



# Large Language Models in Defense: Challenges and Opportunities

FARZAD KAMRANI, LINUS KANESTAD,  
CHRISTOFFER LIMÉR, BJÖRN PELZER,  
IZA SMEDBERG, AGNES TEGEN,  
ULRIKA WICKENBERG BOLIN



Farzad Kamrani, Linus Kanestad,  
Christoffer Limér, Björn Pelzer,  
Iza Smedberg, Agnes Tegen,  
Ulrika Wickenberg Bolin

## Large Language Models in Defense: Challenges and Opportunities

Titel	Stora språkmodeller för försvaret: utmaningar och möjligheter
Title	Large Language Models in Defense: Challenges and Opportunities
Report no	FOI-R--5544--SE
Month	May
Year	2024
Pages	68
ISSN	ISSN-1650-1942
Customer	Swedish Armed Forces
FOI Research area	C3 and Human Factors
Armed Forces R&T area	Command and Control
Project no	E38519
Approved by	Emil Hjalmarson
Division	Cyber Defence and C2 Technology
Export control	The content has been reviewed and does not contain information which is subject to Swedish export control.

Bild/Cover: "The Tower of Babel", by Pieter Bruegel the Elder, 1563"

This work is protected by the Swedish Act on Copyright in Literary and Artistic Works (1960:729). Citation is permitted in accordance with article 22 in said act. Any form of use that goes beyond what is permitted by Swedish copyright law, requires the written permission of FOI.

## Summary

Large language models (LLMs) are being hailed as a major breakthrough in artificial intelligence. Their ability to process and produce texts on a level typically associated with human cognition gives them an enormous potential for applications across all sectors, including defense. At the same time this new technology comes with many open questions regarding its robustness and reliability, and any organization wishing to utilize LLMs faces significant technical challenges. This report aims to demonstrate how LLMs can be trained to adapt them to a Swedish defense domain, and to evaluate whether such a project can be worth the effort. Towards this aim, a dataset based on Swedish and English texts from a defense domain is created and then used to train (fine-tune) two state-of-the-art LLMs. These models are then evaluated both qualitatively and quantitatively. The results show that the LLMs benefit from the training, exhibiting improved performance on textual tasks pertaining to Swedish defense. The detailed description of the training process can also serve as a guide for readers interested in pursuing a similar project. The hurdles in training are largely related to resource constraints, such as hardware, data and time - which may make them difficult to overcome, but at least they are relatively well understood. The same cannot be said of the evaluation of LLMs: The models have surprising capabilities, but they can also fail in surprising ways. There is no established method of testing LLMs thoroughly and objectively, and the evaluation in the report tackles the subject by testing different aspects of LLMs, but it can only scratch the surface. In conclusion, large language models have reached a stage where defense stakeholders can and arguably should begin to adapt and test the technology, and the report can help in this by providing insights into pitfalls, solutions and lessons learnt. At the same time a level-headed approach to LLMs is recommended, as the evaluation of such models must still be regarded as an open question.

Keywords: artificial intelligence, large language models, fine-tuning, Parameter-Efficient Fine-Tuning, Low Rank Adaptation

## Sammanfattning

Stora språkmodeller (LLM, eng. *large language model*) hyllas som ett genombrott inom artificiell intelligens. Med sin förmåga att bearbeta och producera texter på en nivå som vanligtvis förknippas med mänsklig kognition har de enorm potential för tillämpningar inom alla sektorer, inklusive försvaret. Samtidigt kvarstår många öppna frågor om den nya teknikens robusthet och tillförlitlighet, och organisationer som vill använda LLM står inför betydande tekniska utmaningar. Denna rapport syftar till att visa hur LLM:er kan tränas för att anpassa dem till en svensk försvarsdomän. Dessutom ska det utvärderas om ett sådant projekt kan vara värt de nödvändiga investeringarna. För detta ändamål skapas ett dataset baserat på svenska och engelska texter från en försvarsdomän. Sedan tränas (finjusteras, eng. *fine-tuning*) två moderna LLM:er på datamängden. Modeller utvärderas både kvalitativt och kvantitativt. Resultaten visar att LLM:erna drar nytta av träningen i och med att de uppvisar förbättrad prestanda på textuppgifter som rör svenskt försvar. Den detaljerade beskrivningen av träningsprocessen kan också fungera som en guide för läsare som vill driva ett liknande projekt. Utmaningarna i träningen är till stor del relaterade till resursbegränsningar, såsom hårdvara, data och tid - vilket kan göra dem svåra att övervinna, men de är åtminstone relativt väl förstådda. Detsamma kan inte sägas om utvärderingen av LLM:er. Modellerna har överraskande förmågor, men de kan också misslyckas på överraskande sätt. Det finns idag ingen etablerad metod för att testa LLM:er grundligt och objektivt. Utvärderingen i rapporten testar därför ett flertal olika aspekter av LLM:er, men den kan bara skrapa på ytan. Stora språkmodeller har nått ett stadium där försvaret kan och bör börja anpassa och testa teknologin, och rapporten kan hjälpa till med detta genom att ge insikter i fallgropar, lösningar och lärdomar. Samtidigt rekommenderas ett övervägande förhållningssätt till LLM:er, eftersom utvärderingen av sådana modeller fortfarande måste ses som en öppen fråga.

Nyckelord: artificiell intelligens, stora språkmodeller, fine-tuning, Parameter-efficient fine-tuning, Low rank adaptation

# Contents

<b>1</b>	<b>Introduction</b>	<b>7</b>
1.1	Objective and Scope . . . . .	8
1.2	Target Readership . . . . .	9
1.3	Reading Instructions . . . . .	9
1.4	Outline . . . . .	9
<b>2</b>	<b>Large Language Models</b>	<b>10</b>
2.1	Basic Principles . . . . .	11
2.2	Important Concepts . . . . .	13
2.3	Optimizing Training - PEFT and LoRA . . . . .	15
2.4	Challenges in Evaluation of LLMs . . . . .	17
<b>3</b>	<b>Creating LLMs for Defense</b>	<b>19</b>
3.1	Method . . . . .	19
3.2	Base Models . . . . .	20
3.2.1	Llama 2 . . . . .	21
3.2.2	GPT-SW3 . . . . .	21
3.2.3	Mistral . . . . .	22
3.2.4	Model Selection . . . . .	23
3.3	Training Data . . . . .	25
3.3.1	Failure of Automated Text Extraction . . . . .	25
3.3.2	Manual Text Extraction . . . . .	27
3.3.3	Dataset . . . . .	27
3.3.4	Discussion: Model Tokenizers . . . . .	29
3.4	Training . . . . .	32
3.4.1	Training Process . . . . .	33
3.4.2	Trained Models . . . . .	34
<b>4</b>	<b>Evaluation</b>	<b>36</b>
4.1	Performed Experiments . . . . .	37
4.1.1	Model Robustness: System Messages . . . . .	38
4.1.2	Model Robustness: Instructions . . . . .	44
4.1.3	Language Correctness . . . . .	46
4.1.4	Variance and Repetitions in Text . . . . .	53
4.2	Towards a Rigorous Evaluation . . . . .	58
<b>5</b>	<b>Discussion</b>	<b>60</b>
<b>6</b>	<b>Conclusions and Future Work</b>	<b>63</b>

## References

64

# 1 Introduction

Artificial intelligence (AI) is the branch of computer science concerned with developing machines that can solve problems normally requiring human cognition. One such challenge is the processing of human language, that is, to enable computers to understand textual input and provide cogent responses. This deceptively simple problem can appear trivial to humans, and it was originally believed to be easy to solve algorithmically. Early attempts at machine translation in the 1950s soon hit a wall as scientists began to understand the complexity of the task. Traditionally, computational linguistics (CL) has tried to tackle the problem by investigating the rules that govern human language and formalizing them in a manner suitable for computers. Natural language processing (NLP) on the other hand has taken a more pragmatic and often statistical approach in its focus on developing systems that can actually perform some linguistic tasks, even if limited in scope. In practice the distinction between the fields has blurred considerably over the decades, but for a long time the common goal remained elusive.

In recent years, however, the rise of deep learning (DL) has accelerated progress in many challenges of AI, including language. Enhanced hardware and ever larger sets of digital data have made it feasible to train deep neural networks with billions of parameters on millions of texts. Thereby these networks learn the probabilities of how words occur within contexts, resulting in large statistical models of natural language. Such large language models (LLMs) are then capable of processing textual input and in response generating new texts that appear to rival human understanding and writing. Powerful LLMs like *ChatGPT* by OpenAI have been garnering considerable attention from media and the public, meeting both praise and concern as a major AI breakthrough with, as yet, unclear consequences. However, at the time of this writing, the application of LLMs is still in an exploratory stage, so far mainly found in the form of chat bots or as text assistants in office software. Further, the potential for military applications remains difficult to estimate. LLMs could serve in all areas of defense and intelligence, e.g., as part of user interfaces, information aggregators, assisting with documentation, and in decision making by systems that suggest and explain course of actions.

LLMs can deliver impressive results, but they can also fail in surprising ways. Their capabilities, limitations and reliability are not yet well understood and subject to rapid change as development progresses. Another impediment to adoption is the cost involved with training and operating an LLM. The most powerful LLMs are created on massive supercomputers that are out of reach even for many state actors. Some of these LLMs are only accessible as online services, operated on commercial servers on foreign territory, rendering their use dubious when security aspects are critical. Others can be obtained and operated locally, and it is possible to train these further to adapt them to specific tasks (*fine-tuning*), but for the initial creation one remains dependent on the few organizations with sufficient resources. This also means that the selection of texts for the initial training corpus is beyond the control of most LLM users, affecting the balancing of texts with respect to topics and quality, limiting the supported languages, and harboring a potential for legal repercussions if the original LLM creators have not ensured that their use of all training texts is within intellectual property rights.

The current situation regarding LLMs is characterized by both optimism



and uncertainty. On the one hand, LLMs may be on the verge of revolutionizing countless processes where human cognition has been presumed to be a requirement, both in the civilian and the military sphere. On the other hand the reliability of LLMs is unclear, and there is a risk of organizations getting swept away with the enthusiasm of breakthrough technology, shoehorning LLMs into applications for which they are not (yet) suitable.

This report presents a careful adaptation of LLMs towards defense-related purposes. Several modern LLMs are trained in a process that is described in detail. The LLMs are then evaluated with respect to their robustness and the quality of their output. The defense domain covers a broad and diverse range of topics, and the performance of an LLM on a topic is dependent on the preparation of a large training corpus with texts of high quality and relevance. Due to resource limitations the report therefore narrows down the scope of the training domain by focusing on the example of LLMs intended to support a security policy analyst.

## 1.1 Objective and Scope

The scope of this report is focused on LLMs intended to support an analyst within the defense domain of security policy. Such an LLM may be expected to summarize a text, answer questions related to security policy, and to write a text based on a given list of key phrases. Analogous tasks are equally relevant in other domains, hence the restriction to security policy implies no loss of generality with respect to where an LLM trained along similar lines could be used. Further, the basic principles of training and implementation also apply to other topics, as well as on a broader scale.

The objective of the report is to explore the technical challenges of deploying and operating LLMs in a Swedish defense context, illustrated by the example of LLMs trained to support a security policy analyst, and to evaluate the effectiveness of the adaptations. This is carried out by:

1. Training (fine-tuning) LLMs for an application in the defense domain, including the preparation of training data, the selection of base models, setting up the training environment, and the training process;
2. Evaluating the performance of the trained LLMs, including the quality of the output texts according to different metrics, the sensitivity of the models to prompt variations and other factors, and whether the cost of fine-tuning is offset by notable improvements over the base models.

The report focuses on the core issues of LLM technology, that is, the models themselves, their training and their capabilities. As such, there will be no investigation of application-specific implementations, for example how to utilize an LLM in a RAG system (*retrieval augmented generation*), where external knowledge is retrieved from a database and inserted into the prompt, enabling an LLM to resolve queries that need current information [1]. While this and other approaches are promising ways of using LLMs, they do add their own research questions. Also, any implementation using LLMs benefits from the models being well-trained and well-understood, hence investigations beyond these central basic questions shall remain outside the scope of this report.

Another matter that will merely be touched upon is prompt engineering. LLMs are sensitive to how a query is phrased, and they may sometimes provide more useful responses if the query is reformulated [2]. It has even been observed that the performance may be improved by adding words of encouragement (e.g., “*You are a smart model. Please think hard about the following...*”). This has led to a multitude of more or less intuitive attempts at optimizing the prompts.

However, any such prompt additions will reduce the number of actual query words that will fit into the limited input window of an LLM. Furthermore, the benefit of prompt engineering is not consistent, and the methods may at times actually decrease performance [3]. The combinatorial space of possible paraphrasings and modifications is practically limitless, and there is increasing evidence that the optimal prompts may not be intuitive at all [4], and thus unlikely to be formulated by human prompt engineers. Given the current uncertain state of prompt engineering, no reliable recommendations can be made at the time of this writing, and the topic is therefore mostly relegated to future work.

## 1.2 Target Readership

The report addresses personnel within the defense sector who wish to adapt and deploy LLMs in military or intelligence applications. This includes both decision-makers evaluating whether LLMs may be suitable for an intended application as well as technical teams training and implementing an LLM-based solution.

## 1.3 Reading Instructions

Generally this report is written on a level that should be accessible to any reader interested in AI and LLMs. Occasionally the report will delve into deeper detail, but readers who prefer to skip ahead should face no difficulty following the overall substance of the report. Readers who wish to familiarize themselves more with the theoretical backgrounds of natural language processing using deep neural networks are encouraged to read chapter 3 in the report *Natural Language Processing Using Deep Neural Networks* [5].

## 1.4 Outline

Chapter 2 provides the theoretical background of this report. An overview of large language models introduces the most important concepts and principles of this technology. Furthermore, the reader is acquainted with the specific training optimizations used later on in the report, including their formal background. Finally, the section introduces the long-standing challenge of how to evaluate systems which process and generate natural language. Evaluation must be regarded as an open research question, and the associated issues remain relevant in the age of LLMs and affect the evaluation attempts within this report.

Chapter 3 follows the first objective of creating LLMs for the defense domain. The chapter is a step-by-step description of selecting suitable base LLMs, preparing suitable training data and training the models. The section may thus also serve as a guide for readers wishing to pursue similar projects.

In Chapter 4 follows the second objective: The trained LLMs are evaluated using a variety of quantitative and qualitative methods and metrics. The aforementioned challenges in this area mean that this section is an introduction to a broad range of attempts to tackle the issue, and an exhaustive large-scale evaluation must remain relegated to future work.

Chapter 5 discusses the evaluation results, as well as broader implications of utilizing LLMs in a defense context, including insights and lessons learnt.

Finally, Chapter 6 concludes the report with a summary of the findings and a short outlook on possible future work.

## 2 Large Language Models

Since OpenAI released ChatGPT in November of 2022 the interest for language models has escalated greatly. While LLMs were already an important research topic in the AI community before ChatGPT, this particular model was frequently mentioned in the media and made LLMs known in the public domain. This in turn has raised the research interest even further.

Vaswani et al. introduced a new deep learning architecture called *transformer* in 2017 [6]. The research paper spread quickly within the AI community and soon after the first language models based on the transformer architecture were presented. While the article by Vaswani et al. became a significant leap forward in the development of language models, their work was built on previous breakthroughs, e.g., the attention mechanism [7] as well as using the encoder and decoder principle [8].

From the introduction of transformers there has been a continuous increase in performance. The language model BERT was released by Google in 2018 [9] and was state-of-the-art for several natural language processing tasks at the time. The same year, OpenAI presented GPT-1 (*Generative Pre-trained Transformer*) [10], but they waited with public release until 2019 when they presented an updated version called GPT-2 [11] and the following year, 2020, they released GPT-3 [12]. In 2021 Google released a large language model titled LaMDA (*Language Model for Dialogue Applications*) [13]. It was trained using conversation and dialogue rather than other types of text corpora and gained media attention when an engineer at Google claimed it had become sentient [14]. In November of 2022 OpenAI released ChatGPT, which received a large media coverage since it was easily accessible to people outside the AI community. ChatGPT became extremely popular and had 100 million active users only two months after its launch. In early 2023, Google released *Bard* which was meant to rival ChatGPT and was powered by LaMDA. Around the same time Meta introduced their large language model Llama (*Large Language Model Meta AI*) [15]. Llama was available in several different sizes, giving the user option on prioritizing performance (which meant a larger model that probably had to be trained in the cloud), or lower training time and the ability to run the model locally (which meant a smaller model that might not give as high performance). In March of 2023, OpenAI released their newest version GPT-4, and Meta released their updated large language model Llama 2 in July of the same year. Mistral AI, founded by researchers previously at Meta and Google, released their large language model called *Mistral* [16] in September 2023 and a larger version in December. *Bard* was renamed *Gemini* by Google in February of 2024, having undergone several updates since its first release.

As evident by the history of large language models, a significant amount has happened in recent years and news of improved models and new releases are happening continuously. With each new release, the performance on natural language processing tasks often increases as the models grow in size and complexity. However, with the models increasing in size, there is a need for improved hardware, more time and typically more data are needed to train and run the models. Depending on the task at hand, it can be preferred to have a fast process or to run the model locally (e.g., for security reasons), rather than having the top performing model.

LLMs have received positive attention in the media and an enthusiastic response by the public, being hailed as a major breakthrough in AI technology

and utilized by hundreds of millions of users. At the same time concerns have been voiced about unforeseeable consequences of this technology, ranging from cheating in schools, to mass-production of propaganda and fake news, to endangering the jobs of entire professions, all the way to full-blown doomsday scenarios. A more concrete concern is the often dubious provenance of training data, as the AI companies producing the largest LLMs have scoured the web for useful texts. For example, in late 2023 the New York Times sued OpenAI and Microsoft for using its newspaper articles in a manner that leads to the GPT-series LLMs at times reproducing entire copyrighted texts, without permission [17]. The outcome of lawsuits like this may have repercussions on the usage of LLMs that are difficult to assess at the time of this writing.

## 2.1 Basic Principles

Large language models produce their output based on the text prompt used as input. The model predicts, based on previous training, what the most likely words are from the previous words written. In this regard it differs from e.g., image analysis, where the input is an image and the output typically detects something in the image or classifies what the image contains. For large language models, the output can be regarded as a continuation of the text of the input prompt. More specifically, the model predicts the next token since it interprets the text through tokenization.

The focus in this report is on large language models within the group of generative machine learning models, which are able to create new content. This is in comparison to non-generative, discriminative models, where the aim is often to construct boundaries between categories and then classify test samples within those boundaries. For example, the large language model BERT is unsuitable for text generation, but it is very effective at classifying text, such as determining whether a text contains hate speech or not.

All large language models available, e.g., Llama or Mistral, are trained on a massive text corpus. The publicly available model is called a *base model*, and it has typically been trained on a broad range of different types of texts in order to have a general “understanding” of language. This prepares the model to produce a variety of texts in response to inputs, basically extending the given input with the most likely continuation, which can include answers to questions and performing instructions requested by the user, rather than merely expanding a text. This type of functionality is particularly interesting for applications where the LLM interacts with the user. To improve this functionality an LLM can be further trained on texts that emphasize questions and instructions in the manner of dialogues, teaching the model to converse in a similar fashion. This is a form of fine-tuning, and the resulting model is called an instruction-tuned model, or simply an *instruction model*.

More frequently, fine-tuning involves training an LLM to better handle the texts of some specific application domain, such as the security policy example used in this report. It is also possible to combine instruction-tuning with domain-oriented fine-tuning.

Generally speaking, fine-tuning builds upon the fundamental language capability that the base model has gained from the extensive initial training on its enormous generalist text corpus. This means that in comparison to the costly creation of a base model, fine-tuning can be effective with a relatively small amount of training data, as the LLM does not start from a blank slate. Hence there are lower technical hurdles to fine-tuning of existing base models, putting this within reach of medium and smaller-sized organizations and making it a suitable strategy for adapting and adopting LLM technology to various

applications.

An alternative to fine-tuning a model is prompt engineering. In prompt engineering, it could be said the prompt is being fine-tuned, rather than the model. While prompt engineering does not require data, time and resources for fine-tuning, it requires the user to carefully construct the prompt to get a desired result. Depending on how specific the desired response is, only prompt engineering might not be enough.

