Contents

1 Introduction

Large Language Models (LLMs) have revolutionized the landscape of natural language processing (NLP). Benefitting from the unprecedented scales of model size and training data, many LLMs have emerged, both open-soruce, like LLaMa [1] and Mistral [2] and closed-source, like ChatGPT [3] and PaLM [4]. Unlike traditional machine learning systems, which are usually trained for a specific task and then exclusively applied to that task, LLMs demonstrate proficiency across a plethora of tasks, both traditional NLP challenges and novel, unanticipated tasks, despite not being explicitly trained for them.

Question Answering (QA) stands as one of the fundamental tasks in NLP, aiming to construct systems capable of autonomously responding to human inquiries in natural language. Historically, QA systems predominantly relied on extractive methodologies, where a question and corresponding context is provided, and a span of text extracted from the context is the anticipated answer. However, with the advent of LLMs, the QA tasks have broadened to include closed-book QA, multi-hop reasoning QA, conversational QA and more. Furthermore, tasks that evaluate other elements of LLMs, such as truthfulness or bias are often also framed as QA tasks.

LLMs have also revolutionized the area of machine learning by introducing new learning paradigms, namely in-context learning. In-context learning refers to the capabilities of LLMs to learn from examples provided during inference, without changing any of the model weights. On the other hand, traditional machine learning methods are still used in improving LLM performance on downstream tasks, such as fine-tuning. Recent studies [5], [6], [7] have shown the efficacy of fine-tuning with a small set of high quality data, a process termed instruction tuning.

One persistent problem in LLM development is the lack of coverage for languages other than English. This includes both pretraining data, which is usually mostly English and a lack of evaluation of model performance on non-English tasks. Some efforts have been made [8] [9] to transfer the capabilities of LLMs to other languages, but the domain is still left largely unexplored.

The aim of this thesis is to investigate the state-of-the-art of question answering in the age of large language models, specifically in the context of QA systems based on a language other than English, namely German. Both mentioned approaches, in-context learning and fine-tuning are explored as possibilities for adapting the base model of Llama-2 to German.

2 Background and Preliminaries

This chapter introduces the concepts and techniques required for understanding the domain of question-answering systems using large language models. First, an outline of the question-answering task is given, followed by an introduction to large language models and techniques used to improve them, namely in-context learning and fine-tuning. Then the topic of LLM evaluation is presented followed by an overview of models and datasets used in the thesis.

2.1 Question Answering

Question answering (QA) is a fundamental task in the field of Natural Language Processing (NLP), where the goal is to build systems that can automatically answer questions posed by humans in natural language. It has a direct application in systems such as search engines, virtual assistants and information retrieval systems. However, it can also be used as a benchmark for measuring machine language understanding for models that will later be used for different downstream tasks.

One way to categorize QA tasks is according to the way the model formulates answers:

1. **Extractive QA** - The model answers the question by extracting relevant parts of the context.
2. **Generative QA** - The model generates free text as an answer.

It is worth mentioning that, as generative QA is a relatively recent QA subtask in the field of NLP, historically, most QA benchmarks were developed with extractive question answering in mind. This means that to compare both extractive and generative models on the same benchmark it is often necessary to constrain generative models to answer only with context extractions. This results in the generative model being evaluated on both the task of instruction-following and question-answering at the same time which can confound the results. One possible way of dealing with this is presented in chapter 2.5.

Another way of QA task categorization is according to the availability of context for the given question:

1. **Open-book QA** - Context is provided to the model and the model uses is to answer the given question.
2. **Closed-book QA** - No context is provided, the model must answer the question using knowledge it already has.
3. **Retrieval Augmented Generation (RAG)** - Similarly to open-book QA, RAG systems use context to answer a question. However, they have the ability to work with much larger context sizes, such as a whole database of context. They first retrieve a set of relevant documents from the given source (eg. often a vector database) and use the retrieved documents to answer the question.

As mentioned, extractive QA tasks are more suited for model architectures other than decoder-only based LLMs, which are the main focus of this thesis. On the other hand, RAG systems typically utilize larger model architectures beyond the scope of this thesis. Therefore, further focus will be on the investigation of generative open-book and closed-book QA tasks.

2.2 Large Language Models

Language models (LMs) are probabilistic models of natural language. They are used for determining the probability of the next word in a sequence and the probability of the whole sequence. This is formalized in equation 2.1.

$$P(w_1, w_2, ..., w_{n-1}, w_n) = P(w_1^n) \tag{2.1}$$

The most rudimentary language models, word n-gram language models are purely statistical models of language. They are based on the assumption that the probability of the next word in a sequence depends only on a fixed size (n) window of previous words. This is formalized in equation 2.2. The model relies on the frequency of word sequences in large corpora.

