduction (Wang et al., 2021). Change point detection, which is the titular requirement of subtask C, addresses another way in which the use of NLG technology might be obfuscated. In the scenario targeted by the subtask, a human segment ranging from 0% to 50% of the text is concluded by a machine-generated component, making the text as a whole much harder to flag as a generation.

For all subtasks, the data is sourced from the M4 (Wang et al., 2024b) dataset. Task organizers provided generations by GPT variants and the LLaMA series (Touvron et al., 2023), but unfortunately not in high abundance: the training set contained around 5000 texts, and was accompanied by development set containing a little over 500 more. Table 1 offers an overview of the data distribution across the sets. Alongside the general scarcity of training and cross-validation data, one should also note that a relevant portion of the data is entirely machine-generated, meaning the change index is exactly 0. Though this of course preserves the objective of being able to reliably detect fully generated texts, it does cut into the already low amount of true change examples available for training.

The evaluation metric by which the submitted solutions are measured is Mean Absolute Error (MAE), i.e. the average absolute difference in the predicted change-point index and the true value. The baseline suggested by the organizers was based on Longformer (Beltagy et al., 2020), a fine-tuned RoBERTa checkpoint specialized in long texts. This baseline achieved an

impressive 3.5 MAE on the development set – a tough standard – and a (much lower) 21.54 MAE on the test set, though the latter was of course not known at development time.

Domain	Generator	Train	Dev	Test	Total
PeerRead	ChatGPT	**3,649 (232)**	**505 (23)**	**1,522 (89)**	5,676 (344)
	LLaMA-2-7B*	3,649 (5)	505 (0)	**1,035 (1)**	5,189 (6)
	LLaMA-2-7B	3,649 (227)	505 (24)	**1,522 (67)**	5,676 (318)
	LLaMA-2-13B	3,649 (192)	505 (24)	**1,522 (84)**	5,676 (300)
	LLaMA-2-70B	3,649 (240)	505 (21)	**1,522 (88)**	5,676 (349)
OUTFOX	GPT-4	–	–	**1,000 (10)**	1,000 (10)
	LLaMA2-7B	–	–	**1,000 (8)**	1,000 (8)
	LLaMA2-13B	–	–	**1,000 (5)**	1,000 (5)
	LLaMA2-70B	–	–	**1,000 (19)**	1,000 (19)
Total	all	18,245	2,525	11,123	31,893

Table 1 Dataset breakdown for subtask C from Task 8 at SemEval-2024. The number in "()" is the number of examples purely generated by LLMs, i.e., human and machine boundary index=0. LLaMA-2-7B* and LLaMA-2-7B used different prompts. Bold data is used in shared task training development, and test.

Following the example set by the baseline, the solutions we devised did not attempt to predict the boundary change index directly – instead, we performed binary classification at the token level, essentially asking whether any given token belongs to the human or the machine-generated segment. In accordance with the general mission of producing solutions that could run on at least midrange home computers and laptops, large-scale transformer-based approaches, such as the one provided as the baseline, were discarded altogether.

Instead, we presented an ensemble model sourcing its input from two bidirectional long short-term memory, or LSTM (Hochreiter and Schmidhuber, 1997), models. The first LSTM operated at the character level and did not include any elements of pertaining. Preprocessing the input texts for this character-level model involved lowercasing, and mapping numerals and punctuation to a `<NUM>` and a `<PUNCT>` special token respectively. Similarly, whitespace characters of any type (e.g. space, tab, newline) were also mapped to the special token `<WS>`. Classification then proceeds as outlined on the character level, and the first occurrence of the positive label (indicating that the token belongs to the machine segment) is taken as the boundary change index. Naturally, the character-level index is traced back to the word-level position, as required by the task setting, where the position of the word in which the first positively classified character is located is taken as the final label.

The second LSTM relied on pretrained Word2Vec (Mikolov et al., 2013) embeddings sourced from the Wiki2Vec project (Yamada et al., 2020). Static embeddings have fallen out of favor lately due to the emergence of contextual embeddings derived from LLMs, but they offer the significant advantage of not requiring firing up a large language model at inference time, a property that made them well-suited for our objectives in subtask C. While this LSTM performs word-level classification, as opposed to the character-level nature of the first model, many of the preprocessing steps are maintained. Casting texts to lowercase letters is all the more relevant since the Word2Vec mapping is case sensitive, and also maintained mapping numerals, punctuation and white-space to special tokens.

At least on paper, employing pretrained Word2Vec embeddings introduces an important amount of information into the model that would not be possible to obtain solely based on the available training data. However, for the character-level LSTM, there are no such mitigating factors. From very early on, it was clear from intermediate results that the character-level

LSTM struggled significantly, since it had no pretrained semantic context for the input letter sequences. To address this excessive burden placed on the scarce training data to both inform on how human language works, as well as how to distinguish it from generations, we enrich the dataset provided for subtask C with that of subtask A. While deep discussion of the latter is reserved for the later sections of this chapter, it should suffice to say that training data for subtask A contains over 100.000 records, providing ample linguistic material, albeit not targeted to the specific objective of subtask C.

Records imported from the training set for subtask A had the target label set to either 0 (for fully machine-generated texts), or the length of the tokenized text (for fully human texts). Since this operation carries the inevitable side effect of drowning out the far fewer examples where a change occurs mid-text (which are available in subtask C's data only), all models are trained for 5 epochs on the enriched data, and then exclusively on the original task data. Aside from the aforementioned optimizations, hyperparameter tuning over the characteristics of the models yielded that, for both LSTM's, a hidden size of 512 and 2 layers resulted in the highest validation performance.

After training the two base approaches, an ensemble model was also constructed combining the representations of the two LSTM's. This consisted in a non-recurrent feed-forward network (FFN) whose inputs were the concatenated representations at the word level. We applied this FFN head to the representations at each word-level token, outputting a binary label, the same way as the two LSTM's. For the character-level model, this meant averaging the representation at every character for any given word, before passing the representation to the ensemble classifier, along with the Word2Vec-based model's embeddings. Utilizing a simple FFN in this case was deemed to be the most efficient solution, since the interaction between the various elements of the sequence is already captured in the sub-models. This is also similar to how token-level classification is performed with transformers, with a feed-forward classification head applied at each token. The best-performing hidden size for this model was found to be 256. Table 2 offers a summary of the models developed for subtask C.

Model	**Type**	**Number of LSTM layers**	**Hidden size**
Character-level	Long short-term memory	2	512
Word2vec	Long short-term memory	2	512
Joint model	Feed-forward network	-	256

Table 2 Summary of models for subtask C. * Number of layers. ** Hidden size.

5.2 Generated text detection

Subtask A of Task 8 tackles machine-generated text detection in the classical sense, in the way that was discussed in Chapter 4. It offered both a monolingual and a multilingual track, both with their respective leaderboards, but our team decided to only submit for the monolingual track.