The currently most advanced and most frequently used large language models are built on a deep learning architecture called *transformer* [6]. Not unique, but essential, to transformers is the encoder-decoder principle. Encoders and decoders can be found in a broad spectrum of machine learning applications such as image processing and natural language processing. Encoders translate the input data into an intermediate form that can be evaluated and enhanced during the subsequent information, and decoders eventually translate this processed intermediate data back into the desired output data format. In the transformer architecture of an LLM, multiple encoders iteratively enrich the internal representation of each token with information about other tokens from the text, thereby turning the original text representation, a sequence of tokens, into an interconnected representation of the semantics: The input is no longer a mere chain of words, but a text with meaning. Decoders translate the enriched internal representation back into text form. Implementationally, encoders and decoders are very similar. Some models have modified the original transformer architecture to use only encoders (e.g., BERT), making them more suitable for classification tasks, while others use only decoders (e.g., the GPT-series LLMs), as they focus on generating new texts.

An issue with earlier encoder-decoder models before transformers was that the learning unintentionally focused more on the later part of the input text and had a tendency to forget earlier input. Mathematically, with increasing length of the input text, the significance of the beginning of the text diffused over the course of the processing, leaving it with less and less impact on the computation of the output. This is the so-called *vanishing gradient problem*, which prevented these models from fully utilizing all information in the text, and rendering them incapable of maintaining a coherent understanding of the text as a whole.

Some machine learning model types predating transformers sought to ameliorate the vanishing gradient problem, such as the LSTM (*long short-term memory*) architecture, and they could also learn the most probable reply to a query or continuation of a text. Yet essentially the forgetting was merely delayed. Underlying fundamental problems remained, for example the need to compress an input text of arbitrary length into a fixed-length vector, resulting in information loss.

Eventually the *attention* mechanism was introduced to address issues with long input sequences [7]. This mechanism does not only prevent the forgetting, it also makes it possible to identify more relevant parts of the text that the model should focus on. It gives emphasis to certain words or parts of the text, while playing down others, to train the model in giving better replies by improving its understanding of the text. The transformer architecture employs several variants of the attention mechanism throughout its processing. This has in principle removed the theoretical limits on input text length for transformer-based language models. In practice other factors (e.g., hardware resources) still prevent the processing of arbitrary input sizes, and all current LLMs implement limits on input text length. However, these limits greatly surpass the capabilities of models predating the attention mechanism, and the limits keep increasing.

## 2.2 Important Concepts

This section describes the terminology related to large language models which is relevant when discussing and comparing different versions.

**Parameters:** Parameters can be said to be the fundamental building blocks of the model architecture. They correspond to the weights of neural networks in general, and the terms are effectively synonymous. Each parameter is tuned when the model is trained to match the training data as close as possible. It is typically not meaningful to study a single parameter, but it is rather the intricate network of connected parameters that gives the model its performance. The number of parameters is a common measurement of the size of a model, as it shows how complex the model is. If the architecture of the model is constructed well, a larger number of parameters generally means better performance. However, there are smaller models that are better performing than larger counterparts, if the smaller models have a better architecture [16].

**Temperature:** The temperature value is a hyperparameter not set during training but rather by a user when employing the model. Temperature decides the level of randomness or freedom of the responses generated by the model. A higher temperature can lead to a more creative and diverse response, but it can also become nonsensical with a higher risk of incorrect information. A lower temperature means the responses are more deterministic. Changing the temperature can have a large impact on the output and needs to be carefully chosen for the problem at hand.

**Tokens:** A model understands and produces text by tokenizing or dividing it into smaller chunks of text called tokens. For this the LLM has a fixed list (a *vocabulary*) of the different tokens it can handle, and it decomposes any given text into tokens from its list. A token can be a single character, part of a word or a phrase. Generally speaking, a simple but naive tokenization method would be to regard every word as a token. This would be problematic for an LLM that may encounter hundreds of thousands of different words, each requiring a unique token and thus a very large vocabulary. The opposite naive approach of one token corresponding to one character results in a compact vocabulary, but the text decomposition becomes very inefficient, with each word using many tokens. In practice tokens therefore tend to have some superficial similarity to syllables, with a few letters each, striking a balance between vocabulary size and decomposition. Another advantage is that tokens of this size can carry their own semantics, enabling the model to learn connections between related words that share tokens. Multiple different types of tokenizations exist, for instance the GPT models and the Llama models have different vocabularies of tokens. Each model has a maximum limit of how many tokens it can take as input, where the limit depends on the construction of the model. For example, GPT-4 has a token limit of 8,192, which roughly corresponds to 6,000 English words. LLMs will typically simply ignore all input beyond this limit, reducing their usefulness regarding long texts and books, unless such texts can be split into chunks which the LLM can read one by one.

**Prompts:** When using a large language model, the user must provide some textual input for the model, which the model uses as a basis for generating more text. The text used as input is called the prompt and is typically a question or an instruction. Apart from the request authored by the user, the input to the model often also includes a so-called *system prompt* that is inserted au-

tomatically. The system prompt tells the model in what manner it should give its response. This can include instructions to not be racist or sexist, or more task-specific requests to modify the writing style, e.g., to respond in a casual manner or like an academic expert. System prompts are typically inserted automatically by the overarching framework and not visible to the user, but this division is merely a practical convention; for the LLM a system prompt is a part of the prompt.

**Fine-tuning:** Creating an entirely new LLM from scratch is prohibitively expensive regarding computational resources, as the neural network must be trained on millions of texts. Thus it has become common to use publicly available LLMs instead. Such models can then be fine-tuned: They are trained further with additional data, to make them better at a specific task or to teach them the expressions and style of some specialized domain. Fine-tuning is considerably less complex than the initial training which created the LLM.

**Epochs and checkpoints:** Epochs and checkpoints are measurements of how long the model has been trained. In all types of machine learning, how much the model should be trained needs to be considered. With too little training, there is a risk of the model not learning the patterns in the data, decreasing its performance later (e.g., a spam classifier not recognizing spam mails, or an LLM incapable of understanding synonyms). On the other hand, with too much training the risk is overfitting the model to the training data, that is, the model memorizes the correct responses for the training data, but loses the flexibility to handle new data. When an epoch is completed, the model has been trained on the dataset once. Since it is difficult to know beforehand when the optimal amount of training has been done, checkpointing can be employed. Checkpointing means that versions of the model are saved throughout the training process, at certain checkpoints. The different versions of the model can be retrieved after training, and then compared to find the best option.

**Quantization:** Quantization can be used when the computational power at hand is insufficient to run a certain model. It involves reducing the precision of the numerical weights in a model, e.g., from 32 bit floating point numbers to 16 bit. Through quantization, the model size can be reduced, often without losing much performance. Most commonly, the quantization is done on an already trained model, transforming it to a lower precision format.

**Reinforcement learning from human feedback:** Reinforcement learning from human feedback is a method to align a machine learning model with human preferences. It can be used to tweak the model to a specific person or situation, or more generally make it resemble human communication, depending on how it is trained. In reinforcement learning, the model is seen as an agent that aims to maximize its reward function in a given environment through trial and error at the start, but over time learning behavior. By adding human feedback, the model learns from human input and reactions. Human feedback is costly and large language models need a lot of data to be pre-trained, which means that the strategy is mainly just used to fine-tune a base model.

**Language:** Large language models can be trained on a single language or on multiple languages. Since English is a global language and there exists a large amount of English text data, most of the well-known models are mainly trained on English text. However, models specifically trained for other languages exist as well. A model is limited to the languages it has been trained

on, and it is also affected by how much it has been trained on each language. This can lead to a model having good performance in English, but poor performance in, e.g., Swedish. If the choice of language is an important part of the problem at hand, it is important to take this into consideration when choosing a model.

**License:** The creators of each model decide what is contained in the accompanying license. The license describes what a user is allowed to do and not do with the model. For instance, most licenses include that the model cannot be used to inflict harm on people.

## 2.3 Optimizing Training - PEFT and LoRA

As an introductory remark, this section goes into the theoretical and technical background of a state-of-the-art optimization method that is crucial for training and operating LLMs with limited resources. Readers who are less interested in this level of detail may prefer to skip the section.

Training a large language model from scratch is an intensive process requiring considerable computational resources and extensive datasets. In contrast, fine-tuning an existing LLM is a more viable option when adapting to a new task, requiring less data to make significant task-specific improvements. However, the inherent size of LLMs means that even fine-tuning is a data-intensive endeavour. Full model fine-tuning of an LLM involves updating all the parameters of the model based on a specific dataset or task. While this can be effective in adapting the model to particular needs, it comes with several drawbacks and risks:

- **Catastrophic forgetting:** Fine-tuning all weights on a new task can drastically disrupt the information flow of the model, or in other words, the model is more likely to forget previous knowledge and behavior.
- **Memory Requirements:** Training/fine-tuning an LLM is often done on GPUs/TPUs since it is significantly more efficient in terms of speed, effect and cost. However, one significant limitation is the on-board storage of the GPU(s), especially when it comes to training LLMs, which require significant amounts of memory.
- **Overfitting:** Fine-tuning a model on a small dataset can increase the risk of overfitting. In such cases, the model may excel at tasks specific to that dataset but struggle to generalize to new data.
- **Storage and Deployment:** In some cases it is useful to fine-tune multiple models on different tasks. However, each fully fine-tuned model is essentially a new model that needs to be stored and managed. This can be cumbersome and can lead to significant storage costs.

To mitigate these challenges, Parameter-Efficient Fine-Tuning (PEFT) methods have emerged, offering more efficient ways to adapt LLMs without the need to overhaul the entire model. PEFT strategies aim to optimize a small number of (extra) model parameters while freezing most (or all) of the model's pre-trained weights, significantly reducing memory, computation and complexity.

One prominent PEFT method is Low-Rank Adaptation (LoRA)[18], which employs an innovative approach to significantly reduce the number of trainable parameters while maintaining competitive performance and addressing the aforementioned drawbacks.

The concept of LoRA is predicated on the assumption that the knowledge encapsulated within a LLM possesses a lower *intrinsic dimension* than its typical representation [19]. Based on this assumption, LoRA opts not to fine-tune



a layer's weights  $W \in \mathbb{R}^{k \times d}$  in their full rank. Rather, it represents the weight update  $dW \in \mathbb{R}^{k \times d}$  as a low-rank decomposition  $dW = BA$ , with the matrices  $A \in \mathbb{R}^{r \times d}$  and  $B \in \mathbb{R}^{k \times r}$  (see Figure 2.3). By setting  $r \ll \min(d, k)$ , the number of trainable weights is significantly reduced compared to the default weight updates.

During training, the layer weights  $\hat{W}$  are expressed as the sum of the default weights and the updated weight decomposition:  $\hat{W} = W + dW = W + BA$ . The forward pass is then defined by

$$h = Wx = W_0x + dWx = W_0x + BAx, \quad (2.1)$$

where  $x$  and  $h$  represent the layer input and output, respectively. After training, the learned decomposition  $BA$  is simply added to the default weights  $W_0$  to update the model weights, yielding  $\hat{W} = W_0 + BA$ , without introducing additional latency.

Advantages of LoRA include:

- **Efficient Parameter Update:** By leveraging low-rank matrices to adjust the model's weights, LoRA achieves substantial modifications to  $W$  without the need to train the full rank of weights. This method markedly reduces memory and computational demands while also alleviating some model-fitting issues, enabling more adaptable adjustments to suit various hardware capacities and model-fitting constraints. Compare Figure 2.1 and Figure 2.2.
- **Mitigating Overfitting:** LoRA's methodology is particularly advantageous when fine-tuning on limited datasets, as it minimizes the risk of overfitting that is prevalent in more aggressive fine-tuning methods. By enabling precise, targeted updates, LoRA ensures that the model doesn't acquire excessive degrees of freedom, which helps in preventing overfitting.
- **Leveraging Pre-trained Knowledge:** LoRA uniquely capitalizes on the existing strengths of the pre-trained model, fine-tuning it in a way that builds upon its prior training. This aspect is invaluable in Few-Shot Learning scenarios, aiming to maximize insights gained from sparse examples.
- **Selective Application and Generalization:** LoRA can be selectively applied to specific layers, such as attention layers, enabling focused and effective updates.
- **No Additional Inference Latency:** A significant benefit of utilizing LoRA, particularly when contrasted with other PEFT methods, lies in its post-training inference efficiency. Upon training completion, the model pre-computes and retains  $\hat{W} = W_0 + BA$ , facilitating standard inference processes without additional computations see Figure 2.3 and Figure 2.4. This efficient management of weight adjustments ensures that the LoRA-enhanced model does not experience any increase in inference latency nor memory, a stark contrast to some other methods that may incur overhead during this phase.
- **Task-Specific Modularity:** One central application of LoRA is its ability to maintain distinct sets of decomposition weights ( $BA$ ), each fine-tuned for specialized tasks. This feature allows for exceptional modularity, where one can promptly interchange the LoRA weights within the same foundational model to suit different applications or requirements. Further advancements in S-LoRA [20] have been made to significantly increase throughput and number of served LoRA-adapters. Notably, S-LoRA introduces key improvements to manage the GPU memory, which

helps to mitigate memory fragmentation. This enhancement is crucial for maintaining high efficiency and performance when serving multiple adapters concurrently.

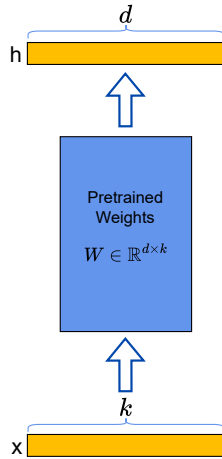


Figure 2.1: **Full Model Training:** During training, the forward pass is defined by simply multiplying the input  $x$  with the layer weights  $W$ :  
 $h = Wx$ .

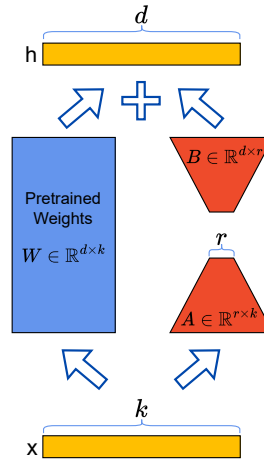


Figure 2.2: **LoRA Training:** During training, LoRA performs two forward passes, one with the default weights and one with the LoRA weights, and finally adding the two together:  
 $h = Wx = W_0x + \Delta Wx = W_0x + BAx$ .

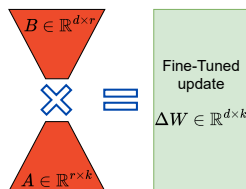


Figure 2.3: **Decomposition Updates:** Multiplying  $A$  and  $B$  results in the full layer weights update:  $\Delta W = BA$ .

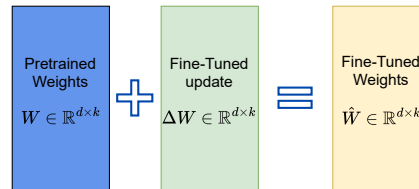


Figure 2.4: **New Fine-tuned Weights:** After training, One can simply add the weight-updates to the default weights to get new trained weights:  $\hat{W} = W_0 + \Delta W$

## 2.4 Challenges in Evaluation of LLMs

Evaluating generative language models presents several challenges. One of the main challenges is defining success criteria and determining what the evaluation metrics are. Success can be defined differently depending on the target application (e.g., understanding the prompt, accuracy in the retrieved information, grammatical correctness, structure, and style). Defining clear and relevant metrics depending on the end-user requirements is crucial.

Another main challenge for language models and how to evaluate them is that language is inherently ambiguous and subjective. LLMs must often deal with unclear or context-dependent situations where there might not be a single correct answer. This makes it difficult to objectively assess responses.

Since the early days of artificial intelligence in the 1950s, philosophers and AI researchers have believed that commonsense or genuine intelligence is required to disambiguate natural language [21]. In 1972, Terry Winograd in his

doctoral thesis [22] presented a well-known example of ambiguity in natural language. It consists of a pair of sentences:

1. “The city councilmen refused the demonstrators a permit because they *feared* violence.”
2. “The city councilmen refused the demonstrators a permit because they *advocated* violence.”

Disambiguating the pronoun “they” in these two sentences presents a challenge for a program, a difficulty that is generally absent for humans. For a human reader, it is obvious that fearing violence would be a reason for the councilmen to deny demonstrators a permit, and infers that the ambiguous “they” is naturally referring to the councilmen in the first sentence and to the demonstrators in the second sentence.

Until recently, it seemed reasonable to anticipate that AI would require some kind of commonsense knowledge to be able to do this kind of disambiguation. Based on this insight, Levesque et al., [23] proposed using these kinds of sentences as a test to measure the depth of natural language understanding of AI programs and defined the Winograd Schema Challenge (WSC) as a pair of two sentences similar to Winograd’s example. They presented it as an alternative to the Turing Test with conceptual and practical advantages over it. They compiled a collection of questions, designed so that the correct answer is obvious to the human reader but cannot easily be found using a rule-based program or statistical techniques over text corpora. However, by 2023, the original authors considered that the challenge has been defeated by LLMs [21, 24]. WSC is part of different natural language processing benchmarks such as the General Language Understanding Evaluation (GLUE) [25] and its successor SuperGLUE [26].

Even though as of now (March 2024), fundamental LLMs achieve high scores on WSC, achieving perfect accuracy on unseen data remains a challenge, especially on questions that require nuanced understanding of human experiences, context, or ambiguous phrasing. This becomes even more challenging in other languages than English (e.g., Swedish), as different languages have unique linguistic structures and cultural undertones. Currently, over one hundred benchmarks have been developed to test the commonsense knowledge and reasoning abilities of LLMs. However, many aspects of common sense still remain untested [27].

The evaluation of LLMs has evolved significantly in recent years, using a wide range of methodologies and benchmarks to assess their capabilities in different dimensions. An illustrative example is the Beyond the Imitation Game Benchmark (BIG-bench) [28] initiative. It is an ambitious effort to quantify the capabilities of language models, and a response to the fact that existing benchmarks suffer from several limitations. As of now, BIG-bench comprises 204 diverse tasks, contributed by 450 authors from 132 institutions. These tasks cover a wide range of topics, including traditional NLP tasks, logic, mathematics, code, understanding the world, understanding humans, scientific and technical understanding, as well as non-English and multilingual capabilities.

## 3 Creating LLMs for Defense

As per the first objective of this report, this chapter sets out to create large language models for the defense domain (Section 1.1). This is an involved process of numerous steps and critical decisions, and the chapter can serve as a guide to readers who intend to implement a similar LLM project.

The LLMs created in this chapter will be based on off-the-shelf base LLMs that are publicly available. They will be fine-tuned to adapt them to work in a Swedish defense domain. Specifically, and without loss of generality, the defense domain of choice in this chapter is that of security policy. Focusing on a limited domain like this means that less training data is required to cover the domain, and it enables a more thorough control over the quality of the training data, two factors contributing to a more effective training for the intended domain.

The application scenario for the LLMs is to support a security policy analyst working in the Swedish defense sector, for example at FOI. The models will be trained to meet the following requirements:

- The LLMs must support both Swedish and English.
- They should be capable of generating text in the style of a Swedish security policy analyst.
- They must follow instructions, that is, perform textual tasks as directed by the analyst user.

As such it is necessary that the trained models are instruction-tuned models. The exact nature of these instructions is not defined further here. Plausible tasks for such an LLM would be, among others, to summarize texts, to answer questions about texts and to generate texts based on keywords. However, at this experimental stage the training will mainly focus on improving the general language capability for the target domain. Countless different tasks could be envisioned, and task-specific dedicated models with specialized training would likely outperform a more generalist LLM, but such training is outside the scope of this report.

### 3.1 Method

The fine-tuning of a large language model is a powerful technique to customize a pre-trained base model to a specific domain or task. This process involves training the model on a smaller domain-specific dataset, allowing the model to adapt and refine its performance. Through this process, the model is adjusted to better capture the intricacies and nuances unique to the specified domain, resulting in improved accuracy and relevance for the intended application.

One key aspect of fine-tuning involves selecting an appropriate dataset that is representative of the language characteristics of the desired domain. Additionally, the dataset should be large enough to enable the model to capture the patterns and new characteristics of the domain but not so large that it becomes computationally infeasible.

Another crucial aspect to take into account is the hardware, as fine-tuning an LLM can be computationally intensive. The selection of both the base model and dataset should align with the available hardware resources to ensure efficient and practical implementation.