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k | w_{k-N+1}^{k-1}) \tag{2.2}$$

With the advent of neural networks, the landscape of language models shifted dramatically. Implementing a simple feed-forward neural network as a language model was first presented in [10], while later advancements include the use of Recurrent Neural Networks (RNNs) with Long Short-term Memory (LSTM) [11] and the Gated Recurrent Unit (GRU) [12].

The next significant milestone was the introduction of the transformer architecture, presented in [13]. The transformer marked the advancement of Language Models (LMs) to Large Language Models (LLMs). Since the transformer is the backbone of the main model that is used in this thesis, a more through look into its architecture is presented in 2.2.1.

2.2.1 The Transformer

Transformers, introduced in [13], have revolutionized the field of natural language processing with their unique architecture. The core idea behind transformers is the use of self-attention mechanisms that directly compute the relationship between all words in a sentence, regardless of their positions. This step enabled parallel instead of sequential processing of the input sequence which was a huge speed-up compared to RNNs. This speed-up in model architecture, aided by improved hardware and the availability and use of huge volumes of data facilitated the move from Language Models to Large Language Models.

The Architecture

The architecture proposed in the original paper is visualized in figure 2.1 and is composed of the following parts:

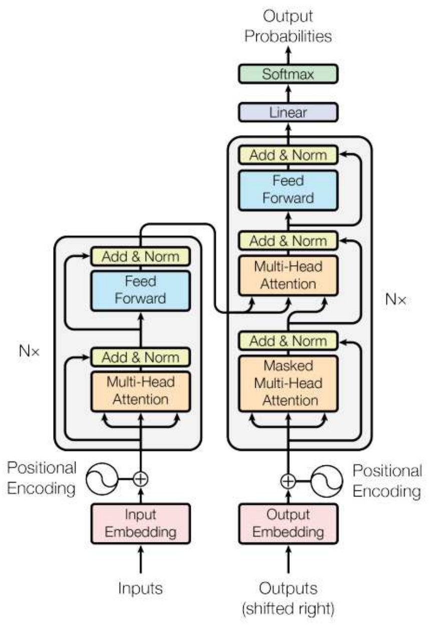

Figure 2.1: The Transformer - model architecture. Image source: [13]

Encoder: The encoder is a stack of 6 identical layers. Each layer consists of a multi-head self-attention mechanism and of a simple, position-wise fully connected feed-forward network. A residual connection is employed around the two sub-layers followed by layer normalization.

Decoder: The decoder again consists of 6 identical layers stacked on top of each other. It again employs the same sub-layers as the encoder, but with an addition of a third sub-layer which performs multi-head attention over the output of the encoder stack. Residual connections are also employed around each sub-layer followed by layer normalization. The self-attention sub-layer is modified so that it doesn't attend to subsequent positions.

Attention: An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key [13]. The formal definition is given in equation 2.3. In practice, the transformer doesn't use only one attention function, but instead the queries, keys and values are linearly projected h times with learned linear projections. This mechanism is called multi-head attention and it allows the model to jointly attend to information from different representation subspaces at different positions.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.3}$$

Positional Encoding: Since the attention mechanism doesn't take into account sequence order, the information about the token position is injected by using "positional encodings". They have the same dimension as the embeddings so that they can be summed. The original paper uses sine and cosine functions of different frequencies to encode word positions.

Architecture Variations

The original transformer model consists of an encoder to process the input and a decoder to generate the output, which was the case for most neural sequence transduction models at the time. Later augmentations of this architecture include using only the encoder or decoder part to process input and generate an output.

Encoder-Only Transformers: Encoder-only transformers, such as BERT (Bidirectional Encoder Representations from Transformers) [14] and RoBERTa (A Robustly Optimized BERT Pretraining Approach) [15], are designed to process input data and generate a meaningful representation of it. These models are usually used for tasks that require understanding of input text, such as sentence classification, named entity recognition, and semantic similarity assessment. They work by reading the entire input all at once and are optimized to encode the context around each word, leading to a deep bidirectional context understanding.

Decoder-Only Transformers: Decoder-only transformers, like GPT (Generative Pretrained Transformer) [16] and its successors, are predominantly used for generative tasks where the goal is to produce new text based on the given input. These models are auto-regressive, meaning they generate an output one part at a time and use their own predictions as part of the input for subsequent steps. This design is ideal for tasks like language generation, story completion, and any scenario where the flow of text continues from an initial prompt.

Encoder-Decoder Transformers: The original transformer model and those like T5 (Text-to-Text Transfer Transformer) [17] employ both an encoder and a decoder. These types of transformers are versatile, being suitable for a wide range of tasks that involve both understanding the input and generating an output based on that understanding. They are particularly powerful in tasks like machine translation, where the encoder processes the source language and the decoder generates the target language, or in summarization tasks where the encoder comprehends the full document and the decoder produces a concise summary.

The task of question-answering is currently best solved by using decoder-only transformer architectures, so they are the focus of this thesis.