As hinted at in the previous section, the training set provided for this subtask contained over 100.000 records, with 5000 more provided for validation, sourced from the M4 (Wang et al., 2024b) dataset. Table 3 offers an overview of the data distribution across the three partitions. All sets offer a balanced mix of generated and human texts, with the training set containing outputs from 4 different models: Davinci-003 (a GPT-3 variant), ChatGPT, Cohere

[6] and Dolly-v2 (Conover et al., 2023). The development set adds BLOOMz (Muennighoff et al., 2023) to the collection, and the test set introduces GPT-4 on top. [7] The intent of the authors to enforce a black-box setting shines through in the way they built the partitions: the development set includes a model unknown to the train set, and the test set contains another model absent from the other splits.

Alongside the partitioned dataset, the task organizers also provided a RoBERTa-based transformer baseline. The evaluation metric for this subtask was accuracy. The transformer baseline achieves an accuracy of 72% on the development set and 88.47% on the test set. [8]

Split	Source	davinci-003	ChatGPT	Cohere	Dolly-v2	BLOOMz	GPT-4	Machine	Human
Train	Wikipedia	3,000	2,995	2,336	2,702	–	–	11,033	14,497
	Wikihow	3,000	3,000	3,000	3,000	–	–	12,000	15,499
	Reddit	3,000	3,000	3,000	3,000	–	–	12,000	15,500
	arXiv	2,999	3,000	3,000	3,000	–	–	11,999	15,498
	PeerRead	2,344	2,344	2,342	2,344	–	–	9,374	2,357
Dev	Wikipedia	–	–	–	–	500	–	500	500
	Wikihow	–	–	–	–	500	–	500	500
	Reddit	–	–	–	–	500	–	500	500
	arXiv	–	–	–	–	500	–	500	500
	PeerRead	–	–	–	–	500	–	500	500
Test	Outfox	3,000	3,000	3,000	3,000	3,000	3,000	18,000	16,272

Table 3 Subtasks A: Monolingual Binary Classification. Data statistics over Train/Dev/Test splits.

It should be highlighted the the composition of the training set proved a unique, and at times unhelpful challenge in tackling the shared task. Evaluating potential solutions based on their ability to detect solely BLOOMz generations could, in theory, be interpreted as an ability to generalize the behavior of some known LMs to unknown ones. However, there is also the possibility that selecting based on this development set achieves a more superficial objective: detecting BLOOMz generations in particular. This moving objective in the setup of the development set has plagued much of the development for this subtask, and will remain somewhat of a theme all the way through the final, post-deadline experiments.

With that premise, the proceedings at the time of the shared task evolved along similar lines as those described above for subtask C. The overarching goal remained to develop a workable solution at a low computational cost. For this purpose, an ensemble approach was devised for subtask A as well, with some modifications compared to the approach to subtask C. Our intuition was that surface-level and stylistic features would be more effective than semantics in discriminating between human and machine-generated text. To build on this idea, we developed three approaches, which were joined at the end in the ensemble.

The first approach involved training a character-level LSTM. We expected the stylistic features of the texts to be good indicators of the generator, and working at the character level is known to capture this information well. For example, character n-gram models have been used successfully in the field of authorship attribution, which relies heavily on style (Stamatatos et al., 2013).

[6] `https://cohere.com/`

[7] The make-up of the test set was not known during the shared task, it was only made available after the submissions deadline.

[8] Baseline accuracy on the test set was only made public after the submissions deadline.

Following generally the same steps as for the character model in subtask C, the input texts were first tokenized, then all tokens were mapped to their lowercase variants, and lastly numerals and punctuation were mapped to a `<NUM>` and a `<PUNCT>` special token respectively. White-space elements (space, tab, newline) were also mapped to a special token `<WS>`. At this point, the tokenized and transformed inputs were fed through an LSTM, and the representation of the last token was used for prediction. Notably different from subtask C was the model architecture, which was uni-directional in this case. During development, this model was trained on both the classification objective, as well as a language modeling objective, for similar reasons as the dataset enrichment in subtask C. Since this character-level approach did not benefit from any external data that would have latently contributed by means of pre-training, the language modeling objective was added to extract as much linguistic information as possible from the available dataset.

The second approach was constructed along the lines of the first in terms of technical setup, but dealt with words rather than characters. Large transformer-based solutions benefit from vast amounts of pre-training, but at a heavy computational cost – using pretrained embeddings as model inputs appeared to be a good compromise between heavy models and training from scratch, as had been the case with the character-level LSTM. We used the pretrained Word2vec embeddings from the Wikipedia2Vec (Yamada et al., 2020) project to map texts to vectors, but maintained the other steps, such as mapping numerals, punctuation and white-space to special tokens. As opposed to the Word2Vec model in subtask C, this variant also uses a uni-directional LSTM, though no language modeling task is performed in this case.

The third approach was not recurrent in its nature – instead, we used the TextDescriptives pipeline (Hansen et al., 2023) through spaCy (Honnibal et al., 2020) to obtain 66 linguistically motivated features to globally represent the text. Such linguistic features have a long tradition in NLP, for example in the field of readability analysis (for example Vajjala and Meurers, 2012), and have been observed to be valid and cheap-to-compute representations in a variety of settings. Since they are most well known for capturing the style of a text, rather than semantics, they appear to be very well suited for the present task. Additionally, we computed the mean perplexity of the document using GPT-2 (Radford et al., 2019a) and added it to the feature vectors. This follows the idea that the perplexity of a document assigned by a LLM should be higher for human written texts than for machine generated texts (Chaka, 2023). Our third approach computed this global feature vector for the input text, then generated a prediction through a simple MLP. The model consisted of 3 linear layers with Tanh activation functions in between and was trained for 2000 epochs with a learning rate of 0.0003.

An ensemble model was finally formulated, consisting in a feed-forward network which takes as input the final representations (the last hidden states in the case of the LSTM's) of each of the three previous approaches, generating a single prediction. In addition to the standard model weights and biases related to the linear layers, a trainable scalar weight was also specified for each of the three inputs, to allow for dynamic scaling of the three input vectors, which would have been otherwise hard to interpret in tandem. Table 4 offers an overview of the models developed by team TueCICL during the shared task proper.

Results for this and all other experiments will be discussed in detail in Chapter 6, but it makes sense to give a few notes before moving on to the post-deadline contributions. The shared task only allowed one submission per subtask, and out best-performing solution on the development set was the ensemble model, with an accuracy of 85%, beating the baseline (at 72%) by a substantial margin. Despite this, the ensemble model only achieved an accuracy of

Model	Type	Number of LSTM layers	Hidden size
Character-level	Long short-term memory	2	512
Word2vec	Long short-term memory	2	512
Language features	Long short-term memory	3	256
Joint model	Feed-forward network	-	512

Table 4 Summary of models for subtask A.