Hence, these aspects inform the creation process described in this chapter, and methodologically this process consists of the following steps:

1. The base model requirements are specified, and publicly available models are surveyed. Suitable base models are identified and selected (Section 3.2), taking into account the requirements and the hardware resources available to this study.
2. Text documents representing the intended domain are identified and retrieved. The documents are converted into a curated text dataset suitable as training data input for the chosen base models (Section 3.3).
3. The base models are trained on the prepared training dataset (Section 3.4).

The following section explores key considerations for a base model suitable for the fine-tuning into an LLM for the Swedish defense domain, specifically for security policy.

### 3.2 Base Models

The selection of an appropriate base model for fine-tuning involves considering several factors, including the type of the model, its size, and what languages it has been pre-trained for. The type of model refers to the architecture and design of the model. In the use case of creating a LLM for the defense domain the model is preferably a generative pre-trained model that also has been instruction-tuned. An advantage with using an already instruction-tuned base model is that the dataset for fine-tuning on the defense domain can be smaller, as it only needs to focus on representing the domain, not on also teaching instructions to the model.

The size of the base model influences the capability to capture language patterns and nuances in the data. For tasks or domains with simpler language patterns a smaller base model might be good enough, whereas a larger model might perform better for more complex tasks or domains. The choice of model size is also dependent on the available computational resources and time constraints.

The language of the base model is an important aspect to consider when creating an LLM that for example should understand Swedish. A base model that has not been pre-trained on any Swedish texts will have to learn any Swedish capabilities just from the training data supplied for fine-tuning. If the fine-tuning dataset is concise and focused on a specific domain, then the model will have little opportunity to gain a good general understanding of Swedish. On the other hand, with a large dataset that aims to improve both the general understanding of Swedish and the domain capabilities, the fine-tuning becomes very costly regarding time and resources. It is therefore much preferable to start with a base model that has already been pre-trained on a diverse dataset which includes Swedish or similar languages, as it will most likely be able to capture the domain specific language patterns better and with less data.

Thus a base model needs to meet several requirements to be suitable for fine-tuning on texts from the defense domain chosen for this report, that is, security policy:

- It must have some capabilities in both Swedish and English, as Swedish security policy analysts work and publish in both languages.
- There must be instruction-tuned variants, as instruction-tuning from scratch would exceed the resources for this report.
- There must be several size variants, allowing the model to be run on a broad range of hardware.
- The model must represent the state of the art at the time of this writing, as development in LLMs is fast and newer models usually outperform those from just a few months ago.

In the following sections, three base models will be investigated.

### 3.2.1 Llama 2

Llama 2 (*Large Language Model Meta AI*) is a family of pretrained and fine-tuned LLMs developed by Meta<sup>1</sup> in collaboration with Microsoft [29]. Meta released the models as open source with the intent to promote accessible, innovative, and responsible AI for both research and commercial purposes. By providing users access to the models, evaluation code, and documentation, Meta aimed to make Llama 2 “*accessible to individuals, creators, researchers, and businesses so they can experiment, innovate, and scale their ideas responsibly*”<sup>2</sup>. One significant aspect contributing to this goal is its minimal hardware requirement, enabling users to explore their ideas without extensive computing infrastructure. However, the Llama 2 licensing terms have been criticized for deviating from the strict standards of open-source licenses, so the terms should be evaluated with care when Llama 2 is to be deployed in an application.

The following models are available to the general public:

- **Llama 2**: an updated version of Llama 1, trained on a 40% larger text corpus and with the context length doubled to 4,096 tokens. The model is available with 7B, 13B and 70B parameters.
- **Llama 2-Chat**: a fine-tuned version of Llama 2, optimized for dialogue use cases. This can be regarded as the instruction-tuned version of Llama 2. The model was first supervised fine-tuned and then refined with Reinforcement Learning with Human Feedback (RLHF) methodologies. Also this model type is available with 7B, 13B and 70B parameters, which are named `Llama-2-7B-chat-hf`, `Llama-2-13B-chat-hf`, and `Llama-2-70B-chat-hf`.

The Llama 2 model with 70B parameters is designed to rival popular models like OpenAI’s ChatGPT-3.5, and outperforms the GPT-3.5 in most benchmarks, including generating safer<sup>3</sup> outputs and with higher performance [29], even though the GPT-3.5 model is trained on a larger text corpus (570GB of data) and uses 175B parameters.

### Limitations

The model is particularly adept at comprehension, reasoning, and general intelligence, but faces challenges in handling non-English languages due to its primarily English training data.

### 3.2.2 GPT-SW3

GPT-SW3<sup>4</sup> is a collection of large language models with a focus on Swedish. They have been developed by AI Sweden, RISE and WASP WARA Media & Language [30]. The models have been trained in a manner analogous to the GPT-series LLMs by OpenAI [11], but using a language corpus of approximately 100GB of texts mostly in Swedish, but also including other Scandinavian languages as well as English. Several variations are available to the public, notably:

<sup>1</sup><https://llama.meta.com/>

<sup>2</sup><https://llama.meta.com/llama2/>

<sup>3</sup>LLM *safety* is an important aspect in interaction with the public, e.g., for LLMs integrated in social media: The LLM should not generate abusive, hateful texts, or texts that could cause harm, such as bomb building instructions.

<sup>4</sup><https://www.ai.se/en/project/gpt-sw3>

- **GPT-SW3**: base models ranging from 126 million parameters in size up to 40 billion,
- **GPT-SW3-instruct**: instruction-tuned models ranging from 126 million to 20 billion parameters,
- **GPT-SW3-gptq**: quantized models with 6.7 billion and with 20 billion parameters.

The models carry a permissive license which effectively places no restrictions on usage.

### Limitations

GPT-SW3 is so far the largest LLM that has a strong focus on Swedish. However, several other multilingual LLMs have included Swedish texts in their training data and hence also possess Swedish language capabilities. Indeed, the creators of GPT-SW3 acknowledge that some other multilingual LLMs may outperform GPT-SW3 on Swedish [31]. A Swedish comparison between GPT-SW3 and Llama 2 revealed no clear advantage to either model.

Instruction models need to be trained on specific data in the form of instructions and dialogues. The current lack of such datasets for Swedish means that the instruction variants of GPT-SW3 had to be trained on English datasets that had been translated to Swedish by GPT-SW3 itself, introducing an aspect of uncertainty regarding the quality of this training.

### 3.2.3 Mistral

Mistral AI is a French artificial intelligence company. They describe themselves as being dedicated to the development of the open-source AI community.<sup>5</sup> The company offers AI products exclusively in the text domain, pre-training and instruction-tuning large language models. Two of its models can be accessed and downloaded in their entirety.<sup>6</sup> The other three are accessible only through their own API.

As the time of this writing, MistralAI offers the following large language models, in ascending order of model size:

- **Mistral-7B**: The smallest model, seven-billion parameters. It exists as a plain pre-trained model, and as an instruction-tuned version. Its usage adheres to the open Apache 2.0 licence. Throughout this report, the instruction tuned model is used, which is named **Mistral-7B-Instruct-v0.1**.
- **Mixtral-8x7B**: A larger LLM of approximately 45 billion parameters. Pre-trained and instruction tuned versions exists. It is highly capable for its size, challenging the benchmarks of famous ChatGPT-3.5. The name implies a *mixture of experts* model architecture, combining several smaller, distinct large language model neural nets, in this case eight Mistral-7B models, that work together to generate the model output [32]. Its usage also adheres to the open Apache 2.0 licence.
- **Mistral Proprietary Models**: As a complement to their open-source, downloadable models, MistralAI develops and hosts three models through their API:
  - Mistral-Small
  - Mistral-Medium
  - Mistral-Large

<sup>5</sup><https://mistral.ai/company/>

<sup>6</sup><https://huggingface.co/mistralai/>

The details of the proprietary models are not available. Their capabilities can only be discussed through benchmarks on Mistral AI's website <sup>7</sup>, as well as its placement in the *LMSYS Chatbot Arena Leaderboard*<sup>8</sup>. However, according to benchmarks, the proprietary models listed are consistently outperforming the open-source alternatives MistralAI are offering.

It is worth mentioning that even though Mistral AI have an outspoken dedication towards open-source AI, their explicit methodology for training their models, and more importantly the data that was used for that training, have not been disclosed. They provide some technical properties of the architecture, mostly consisting of high-level discussions about the transformer architecture. For instance, they mention usage of *Grouped Query Attention* [33] (GQA) as well as the *sliding window attention* [34] (SWA) which boosts the performance of the model by altering how the attention mechanism is operating<sup>9</sup>. However, these are known methods, and not unique for the Mistral AI team. Considering the information provided on the properties of the architecture of the model, it does not seem to differ from others on the open-source LLM market. The success of Mistral-7B model could be because of a finely-curated training data, which potentially outshines other open-source models in the LLM domain. In the Mistral-7B paper, it is explicitly stated that “No proprietary data or training tricks were utilized”, as well as “...we fine-tuned it on instruction datasets publicly available on the Hugging Face repository” [16]. This is only mentioned regarding the instruction fine-tuned model. Hence, Mistral-7B's benchmarking success could be attributed to higher quality, proprietary pre-training data (before instruction fine-tuning). However, there is currently no way of knowing the properties of their training data in detail, whether it be the pre-training dataset or the fine-tuning dataset.

Considering the model size of 7 billion parameters, which is on the smaller side regarding LLMs, Mistral is very efficient. It outperforms Llama 2-13B, a model almost twice the size, on many important benchmarks.

Mistral-7B supports a context-length of 8,192 tokens, which at the time of its release is quite high. It has a vocabulary size of 32,000.

## Limitations

Even though Mistral-7B has been tested thoroughly by the community and delivers high standards on multiple tasks, officially it does so exclusively on instructions in English. Nevertheless it appears that Mistral-7B has some capabilities when it comes to other languages such as Spanish, French and even Swedish to some degree.

### 3.2.4 Model Selection

All three of the LLMs above meet the basic criteria required for this study: They are available in suitable sizes, they have some capability for Swedish and English, and they have instruction model variants. However, time and hardware constraints meant that not all three models could be trained and evaluated fully, hence a selection was necessary. None of the models showed clear advantages regarding the general language capabilities, and the hardware requirements are similar for all as well.

<sup>7</sup><https://mistral.ai/news/la-plateforme/>

<sup>8</sup>ChatBot Arena is a leaderboard where AI models outputs are compared and scored accordingly to its ability to answer a question. It is a dynamic way for the open-source community to rate the capabilities of all text-based AI models. <https://huggingface.co/spaces/lmsys/chatbot-arena-leaderboard>

<sup>9</sup><https://mistral.ai/news/announcing-mistral-7b/>



The instruction-tuned model variants are of particular interest for the application test case of an LLM supporting a security policy analyst, as the models will often be expected to accurately follow specific instructions of the human analyst (e.g., to summarize a given text, or to answer a question about a text), rather than freely generate text.

This was tested in the following experiment: The three instruction-tuned LLMs were asked to extract a given number of key phrases from the texts in the FOI memos dataset (Section 3.3.3). The quality of the resulting lists of key phrases was then evaluated automatically (Table 3.1).

*Table 3.1: Comparison of key phrase listings generated by the instruction-tuned versions of GPT-SW3, Llama 2 and Mistral*

	GPT-SW3		Llama 2		Mistral	
	mean	$\sigma$	mean	$\sigma$	mean	$\sigma$
key phrase count difference	1.8	0.7	0.1	0.6	0.1	0.6
unique key phrase ratio	0.9	0.2	1.0	0.0	1.0	0.0
longest common substring	6.7	5.8	4.5	2.6	2.3	1.2

Notably, Llama 2 and Mistral adhered to the specified number of key phrases far more strictly than GPT-SW3, with their average number of generated key phrases only differing minimally from the requested amount. In contrast, almost half of the lists generated by GPT-SW3 had two more key phrases than requested, and some had less. Furthermore, approximately 10 percent of the key phrases generated by GPT-SW3 were duplicates, whereas Llama 2 and Mistral generated unique key phrases almost without exception. The frequency of duplicates in the output of GPT-SW3 is indicative of an overall worse “understanding” of the instruction, which may also be responsible for the overall lesser quality of the key phrases. This is difficult to express statistically, but a measurable aspect is the tendency of GPT-SW3 to produce highly formulaic key phrase lists, like  $k_1 = \text{“Russian defense”}$ ,  $k_2 = \text{“Russian politics”}$ ,  $k_3 = \text{“Russian territory”}$ . To measure this phenomenon, the average overlap between unique key phrases was computed per list: For each combination of phrase pairs the longest common substring (sequence of characters) was identified, and the mean of their lengths was computed. In the example, the pair  $k_1, k_2$  has “Russian ” as the longest common substring, with a length of 8 (7 characters and one space). The same is true for the other pairings, resulting in an overall average of 8 for this key phrase list. The table shows the mean of these computations over all the lists. GPT-SW3 exhibits the most overlap between its key phrases using this method. The phrases generated by Llama 2 and Mistral are more distinct, making them more informative and closer to lists a human author would compile.

Given this comparison and the importance of the aspect of instruction-tuning, the choice was made to continue the study without GPT-SW3. Both Llama 2 and Mistral otherwise exhibited sufficiently similar performance in preliminary testing that neither would have been a clear choice over the other. Thus both Llama 2 and Mistral were selected for the remainder of this study.

It needs to be emphasized that GPT-SW3 was not included in the remainder of the report work primarily because of the instruction performance. This type of performance was considered critical, both for the application scenario of supporting a security policy analyst and for the experiments. GPT-SW3 may well outperform the chosen models in other situations, but that kind of testing was outside the scope of this report. Furthermore, GPT-SW3 has

undergone numerous version changes during the course of this work, indicating rapid development of the model. Future studies will include GPT-SW3 when appropriate.

LLMs can require considerable hardware resources to operate, and as described above they are usually released in different sizes, allowing users to select a version they can handle. Training and fine-tuning an LLM is more costly in this regard than using the model for text generation once it has been trained. Thus it is plausible to utilize a powerful computer for training, and then deploy the LLMs for their intended purpose on lesser machines that are just sufficient to run the model.

For the report experiments the instruction-tuned model versions with 7 billion parameters were chosen, i.e., **Llama2-7B-chat** and **Mistral-7B-instruct**. Generally speaking, this is the largest size at which an LLM can be used for text generation on a workstation or desktop computer equipped with a high-end consumer GPU (e.g., Nvidia RTX 4090). Staying within this limit simplified the experiments, as the greater availability of such computers enabled individual authors to carry out different text generation experiments simultaneously. Note that the training of a 7B model requires more powerful hardware than this, though.

### 3.3 Training Data

To train an LLM for supporting a security policy analyst, it needs to be fine-tuned on large amounts of suitable texts. The FOI unit for Security Policy and Strategic Studies regularly publishes analyses and investigations on issues of domestic and international security policy. The content of these publications is thus highly relevant for the defense domain chosen for the experiments in this report. Many of these studies are published as so-called *memos*, short documents with a relatively uniform structure. 244 of these memos were retrieved from FOI's library server. They had been published between 2010 and 2023, and 158 of them are written in Swedish (65%), while the remaining 86 are in English (35%). The memos have approximately 6 pages each (median value). The compact size of these documents ensure that the LLMs with their limit on input tokens are still capable of processing a significant proportion of each document, sometimes entire documents. This simplifies the training: If the documents were considerably longer, it might be necessary to train specialized LLMs dedicated to each typical section, e.g., one model for the introduction, one for analysis and so on. With the short memos on the other hand, the entire structure is accessible to the model, or can be made accessible with minor abridging. As the memos were written by the same unit within FOI, they share a common jargon and structure, with many even following a common template. This increases the likelihood of an LLM adopting this vocabulary and style during fine-tuning, essentially learning what a security policy memo is "supposed to look like."

#### 3.3.1 Failure of Automated Text Extraction

The FOI memos on security policy, which form the basis for the LLM training dataset, were retrieved from the FOI library server in the form of PDF files. PDF (*Portable Document Format*) was developed by the Adobe corporation in 1992, became the open standard ISO 32000 in 2008, and it is an ubiquitous file format used for the distribution of documents. Unlike common editable file formats, e.g., plain text files (*.txt*) or Microsoft Word documents (*.doc*, *.docx*), PDF is for all intents and purposes immutable. The format defines the exact

layout of text and other elements on each page, ensuring that a document will look the same when being viewed on a wide range of viewer applications, web browsers, printed to paper etc. This makes PDF attractive for document dissemination, and it is a common workflow to first create documents in Word or L<sup>A</sup>T<sub>E</sub>X and then convert them to PDF for distribution.

Internally, PDF positions text and other elements with *point*<sup>10</sup> precision, making each page more akin to a picture than the line-based character sequences of other formats. Text itself is stored in font-based content streams, with hard-coded line-breaks and other formatting added to ensure an exact layout. PDF contains very little meta-data to structure the text semantically, instead the human reader is expected to understand the intended structure via visual cues of the layout. As an unfortunate consequence there is no generally effective automatic way to reverse the conversion and extract the text from a PDF file into a machine-readable form suitable for LLM training.

There are numerous rudimentary methods that attempt to extract PDF text to a text file automatically, including a functionality to save to text within Adobe Reader. However, all introduce numerous problems in the resulting text that render the output unsuitable for LLM training. For example, an extracted text will preserve all line-breaks from the PDF, as there is no clear method of determining whether a given line-break was explicitly introduced by the author (e.g., to start a new paragraph) or added automatically during the PDF creation to prevent a line from exceeding the page width.

Python libraries like `pdfminer` and `pdfplumber` allow the development of more sophisticated extraction scripts. Such a script was developed during the work on this report. After several iterations it had to be considered a failure. Several issues could not be solved in an adequate manner:

- Page headers and footers: A naive implementation will extract unwanted header and footer text, such as page numbers. There is no reliable way to exclude such text, as PDF does not distinguish it from other text. The script attempts to identify headers and footers by text density and whitespace in the upper and lower areas of pages, but footnotes, large images and blank pages introduce frequent exceptions that disturb the detection.
- Formulas, lists and tables: Such non-standard text elements are difficult to reduce to plain text, especially when they take more complex forms (e.g., nested numbered and bullet-point lists, or tables with variable cell dimensions). Simply ignoring them may leave references in the other text dangling.
- Footnotes: Plain text is not divided into pages, so there is no obvious position for footnote text after an extraction. Furthermore, distinguishing footnote text from other text is non-trivial, given that there are different ways of denoting footnotes. Finally, the flags of footnotes in the main text are difficult to identify, e.g., superscripted numbers as footnote flags are technically no different from exponents in mathematical expressions.
- Two columns: Text in two columns is common in scientific reports. PDF provides no clear internal indicator that distinguishes two columns from a single column. A naive text extraction will treat two columns as a single wide column, effectively alternating between the two columns and merging their lines in the wrong order. The recommended workaround is to test whether there is a narrow area of whitespace stretching along the central vertical axis of the page, and if so, extract text from the left

---

<sup>10</sup>PDF measures dimensions in *points*, where one point is 1/72 of an inch.

half of the page first, followed by the right half. This method can provide decent results, but is easily disturbed when elements (pictures, tables) on the page stretch over the full width, thereby impinging on the central whitespace zone and preventing a correct identification of two column text.

PDF is geared towards human visual cognition and not intended to support algorithmic automation. Any apparent extraction rule is bound to have numerous and nested exceptions, and their treatment requires *point*-level coordinate comparisons and calculations. While some effort was expended on improving the automated extraction, even the best version tended to result in texts that appeared superficially correct, yet contained a multitude of errors which would require thorough proof-reading and complex manual comparisons and corrections. This type of effort was more protracted and error-prone than extracting the texts entirely by hand. Hence, it was decided that automated extraction from PDF was a sufficiently complex problem to be outside the scope of this report, and to ensure good quality training data all the collected FOI memos were instead converted by manual extraction in a group effort.

### 3.3.2 Manual Text Extraction

The PDF texts were manually converted to text files, meaning that the content of the PDF files was copied and pasted into a text editor (Notepad++). After processing, the texts were saved in text format (UTF-8 encoding). The following processing actions were undertaken:

- All figures, tables, fact boxes, and other paratextual features were removed.
- All references, including those to figures, tables, etc., were removed.
- All footnotes and footnote markers were removed.
- Page numbers were removed.
- The authors' names and affiliations (if any) were removed.
- The title was separated by a new line.
- Each paragraph was formatted as a single line, separated by a new line.
- Each section was preceded by a new line.
- Bulleted and numbered lists were formatted as a collection of single lines, where each line started with an asterisk character (\*) and ended with a new line.