2.2.2 Large Language Model Training

There are three main stages used in LLM training of state-of-the-art decoder-only LLMs:

1. Pretraining: LLM pretraining is done using self-supervised learning, where the model is trained on next token prediction of vast amounts of text data. Given a large corpus $\mathcal{D}$ the training objective is to minimize the following loss:

$$\mathcal{L}_{\text{pretrain}} = \sum_{x \in D} \sum_{i} \log p_{\theta}(x_i | x_1, \ldots, x_{i-1}), \tag{2.4}$$

2. Supervised fine-tuning: Supervised fine-tuning is the process of aligning an LLM to downstream tasks. Instruction tuning is a widely used version of fine-tuning which attempts to bridge the gap between the next token prediction task and user instructions. It is described in more detail in section 2.3.

3. Reinforcement learning from human feedback: RLHF is a model training procedure that is applied to a fine-tuned language model to further align model behavior with human preferences and instruction following.

2.3 Supervised Fine-Tuning

Supervised fine-tuning consists of training a model on input-output pairs for a particular task. This task is again framed as next token prediction, but this time, instead of unlabeled text, the data is formatted as (INPUT, OUTPUT) pairs. The most common example of this is instruction tuning, introduced in [18], which aims to enhance zero-shot learning abilities of LLMs through fine-tuning on a collection of tasks described via instructions. This process can also be described as aligning LLM behaviour to human intent.

Formally, given an instruction dataset $\mathcal{D}' = (I, Y)$, where I represents a task instruction and Y the desired response, the training objective is to minimize the following loss:

$$\mathcal{L}_{\text{ins}} = -\log p_\theta(Y|I) \tag{2.5}$$

2.3.1 Parameter Efficient Fine-tuning

At inference time, all of the LLM's weights need to be loaded onto the GPU. This already poses a problem, as LLMs are, as their name suggests, large. For fine-tuning however, the memory requirements are roughly doubled, as both the model parameters and their respective gradients need to be stored at the same time. These requirements are often prohibitively costly. Therefore, great efforts are being made to make fine-tuning LLMs less resource intensive.

Parameter-Efficient Fine-Tuning (PEFT) methods enable efficient adaptation of pre-trained language models to various downstream applications without fine-tuning all the model's parameters. This decreases the computational and storage costs. State-of-the-art PEFT techniques achieve performance comparable to that of full fine-tuning.

Some of the popular methods include:

Prefix-Tuning: Introduced in [19], prefix tuning is a PEFT method which freezes all LLM parameters and instead optimizes a sequence of continuous tasks-specific vectors which are added as a prefix to the normal task prompt. While this saves a lot of resources, the method is difficult to optimize and effectively shortens the usable input sequence length.

Adapters: Introduced in [20], adapters are small layers added to every Transformer block. These layers are then trained, while the rest of the parameters stay the same. The problem arises, because even though the adapters themselves are very small, they can't be parallelized on hardware and therefore introduce a noticeable latency.

Low-rank Adaptation: Introduced in [21], Low-Rank Adaptation (LoRA) freezes the model weights and injects trainable rank decomposition matrices into each layer of the Transformer architecture. Since this is the method used in this thesis, it is described in more detail in section 2.3.1.

Low-Rank Adaptation

Low-Rank Adaptation (LoRA) [21] is currently one of the most widespread PEFT methods.

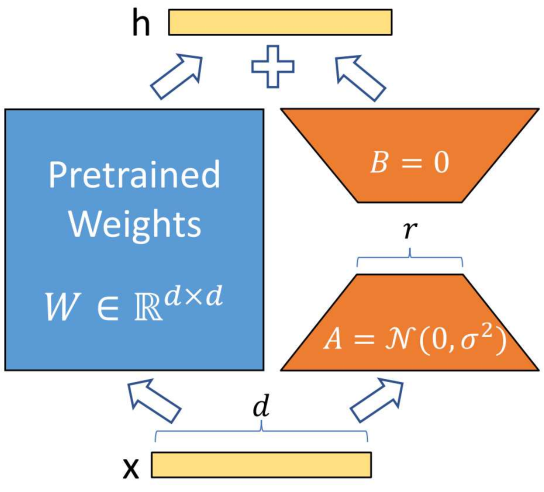
h
Pretrained Weights
$W \in \mathbb{R}^{d\times d}$
$B = 0$
r
$A = \mathcal{N}(0, \sigma^2)$
d
x

Figure 2.2: Reparametrization used in LoRA. Image source: [21]

Instead of modifying all model weights, LoRA freeezes the original model weights and adds a separate set of weights. These weights, after fine-tuning, represent the difference that needs to be added to the original model weights to improve model performance on the fine-tuning task. This separate set of weights uses low-rank decomposition by representing the original matrix with two matrices A and B, which are both smaller than the original weight matrix. If the original matrix is of size dxk, matrix B is of size dxr and matrix A is of size rxk. The actual intrinsic rank of the weight matrix is unknown, so r is a hyperparameter. The matrix A is initialized from a normal distribution and B with zero and backpropagation is used to determine the final values. The figure 2.2 visualizes the described decomposition. The new forward pass is formally written down in equation 2.6.

$$h = W_0x + \Delta Wx = W_0x + BAx \tag{2.6}$$

One of the great advantages of LoRA is that there is no additional latency at inference time, as the LoRA adapter can be merged into the original model so the number of parameters stays the same. This is an improvement over the already mentioned other adapter methods.