54.57% accuracy on the test set, far behind the baseline's 88.47%. This observation remains true even at the level of the three sub-models, neither of which performs convincingly on the test set. Surprisingly, evaluations made after the deadline, once the test set labels were published, revealed that our best submission came from a model that was never intended to be the front-runner in our research, namely a TF-IDF logistic regression approach that only reached 60% accuracy on the development set, but achieved an impressive 87% accuracy on the test set, and was thus our only model that managed to beat the baseline.

After looking at our various models' performance on the test set, it strongly appeared that solid performance on the development set did not translate into a good grasp of the test set. Instead, our solutions had become presumably more optimized at detecting BLOOMz generations, as opposed to machine-generated text in general. The best performing model on the test set ended up being one that did not show almost any promise in development.

In summary, two important lessons were drawn from the shared task results that informed the subsequent work on this subtask, which will be presented in detail in the next subsection. First, the models did not achieve a high enough degree of sophistication, and failed to capture the characteristics of generated texts – new solutions needed to be more firmly grounded in the current research around the best and most efficient detection systems. Second, good performance on the provided development set did not indicate a good system overall. Finding a more suitable evaluation method for the models in subsequent training iterations was essential to improve performance, otherwise the cross-validated definition of good detector and a good *BLOOMz* detector would remain indistinguishable.

5.3 Generated text detection: post-deadline additions

Ensemble solutions to the problem of machine-generated text detection are far from rare in the scientific literature on the subject. For example, Fröhling and Zubiaga (2021) propose ensemble approaches over linguistic features, TF-IDF models, and neural classifiers. This could be seen as a more refined implementation of a similar intuition to what was presented in the previous section for subtask A, with two notable departures from that workflow: pretrained static embeddings like Word2Vec are not used, and the inputs to the ensemble are not long vector representations, but individual model probabilities associated with the positive label. Results obtained by Fröhling and Zubiaga (2021) in the multi-generator setting (the same situation as subtask A, where the data splits contain generations from multiple models) suggested that their detector might be developing specialized sub-detectors for each individual data source, instead of forming a generalized comprehension of the distinction between human-written text and machine outputs. This hypothesis is echoed by other studies as well, which observe a generally low transferability of a detector's ability to detect *some* model's to generations to detecting generated text in general. For example, Mitchell et al. (2023a) also observe a steep drop in detection performance in the black-box setting compared

to the white-box setting, suggesting that ensemble solutions targeting different but *particular* models might constitute a promising way forward in the field.

The work put forward in the following pages of this Master Thesis is developed along the guidelines mentioned in these recent research approaches. These additions are made in the context of this thesis, and are distinct from the shared task at SemEval-2024 and do not steam from a team effort like TueCICL's submissions in the main event, discussed in the previous section.

A first direction for the post-deadline work was provided by the realization that the models developed for the shared task did not fully utilize the information provided in training and validation. In fact, aside from the text, the domain (Wikipedia, Wikihow, et cetera) and the label (machine or human) of every record, the exact generator model was also known for machine-generated text. Jointly with the evidence that detectors are much better suited to detect *particular* models, as opposed to generated text in general, it was determined that the contributors to the final ensemble model in this second stage would be specialized models, trained to detect only one particular generator.

Of the original strategies devised for the shared task, the Word2Vec and the character-level approaches were discarded, since they underperformed their counterparts in the shared task submissions, which is a confirmation of their relatively lower representation in the literature. Despite half of the previous approaches being discarded in this stage, it still makes sense to continue working with linguistic features. They appear to give valuable contributions to detectors in the relevant research (Fröhling and Zubiaga, 2021), and they were perhaps not sufficiently explored for the shared task submissions. With linguistic features, it is possible to hand-pick a limited selection to ensure explainability of the results, but for pure performance it is generally preferred to aim for a higher quantity of metrics. For the shared task, we computed the feature vectors through the TextDescriptives pipeline (Hansen et al., 2023) for spaCy, which yield about 60 distinct measures per text. In the second stage, we swapped the extraction library to LFTK (Lee and Lee, 2023), a more recent and mature feature collection and framework, and also the largest off-the-shelf solution at the time of writing.

In addition, since the best-performing model for the original task unexpectedly ended up being TF-IDF, it seems fair to conclude that the information captured by these document matrices can be fruitfully applied to the tasked at hand. Fröhling and Zubiaga (2021) also employ TF-IDF to successfully boost detection performance.

Lastly, large-scale transformer-based solutions such as RoBERTa are still off the table, but excluding all transformers altogether was perhaps overly hasty. DistilBERT (Sanh et al., 2020) is a BERT version that retains much of its larger siblings' language capability at a lower computational cost. Since inference with DistilBERT is still manageable with the target end-user systems, it is included as an ensemble component in this work, with the aim to obtain some of the predictive power of state-of-the-art transformer approaches.

This work breaks down the problem of black-box machine-generated text detection into a set of smaller problems, consisting in detecting the output of one generator at a time. A number of classifiers are trained for each model contained in the train set: Davinci, ChatGPT, Dolly, and Cohere. Each individual classifier is trained with one of three general strategies and is only optimized to detect one model. To summarize the model strategies outline above, a detector model is trained for each generator model, sourcing its inputs either from linguistic features computed through LFTK, TF-IDF vectors, or DistilBERT.

To achieve the training objective for each sub-classifier, a secondary development is derived from the training set, containing 3000 records for each generator, and an extra 12.000 human texts, which yields balanced classes. This is done both to set up a validation set that allows

the best models to be selected only based on their ability to detect their own generations, as well as to maintain consistent train and development data for the sub-classifiers, in order to maintain a validation data split whose is unknown to all classifiers, which would not have been the case had a custom validation set been sampled individually for each classifier. Random state for every step is fixed at the same arbitrary constant value, and there was no supervised search performed for any step described below.

To train the TF-IDF and language feature-based classifiers, the process runs in a similar fashion. First, all documents produced by the target generator are extracted from the train set, yielding around 10.000 documents. The same amount of human records is sampled from the train set. This means that in the training process for each classifier, human texts are significantly undersampled compared to all of the available data. This is done to ensure balance between classes, and since the sampling is random for each classifier, it is statistically likely that all of the human texts will have been used by the time all classifiers are trained. For TF-IDF, the documents are vectorized with lowercasing, stopword removal, and the top 100.000 most frequent terms are kept in the unigram-bigram range. For the language features, the pre-computed feature vectors are taken from a database for time efficiency, but computing them at inference time in downstream applications would not be a technical hurdle. A Random Forest Classifier is then fit to the input vectors, with standard hyperparameter tuning through grid search to evaluate a variety of configuration constellations. Cross-validation for the classifier is not carried out through ten-fold cross-validation, instead the custom validation set (3000 human texts and 3000 documents generated by the target classifier) is supplied during training to obtain the best model.

For the DistilBERT fine-tuned classifier, the data partitions used in training are the same as for the statistical models. The difference is of course in the model architecture, as well as the preprocessing, which is done by the associated DistilBERT tokenizer. The training proceeds for 5 epochs, with the best model selected based on accuracy on the custom validation set.