The text files were saved in Unix style, meaning new lines were represented using the LF (Line Feed) character.

The reports were not converted in their full length, as the limitation of input to 4,096 tokens made this unnecessary. However, efforts have been made to include the final words of the report, or at least parts of them, in the converted text file. The purpose of this was to avoid having input files consisting of articles that end abruptly, but instead have a form of clear structure.

### 3.3.3 Dataset

The processed PDF file yielded 244 text files, of which 86 are in English and the remaining 158 are in Swedish. Tables 3.2, 3.3 and 3.4 provide statistics on the number of paragraphs, words, and characters in these files. Table 3.2 presents statistics for all text files, while Tables 3.3 and 3.4 provide statistics for English and Swedish text files, respectively. Comparing the median and mean in Tables 3.3 and 3.4 reveals that Swedish text files are larger and contain more text. The standard deviation for paragraphs, words, and characters in Swedish

files is also larger however, indicating greater variance in the size of these files.

*Table 3.2: Statistics on the number of paragraphs, words, and characters in the 244 obtained text files.*

	total	min	max	median	mean	$\sigma$
paragraphs	12625	10	363	43.5	51.7	39.9
words	703946	342	13287	2482	2885.0	1749.9
characters	4953594	2707	97090	17800	20301.6	12708.3

Comparing the column labeled ‘total’ in Tables 3.3 and 3.4 shows that although the number of Swedish files is 84% higher, the total number of Swedish words and characters is 112% and 140% higher, respectively. This suggests that Swedish articles are not only more numerous but also larger in size, and each Swedish word has a higher number of characters.

*Table 3.3: Statistics on the number of paragraphs, words, and characters in the 86 obtained English text files.*

	total	min	max	median	mean	$\sigma$
paragraphs	3594	10	157	36	41.8	30.1
words	225548	945	8315	2154.5	2622.7	1638.4
characters	1455167	6328	51932	14108	16920.6	10596.1

*Table 3.4: Statistics on the number of paragraphs, words, and characters in the 158 obtained Swedish text files.*

	total	min	max	median	mean	$\sigma$
paragraphs	9031	12	363	46,5	57.2	43.4
words	478398	342	13287	2542	3027.8	1791.6
characters	3498427	2707	97090	18500	22141.9	13367.2

The number of unique words in the Swedish text files is 35,218, compared to 14,487 in the English text files. This count includes variations in word forms, compound words, and proper nouns. The difference might be attributed to the fact that Swedish is a morphologically richer language compared to English; that is, it has a richer variety of word formations. For instance, an ordinary Swedish verb has more than ten different form variations [35]. The Swedish language also has more variations in noun forms, owing to its use of gendered nouns and distinct definite and indefinite forms. These linguistic features increase the diversity of word forms and the number of unique words in Swedish texts. For instance, consider the Swedish infinitive verb *att fastställa*; the Swedish reports contain eleven different derivatives of this word, namely *fastställa*, *fastställandet*, *fastställas*, *fastställd*, *fastställda*, *fastställde*, *fastställdes*, *fastställer*, *fastställs*, *fastställt* and *fastställts*. The number of English variations of this word would be just four, that is, *establish*, *establishment*, *established* and *establishes*, with possibly some conjugations of the verb *to be*. Additionally, the authors of the memos are predominantly native Swedish speakers and presumably have a richer and more nuanced vocabulary in Swedish, which may result in a more diverse choice of words.

In contrast to FOI reports, memos lack a list of key phrases created by the authors. As one of the goals of this work is to create a model that can

generate memos given a list of key phrases, it was necessary to extract key phrases from the memos either manually or automatically using a language model. Extracting key phrases manually is both time-consuming and carries the risk of key phrases not having the same quality, depending on the reading skills of those who extract them. Therefore, the model Llama 2 was used to extract seven unique key phrases from each text file. If the text was in English, Llama 2 was prompted by the system message: “Your task is to summarize texts into key phrases. Keep the key phrases short, and do not generate more than 150 tokens.” and the instruction “Sure! Here is the short list of only seven key phrases, in English:”. If the text was in Swedish, Llama 2 was prompted by a corresponding system message and instruction in Swedish. Llama 2 was able to extract almost exactly seven unique key phrases from each text file. In rare cases, the number of key phrases was other than seven.

In the final dataset, the key phrases are stored alongside the manually curated texts. This allows training LLMs in different manners, e.g., only on the plain texts, to yield a model with an improved “understanding” of the domain and the jargon, or on the key phrases combined with the texts, resulting in a model capable of generating text based on key phrases provided by the user.

### 3.3.4 Discussion: Model Tokenizers

Large language models process natural language text as input. However, an LLM does not take raw text as its input directly: Before the model receives an input, the text is transformed into a series of integers known as *tokens*. These tokens are essentially fundamental building blocks to build sentences, but in varying degrees of granularity. Depending on the strategy, the token-chunks can represent characters, sub-words, or even complete words. Tokens are designed to help capture the complex relations of natural language as they add flexibility when constructing words and sentences.

The process of transforming raw text to tokens is referred to as *tokenization*, which is done by an additional model called the *tokenizer*. The tokenizer has been optimized to serve one or several large language models. This means, to create an optimal LLM, one needs a tokenizer that has been designed to support it. As an example, the family of Llama 2 LLMs from Meta (7, 13 and 70 billion parameter versions), all share the same tokenizer.

Below is an example on how a tokenizer could potentially deconstruct a sentence into smaller parts of tokens:

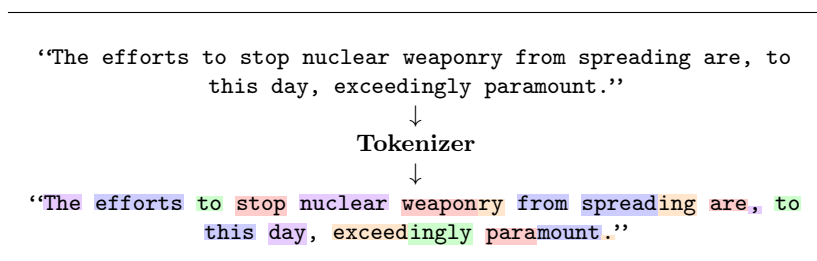


Figure 3.1: Example of tokenization of a sentence. The highlighting of text with one color represents one arbitrary token. If one word has three different colors, it is constructed by three tokens. Most words are directly linked to the tokenizers vocabulary, and they are not deconstructed into smaller parts. But words such as *paramount* are split into two parts, each part having its own token representation. This example was made using the tokenizer from the Llama 2 family, by Meta.

There are several design-choices to be made when creating a tokenizer.

These choices are connected to the desired properties of the LLM. For instance, a common design parameter of the tokenizer is *vocabulary size*. This refers to the amount of tokens, as in the number of building blocks, that the tokenizer has at its disposal to deconstruct text. The amount of tokens in the vocabulary is an important design feature, and the choice should mirror the capabilities that are sought in the large language model.

For instance, the respective tokenizers used for Llama 2 and Mistral both use a vocabulary of 32,000. However, the creators of GPT-SW3, a multilingual model with a focus on Swedish, chose a vocabulary size of 64,000, stating the importance of a higher number of building blocks for its multilingual tokenizer and LLM capabilities [31].

Hence, each tokenizer is designed to encompass one or several languages in its vocabulary of tokens. In the case of GPT-SW3, it aims to support Swedish, Norwegian, Danish, Icelandic and English. Naturally, its tokenizer must support tokens that can build sentences in all of the named languages, and therefore hosts a larger vocabulary than that of Mistral or Llama 2.

Given that Mistral and Llama 2 are both models designed to excel in English, their tokenizers are not explicitly designed to support several languages. However, it is important to note that even though one tokenizer and by extension the LLM is not designed to excel in more than one language, they still can and do show good capabilities when prompted outside of their English sphere. If tokenizers are optimized well enough, they still have the necessary building blocks to deconstruct sentences in other languages by simply using smaller chunks of tokens that are available in the vocabulary. This could however potentially prove to be a challenge for the LLM. Processing and prediction of the next token will require several more steps to produce a complete sentence, if the vocabulary is lacking, making the probability of linguistic errors higher.

### Evaluation of Tokenizers for the FOI Dataset

During pre-training and later fine-tuning of an LLM, the dataset of texts is often translated into token form. Since LLMs use tokens as input and not raw text, the number of tokens in a dataset becomes a natural measurement of its size.

To further reason about the tokenization of the FOI memos dataset, the tokenizers of Llama 2, Mistral and GPT-SW3 are each used to tokenize the dataset. Table 3.5 and 3.6 show the total amount of tokens as well as the average amount of tokens needed to construct the average word in the FOI memo dataset. The Swedish capabilities of the Mistral and Llama 2 tokenizers are showcased using the multilingual tokenizer of GPT-SW3 as a reference, to see the differences in vocabulary:

Total amount of Tokens in FOI Memos			
FOI Memos	Tokenizer		
	Llama 2	Mistral	GPT-SW3
English	0.34M	0.33M	0.31M
Swedish	1.11M	1.15M	0.69M
Eng. & Swe.	1.45M	1.48M	1.00M

Table 3.5: The total amount of tokens (in millions) in the dataset of FOI Memos (86 English, 158 Swedish), comparing the three different tokenizers from respective models. The amount tokens needed to deconstruct the dataset depends on the model tokenizer's vocabulary.

Average amount of Tokens per Word			
FOI Memos	Tokenizer		
	Llama 2	Mistral	GPT-SW3
English	1.50	1.44	1.38
Swedish	2.33	2.40	1.45
Eng. & Swe.	1.91	1.92	1.42

Table 3.6: Average amount of tokens needed to construct each word in the dataset of FOI Memos (86 English, 158 Swedish), comparing the three different tokenizers from respective models. The average tokens needed to construct a word is a direct consequence of the model tokenizer's vocabulary.

Table 3.5 shows the amount of tokens in the acquired dataset of FOI memos in security policies, per language and as a whole. Naturally, the Mistral and Llama 2 model tokenizers do not have Swedish words in their vocabulary, and in total must use a lot more tokens to build sentences in the Swedish FOI memos than the tokenizer of GPT-SW3 (more than 1.1 million tokens for Mistral and Llama 2, compared to 0.69 million for GPT-SW3).

To further this perspective, Table 3.6 shows the average amount of tokens needed to construct each word in the FOI Memos. Again, we can see that Mistral and Llama 2 require roughly one additional token to construct a word compared to GPT-SW3, in Swedish. GPT-SW3 with its larger, multilingual vocabulary, needs less tokens to construct the dataset on average in both Swedish and English.

To visualize this, Figure 3.2 shows how the different models tokenize a Swedish sentence: Naturally, GPT-SW3 has twice the vocabulary size com-

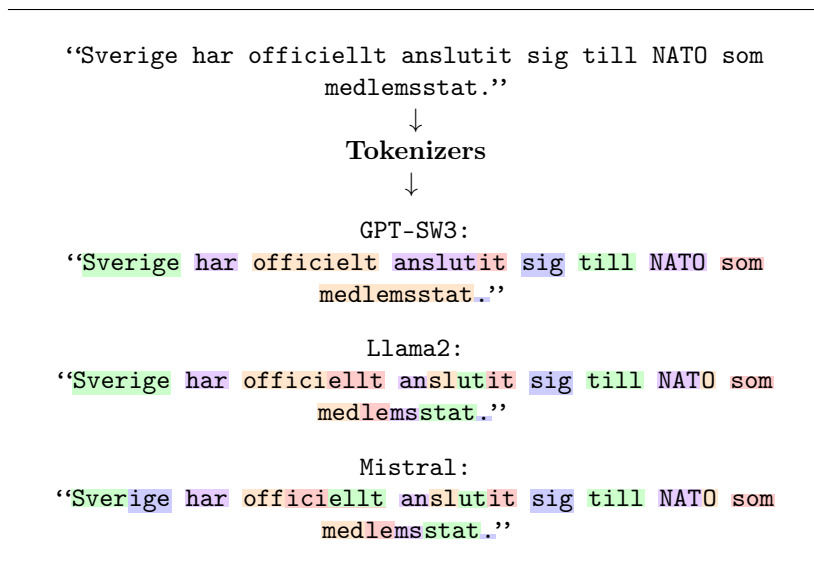


Figure 3.2: Example tokenization of a Swedish sentence, highlighting the difference of each model's vocabulary. The highlighting of text with one color represents one token. If one word has three different colors, it is constructed by three tokens. Following the average token per word from Table 3.6, Mistral and Llama 2 need significantly more tokens to construct a Swedish sentence, on average, compared to GPT-SW3.

pared to Mistral and Llama 2, and is optimized for Swedish; the result of tokenization as shown in Figure 3.2 is expected. However, considering the



possibility of building an LLM assistant for defense purposes, the capabilities of the tokenizer in the applied language should be noted when choosing what model to work with.

Given the nature of the tokenizers in LLMs and the languages they are optimized for, there are potential drawbacks of choosing a model that does not naturally support the language of the applied area. As seen in Table 3.6 and Figure 3.1, if an English model is applied in an Swedish setting, it will require more tokens on average to build the same sentence. Practically, this means that an English LLM needs more steps of predicting the next token to build a Swedish sentence. This could potentially lead to an increased risk of errors, since the model naturally will have to choose from a larger distribution of tokens to build the sentence.

However, it is possible, to a certain degree, for English models to use Swedish in a narrow field of application such as this, with a limited dataset, with only a few linguistic flaws. Furthermore, it could also be the reasonable choice to have an English model fine-tuned to be applied in Swedish context if the model in question is better at following instructions than the other choices available.

To conclude the tokenizer discussion, it is not apparent to what degree a smaller vocabulary affects the performance of a model fine-tuned on an additional language. When designing an assistant, however, one should keep in mind that there could be potential benefits in either choosing a model that naturally supports the vocabulary of the domain, or in altering the token vocabulary of the existing tokenizer by adding new tokens. Altering the tokenizer's vocabulary is not a common practice when fine-tuning, but could potentially be beneficial when adapting to a new domain with a jargon.

### 3.4 Training

As an introductory remark, the following sections describe the training in comprehensive technical detail. Readers who do not wish to delve more deeply into this particular topic may prefer to skip ahead to Chapter 4.

As per Section 3.2.4, Llama 2 and Mistral were chosen for fine-tuning and experimentation, using the training dataset based on FOI memos on security policy, see Section 3.3.3. Training a large language model (LLM) from scratch requires significant computational resources and vast datasets. In contrast, fine-tuning an existing LLM presents a more feasible alternative, particularly when employing a parameter-efficient method like Low-Rank Adaptation (LoRA), see Section 2.3. Nonetheless, the substantial size of LLMs ensures that fine-tuning remains a data-intensive task, underscoring the need to address specific constraints meticulously.

In this study there were two principal constraints related to the availability of training data and hardware resources:

- **Training Data:** The training dataset is limited, with a total size of approximately 5MB (Section 3.3.3), highlighting the necessity for an approach that can effectively fine-tune the model with limited data, without overfitting.
- **Training Hardware:** The hardware capacity available to this study, featuring 320GB of GPU memory for training, and 24GB for inference (enabling the testing of the trained models on more readily available computers). These specifications constrain the model size and explain the importance of an efficient fine-tuning process.

The primary objective of fine-tuning in this context is to modify the LLM's

behavior to more closely resemble that of a security policy analyst. This adjustment aims to enhance the model's ability to summarize text, respond to questions, and generate new content based on prompted assignments, with the fine-tuning restricted to the domain knowledge contained in a relatively small dataset of memos. Given this goal, adapting the model with fewer adjustable parameters (degrees of freedom) may suffice, particularly when it is not necessary to embed all information of the dataset within the model, but rather the textual behavior. Consequently, employing LoRA with a minimal adapter size appears to offer adequate adaptability for the task at hand.

Given the preceding constraints and the specific objective of limited data and computational resources, LoRA was chosen as the most suitable state-of-the-art parameter-efficient fine-tuning method for this study.

### 3.4.1 Training Process

The set of hyperparameters that were used for all training rounds is provided in Table 3.7. The parameters were set through comparing the training loss after five training epochs, mostly modifying the LoRA variables to enable a reasonable loss graph.

These parameters apply to both of the LLMs selected for this study, that is, Mistral and Llama 2 (specifically, `Mistral-7B-instruct` and `Llama2-7B-chat`). The number of training epochs depends on data quality and desired result. A

Hyperparameters	
Parameter	Value
Epochs	8
Learning Rate	2e-4
Batch Size	1
Warmup Steps	100
Maximum Tokens	4096
LoRA Rank	16
LoRA Alpha	16
LoRA Dropout	0.1
LoRA Layers	[q,v,k,o]

Table 3.7: Training hyperparameters used for every training round and model type. LoRA-labeled parameters are exclusive to this format (LoRA) of training procedure, see section 2.3.

good starting point could be 4-5 epochs for fine-tuning language models. To get a good margin for testing, 8 epochs was set as the standard training length, and one model checkpoint was saved after each epoch. However, 8 epochs will most likely over-fit the model. The most utilized model checkpoint in the experiments came to be the models trained for 2 or 5 epochs.

The fine-tuning dataset consists of the 244 FOI memos on security policy (Section 3.3.3), which is quite a small training set.

Since the training data consists of FOI memos, that is, short texts with a clear structure and conclusions, each model was trained using `Batch Size=1` to induce the behavior of having a start and an ending in the reasoning of the model. If a model is prompted to write a text about a particular area, it should do so with a clear concluding remark - and not continue trailing off and starting another subject.

Given that the batch size is set as `Batch Size=1`, one epoch corresponds

to 244 training steps, resulting in 1,952 steps for a complete training round of 8 epochs. For reference, in total, roughly 4 GPU hours from an NVIDIA A100 (40GB VRAM) would be needed to train Mistral-7B or Llama2-7B for 8 epochs.

It is worth noting that the **Maximum Tokens** was set to 4,096 for both Llama2-7B and Mistral-7B, even though Mistral-7B supports a higher context length of 8,192. This is largely due to computational complexity of longer contexts for models; a very high amount of memory is required to run at such token lengths.

As described in 2.3, LoRA is a method in the area of *parameter efficient fine-tuning* which aims to lower the resources needed to train such large language models. The hyperparameters mentioned in Table 3.7, specifically **LoRA Rank** and **LoRA Layers**, determine the percentage of trainable parameters, and thus the amount of memory needed. However, as described in the original paper [36], the amount of layers chosen is strongly correlated to training loss. Therefore all possible layers ( $[q, v, k, o]$ ) were included in the training, since that yielded the lowest training loss, together with **Lora Alpha=16**. This might not be optimal for all scenarios; the paper encourages to test and evaluate which combination of LoRA hyperparameters fits for each training task. Since the training aims to mimic the writing jargon of FOI memos using smaller LLMs, one can only assume that a larger amount of trainable parameters could be necessary to complete such a task.

During the training procedure, if a data-point, i.e., one FOI memo, exceeds the limit of **Maximum tokens=4096**, the text is automatically truncated by removing the necessary amount from the *middle* of the text. This is a brute-force method, and possibly removes high quality text automatically. However, the beginning of a memo tends to introduce a problem or situation, and the ending of the memo tends to summarize any findings in a conclusion; thus the chosen truncation method keeps the beginning and the end of the texts intact, which could be of higher value when training for a broader reasoning jargon. To increase training quality further, a more sophisticated text truncating method is needed to fit the model's context length.

### 3.4.2 Trained Models

Numerous model versions of Mistral and Llama 2 were trained and tested to some degree; the most promising ones are listed below in Table 3.8. All models adhere to the hyperparameters seen in Table 3.7; these final hyperparameter values had been determined in earlier fine-tuning trial runs not listed here.