Each individual classifier is then evaluated on all three non-train data partitions: the custom development set containing 24.000 records originally in the train set, the *true* validation set with 5000 (2500 BLOOMz + 2500 human) generations, and the test set. Table 5 reports the accuracy achieved by the individual classifiers against the data partitions available at this stage. Deeper discussion about the results can be found in Chapter 6, but the results are included here for an intermediate sanity check.

Model	Dev Set			True Dev Set			Test Set		
	TF-IDF	Lang	BERT	TF-IDF	Lang	BERT	TF-IDF	Lang	BERT
ChatGPT	0.74	0.86	0.88	0.58	0.50	0.60	0.86	0.82	0.83
Davinci	0.79	0.95	0.88	0.65	0.59	0.63	0.89	0.76	0.81
Cohere	0.71	0.66	0.87	0.66	0.57	0.60	0.55	0.56	0.57
Dolly	0.72	0.85	0.89	0.64	0.54	0.69	0.76	0.42	0.64

Table 5 Classification accuracy by the individual classifiers on the various data partitions. The "Dev Set" is a split-off segment of the train set, the "True Dev Set" is the initial development set provided by the task organizers. "Lang" refers to the classifiers trained on linguistic features.

There are a few interesting observations that can be made in relationship with the classifiers' performance. One aspect that was hard to miss among the test set results is that the TF-IDF classifier trained to detect the Davinci generator is the best solution ever to

emerge for the shared task, beating all potential submissions – though this of course it is only possible to say this after the fact, with access to the test set labels. Without these, there is nothing in the runs on the validation set that would make this solution stand out. Accuracy on the BLOOMz development set remained consistently low for all classifiers, but the worst classification performance among all was the features-based classifier trained for Dolly on the test set, which dipped below random chance. Overall, the consistently higher results on the test set compared to the true development set reinforce the idea that trying to optimize for the original development set might be a fool's errand. The results in this case clearly reflect the much higher similarity of the train set to the test set. Generally, the feature and TF-IDF based models did not underperform DistilBERT on either set, which might have perhaps been the expectation, and TF-IDF even outperforms DistilBERT by a significant margin for the Dolly detectors, and generally do better for three out of four models on the test set. This inspires some initial confidence in the new approach.

After obtaining the model classifiers, an ensemble model is trained sourcing the inputs from them. The data used for training the ensemble is the custom development set mentioned earlier, containing 3000 examples from each generator and 12.000 human texts. This is setup helps ensure that individual generator detectors haven't already seen the training example. In truth, training the ensemble on the full train set or on this split-off segment wasn't observed to result in a different outcome in terms of performance on the original validation set, which was used to select the best model in this stage. Following Fröhling and Zubiaga (2021), a probability is inferred with each classifier for each target document, resulting in a 12-length input vector, with each value referring to the probability associated to the positive label according to one classifier. Two model architectures are explored for the ensemble: a random forest classifier, trained with grid search in a similar fashion as the TF-IDF and language feature models, and a neural feed-forward network. Table 6 presents the resulting ensemble classifiers.

Model	Description	Dev accuracy	Test accuracy
Statistical	Random Forest with Grid Search	0.89	0.82
FFN	Neural feed-forward network, with 3 linear layers and a hidden size of 64	0.65	0.79

Table 6 Accuracy of the ensemble models on the task development set (BLOOMz generator only) and the test set.

The final ensemble classifiers do not represent a bad result in and of themselves – in fact, submitting either for the shared task would be constitute a 30% improvement in accuracy on the official submissions, and would have placed better in the final leaderboard. However, there is also reason to be disappointed. Firstly, neither ensemble performs better than the best individual classifier on the test set, where the Davinci-optimized TF-IDF classifier achieved 89%. In addition, this second stage of experimentation also failed to produce a detector better or equalling the 90% mark in classification performance. There is nothing inherently special about this threshold, other than that it is about the minimum value that beats the baseline, but literature analysis and earlier results would have suggested that reaching is not only possible, but, at least in theory, likely. Another important observation is that the best result ever observed on the development set, 89% by the random forest ensemble, did not translate to a convincing result on the test set, again indicating that the development set acts more as an obstacle to overcome, rather than a true indication of performance, at least

in the context of the shared task.

As a final round of experiments to check whether swapping the development set would indeed improve performance on the shared task, all steps described above are done anew, with a few modifications to the data partitions. The original train and development sets, totalling a little more than 122000 documents combined, are merged, and a new development set is sampled from this combined partition. After this process, the new development set contains 13000 human texts, 3000 text from each of Cohere, ChatGPT, Davinci, Dolly, and finally 1000 BLOOMz generations. The new train set contains all remaining documents, i.e. 52851 human texts, 11343 from Davinci, 11339 from ChatGPT, 11046 from Dolly, 10.678 from Cohere, and finally 1500 from BLOOMz.

All single-generator classifiers were re-trained on these new data partitions. For most classifiers, this did not introduce any change, since they were still only trained and evaluated on a single generator, therefore, for example, the ChatGPT classifiers were still only seeing human and ChatGPT texts. A complete novelty compared to the earlier models are the three BLOOMz-based classifiers. These models don't have nearly as much to train and evaluate on as their siblings, but their contribute may still reveal itself to be valuable. It's worth mentioning that, just like in the earlier iteration of this experiment, the single-generator models' ability to detect their target model's generation is near-perfect across the board.

After obtaining the single-generator detectors, a set of new ensemble models are trained. The ensemble inputs remain the probability of the positive label according to each single-generator classifier, since this was unlikely to be limiting factor in the earlier run. This second run of experiments yielded much more convincing results, which will be explored in-depth in Chapter 6.

6 Discussion of SemEval results

6.1 Subtask A: Shared Task analysis rankings

Model	Development set	Test set	Ranking
Baseline	0.72	0.88	20
Character-level	0.85	0.55	127
Word2vec*	0.82	0.72	85
Language features*	0.63	0.88	21
Joint model*	0.83	0.69	96

Table 7 Results for SemEval-2024 Task 8, subtask A. Dev and Test columns report the accuracy on the respective data partitions. The ranking column refers to the model ranking in the shared task competition. The scores and ranking of the unofficial submissions were not provided by the organizers and computed by team TueCICL. There was a total of 137 submissions.
* unofficial submissions

Table 7 shows the results for each model submitted during the proceedings of the shared task, for subtask A. On the development set, almost all models outperform the transformer baseline provided by the organizers. The best performing model was the character-level model, with an accuracy of 0.85 – this was our final submission for the shared task. While the two recurrent models and the joint model do not differ very much from one another, the FFN built on linguistically motivated global feature vectors sets itself apart in that it is the worst performing model on the development set.