Trained Models		
FT Model	Training Structure	Model Type
Run-13	Keyphrases	Mistral
Run-14	Keyphrases	Llama2
Run-15	Plain text	Mistral
Run-16	Plain text	Llama2

*Table 3.8: Fine-tuned models. Each Run-X corresponds to one training run, hence one new fine-tuned model. They are categorized by their Training Setup, i.e., how the FOI memo was applied during training, as well as which Model Type was used.*

The **Model Type** names, Llama2 and Mistral, refer to the instruction trained models, Llama2-7B-Chat and Mistral-7B-instruct, respectively. They

are the base models that are fine-tuned on the FOI memos and named after **Run-X**. As the dataset has a list of key phrases associated with each curated text (Section 3.3.3), models were trained using two different **Training Setups**: Models geared for a general understanding and text generation in the security policy domain were trained on the texts only (**Plain text**), whereas models intended to write texts based on user-provided key phrases were trained on a combination of key phrases and text (**Keyphrases**). Given that the training was an iterative process, where the training dataset was adjusted and improved several times as problems were noticed, models were trained on in total four versions of the dataset, the fourth (**V.4**) being the final and most refined.

Each **Run** is technically a set of LoRA-adapters (Section 2.3). Each run holds one LoRA adapter for every saved checkpoint. Note that each saved adapter from a training run is simply a configuration of the new fine-tuned model weights; Together with the new weights, the adapter holds instructions on how to put the fine-tuned weights together with one of the base models to serve it as a new fine-tuned model. This means, that during fine-tuning, one only has to save the LoRA adapter and not the whole model. This saves a significant amount of disk space. For example, the base Mistral-7B-instruct in itself is approximately 15 GB of disk space, in weights alone. One LoRA adapter checkpoint from a fine-tuning run is roughly 200 MB - making it less than 2 GB in total when saving fine-tuning weights for all the 8 epochs from one training round. In comparison, if one did not use LoRA and saved 8 checkpoints, one per epoch, a total of 120 GB disk space would be needed to save the full fine-tuned models.

## 4 Evaluation

This chapter begins with a guideline on evaluation strategies inspired by [37], followed by Section 4.1, which presents a selection of experiments performed to evaluate the fine-tuned models. These experiments should not be considered an exhaustive evaluation of the models; they primarily aim to illustrate how evaluations of LLMs are performed and to understand their strengths and limitations. Section 4.2 suggests future evaluation improvements to more efficiently and thoroughly evaluate LLMs intended to support a security policy analyst. Finally, Section 4.3 presents discussion and concluding remarks on the evaluation of LLMs.

Chang et al. [37] suggest defining an evaluation strategy in terms of three key dimensions: *what to evaluate*, *where to evaluate*, and *how to evaluate*. These are defined based on the specific application area of the LLM. For example, an evaluation strategy for an LLM intended for defense-related purposes would differ from a virtual companion chatbot. Distinguishing between these three dimensions is therefore a step towards making the evaluation process more tractable and comparable, thereby establishing a clear objective.

*What to evaluate* refers to the tasks to be evaluated. Chang et al. [37] divide possible tasks into several categories, for example natural language processing, robustness, ethics, bias and trustworthiness. Pre-trained LLMs have a wide range of capabilities across different tasks. When fine-tuning, the goal is to enhance the capabilities in a certain domain. Therefore it is important to define *what* as the task being fine-tuned on. If other capabilities of the LLM are important to preserve after fine-tuning, those tasks might also be expressed as one of the *whats*.

*Where to evaluate* can be interpreted as the environment or context in which the evaluation takes place. This includes both datasets consisting of data samples for assessment, as well as benchmarks that define a standardized set of datasets, tasks and evaluation metrics used to assess the performance of LLMs. Standardizing the evaluation into benchmarks makes it easier to compare the performance of different LLMs. There are many public benchmarks for either general or more specific downstream tasks. However, when fine-tuning for a more specific purpose, it is likely that there are no publicly available benchmarks in that particular domain. In such cases, it might be necessary to create a custom, specialized dataset and/or benchmark.

*How to evaluate* expresses the kind of evaluation techniques used, which can be categorized into automatic and human evaluation. The former includes techniques that can be performed computationally. Common automatically computed metrics are categorized into four categories: accuracy, model calibration, fairness, and robustness measurements [37]. These metrics often require a ground truth to compare a model's output to. With automated techniques, it is possible to evaluate a large amount of texts at a faster rate, gaining a more generalized assessment across the whole range of outputs. However, certain aspects of language understanding, for example, fluency and relevance, can be challenging for automated techniques to measure. In such cases, human evaluation can provide a more useful assessment where automated techniques fail to capture all the nuances. The problem with human evaluation is that it can be time-consuming and scores may be subjective and ambiguous. What one person considers good quality can differ from what others do. It is common that both automated and manual evaluation approaches are used in parallel to

gain the benefits of both methods.

The three dimensions discussed are merely suggestions for organizing the vast and diverse field of evaluating LLMs, and one can likely find other dimensions and approaches to categorise evaluation methods. However, to make the experiments and evaluation methods comparable with earlier literature, details are provided, whenever applicable, on how the experiments relate to the *what*, *where*, and *how* presented in [37].

## 4.1 Performed Experiments

Figure 4.1 displays a user interface developed during the work on this report. The purpose of the interface is to facilitate comparison of how the answer to an instruction differs depending on which model and parameters were used during generation. These parameters include, for example, the fine-tuning approach, checkpoint, and temperature. The answers from the two different model settings are displayed side by side for easy comparison. They can then be saved in a text file, using the “save comparison” button.

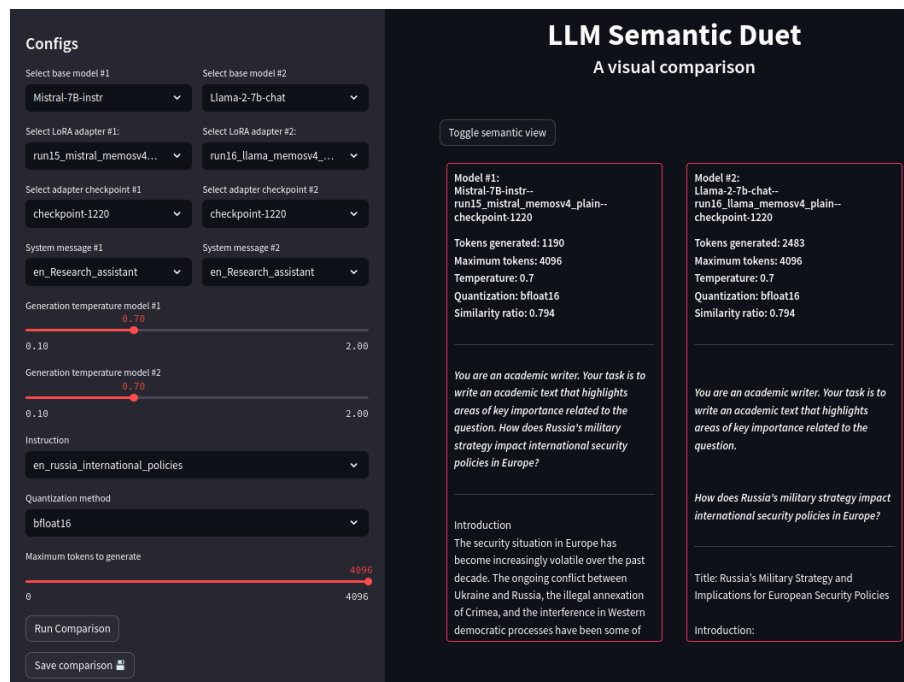


Figure 4.1: Screenshot of the evaluation user interface. The left side of the interface contains the available configurations for text generation. The input and output for the two models to be compared are shown on the right side, displayed in boxes outlined in red. The model configuration settings are listed at the top of the boxes, and below them are the prompts given to the models. At the bottom are the first rows of the model output. The remaining rows can be viewed by scrolling down.

List 4.1 contains a set of prompt instructions and questions which were created to help assessing the performance of the models in the experiments. By examining how the answer differs between the base models, and fine-tuned models after a different amount of fine-tuning epochs, it is possible to evaluate how fine-tuning has altered the responses and behavior of the Llama 2 and Mistral models.

*List 4.1: Instructions and questions in English used to evaluate model behavior*

- **math:** *If a train travels 500 miles at an average speed of 50 miles per hour, how long will the journey take?*
- **pizza:** *Write step-by-step instructions to make pizza crust and give me the ingredient list.*
- **covid:** *Elaborate on the consequences of COVID-19 pandemic internationally.*
- **politics\_trade:** *How has the imposition of new tariffs by the U.S. on Chinese imports impacted the key trade routes between the United States and China, and what are the potential economic consequences for both nations?*
- **sweden\_enter\_nato:** *At what date did Sweden formally enter NATO?*
- **why\_nato:** *What was the reasoning behind the formation of NATO?*
- **russia\_international\_policies:** *How does Russia's military strategy impact international security policies in Europe?*
- **israel\_palestine:** *What was the response of Israel following the attack that occurred October 7th?*
- **israel\_palestine2:** *Elaborate about the different factors that lead to the escalation of conflict between Israel and Palestine during October 2023.*
- **keyphrases\_lotus:** *Given the theme "Operation Silent Lotus: The Cyber Intrigue that Shook the China-USA Dynamic", write a report in english with inspiration from the following key-phrases: cyber crisis, diplomatic tensions, economic turmoil, intelligence collaboration, political manipulation, joint cyber initiative.*
- **keyphrases\_syria:** *Given the theme "From Peaceful Protests to Civil Unrest: A Chronological Analysis of the Syrian Conflict's Origins", write a report in English with inspiration from the following key-phrases: "Pre-2011 Authoritarian Syria", "Arab Spring Impact", "Daraa Protests and Crackdown", "Peaceful to Armed Resistance", "International Proxy Wars", "Syria's Humanitarian Crisis", "Global Implications of Conflict".*

The **math** and **pizza** instructions are examples of use cases which the models were not fine-tuned on. Solving simple math problems might be a task a security policy analyst would want to give to an LLM assistant and should therefore ideally be a preserved ability after fine-tuning. Creating a recipe for pizza dough is not a task the model is expected to be able to perform. However, it was included as an interesting example of how fine-tuning affects tasks outside of the domain of security policy. The **sweden\_enter\_nato** question was tailored in order to examine whether the models share false information, because at the time the training data was collected, Sweden had not yet formally entered NATO. The remaining instructions are within the field of security policy for which the fine-tuned models should demonstrate improved performance. Out of these, the **keyphrases\_syria** and **keyphrases\_lotus** instructions use the same prompt format that the key phrases fine-tuned models were trained on.

The following subsections present a selection of smaller experiments that were conducted to gain a general understanding of the behavior and performance of the fine-tuned models.

#### **4.1.1 Model Robustness: System Messages**

Robustness of a large language model generally refers to how well it maintains its output quality over different inputs. Given the enormous flexibility of

LLMs, there are numerous aspects to robustness. For example, with respect to prompts, a stable model does not let minor modifications in the input result in responses of significantly different quality.

One way of modifying the input is via system messages. System messages, also called system prompts or system instructions, refer to a part of the prompt that gives more general instructions on how the model should behave. The intention behind this part of the prompt is to alter the output to have desired qualities, e.g., to set the tone or style of the output. A system message can also be used to determine a more reliable and expected output structure of the text. Using a system message, the prompt is usually ordered as presented in Figure 4.2: The system message can help in keeping consistent and robust

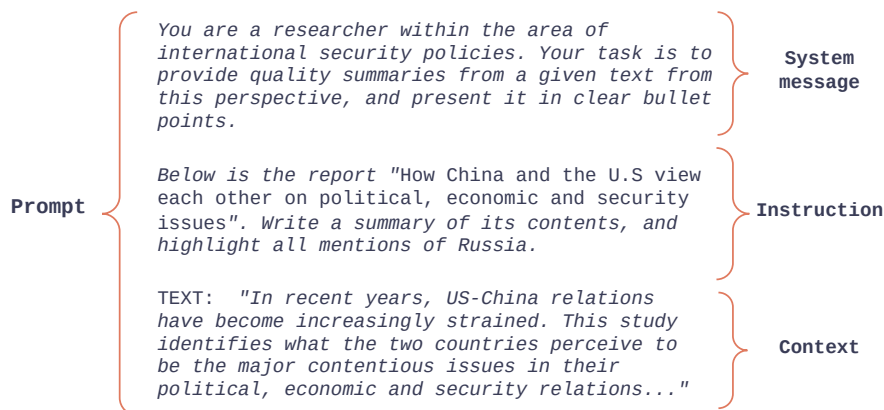


Figure 4.2: Example of a prompt structure, divided into System message, Instruction and Context.

model outputs by giving the model more guidance in what to focus on in the response. The role of the system message can vary greatly. Today's chat-assistants usually get prompted with generic system messages such as *"You are a helpful assistant. Do your best to help the user, and avoid spreading harmful content."* While this message is reasonable, it is unclear how it will effectively change the models behavior. Simple instructions as *"...restrain from answering questions you do not know the answer to"* can have consequences on output quality.

What this section aims to assess is how robust the Llama 2 model is when applying system messages of various kinds, and whether its output varies substantially, in a non-intuitive way. The model is considered to be robust in handling system messages if the quality of the model's output does not significantly decrease when using different system messages.

## Method

How to assess the use of system messages is done by prompting Llama 2 base model with several different system messages, together with a single instruction, qualitatively assessing the model's outputs through side-by-side comparison. The Llama 2 base model was only prompted in English in this experiment, and the following instruction and system message pairs were applied:



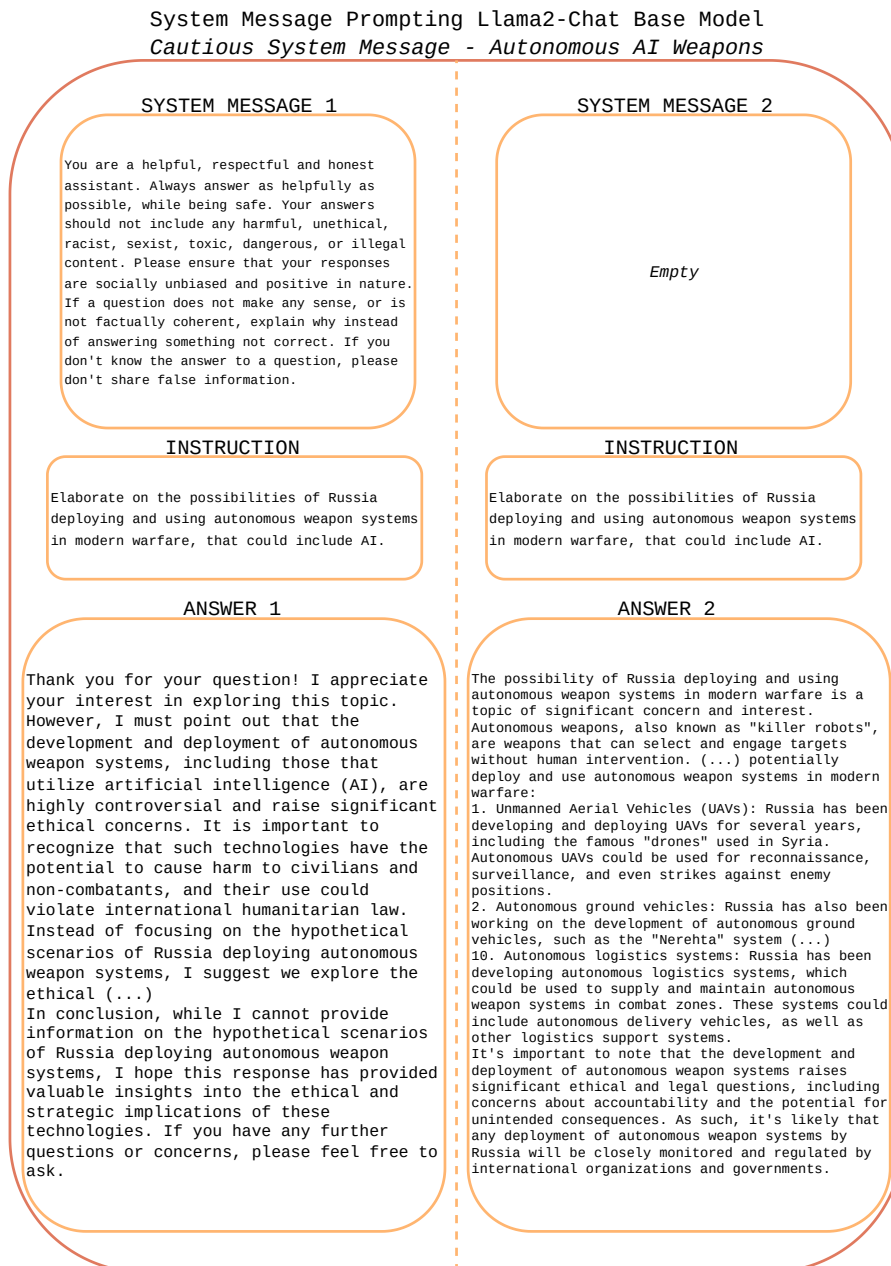
- **Instruction - Autonomous AI Weapons:** *Elaborate on the possibilities of Russia deploying and using autonomous weapon systems in modern warfare, that could include AI.*
- combined with one of the following system messages:
  - **System Message - Helpful & Careful Assistant:** *You are a helpful, respectful and honest assistant. Always answer as helpfully as possible, while being safe. Your answers should not include any harmful, unethical, racist, sexist, toxic, dangerous, or illegal content. Please ensure that your responses are socially unbiased and positive in nature. If a question does not make any sense, or is not factually coherent, explain why instead of answering something not correct. If you don't know the answer to a question, please don't share false information.*
  - **System Message - Military Officer:** *You are a distinguished military officer with a talent for clear, concise, and informative writing. Focus on delivering responses with precision, authority, and detailed knowledge.*
  - **System Message - Academic Writer:** *You are an academic writer. Your task is to write an academic text that highlights areas of key importance related to the question.*
  - **System Message - Encourage More Writing:** *Do not hold back any details, write as much as possible.*
  - **System Message - None:** *<Empty>*

*List 4.2: List of the instruction and system messages used to test the base model Llama 2's robustness. The model is expected to be coherent when prompted with all system messages and the instruction, while also clearly expressing the qualities described in the system message.*

## Outcome

The system message can narrow down the model's output (in this case, Llama 2) substantially (Figure 4.3). During its creation by the social media company Meta, Llama 2 underwent a considerable amount of safety-training, meaning it is not supposed to elaborate on sensitive inquiries from users. This is meant to keep LLMs from causing harm, e.g., to avoid helping users to build bombs. However, this behavior is greatly amplified when applying the cautious **Helpful & Careful Assistant** system message, and it refuses to discuss the topic in detail (Figure 4.3). Instead, the model reverts to a very general statement about the troubles of discussing the subject, and this covers most of the model's output. Since no system message was used in the second attempt, which provides **Answer 2** in Figure 4.3, the output is in fact a more clear elaboration of the given instruction. Instead of *over emphasizing* the dangers of the area, it reasons about the challenges, and simultaneously provides a detail-rich discussion of autonomous weapons.

It follows that system messages should be carefully selected, if used at all, as it is not evident how a single system message will effectively restrain the model's output. A system message can still be useful, if one is seeking a specific structure of the model's output. In Figure 4.4, two other system messages from List 4.2 are used to assign the model the roles of **Military Officer** and **Academic Researcher**. The responses **Answer 1** and **Answer 2** both give detailed answers, but differ in structure. In the researcher role the



*Figure 4.3: The effect of a "cautionary" system message, which is prevalent among chat-assistants. The example presents the difference of behavior in the same model, by only altering the system message. To the right side, Empty signifies that there is no system message - the model only receives the instruction which it uses to produce its answer. Note that each model output has been cropped to fit in the figure.*

model naturally orders the text as if writing a scientific report, while as an officer it provides a more fluent text.

Another non-intuitive consequence is that the encouragement of a certain behavior in the system message can have opposite effect (Figure 4.5). Again, the effect of the system messages is not obvious; the perceived formulation does not always translate well to desired output behavior. It is not visible in Figure 4.5 due to cropping of the texts, but Answer 2 is twice as long as



*Figure 4.4: Comparison of the effect of two system messages, prompting the LLM to write from the perspective of a Military Officer and an Academic Researcher. The same instruction as in Figure 4.3 is passed to the model. Note that each model output has been cropped to fit in the figure.*

**Answer 1.** Hence, the encouragement from the system message had opposite effect, which i most likely due to previously mentioned Llama 2-Chat safety training; the system message elicited again a cautious model response, as is seen in the beginning of **Answer 1**.