Perhaps the most important lesson here is that there's clearly diminishing returns in closely fitting the development set – perhaps even negative returns, as the worst-performing model in development is the best-performing one on the test set by a big margin. At any rate, it wouldn't be honest to put the blame of this middling performance on the development set alone. Selecting character-level and Word2Vec-based solutions appears after the fact to have been a misfire as well. For further information, the original task report (Stuhlinger and Winkler, 2024) contains an in-depth rundown of the shared task, while the following pages are dedicated to how these lessons can help improve performance on the task, with little adjustments to the methods, that nonetheless preserve the mission of computationally efficient, locally runnable detectors.

6.2 Subtask A: post-deadline improvements

Having taken note of the disappointing performance of the official submissions to the shared task, a new batch of experiments was conducted in the context of this Master Thesis. These had the objective of being more firmly grounded in research, while still maintaining the objective of developing detectors which end-users would be able to own and run on their own machines. A new approach was developed, targeting generator models one at a time with different strategies, since machine-generated detection seems to be an elusive target when targeting all generators at once. Instead, the work was split among many different classifiers, attempting to differentiate between human texts and only one generator (Davinci, ChatGPT, Cohere, Dolly, BLOOMz).

The first batch of experiments resulted in the single-generator classifiers described in Table 5. Among these models was the one showing the best ever performance on the test set, with an 89% classification accuracy by the TF-IDF model on Davinci, though this model would have been impossible to find without access to the test set labels, since its performance on the development does not stand out. After running the full experiment, which ended in the development of the ensemble model, it was also observed that the approach resulted in good, but not excellent classification performance. Table 6 describes the two ensemble models, one neural and one trained with a random forest base. Unfortunately, neither ensemble beats the baseline, and both fall short of the best single-generator classifier on the test set. They also both underperform the best unofficial submission to the shared task, which was developed by team TueCICL, achieving 88% test set accuracy with only linguistically motivated features.

Chapter 5 concluded with the description of the last experimental run, which followed the general lines of the first, with a re-sampled development set. The single-generator model lineup was also extended to include the three BLOOMz classifiers, one with a fine-tuned DistilBERT model, and two statistical approaches with TF-IDF and language feature representations respectively. The remainder of this section is dedicated to the analysis of this last experimental run, which resulted in a final ensemble with a classification accuracy of 97% on the test set.

The first task in the newly formulated approach to the task of machine-generated text detection is obtaining a set of single-generator classifiers. As a reminder, a total of 15 such classifiers, one for each model and strategy combination, are trained on a subset of the train data, containing only human texts and generations from the target model. The validation set for these models is constructed along the same lines, containing no generations from other models, to ensure each single-generator classifier is selected based solely on its ability to detect its target. Probability scores associated to the positive label are then used as input to the final ensemble classifier.

The objective for single-generator classifiers is to be as specialized as possible in detecting generations from a single model. In other words, they should be geared towards high precision, rather than high recall. Table 8 provides a summary of the single-generator classifiers, with precision and recall for the positive label (i.e. for the machine-generated class) over the re-sampled development set, and accuracy on the test set. There are several takeaways from this table that are worth mentioning. Initially, it can be noted that precision is high for nearly all classifiers, which is in concordance with the models' training goals. For some single-generator classifiers, even the recall value for the positive label is respectable, meaning that the models display good ability to detect generations from other models as well. Another interesting aspect is that for all generators, at least one strategy displays high precision, even when other strategies struggle. For example, BLOOMz-targeted models struggle when using TF-IDF and language features, but are rescued by their DistilBERT sibling. Similarly, detectors for Dolly in the TF-IDF and DistilBERT strategies do not inspire high confidence, but the features-based classifier appears to have specialized much more than the others, and could come to the rescue for difficult scenarios despite its low test-set performance. Feature-based detectors generally perform above expectations, with all except BLOOMz developing a highly specialized toolkit, resulting in especially high precision. In this sense, they are perhaps the strategy that best captures the objective of the single-generator models: highly specialized systems, that can detect one specific model with very high precision. As will be seen later, this is likely why they are a crucial driver for performance in the ensemble classifiers.

When it comes to performance on the test set, the results of course do not differ from

Model	Strategy	Development set		Test set
		Preicision	Recall	Accuracy
ChatGPT	DistilBERT	1.00	0.66	0.83
ChatGPT	TF-IDF	0.97	0.55	0.88
ChatGPT	Language Feats	0.98	0.69	0.83
Davinci	DistilBERT	0.95	0.82	0.77
Davinci	TF-IDF	0.94	0.68	0.89
Davinci	Language Feats	0.97	0.87	0.97
Cohere	DistilBERT	0.96	0.82	0.69
Cohere	TF-IDF	0.94	0.66	0.70
Cohere	Language Feats	0.96	0.65	0.45
Dolly	DistilBERT	0.87	0.92	0.62
Dolly	TF-IDF	0.89	0.81	0.87
Dolly	Language Feats	0.99	0.30	0.56
BLOOMz	DistilBERT	0.92	0.10	0.55
BLOOMz	TF-IDF	0.80	0.32	0.54
BLOOMz	Language Feats	0.70	0.57	0.56

Table 8 Performance metrics for final single-generator classifiers. The development set referenced here is the one derived from the train set and described in Chapter 5, not the original development set. Precision and recall refer to the **positive** label, not to the average of the metric over the two classes.

those already reported in Table 5, since these are the same models, evaluated on the same data. For the 12 single-generator classifiers that were already discussed, only the precision and recall metrics were new introductions to the conversation. In that sense, the metrics paint a positive picture, with a good degree of specialization among models, with high precision being an indicator of success even in the face of low test-set accuracy. The BLOOMz-targeted models were trained with only 1500 positive examples, with a further 1000 reserved for validation. Predictably, this led to the non-pretrained models being at a disadvantage, since TF-IDF and feature-based models need more data than fine-tuning to achieve a good fit. Indeed, the DistilBERT-based detector is the best-performing classifier in terms of precision, with the two other models lagging behind. The scarcity of data appears to have hurt the features-based classifier the most, even though this strategy is very high-performing when detecting other generators. Not only is the fine-tuned classifier the most precise, but it also appears to be the most highly specialized. It achieved a recall of only 10%, which one might mistakenly consider to be a negative measurement. BLOOMz texts were only responsible for 3% of generations in the re-sampled development set, meaning that the low recall measured for this model is a positive sign of highly optimized detection behavior, which again is the goal of the single-generator classifiers. The TF-IDF and feature-based classifiers for BLOOMz do not display similar behavior, as their lower precision and higher recall indicate that they take more shots in the dark, which constitutes and undesirable, if perhaps inevitable outcome. If any conclusion is to be drawn from the results in Table 8, it would be that the process has seemingly produced at least one highly specialized detector for all generators, with some less precise but hopefully good auxiliary models. It remains to be seen whether it is possible to obtain an ensemble that properly leverages the properties of the single-generator models.

Aside from verifying the models' performance on their own targets, it should also help to check in more detail how well these classifiers generalize to unknown generators. Table 9 presents generalization metrics for the same models as above. Each value in the table represents the recall score, i.e., the proportion of documents from a particular generator that a model correctly flagged as machine-generated, with detector-generator model concordance being bolded.

Model	Strategy	ChatGPT	Davinci	Cohere	Dolly	BLOOMz
ChatGPT	DistilBERT	**1.00**	0.74	0.68	0.39	0.15
ChatGPT	TF-IDF	**1.00**	0.64	0.43	0.25	0.13
ChatGPT	Language Features	**1.00**	0.60	0.50	0.86	0.03
Davinci	DistilBERT	0.99	**0.99**	0.78	0.63	0.47
Davinci	TF-IDF	0.85	**1.00**	0.60	0.40	0.27
Davinci	Language Features	0.94	**1.00**	0.87	0.90	0.23
Cohere	DistilBERT	0.96	0.80	**1.00**	0.68	0.27
Cohere	TF-IDF	0.66	0.59	**1.00**	0.51	0.35
Cohere	Language Features	0.71	0.69	**1.00**	0.35	0.16
Dolly	DistilBERT	0.98	0.86	0.97	**0.99**	0.53
Dolly	TF-IDF	0.78	0.73	0.83	**1.00**	0.57
Dolly	Language Features	0.23	0.04	0.00	**1.00**	0.13
BLOOMz	DistilBERT	0.01	0.03	0.03	0.02	**0.99**
BLOOMz	TF-IDF	0.18	0.25	0.36	0.24	**1.00**
BLOOMz	Language Features	0.00	0.16	0.08	0.10	**1.00**
ChatGPT	Statistical Tandem	**1.00**	0.77	0.75	0.88	0.16
Davinci	Statistical Tandem	0.98	**1.00**	0.94	0.92	0.39
Cohere	Statistical Tandem	0.86	0.79	**1.00**	0.65	0.39
Dolly	Statistical Tandem	0.79	0.74	0.83	**1.00**	0.65
BLOOMz	Statistical Tandem	0.18	0.34	0.39	0.30	**1.00**

Table 9 Generalization metrics for each single-generator classifier. The reported value is equal to the ratio of correctly flagged documents for a generator, over all documents produced by that generator. In other words, a measure of recall is shown for each detector-generator combination. Bolded values refer to instances where the detector is trained on the productions of the generator. Statistical tandem indicates a combination of the Features and TF-IDF models, where a correct classification by either is counted as a success.

Unsurprisingly, self-detection performance (bolded values) is the best in every case, showing that models are most effective at detecting text generated by the model they were trained on. On the contrary, cross-model generalization is inconsistent, but there is much more differentiation between models in this case. For example, ChatGPT detectors generalize moderately well to other models, especially when using the fine-tuning strategy (e.g., 0.68 on Cohere and 0.74 on Davinci). On the other end of the spectrum, Dolly and BLOOMz detectors perform significantly worse when detecting texts from other generators. For example, the BLOOMz detector using DistilBERT has almost negligible recall on ChatGPT (0.01) and Cohere (0.03), while Dolly-trained models using language features have very low recall scores on all other generators (e.g., 0.23 for ChatGPT). In general, the BLOOMz targeted models barely generalize at all, but this is understandable given the low amount of training data.

But more importantly, other models also generalize very poorly to BLOOMz generations, with the best recall being 0.53 (excluding, for now, the tandem models), achieved by Dolly on TF-IDF. Considering that the test set is made up by around 9% of BLOOMz models, and that the first attempt at the ensemble strategy did not have BLOOMz-optimized detectors, it is easy to see how that segment of the test set could have ended up mostly misclassified in the first iteration. Even though the BLOOMz detectors generalize poorly, they still clearly capture something about the generations by their target model (and, hopefully, the class of generators to which BLOOMz belongs in general) that all other classifiers fail to.

DistilBERT models distinguish themselves as the best-generalizing models across the board, especially those targeting ChatGPT and Davinci. One possible explanation might be that these OpenAI models are either the largest or the most general-purpose models, thus their properties extend father away, whereas other models like Dolly describe themselves as instruction-tuned, thus making their characteristics potentially less general. In addition, the Davinci and ChatGPT-based detectors share an interesting asymmetrical relationship, where Davinci detectors generalize very well to ChatGPT generations, but not vice versa. This might also be explained in the context of the degree to which the model is close to the original concept of a general-purpose language model. Davinci, in essence GPT-3, can be used for all types of completions, whereas ChatGPT is optimized as a chatbot, and takes specialized input prompts as a result. Another standout performance is the DistilBERT classifier trained on Dolly generations, which boasts high recall on all other generators except BLOOMz. The TF-IDF variant of the same detector loses a large chunk of recall across the board, and the feature-based classifier impresses with its ability *not* to generalize. It is important that some detectors show generalization to unknown generators, as this is after all a black-box detection task, where the test set contains a model not seen during training, namely GPT-4. At the same time, a detector can also contribute by being excellent at detecting only its target, since it would be pointless to aim for unknown generators at the cost of dwindling prediction power over known ones. The picture painted by Table 9 is of a good balance between the two specializations.

Composition	Development accuracy	Test accuracy
TF-IDF only	99.527%	87.179%
Language features only	98.777%	85.466%
DistilBERT only	96.250%	74.790%
TF-IDF + features	99.927%	95.544%
Full Model	99.892%	97.106%
Winning model	NA	96.88%

Table 10 Summary of final ensemble performance across different configurations. The best-performing model is the full ensemble, beating the task-winning model, whose reported accuracy is included for comparison.

6.3 Subtask C: Shared task analysis and rankings

7 Conclusion

References

D. I. Adelani, H. Mai, F. Fang, H. H. Nguyen, J. Yamagishi, and I. Echizen. Generating sentiment-preserving fake online reviews using neural language models and their human- and machine-based detection, 2019. URL https://arxiv.org/abs/1907.09177.

Alec Radford and Jeff Wu and Jong Wook Kim. GPT-2 Output Dataset, 2019. URL https://github.com/openai/gpt-2-output-dataset.

J. A. Anderson. *An introduction to neural networks*. MIT press, 1995.

I. Asimov. *Foundation*. Foundation series. Gnome Press, 1951. URL https://books.google.it/books?id=gFpQswEACAAJ.

S. Badaskar, S. Agarwal, and S. Arora. Identifying real or fake articles: Towards better language modeling. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*, 2008.

S. Baki, R. Verma, A. Mukherjee, and O. Gnawali. Scaling and effectiveness of email masquerade attacks: Exploiting natural language generation. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, page 469–482, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349444. doi: 10.1145/3052973.3053037. URL https://doi.org/10.1145/3052973.3053037.

R. Barzilay and M. Lapata. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34, 2008.

I. Beltagy, M. E. Peters, and A. Cohan. Longformer: The long-document transformer, 2020. URL https://arxiv.org/abs/2004.05150.

P. Bhatt and A. Rios. Detecting bot-generated text by characterizing linguistic accommodation in human-bot interactions, 2021. URL https://arxiv.org/abs/2106.01170.