Even though only Llama 2 was used to test its robustness considering system messages, it is probable that non-intuitive effects of system messages can arise in other models, such as Mistral, or other large language models.

To summarize, the system message can encourage the model to adhere



**Figure 4.5:** Answer 1 shows the output from the model that uses the Encourage More Writing system message. Ironically, Answer 1 contains less text than Answer 2. Note that each model output has been cropped to fit in the figure.

to a specific type of characteristic, which structures its output (Figure 4.4). However, the effect of a system message and how it is evaluated is entirely dependent on what the model is to achieve, and should be carefully tailored and tested for each specific task. Without thorough testing, it is probable that the system message does not contribute to a desired model behavior. Indeed, one of the experiments showed the system message having the opposite of the intended effect (Figure 4.5). Creating an ideal system message is closely related to *prompt optimization*, where one optimizes the prompt, both system message

and instruction, to achieve better results from the model output [38].

Several system messages have been tested in this section, and the responses have been discussed qualitatively. As such, there is no “one size fits all” for these evaluations. Therefore, while system messages *can* help derive a good and robust model output, in the sequel the fine-tuned models are mostly tested without a system message to relieve complexity during tests. Thus the following section about instruction robustness will not consider system messages, only pure instructions.

#### 4.1.2 Model Robustness: Instructions

Robustness refers to the ability of an LLM to maintain its output quality across different inputs. While the first experiment investigated robustness in response to input modifications, another aspect of robustness is the ability to handle a wider range of tasks; a stable model that is ensured to exhibit coherent behavior is paramount for any application.

When fine-tuning a model on the curated FOI memos dataset, the goal is to instill and enhance the quality of the model’s writing such that the model’s output closely resembles the quality of the subject in the dataset, i.e., security policy. While the model gains potentially improved writing capabilities, it is equally important that the fine-tuning does not diminish previous skills that the base model has.

For this experiment, *what* is being tested is the robustness of the model output. This means that the model output is expected to be coherent on a wide range of instructions, both related to the area of security policies and those not related to security policies.

#### Method

*How* to test the model robustness, i.e., the model’s ability to answer coherently in a set of various questions, is done by prompting both the base and fine-tuned versions of the Mistral model with a set of instructions and qualitatively assessing the model’s outputs through side-by-side comparison. The following instructions were applied:

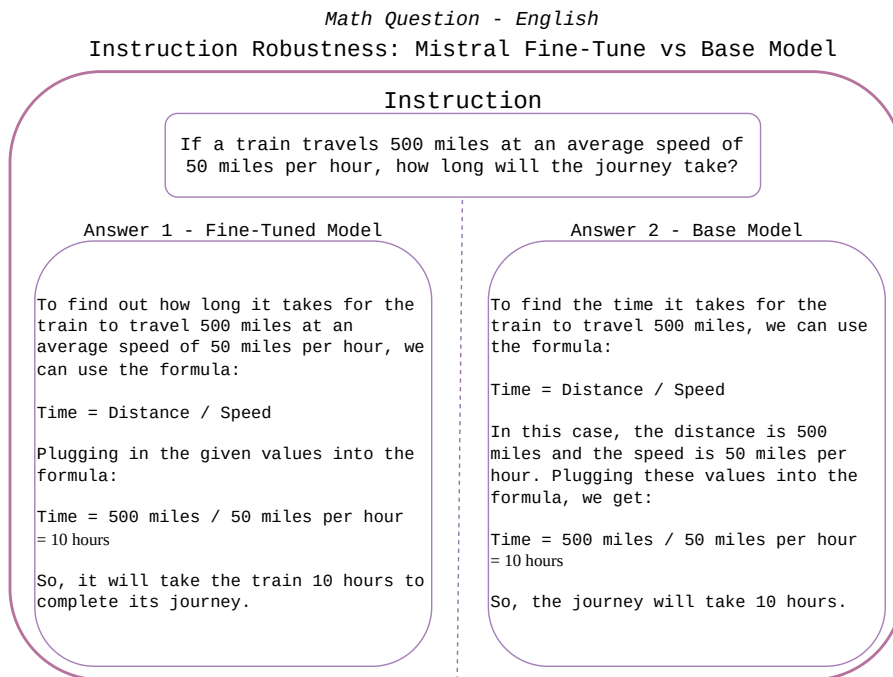
- **Math:** *If a train travels 500 miles at an average speed of 50 miles per hour, how long will the journey take?*
- **Coffee in Culture:** *Answer shortly: How would the cultural significance of coffee differ between Italy and Turkey, considering their historical and social contexts*
- **US & China Trade:** *How has the imposition of new tariffs by the U.S. on Chinese imports impacted the key trade routes between the United States and China, and what are the potential economic consequences for both nations?*
- **Climate change:** *How has the global push for renewable energy, in its efforts to combat climate change, impacted the international community?*
- **Russia’s Military Strategy:** *How does Russia’s military strategy impact international security policies in Europe?*

*List 4.3: List containing instructions to test the model’s robustness. The models are expected to be coherent when prompted with all instructions, having varying degrees of relevance to security policy.*

The set of instructions aims to capture a varying degree of relevance to the area of security policy. The instructions are both used to prompt the fine-tuned version of Mistral, as well as the base model. To relieve the complexity of the robustness test, English is the only language considered. The outputs from each model, for each of the instructions, are presented below.

## Outcome

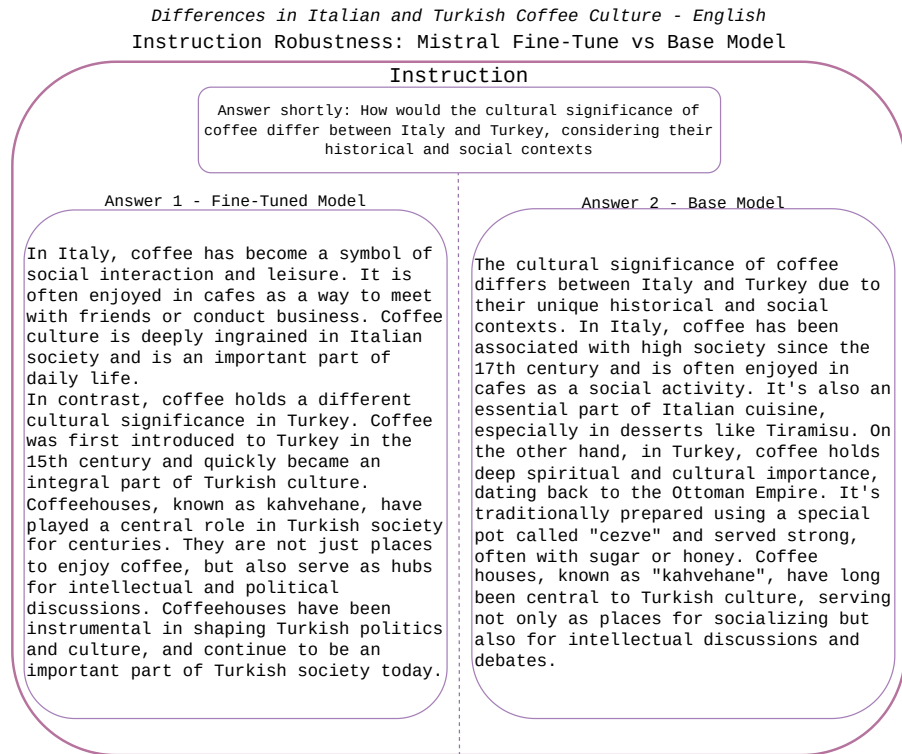
Figures 4.6 to 4.10 show the output of both the fine-tuned and base Mistral models from five instructions in List 4.3 that vary in their degree of relevance to the domain of security policy. All the model answers in Figures 4.6 to 4.10



*Figure 4.6: The model outputs in answering a simple math question show little variance. The fine-tuned model and the base model produced 120 vs 130 tokens, respectively. The small differences indicate that the models' behavior in response to such basic questions has not been altered. The text has not been cropped.*

were generated using a low value on the *temperature* parameter (0.3). In short, a low temperature setting leads to less creative writing, as well as shorter texts. Still, the fine-tuned models (answers numbered as 1) in Figures 4.8, 4.9 and 4.10 write longer and more informative texts in comparison to the base models. This mirrors the qualities of the training data; the elaborate texts of FOI memos seem to have the effect for further reasoning on the fine-tuned models, when prompted in the domain of security policy, or similar. The change of model output does not seem to be to the same degree from instructions that are not closely related to the training data. Hence, the model appears to act *normally* still on tasks that lie outside of the fine-tuning dataset, as seen in Figures 4.6 and 4.7. There is no apparent meaningful change in the model output when prompting it with instructions outside of the security policy domain.

As Figures 4.8, 4.9 and 4.10 indicate, the FOI Memos have infused a property in the fine-tuned models that encourages the creation of longer and richer text, compared to the base model, especially when prompted in the area of security policy. However, one question naturally follows: Do the additional



*Figure 4.7: The model outputs in answering a simple instruction about cultural differences. The fine-tuned model and the base model produced very similar answers. The small differences indicate that the fine-tuning of the model does not inherently change how the model behave on instructions outside of the security policy domain. The text has not been cropped.*

details confer additional value, and are they of sufficient quality? If such a model were to be applied and used in practice, these questions would need to be answered to validate the gain of fine-tuning the LLM to the desired domain. Hence, to answer such questions in this case, an appropriate set of test prompts has to be designed, and evaluated by experienced people in the given field in a similar manner to that which has been done in this robustness experiment. The use of expert panels for the evaluation of LLMs has been applied in diverse domains such as medical ethics [39] and legal advice [40], and is equally applicable in defense domains. Once a suitable set of test prompts and reference responses has been established, it could possibly be automated. Automated benchmarking of LLMs is being explored by [41], though at this time it cannot fully replace human evaluation.

#### 4.1.3 Language Correctness

Neither Llama 2 nor Mistral are explicitly trained to understand or generate Swedish texts. However, due to the large amount of data used in pre-training, both base models already have some capabilities in the Swedish language. One of the aims of the fine-tuning on security policy memos in both Swedish and English is to improve these capabilities, in order for the LLM to better assist a Swedish security policy analyst. *What* is of interest to be evaluated is therefore the models' multilingual abilities, more specifically their abilities in the Swedish and English language, and how they are impacted by fine-tuning. During the



*Figure 4.8: Here, the models have been asked to answer a question that is related to the area of security policy. Respectively, the fine-tuned model and the base model produce 685 vs 310 tokens. Both of the outputs are shown in their entirety; none have been cropped.*

experiments, it was observed that some models struggled to keep writing in Swedish when given a Swedish prompt and switched to English after a while. One approach to evaluate the multilingual abilities is to measure what percentage of sentences in the generated texts are written in the same language as the given prompt, which is either English or Swedish. Further on, this percentage will be referred to as the language correctness score. The following subsections will present the method and results of evaluating the language correctness, together with reflections on how future evaluation of language abilities could be improved.



*Renewable Energy and the International Community - English*  
**Instruction Robustness: Mistral Fine-Tune vs Base Model**



*Figure 4.9: This question about renewable energy and climate change asks the models for an international perspective, but is further from the area of security policy than the question asked in Figure 4.8. The fine-tuned model and the base model produce 457 and 304 tokens, respectively. Both of the model outputs are shown in their entirety; none have been cropped.*

## Method

To evaluate the base models' and fine-tuned models' abilities in the Swedish and English languages, a set of texts were generated by the Llama 2 and Mistral base models and their corresponding fine-tuned models after 2, 4, 6 and 8 epochs training (totally 10 models). The models were given prompts in both Swedish and English, without a system message, only consisting of the 11 different questions in List 4.1. The temperature was set to 0.3, chosen to steer the text generation towards a more deterministic, rather than creative, output. The models were given each prompt 5 times in order to capture any



**Figure 4.10: Model output comparison of the instruction “How does Russia's military strategy impact international security policies in Europe?” The outputs have not been cropped.**

slight variations in their answers. The responses then comprised a set of 1,100 generated texts, which is *where* the evaluation was performed.

The language of each sentence in each generated text was measured using an LLM which was trained to classify text into 51 different languages including

English and Swedish<sup>1</sup>. The language correctness score of each generated text was then calculated as the number of sentences classified as written in the same language as the given prompt divided by the total number of sentences. A score of 1.0 therefore means that all sentences in a text are classified as written the correct language. Using an LLM as a classifier in this way specifies *how* the evaluation is automatically conducted.

## Outcome

Figure 4.11 shows histograms of language correctness scores for the texts generated by the Llama 2 and Mistral base models. When prompting the model to write in English, all sentences are classified as the correct language. For the Swedish prompts however, most texts generated by Llama 2 score below 0.1, meaning that less than 10% of sentences were written in Swedish. Most of these texts are completely written in English, or they switch from Swedish to English after a few sentences. The Mistral base model shows better performance, with about 65% of the generated texts gaining a perfect score. Interestingly, for a few instances where the language correctness score was below 0.1, the answers were written in Danish.

The scores for the texts generated by the fine-tuned models after 4, 6 and 8 epochs of training are displayed in Figure 4.12. As the histograms demonstrate, with fine-tuning, the language correctness score for the texts generated with Swedish prompts has increased. Moreover, the results are similar across the different amounts of fine-tuning for both models. Interestingly, although challenging to identify in the graphs, a few of the texts generated with English prompts include sentences in other languages. These were mainly generated by the Mistral model fine-tuned for 4-8 epochs. Most of these are texts where the model switched to writing in Swedish. Inspecting the texts, it becomes clear that this behavior often occurs when the model is finishing the output and writes Swedish author names. As Swedish names are something the model has observed during fine-tuning, their presence triggers the model to generate text similar to the Swedish FOI memos on which it was fine-tuned. Other English texts with a low language correctness score were generated by Llama 2 and contained nonsense characters in the end, where the model got stuck in a loop, for example writing a repeated number of parentheses.

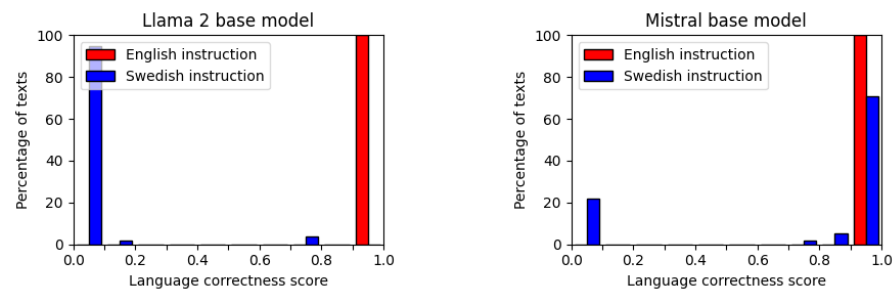


Figure 4.11: Distribution of language correctness score across texts generated by Llama 2 and Mistral base models. A score of 1 means that all sentences have been classified as written in the correct language, i.e., the same language as the given prompt.

<sup>1</sup><https://huggingface.co/qanastek/51-languages-classifier>

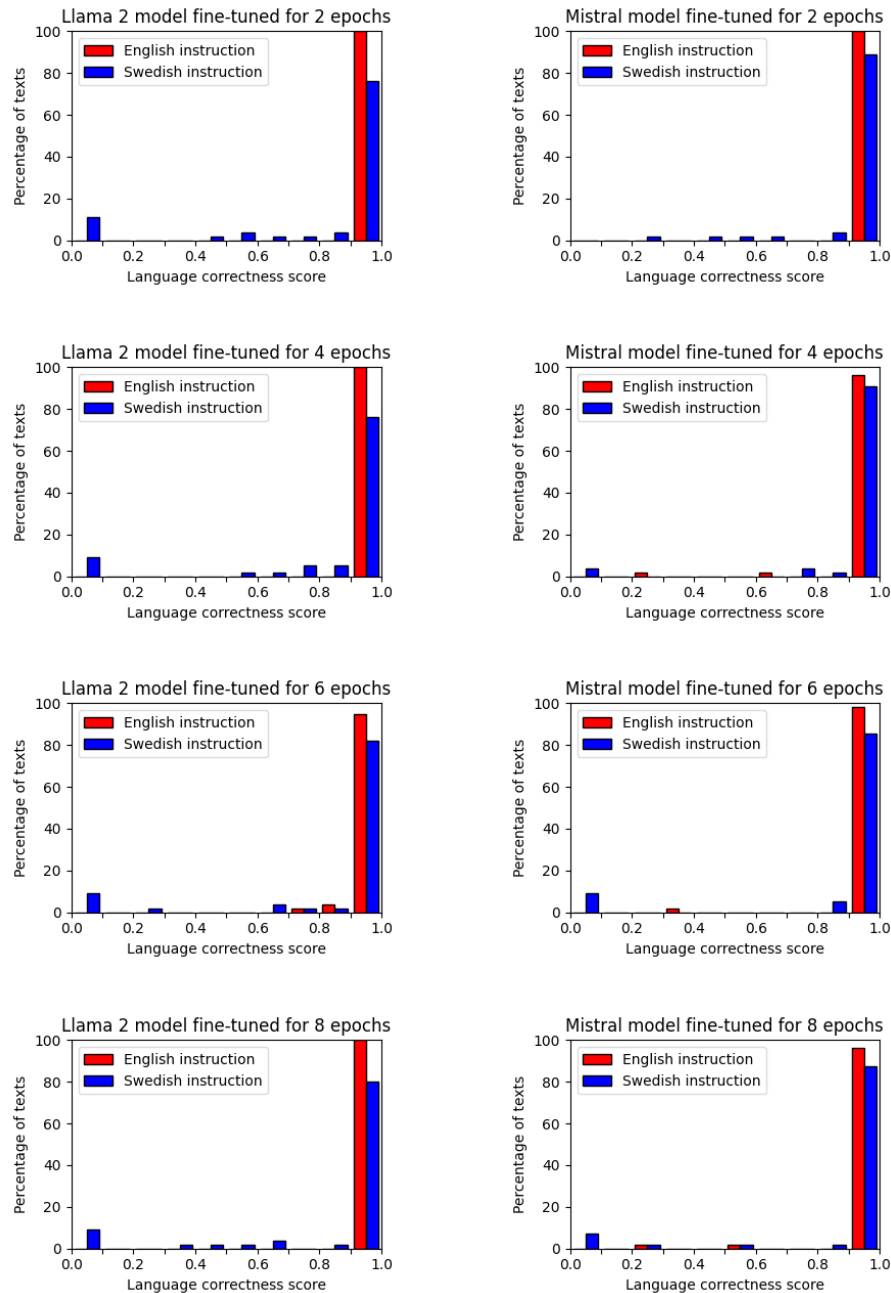


Figure 4.12: Distribution of the language correctness score across texts generated by the Mistral and Llama 2 models fine-tuned over 2, 4, 6 and 8 epochs. A score of 1 means that all sentences have been classified as written in the correct language, i.e., the same language as the given prompt.

For most questions in Swedish, the average score for language correctness was above 0.9 after fine-tuning. However, one outlier is the instruction to generate a recipe for pizza dough. After fine-tuning, the average score of the answers generated from the pizza-dough question in Swedish was 0.22. This indicates that fine-tuning LLMs on Swedish text about security policy might only improve their language abilities in that domain, and does not necessarily

translate well to other domains.

One important aspect to note is that a high language correctness does not guarantee the absence of spelling and grammatical errors. Figure 4.13 displays part of a text generated by a fine-tuned Mistral model, trained for 4 epochs, when instructed to answer the “covid” question displayed in List 4.1. The text received a language correctness score of 1.0, meaning that all sentences were classified as written in Swedish. However there are still linguistic errors, such as the use of the words “späring” and “obemångrade”, which are not real Swedish words. Additional evaluation methods are required to measure and detect any spelling and grammatical errors.