A. P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners, 2020.

B. Buchanan, A. Lohn, M. Musser, and K. Sedova. Truth, lies, and automation. *Center for Security and Emerging technology*, 1(1):2, 2021.

G. Cabanac and C. Labbé. Prevalence of nonsensical algorithmically generated papers in the scientific literature. *Journal of the Association for Information Science and Technology*, 72 (12):1461–1476, 2021.

D. E. Cahyani and I. Patasik. Performance comparison of tf-idf and word2vec models for emotion text classification. *Bulletin of Electrical Engineering and Informatics*, 10(5): 2780–2788, 2021.

C. Chaka. Detecting ai content in responses generated by chatgpt, youchat, and chatsonic: The case of five ai content detection tools. *Journal of Applied Learning and Teaching*, 6 (2), 2023.

M. Chen et al. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394, 1999.

M. Conover, M. Hayes, A. Mathur, J. Xie, J. Wan, S. Shah, A. Ghodsi, P. Wendell, M. Zaharia, and R. Xin. Free dolly: Introducing the world's first truly open

instruction-tuned llm, 2023. URL https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm.

E. Crothers, N. Japkowicz, H. Viktor, and P. Branco. Adversarial robustness of neural-statistical features in detection of generative transformers. In *2022 International Joint Conference on Neural Networks (IJCNN)*, volume 21, page 1–8. IEEE, July 2022a. doi: 10.1109/ijcnn55064.2022.9892269. URL http://dx.doi.org/10.1109/IJCNN55064.2022.9892269.

E. Crothers, N. Japkowicz, H. Viktor, and P. Branco. Adversarial robustness of neural-statistical features in detection of generative transformers. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2022b.

E. Crothers, N. Japkowicz, and H. Viktor. Machine generated text: A comprehensive survey of threat models and detection methods, 2023. URL https://arxiv.org/abs/2210.07321.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

E. Dinan, V. Logacheva, V. Malykh, A. Miller, K. Shuster, J. Urbanek, D. Kiela, A. Szlam, I. Serban, R. Lowe, S. Prabhumoye, A. W. Black, A. Rudnicky, J. Williams, J. Pineau, M. Burtsev, and J. Weston. The second conversational intelligence challenge (convai2), 2019. URL https://arxiv.org/abs/1902.00098.

T. Fagni, F. Falchi, M. Gambini, A. Martella, and M. Tesconi. Tweepfake: About detecting deepfake tweets. *Plos one*, 16(5):e0251415, 2021.

L. Feng, M. Jansche, M. Huenerfauth, and N. Elhadad. A comparison of features for automatic readability assessment. In *Coling 2010: Posters*, pages 276–284, 2010.

G. A. Fowler. Say what, Bard? What Google's new AI gets right, wrong and weird, 2023. URL https://www.washingtonpost.com/technology/2023/03/21/google-bard/.

L. Fröhling and A. Zubiaga. Feature-based detection of automated language models: tackling gpt-2, gpt-3 and grover. *PeerJ Computer Science*, 7:e443, 2021.

M. Gallé, J. Rozen, G. Kruszewski, and H. Elsahar. Unsupervised and distributional detection of machine-generated text. *arXiv preprint arXiv:2111.02878*, 2021.

C. A. Gao, F. M. Howard, N. S. Markov, E. C. Dyer, S. Ramesh, Y. Luo, and A. T. Pearson. Comparing scientific abstracts generated by chatgpt to original abstracts using an artificial intelligence output detector, plagiarism detector, and blinded human reviewers. *BioRxiv*, pages 2022–12, 2022.

J. Gao and C.-Y. Lin. Introduction to the special issue on statistical language modeling, 2004.

S. Gehrmann, H. Strobelt, and A. M. Rush. Gltr: Statistical detection and visualization of generated text, 2019. URL https://arxiv.org/abs/1906.04043.

A. Giaretta and N. Dragoni. *Community Targeted Phishing: A Middle Ground Between Massive and Spear Phishing Through Natural Language Generation*, page 86–93. Springer International Publishing, Mar. 2019. ISBN 9783030146870. doi: 10.1007/978-3-030-14687-0_8. URL http://dx.doi.org/10.1007/978-3-030-14687-0_8.

L. Hansen, L. R. Olsen, and K. Enevoldsen. TextDescriptives: A Python package for calculating a large variety of metrics from text. *Journal of Open Source Software*, 8(84): 5153, Apr. 2023. ISSN 2475-9066. doi: 10.21105/joss.05153. URL http://dx.doi.org/10.21105/joss.05153.

J. Hargrave. SCIgen - An Automatic CS Paper Generator, 2005. URL https://pdos.csail.mit.edu/archive/scigen/.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd. spaCy: Industrial-strength Natural Language Processing in Python. 2020. doi: 10.5281/zenodo.1212303.

J. Howard and S. Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

D. Ippolito, D. Duckworth, C. Callison-Burch, and D. Eck. Automatic detection of generated text is easiest when humans are fooled. *arXiv preprint arXiv:1911.00650*, 2019.

A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed. Mistral 7b, 2023.

S. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE transactions on acoustics, speech, and signal processing*, 35(3): 400–401, 1987.

M. Khonji, Y. Iraqi, and A. Jones. Phishing detection: A literature survey. *IEEE Communications Surveys & Tutorials*, 15:2091–2121, 2013. URL `https://api.semanticscholar.org/CorpusID:4133482`.

K. Krishna, Y. Song, M. Karpinska, J. Wieting, and M. Iyyer. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *Advances in Neural Information Processing Systems*, 36, 2024.

M. Latah. Detection of malicious social bots: A survey and a refined taxonomy. *Expert Systems with Applications*, 151:113383, 2020.

B. W. Lee and J. H.-J. Lee. Lftk: Handcrafted features in computational linguistics, 2023. URL `https://arxiv.org/abs/2305.15878`.

C.-C. Lin, A. Jaech, X. Li, M. R. Gormley, and J. Eisner. Limitations of autoregressive models and their alternatives, 2021. URL `https://arxiv.org/abs/2010.11939`.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach, 2019a.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach, 2019b. URL `https://arxiv.org/abs/1907.11692`.

D. Macko, R. Moro, A. Uchendu, I. Srba, J. S. Lucas, M. Yamashita, N. I. Tripto, D. Lee, J. Simko, and M. Bielikova. Authorship obfuscation in multilingual machine-generated text detection. *arXiv preprint arXiv:2401.07867*, 2024.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space, 2013. URL `https://arxiv.org/abs/1301.3781`.

N. Mireshghallah, J. Mattern, S. Gao, R. Shokri, and T. Berg-Kirkpatrick. Smaller language models are better black-box machine-generated text detectors. *arXiv preprint arXiv:2305.09859*, 2023.

E. Mitchell, Y. Lee, A. Khazatsky, C. D. Manning, and C. Finn. Detectgpt: Zero-shot machine-generated text detection using probability curvature, 2023a.

E. Mitchell, Y. Lee, A. Khazatsky, C. D. Manning, and C. Finn. DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature, 2023b. URL `https://arxiv.org/abs/2301.11305`.

N. Muennighoff, T. Wang, L. Sutawika, A. Roberts, S. Biderman, T. L. Scao, M. S. Bari, S. Shen, Z.-X. Yong, H. Schoelkopf, X. Tang, D. Radev, A. F. Aji, K. Almubarak,

S. Albanie, Z. Alyafeai, A. Webson, E. Raff, and C. Raffel. Crosslingual generalization through multitask finetuning, 2023. URL https://arxiv.org/abs/2211.01786.

K. P. Murphy et al. Naive bayes classifiers. *University of British Columbia*, 18(60):1–8, 2006.

J. Ni, J. Li, and J. McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 188–197, 2019.

OpenAI et al. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

M. Ott, Y. Choi, C. Cardie, and J. T. Hancock. Finding deceptive opinion spam by any stretch of the imagination. *arXiv preprint arXiv:1107.4557*, 2011.

Papers With Code. Natural language processing benchmarks, 2024. URL https://paperswithcode.com/area/natural-language-processing.

A. Radford. Improving language understanding by generative pre-training. 2018.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019a.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019b.

C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023. URL https://arxiv.org/abs/1910.10683.

J. Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer, 2003.

J. D. Rodriguez, T. Hay, D. Gros, Z. Shamsi, and R. Srinivasan. Cross-domain detection of gpt-2-generated technical text. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: human language technologies*, pages 1213–1233, 2022.

J. Salminen, C. Kandpal, A. M. Kamel, S.-g. Jung, and B. J. Jansen. Creating and detecting fake reviews of online products. *Journal of Retailing and Consumer Services*, 64:102771, 2022.

V. Sanh, L. Debut, J. Chaumond, and T. Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter, 2020. URL https://arxiv.org/abs/1910.01108.

A. See, A. Pappu, R. Saxena, A. Yerukola, and C. D. Manning. Do massively pretrained language models make better storytellers? *arXiv preprint arXiv:1909.10705*, 2019.

S. Selva Birunda and R. Kanniga Devi. A review on word embedding techniques for text classification. *Innovative Data Communication Technologies and Application: Proceedings of ICIDCA 2020*, pages 267–281, 2021.

A. Shostack. *Threat Modeling: Designing for Security*. Wiley Publishing, 1st edition, 2014. ISBN 1118809998. URL https://www.wiley.com/en-us/Threat+Modeling%3A+Designing+for+Security-p-9781118809990.

I. Solaiman, M. Brundage, J. Clark, A. Askell, A. Herbert-Voss, J. Wu, A. Radford, G. Krueger, J. W. Kim, S. Kreps, et al. Release strategies and the social impacts of language models. *arXiv preprint arXiv:1908.09203*, 2019.

P. D. Stamatatos et al. On the robustness of authorship attribution based on character n-gram features. *Journal of Law and Policy*, 21(2):7, 2013.

D. Stuhlinger and A. Winkler. TueCICL at SemEval-2024 task 8: Resource-efficient approaches for machine-generated text detection. In A. K. Ojha, A. S. Doğruöz, H. Tayyar Madabushi, G. Da San Martino, S. Rosenthal, and A. Rosá, editors, *Proceedings of the*

18th International Workshop on Semantic Evaluation (SemEval-2024), pages 1597–1601, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.semeval-1.227. URL https://aclanthology.org/2024.semeval-1.227.

G. Team et al. Gemini: A family of highly capable multimodal models, 2024. URL https://arxiv.org/abs/2312.11805.

J. Tourille, B. Sow, and A. Popescu. Automatic detection of bot-generated tweets. In *Proceedings of the 1st International Workshop on Multimedia AI against Disinformation*, pages 44–51, 2022.

H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. LLaMA: Open and efficient foundation language models, 2023.

S. Vajjala and D. Meurers. On improving the accuracy of readability classification using insights from second language acquisition. In J. Tetreault, J. Burstein, and C. Leacock, editors, *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 163–173, Montréal, Canada, June 2012. Association for Computational Linguistics. URL https://aclanthology.org/W12-2019.

R. Van Hout and A. Vermeer. Comparing measures of lexical richness. *Modelling and assessing vocabulary knowledge*, 93:115, 2007.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

B. Wang, C. Xu, S. Wang, Z. Gan, Y. Cheng, J. Gao, A. H. Awadallah, and B. Li. Adversarial glue: A multi-task benchmark for robustness evaluation of language models. *arXiv preprint arXiv:2111.02840*, 2021.

Y. Wang, J. Mansurov, P. Ivanov, J. Su, A. Shelmanov, A. Tsvigun, O. M. Afzal, T. Mahmoud, G. Puccetti, T. Arnold, et al. Semeval-2024 task 8: Multidomain, multimodel and multilingual machine-generated text detection. *arXiv preprint arXiv:2404.14183*, 2024a.

Y. Wang, J. Mansurov, P. Ivanov, J. Su, A. Shelmanov, A. Tsvigun, C. Whitehouse, O. Mohammed Afzal, T. Mahmoud, T. Sasaki, T. Arnold, A. Aji, N. Habash, I. Gurevych, and P. Nakov. M4: Multi-generator, multi-domain, and multi-lingual black-box machine-generated text detection. In Y. Graham and M. Purver, editors, *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1369–1407, St. Julian's, Malta, Mar. 2024b. Association for Computational Linguistics. URL https://aclanthology.org/2024.eacl-long.83.

J. Weizenbaum. ELIZA — a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.

A. Wendland, M. Zenere, and J. Niemann. Introduction to text classification: impact of stemming and comparing tf-idf and count vectorization as feature extraction technique. In *Systems, Software and Services Process Improvement: 28th European Conference, EuroSPI 2021, Krems, Austria, September 1–3, 2021, Proceedings 28*, pages 289–300. Springer, 2021.

I. Yamada, A. Asai, J. Sakuma, H. Shindo, H. Takeda, Y. Takefuji, and Y. Matsumoto. Wikipedia2Vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from Wikipedia. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 23–30. Association for Computational Linguistics, 2020.

R. Zellers, A. Holtzman, E. Clark, L. Qin, A. Farhadi, and Y. Choi. Evaluating machines by their real-world language use. *arXiv preprint arXiv:2004.03607*, 2020a.

R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner, and Y. Choi. Defending against neural fake news, 2020b. URL https://arxiv.org/abs/1905.12616.

J. Zhang. Gradient descent based optimization algorithms for deep learning models training, 2019. URL https://arxiv.org/abs/1903.03614.

W. Zhang, T. Yoshida, and X. Tang. A comparative study of tf* idf, lsi and multi-words for text classification. *Expert systems with applications*, 38(3):2758–2765, 2011.

Selbständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Masterarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Ort, Datum Unterschrift