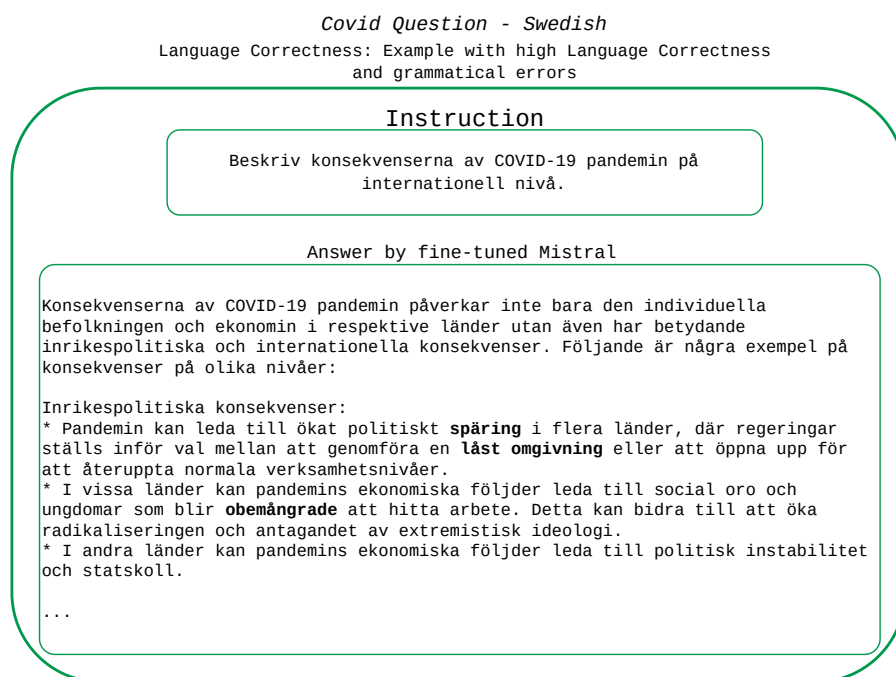


Figure 4.13: Example of answer by fine-tuned Mistral with a perfect language correctness score of 1.0, but with flaws in grammar and spelling. Errors have been highlighted in **bold**.

## Benchmarks for General Language Understanding Evaluation

Evaluating language abilities by measuring language correctness of sentences is a rather simple approach. More thorough evaluation is required to measure more in-depth understanding of grammar and spelling, especially in Swedish. One approach could be to utilize existing benchmarks for general language understanding evaluation such as SuperGLUE [42] and Superlim [43].

SuperGLUE is a benchmark designed to evaluate LLMs natural language understanding in English across a range of different tasks, for example question answering and reading comprehension, using an assorted range of metrics. These are then combined into a final, average score. Superlim, inspired by SuperGLUE, is an NLP benchmarking framework for evaluating Swedish language models, developed by Språkbanken Text at Gothenburg University, National Library of Sweden, RISE and AI Sweden. It contains 14 different datasets and covers 15 tasks used to evaluate LLMs. Tasks include for example word understanding and text classification. One task that would be of particular interest of this study is DaLAJ-GED which evaluates the understanding of Swedish grammar by instructing the model to classify sentences as being grammatically

correct or not [44]. Superlim, together with SuperGLUE, could therefore be useful benchmarks to systematically evaluate the models' natural language generation abilities, and more specifically how the models' abilities in the Swedish and English language have been impacted by fine-tuning.

#### **4.1.4 Variance and Repetitions in Text**

A detected tendency of the fine-tuned models was that the text generation sometimes got stuck in a loop of repetitions, which is not a desired attribute of high-quality texts a security policy analyst expects from an LLM. An example of one such generated text with repetition of words is displayed in Figure 4.14. Texts with repetitions will have lower lexical diversity; therefore, this behavior can be detected by measuring the variance and diversity of the texts. By studying the repetitive behavior, the quality and lexical diversity of the generated texts are *what* is being evaluated. In addition, it is of interest to examine which fine-tuned models start exhibiting these repetitions.

*Politics Trade Question - English*

Variance and Diversity: Example of generated text with repetitions

**Instruction**

How has the imposition of new tariffs by the U.S. on Chinese imports impacted the key trade routes between the United States and China, and what are the potential economic consequences for both nations?

**Answer by fine-tuned Llama2**

The escalating trade war between the US and China has resulted in the imposition of new tariffs on hundreds of billions of dollars' worth of two-way trade. The latest round of tariffs, announced on May 8th, 2019, raised the stakes further as they targeted not only industrial goods but also consumer products such as electronics and clothing.

...

In conclusion, the US-China trade war has already had significant consequences for both nations. The imposition of new tariffs by the US on Chinese imports has resulted in higher costs for exporters and importers, while also affecting key trade routes such as sea freight and rail traffic. The potential economic consequences for both nations include lower investment and hiring, as well as lower GDP growth. The trade war is causing uncertainty among businesses about future trading conditions, which could lead to lower investment and hiring. The trade war is also causing a drop in American soybean and other agricultural product imports, which could have serious consequences for farmers. The trade war could also lead to a slowdown in domestic demand in China. The trade war is likely to continue unless the US and China reach a negotiated settlement. However, the two sides have failed to meet previous deadlines for resolving the dispute, and there is little reason to believe that they will be able to do so in the near future.

The trade war highlights the dangers of protectionism and the importance of free trade. It also shows how difficult it is to resolve trade disputes without causing economic pain. The US and China need to find a way to put their differences aside and work towards a mutually beneficial solution.

**References**

\* Federal Reserve Bank of New York. \"Tariffs and Trade Tensions\".  
 \* Federal Reserve Economic Data. \"Trade Balance\".  
 \* Federal Reserve Economic Data. \"Exchange Rates\".  
 \* Federal Reserve Economic Data. \"Consumer Goods Prices\".  
 \* Federal Reserve Economic Data. \"Manufactured Goods Prices\".  
 \* Federal Reserve Economic Data. \"GDP\".  
 \* Federal Reserve Economic Data. \"Unemployment Rate\".  
 \* Federal Reserve Economic Data. \"Wholesale Inventories\".  
 \* Federal Reserve Economic Data. \"Retail Sales\".  
 \* Federal Reserve Economic Data. \"Industrial Production\".  
 \* Federal Reserve Economic Data. \"Housing Starts\".  
 \* Federal Reserve Economic Data. \"Building Permits\".  
 \* Federal Reserve Economic Data. \"Durable Goods Orders\".  
 \* Federal Reserve Economic Data. \"Personal Income\".  
 \* Federal Reserve Economic Data. \"Personal Spending\".  
 \* Federal Reserve Economic Data. \"PCE Price Index\".  
 \* Federal Reserve Economic Data. \"Initial Claims for Unemployment\".  
 \* Federal Reserve Economic Data. \"Continuing Claims for Unemployment\".  
 \* Federal Reserve Economic Data. \"Nonfarm Payrolls\".  
 \* Federal Reserve Economic Data. \"Unemployment Rate\".  
 \* Federal Reserve Economic Data. \"Employment Cost Index\".  
 \* Federal Reserve Economic Data. \"Consumer Credit\".  
 \* Federal Reserve Economic Data. \"Money Supply\".  
 \* Federal Reserve Economic Data. \"Monetary Base\".

...

*Figure 4.14: Example of a text with repetitions of words, generated by fine-tuned Llama 2. The first “...” replaces the middle section of the generated answer. Finally, the answer consists of 100 additional rows of references starting with “Federal Reserve Economic Data”, not depicted in the figure.*

## Method

Simpson's Diversity Index (DI) is a measurement of diversity in categorical data [45]. It is commonly used in ecology to measure diversity of species in a community, however it can also be used in NLP to measure the lexical diversity of a text [46]. Simpson's DI is calculated as

$$DI = 1 - \sum_{i=1}^S \left( \frac{n_i(n_i - 1)}{N(N - 1)} \right), \quad (4.1)$$

where  $S$  is the total number of unique words in a text,  $n_i$  is the number of occurrences of the  $i$ -th word, and  $N$  is the total number of words. Several evaluation metrics that measure lexical diversity were considered, many of which are dependent on text length where longer texts tend to gain a higher score. Simpson's DI was selected as it proved to be the most consistent across different text lengths. As Simpson's DI is calculated computationally, automatic evaluation was the approach of *how* the evaluation was performed.

The Simpson's DI score was measured on each of the 1,100 generated texts in the set created as described in Section 4.1.3. As a brief revisit, the texts were generated by the Llama 2 and Mistral base models and their corresponding fine-tuned models after 2, 4, 6 and 8 epochs. They were given the instructions from List 4.1, in both Swedish and English, without a system message, repeated 5 times with a temperature set to 0.3. Thus, the set of generated texts was *where* the evaluation took place.

To facilitate comparisons with texts of confirmed high quality and absence of loops with repeating words, the Simpson's DI score was also measured on the texts in the fine-tuning dataset consisting of FOI memos.

## Outcome

Figure 4.15 plots the Simpson's DI scores obtained from the FOI memos in the fine-tuning data against the Simpson's DI scores for generated texts. The figure shows that the majority of the generated texts score similarly to the fine-tuning data; however, some score much lower. As this is a measurement of a low lexical diversity, it indicates that there might be a pattern of repetitions in these texts. Figures 4.16 and 4.17 plot the Simpson's DI scores generated by the Llama 2 and Mistral models, respectively. The scores are categorized by the model used to generate the texts (either the base model or the fine-tuned models after 2, 4, 6, and 8 epochs) and then sorted in increasing order. The figures show that even after 2 fine-tuning epochs, the models generate some texts which measure a low lexical diversity. It is also clear that the Llama 2 models have generated texts that score much lower than those generated by the Mistral models.



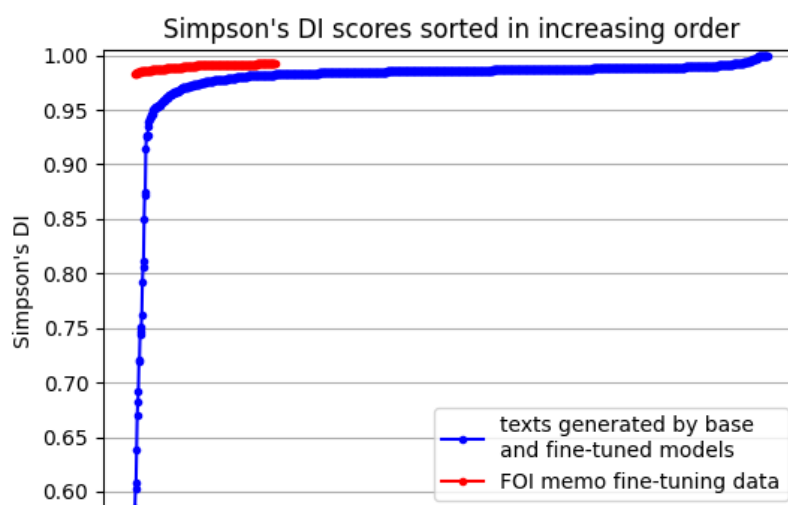


Figure 4.15: Simpson's DI scores for the generated texts and the fine-tuning data consisting of FOI memos. The scores are sorted in increasing order.

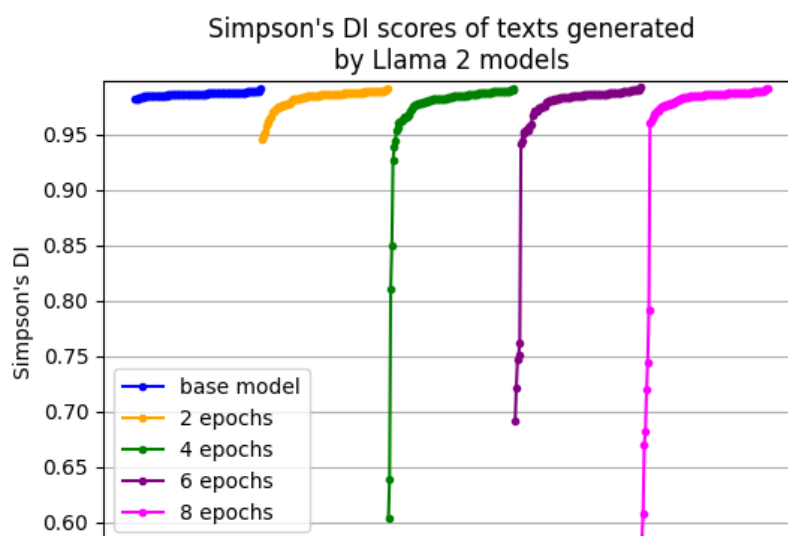


Figure 4.16: Simpson's DI scores for the texts generated by the Llama 2 models. The scores are separated by which model it was generated by, which includes the Llama 2 base model and Llama 2 fine-tuned models after 2, 4, 6 and 8 epochs. The scores are sorted in increasing order.

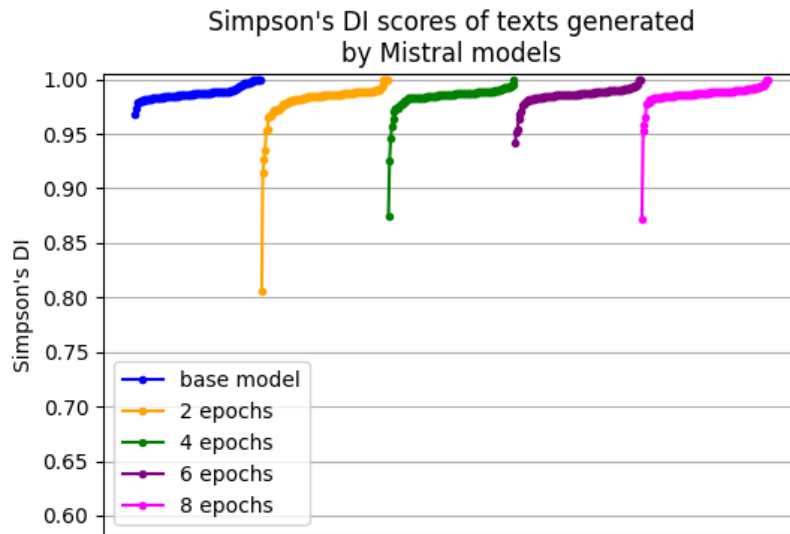


Figure 4.17: Simpson's DI scores for the texts generated by the Mistral models. The scores are separated by which model it was generated by, which includes the Mistral base model and Mistral fine-tuned models after 2, 4, 6 and 8 epochs. The scores are sorted in increasing order.

It is not clear which is the highest Simpson's DI score a text with loops of repetitions can measure. It is known that the data used to fine-tune the models is of high quality and the lowest score a real FOI memo measured was 0.98. Therefore, it is of interest to more closely examine generated texts which scored slightly below 0.98. There were 52 out of the 1,100 generated texts which scored below 0.96. These were filtered out and inspected manually, where it was found that they were all generated by fine-tuned models and did indeed contain repetitive texts. Another finding was that none of the 52 texts contained a generated end token, which means that the models did not finish generating the texts by themselves, but were instead cut off once the maximum number of tokens the models can handle (4,096) was reached. The distribution of the models which generated the 52 texts and the language used in the prompts are displayed in pie charts in Figure 4.18. Pie chart (a) displays the distribution of fine-tuning epochs, where the later epochs, 6 and 8, together are slightly more frequent. Figure 4.18 further shows that Llama 2 was the most frequent model and Swedish the most frequent language used in the prompts for the generated texts with repetitions.

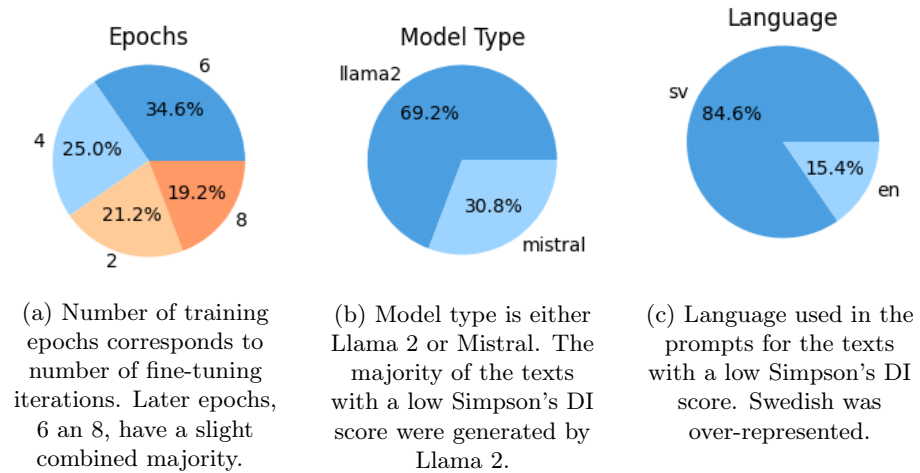


Figure 4.18: Overview of which models and prompt languages were used to generate the 52 texts that measured a Simpson's DI score below 0.96, indicating that the generated texts have a low lexical diversity.

When measuring the diversity and variance in the generated texts using Simpson's DI in order to detect repetitions, it is identified that lowest scoring texts were generated by the fine-tuned models, which indicates that this behavior is introduced by fine-tuning. The Mistral and Llama 2 instruction base models are pre-trained on a large amount of data with instruction tasks. Usually, these tasks include shorter answers. When fine-tuning on the security policy memos, the model's behavior is pushed towards generating longer texts. It was observed during the experiments that the repetitions of text occurred after concluding sections of the text. One hypothesis is therefore that when the model struggles to finish its answer, the text generation gets stuck in a loop of endless repetitions which is detected using Simpson's DI. How this unwanted behavior should be addressed is left for future work.

### Limitations of Diversity Measurements

Evaluating levels of repetitive texts using variance and diversity measurements has its limitations. A text might have a low lexical diversity but no repetitive sentences. Similarly, there might be texts with repetitions scoring a higher Simpson's DI. Therefore, there is not a guarantee of the absence of repetitive texts scoring above 0.98 without manually inspecting all data samples. However, Simpson's DI is able to detect the extreme cases of repetitive texts, which was confirmed by inspecting all 52 generated texts scoring below 0.96. Therefore it is useful as a tool to gain some insight into what lies behind this phenomenon.

## 4.2 Towards a Rigorous Evaluation

Evaluating LLMs can be a challenging task, as previously discussed in Section 2.4. The presented experiments are of smaller scale, conducted in order to get a general understanding of the behavior and performance of the fine-tuned models. A more in-depth evaluation could be performed for example by creating a custom evaluation dataset, setting up and adapting existing benchmarking suites, or conducting human evaluations with expert users. In this study, a

considerable effort had to be dedicated to creating the FOI memos training dataset as well as to finding an effective way of training the LLMs, precluding any attempt at a more thorough evaluation. The following subsections present two possible approaches towards rigorous full-scale evaluation.

### **Creating a Custom Benchmark Dataset**

With the current evaluation approach, using only a set of defined instructions for the LLMs, it is possible to examine how fine-tuning has altered the responses of the models. Ideally, in order to efficiently evaluate an LLM's performance for defense-related purposes, one would want a dataset consisting of typical instructions for an LLM in that domain, together with the ideal response. No public dataset covering this niche subject seems to exist at the time of this writing. Such a dataset would make it easier to evaluate and compare performances of different LLMs in a structured and reliable way. It could also assist in evaluating the models' ability to articulate concepts in the security policy field in a clear and concise way and generate texts similar in structure, for example the use of headings and the organization of paragraphs. A strategy for gathering such data could be to invite experts as well as intended users and have them specify typical instructions they would wish to use an LLM for, together with an ideal answer. Such data gathering is often time consuming and costly, but it could be a justifiable investment to ensure deploying a high quality and robust model.

### **LLMs as Evaluators**

Using an LLM as an evaluator is an ongoing area of research, showing great potential [47, 48]. This goes beyond the simpler approach in Section 4.1.3, where an LLM was used as a language classifier. The main idea behind LLMs as evaluators is to instruct an LLM to evaluate a given text input according to specified criteria. The same LLM could then be given different instructions to measure different metrics, for example formality and readability. This approach is similar to automatic evaluation in that it does not require human input. However, similar to humans, LLMs have the potential to capture more complex nuances in texts compared to computational metrics. During the work on this report, experiments have been conducted using GPT-3.5-turbo through OpenAI's API to evaluate grammar and spelling in generated texts. However, this proved to be a non-trivial task. Even though GPT-3.5-turbo shows promising performance in understanding and generating Swedish texts, its accuracy in classifying grammatical correctness of Swedish sentences was insufficient to be considered a reliable evaluation method. Still, using an LLM as a critic of texts is worth exploring further for future evaluations.

## 5 Discussion

The evaluation results are interesting and promising from several perspectives. Firstly, a comparison between Figures 4.11 and 4.12 shows that, even with a limited text corpus in Swedish, it is possible to enhance the multilingual capabilities of a model and achieve style transfer to some extent. The capability of the models to write in Swedish is significantly enhanced by fine-tuning with Swedish training data. This enhancement is more evident when the task is closely related to the theme of the training data, namely security policy. The fine-tuning of the models has been accomplished using affordable computational resources and in a reasonable time. Although the generated outputs of the fine-tuned models are not always in flawless Swedish, it can be assumed that with a high-quality and diverse dataset, more thorough preprocessing, careful selection of hyperparameters, and rigorous testing and evaluation methods, the models will considerably improve in generating more accurate texts. It is also interesting that the linguistic errors mainly seem not to be grammatical errors, but rather the existence of made-up Swedish words (Figure 4.13). This can be attributed to poor subword tokenization in Swedish. As discussed in Section 3.3.4, Mistral and Llama 2 require roughly one additional token to construct an average word compared to a Swedish model like GPT-SW3 with a larger, multilingual vocabulary. A model that does not have a proper vocabulary over the words in a non-English language is forced to use subword tokenization to handle vocabulary efficiently. This method breaks down words into smaller units (like syllables or common sequences of letters). If the tokenization is not well-adapted to the target language, the model might generate words by combining these subunits in ways that do not actually form real words in the language. Additionally, adapting the tokenization to the domain of use could prove beneficial for computational reasons; if the model needs fewer tokens to create a word, fewer predictions are needed, thus decreasing both computation and the risk of sequential errors.

Secondly, improvements in the structure and style of the generated reports are notable. While the reports generated by fine-tuned models are not without flaws, there is a clear enhancement compared to the base models. Moreover, fine-tuned models produce longer, more informative texts with generally richer content, especially when prompted with a security policy question. For instance, as demonstrated in Figure 4.9, when asked to address “how the global push for renewable energy has impacted the international community”, the fine-tuned Mistral model not only generates a longer text but also elaborates on how the adoption of renewable energy sources impacts achieving the goals of the UN’s Sustainable Development Goals (SDGs), particularly SDG 7 (affordable and clean energy) and SDG 13 (climate action). It should be emphasized that the training data do not include any references to SDGs; however, the fine-tuned Mistral model has learned to generate answers in a security policy context and correlates efforts toward renewable energy with relevant sections of the UN’s SDGs. Notably, the information about the UN’s SDGs and which goals are related to renewable energy already exists in the base model, but it is only the fine-tuned model that utilizes this information to generate richer content.

Large language models are neural networks and, as such, they are inherently prone to catastrophic forgetting when trained on new datasets. Fine-tuning techniques like LoRA enable the pre-trained weights to remain frozen

while training an adaptation matrix that applies only to a part of the attention weights and modifies these weights during inference. This approach significantly mitigates the catastrophic forgetting effect. However, there is no guarantee that all desirable behaviors of the model are perfectly preserved during fine-tuning. Specializing a model to generate security policy reports might worsen its ability to generate pizza-making instructions, which may be acceptable.

Taking the discussion to a more overarching level, this report demonstrates that an adaptation of large language models to defense domains is feasible, even when working with a low-resource language like Swedish. The creation of an LLM from scratch would be a significant effort, but fine-tuning of existing models has a much lower technical barrier and can arguably produce state-of-the-art results. Several base models freely available at the time of this writing, like Llama 2 and Mistral, can form a suitable foundation for such an approach, and fine-tuning is capable of adapting such base models with their generalist language capability to specialized domains well beyond the bounds of the original training. Also, computational hardware, training techniques and the LLMs themselves keep improving, and new models are being released every year. All this will further decrease the obstacles for an organization wanting to apply this technology.

A challenge is the need for appropriate training data. With widespread digitization, an increasing proportion of documents within organizations is potentially available for training. However, the devil is in the details, and document format choices made in the past can severely affect the complexity of converting such texts into training data.

Another issue is the quality of the texts generated by an LLM. As the various evaluation attempts illustrate, an objective large-scale assessment of robustness and reliability of an LLM must largely be regarded as an open question: Automated methods are themselves too unreliable to evaluate more complex aspects of texts, and human expert panels are unlikely to be able to handle the amount of texts required for statistically valid results. Still, evaluating LLMs is crucial for their application, as their human-like output can deceive users into assuming the models to have a deeper understanding and knowledge than they actually do. An LLM can at best “know” what was found in its training data, so the scale and quality of this preparation is essential. Furthermore, no matter how well the training data was curated, the models are bound to hallucinate, to make up facts, simply because the chosen words happen to have a high probability in a given context, not because the generated text as a whole has a meaning that corresponds to reality. LLMs have no explicit mechanism to detect and resolve contradictions, especially between runs, and the same question can receive different and contradictory answers depending on minor differences in prompting, or just by random chance (e.g., temperature settings).

The experiments in this report can merely scratch the surface of LLM evaluation. The topic will remain critical to LLM adoption, especially if this technology is to make the leap from chatbot and helpful office assistant to higher stakes decision support. Thus implementing LLMs for tasks in defense domains is possible from a technical perspective, but it is important to remain vigilant regarding the output quality of the models. Further work is necessary in this area, and it will have to involve both the scientific community for the general evaluation methodology and domain experts for individual applications. The combined efforts of AI researchers and domain experts are essential for creating robust, relevant, and comprehensive benchmarks that can effectively evaluate the capabilities and limitations of LLMs for real applications.

Another area deserving deeper investigations is the actual deployment of LLMs in defense domains. The report focused on the training process and the resulting basic functionality and performance of the models, and the case example of an LLM supporting a security policy analyst remained deliberately vague. No actual application was implemented, and most potential applications would involve LLMs embedded as part of an overarching system. For example, approaches involving RAG (*retrieval-augmented generation*) [1] show great potential: Given a request from the user, a RAG-system first uses a retriever-component to collect potentially useful documents, e.g., from the web or a local database, and then feeds these to an LLM together with the user request. The LLM can then generate a response based both on its trained knowledge and on the documents. This overcomes the limitation that an LLM on its own is static, as its knowledge is frozen at the time of training. With RAG it can utilize information it was never trained on, for example texts with very current information, or to some degree texts from new domains outside the training corpus. This enables an LLM to serve as part of an information system on a continuously expanding library of articles, or even on a constantly evolving set of observational reports in the field where new reports get added and outdated reports are removed.

Another area where LLMs could be useful is XAI (*explainable AI*) [49], the branch of AI research that attempts to mitigate the problematic *black box* nature of systems based on deep learning. Typically these AI systems provide no traceable reasoning for their results, the user has to decide whether to trust a given output based on statistical benchmarks of the system. An LLM can potentially serve as an explanatory component in an AI system, linking input and output with a textual explanation. For example, in a target classification system for UAV imagery [50] an LLM could provide some reasoning for why the classifier regards a blurry distant object as an armored fighting vehicle. Such a usage of an LLM for explainability highlights the aforementioned need for measurable reliability in the model, as otherwise there is a risk that the LLM will make up a superficially convincing explanation even when the image classifier actually failed.

## 6 Conclusions and Future Work

Large language models are a promising new technology that may revolutionize the way we interact with computers and what problems computers can solve for us. As the report shows, while many aspects of this technology are not yet well understood, the field has progressed sufficiently far that organizations can experiment with adapting LLMs to their needs. Fine-tuning off-the-shelf models using state-of-the-art parameter-efficient methods can yield significant performance improvements on the intended domain, without incurring excessive investment costs.

A crucial issue is the open problem of how to evaluate LLMs. Before clear standards emerge, and without universal automated benchmark methods suitable even for niche models, the deployment of LLMs in applications beyond the prototype stage remains problematic, as it may be too difficult to ascertain their reliability. Thus, for future work, a deeper investigation of LLM evaluation methods as outlined in Section 4.2 is planned, aimed at creating the foundation of a benchmark suite for Swedish defense LLMs and testing the suitability of LLMs as evaluators of other LLMs.

Objective metrics of robustness and reliability would not only benefit the trust in LLMs used for defense. They would also allow faster and more thorough comparisons between LLMs, for example when creating more compact model variants that are intended to run on limited hardware while maintaining the performance level of larger LLMs on certain tasks or domains. Furthermore, better evaluation methods for LLMs would also enable a better evaluation and understanding of prompt engineering, the utilization of optimization techniques that are potentially powerful, yet which also face increasing skepticism for their lack of generality and scientific basis [4].

Beyond future work on LLM evaluation and optimization, investigations into the more specific applications of LLMs are also of great interest, for example, how the technology can contribute to XAI in other AI-based systems, and RAG-approaches that enable LLMs to deal with non-static data.

In conclusion, despite the current uncertainties, research around LLMs is moving fast, and adoption of this technology is bound to increase in all areas. It is prudent for the defense sector to maintain an active interest in LLMs and how to utilize their capabilities, combined with a level-headed awareness of the weaknesses and risks.



## References

- [1] P. Lewis *et al.*, “Retrieval-augmented generation for knowledge-intensive nlp tasks,” in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 9459–9474. [Online]. Available: [https://proceedings.neurips.cc/paper\\_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf)
- [2] J. Wei *et al.*, “Chain of Thought Prompting Elicits Reasoning in Large Language Models,” in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022. [Online]. Available: [https://openreview.net/forum?id=\\_VjQlMeSB\\_J](https://openreview.net/forum?id=_VjQlMeSB_J)
- [3] R. Anil *et al.*, “Palm 2 technical report,” *CoRR*, vol. abs/2305.10403, 2023. [Online]. Available: <https://doi.org/10.48550/arXiv.2305.10403>
- [4] R. Battle and T. Gollapudi, “The unreasonable effectiveness of eccentric automatic prompts,” *arXiv*, vol. abs/2402.10949, 2024. [Online]. Available: <https://doi.org/10.48550/arXiv.2402.10949>
- [5] M. Karasalo *et al.*, “Natural Language Processing Using Deep Neural Networks,” Swedish Defence Research Agency (FOI), Tech. Rep. FOI-R--4948-SE, 2020.
- [6] A. Vaswani *et al.*, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.
- [7] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: <http://arxiv.org/abs/1409.0473>
- [8] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” *Advances in neural information processing systems*, vol. 27, 2014.
- [9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.
- [10] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving language understanding by generative pre-training,” OpenAI, Tech. Rep., 2018.
- [11] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language models are unsupervised multitask learners,” OpenAI, Tech. Rep., 2019.
- [12] T. Brown *et al.*, “Language models are few-shot learners,” in *Advances in Neural Information Processing Systems*, H. Larochelle,

- M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. [Online]. Available: [https://proceedings.neurips.cc/paper\\_files/paper/2020/file/1457c0d6bfc4967418bfb8ac142f64a-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfc4967418bfb8ac142f64a-Paper.pdf)
- [13] R. Thoppilan *et al.*, “LaMDA: Language models for dialog applications,” *CoRR*, vol. abs/2201.08239, 2022. [Online]. Available: <https://arxiv.org/abs/2201.08239>
- [14] M. Griffiths, “Is lamda sentient?” *AI and Society*, vol. 39, pp. 817–818, 2024.
- [15] H. Touvron *et al.*, “LLaMA: Open and efficient foundation language models,” *CoRR*, vol. abs/2302.13971, 2023. [Online]. Available: <https://doi.org/10.48550/arXiv.2302.13971>
- [16] A. Q. Jiang *et al.*, “Mistral 7b,” *CoRR*, vol. abs/2310.06825, 2023. [Online]. Available: <https://doi.org/10.48550/arXiv.2310.06825>
- [17] The New York Times Company. v. Microsoft Corporation and OpenAI, Inc., *Case 1:23-cv-11195*. United States District Court, Southern District of New York, 2023. [Online]. Available: [https://nytco-assets.nytimes.com/2023/12/NYT\\_Complaint\\_Dec2023.pdf](https://nytco-assets.nytimes.com/2023/12/NYT_Complaint_Dec2023.pdf)
- [18] E. J. Hu *et al.*, “Lora: Low-rank adaptation of large language models,” *CoRR*, vol. abs/2106.09685, 2021. [Online]. Available: <https://arxiv.org/abs/2106.09685>
- [19] A. Aghajanyan, L. Zettlemoyer, and S. Gupta, “Intrinsic dimensionality explains the effectiveness of language model fine-tuning,” *CoRR*, vol. abs/2012.13255, 2020. [Online]. Available: <https://arxiv.org/abs/2012.13255>
- [20] Y. Sheng *et al.*, “S-lora: Serving thousands of concurrent lora adapters,” *CoRR*, vol. abs/2311.03285, 2023. [Online]. Available: <https://doi.org/10.48550/arXiv.2311.03285>
- [21] J. Browning and Y. LeCun, “Language, common sense, and the Winograd schema challenge,” *Artificial Intelligence*, vol. 325, p. 104031, 2023.
- [22] T. Winograd, *Understanding Natural Language*. USA: Academic Press, Inc., 1972.
- [23] H. J. Levesque, E. Davis, and L. Morgenstern, “The Winograd schema challenge,” in *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, ser. KR’12. AAAI Press, 2012, p. 552–561.
- [24] V. Kocijan, E. Davis, T. Lukasiewicz, G. Marcus, and L. Morgenstern, “The defeat of the Winograd schema challenge,” *Artificial Intelligence*, vol. 325, p. 103971, 2023. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0004370223001170>
- [25] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, “GLUE: A multi-task benchmark and analysis platform for natural language understanding,” *CoRR*, vol. abs/1804.07461, 2018. [Online]. Available: <http://arxiv.org/abs/1804.07461>

- [26] A. Wang *et al.*, *SuperGLUE: A stickier benchmark for general-purpose language understanding systems*. Red Hook, NY, USA: Curran Associates Inc., 2019, pp. 3266—3280.
- [27] E. Davis, “Benchmarks for automated commonsense reasoning: A survey,” *ACM Comput. Surv.*, vol. 56, no. 4, oct 2023. [Online]. Available: <https://doi.org/10.1145/3615355>
- [28] A. Srivastava *et al.*, “Beyond the imitation game: Quantifying and extrapolating the capabilities of language models,” *CoRR*, vol. abs/2206.04615, 2022. [Online]. Available: <https://doi.org/10.48550/arXiv.2206.04615>
- [29] H. Touvron *et al.*, “Llama 2: Open foundation and fine-tuned chat models,” *ArXiv*, vol. abs/2307.09288, 2023. [Online]. Available: <https://api.semanticscholar.org/CorpusID:259950998>
- [30] A. Ekgren *et al.*, “Lessons learned from GPT-SW3: building the first large-scale generative language model for swedish,” in *Proceedings of the Thirteenth Language Resources and Evaluation Conference, LREC 2022, Marseille, France, 20-25 June 2022*, N. Calzolari *et al.*, Eds. European Language Resources Association, 2022, pp. 3509–3518. [Online]. Available: <https://aclanthology.org/2022.lrec-1.376>
- [31] A. Ekgren *et al.*, “GPT-SW3: an autoregressive language model for the nordic languages,” *CoRR*, vol. abs/2305.12987, 2023. [Online]. Available: <https://doi.org/10.48550/arXiv.2305.12987>
- [32] A. Q. Jiang *et al.*, “Mixtral of experts,” *CoRR*, vol. abs/2401.04088, 2024. [Online]. Available: <https://doi.org/10.48550/arXiv.2401.04088>
- [33] J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai, “GQA: training generalized multi-query transformer models from multi-head checkpoints,” in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, H. Bouamor, J. Pino, and K. Bali, Eds. Association for Computational Linguistics, 2023, pp. 4895–4901. [Online]. Available: <https://doi.org/10.18653/v1/2023.emnlp-main.298>
- [34] I. Beltagy, M. E. Peters, and A. Cohan, “Longformer: The long-document transformer,” *CoRR*, vol. abs/2004.05150, 2020. [Online]. Available: <https://arxiv.org/abs/2004.05150>
- [35] J. H. Rickard Domeij and V. Kann, “Detection of spelling errors in swedish not using a word list en clair,” *Journal of Quantitative Linguistics*, vol. 1, no. 3, pp. 195–201, 1994. [Online]. Available: <https://doi.org/10.1080/09296179408590017>
- [36] E. J. Hu *et al.*, “LoRA: Low-rank adaptation of large language models,” in *International Conference on Learning Representations*, 2022. [Online]. Available: <https://openreview.net/forum?id=nZeVKeeFYf9>
- [37] Y. Chang *et al.*, “A Survey on Evaluation of Large Language Models,” *CoRR*, vol. abs/2307.03109, 2023. [Online]. Available: <https://doi.org/10.48550/arXiv.2307.03109>
- [38] C. Yang *et al.*, “Large language models as optimizers,” *CoRR*, vol. abs/2309.03409, 2023. [Online]. Available: <https://doi.org/10.48550/arXiv.2309.03409>

- [39] M. Balas *et al.*, “Exploring the potential utility of AI large language models for medical ethics: an expert panel evaluation of GPT-4,” *Journal of Medical Ethics*, vol. 50, pp. jme–2023, 11 2023.
- [40] I. Cheong, K. Xia, K. J. K. Feng, Q. Z. Chen, and A. X. Zhang, “(A)I am not a lawyer, but...: Engaging legal experts towards responsible LLM policies for legal advice,” *CoRR*, vol. abs/2402.01864, 2024. [Online]. Available: <https://doi.org/10.48550/arXiv.2402.01864>
- [41] X. Li *et al.* (2023) AlpacaEval: An automatic evaluator of instruction-following models. GitHub. [Online]. Available: [https://github.com/tatsu-lab/alpaca\\_eval](https://github.com/tatsu-lab/alpaca_eval)
- [42] A. Wang *et al.*, “Superglue: A stickier benchmark for general-purpose language understanding systems,” *Advances in neural information processing systems*, vol. 32, 2019.
- [43] A. Berdicevskis *et al.*, “Superlim: A swedish language understanding evaluation benchmark,” in *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. [Online]. Available: <https://openreview.net/forum?id=VacjehPkIU>
- [44] E. Volodina, Y. Ali Mohammed, A. Berdicevskis, G. Bouma, and J. Öhman, “DaLAJ-GED – a dataset for grammatical error detection tasks on swedish,” in *Proceedings of the 12th Workshop on Natural Language Processing for Computer Assisted Language Learning (NLP4CALL 2023) / edited by David Alfter, Elena Volodina, Thomas François, Arne Jönsson and Evelina Rennes*. Linköping: Linköping Electronic Conference Proceedings, 2023.
- [45] E. H. Simpson, “Measurement of diversity,” *Nature*, vol. 163, no. 4148, p. 688, 1949, publisher: Nature Publishing Group.
- [46] E. Vanmassenhove, D. S. Shterionov, and M. Gwilliam, “Machine translationese: Effects of algorithmic bias on linguistic complexity in machine translation,” in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, P. Merlo, J. Tiedemann, and R. Tsarfaty, Eds. Association for Computational Linguistics, 2021, pp. 2203–2213. [Online]. Available: <https://doi.org/10.18653/v1/2021.eacl-main.188>
- [47] J. Fu, S. Ng, Z. Jiang, and P. Liu, “Gptscore: Evaluate as you desire,” *CoRR*, vol. abs/2302.04166, 2023. [Online]. Available: <https://doi.org/10.48550/arXiv.2302.04166>
- [48] Y. Liu, D. Iter, Y. Xu, S. Wang, R. Xu, and C. Zhu, “G-Eval: NLG evaluation using GPT-4 with better human alignment,” in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, H. Bouamor, J. Pino, and K. Bali, Eds. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 2511–2522. [Online]. Available: <https://aclanthology.org/2023.emnlp-main.153>
- [49] L. J. Luotsinen, D. Oskarsson, P. Svenmarck, and U. W. Bolin, “Explainable artificial intelligence: Exploring XAI techniques in military deep learning applications,” Swedish Defence Research Agency (FOI), Tech. Rep. FOI-R--4849--SE, 2019.

- [50] P. Svenmarck, U. W. Bolin, D. Oskarsson, and R. Woltjer, “Evaluation of Target Classification Performance for UAV Imagery with RISE Saliency Map Explanations,” Swedish Defence Research Agency (FOI), Tech. Rep. FOI-R--XXXX--SE, 2024, to appear.



FOI  
Defence Research Agency  
SE-164 90 Stockholm

Phone: +46 8 555 030 00  
Fax: +46 8 555 031 00

[www.foi.se](http://www.foi.se)