It is worth mentioning that the requirements for fine-tuning LLMs can be brought down even more by using model quantization. Model quantization is a technique to reduce computational and memory costs by representing the weights and activations with low-precision data types like 8-bit integers instead of the usual 32-bit floating points. QLoRA is a version of LoRA that allows for efficient finetuning of quantized LLMs, presented in [22]. However, since the available hardware resources were sufficient for unquantized fine-tuning, QLoRA was not used in this thesis.

2.4 In-context Learning

In-context learning (ICL) is a new paradigm in machine learning that refers to the ability of models to learn and make predictions based on the context provided during inference, rather than solely relying on pre-training. This is a significant departure from traditional machine learning where models could only solve problems they were specifically trained on.

The concept of in-context learning was first defined in [23]: *In-context learning is a paradigm that allows language models to learn tasks given only a few examples in the form of demonstration.* This behaviour was demonstrated on GPT-3 where it achieved strong performance on many classic NLP tasks, such as translation and question-answering, as well as some tasks that require on-the-fly reasoning such as unscrambling words or using a novel word in a sentence.

The figure 2.3 presents the process of in-context learning. A small number of labeled data is modified using a template and turned into demonstration examples. These examples are then prepended to the unlabeled sample that we want the model to label. This prompt is then input into the model which processes it without changing any parameters and generates an answer.

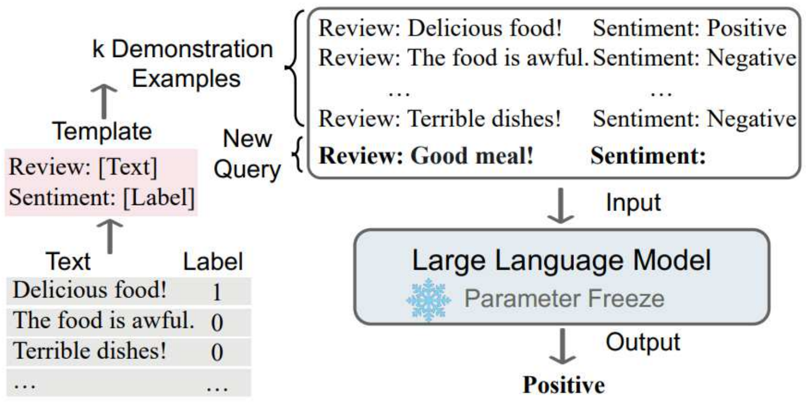

Figure 2.3: Visualization of in-context learning. Image source: [24]

2.4.1 Related Concepts

Prompt learning: According to [24] in-context learning can be regarded as a subclass of prompt learning. Prompt learning, as defined in [25] is taking the original input x and modifying it using a template into a string prompt x'. This prompt is fed into a language model which probabilistically fills in missing information in the prompt and outputs a final string $\hat{x}$. From this string the output y is derived (usually by some postprocessing technique). This is different from traditional machine learning where the input x is used to predict the output y as $P(y|x)$.

Examples of prompt learning include Chain-of-Thought (CoT) prompting [26] and Zero-Shot Chain-of-Thought (CoT) prompting [5]. Both approaches find that prompting LLMs to think step by step improves their complex reasoning abilities. In [26] the authors prompt the model by first providing examples of questions which are answered with the CoT process and then posing the actual question. In [5] the authors achieve similar results by leveraging a simple prompt augmentation: they append the words "Let's think step by step." to the first line of the answer.

Fewshot learning: Another related concept to in-context learning is that of fewshot learning. Fewshot learning is a machine learning paradigm where a model is trained to make accurate predictions with only a small number of examples per class. This differs from in-context learning, as in-context learning is performed directly on pretrained LLMs and doesn't require parameter updates.

2.5 Evaluation

Evaluating Large Language Models (LLMs) presents a complex challenge. Unlike traditional machine learning approaches, where a model is trained and tested on tasks within similar data distributions, LLMs deviate from this norm. These models are typically trained on a vast corpus of data through tasks like next token prediction, and are then expected to excel in diverse tasks that may involve significantly different data distributions. This divergence makes it challenging to encapsulate an LLM's proficiency across varied tasks within a single, or even a limited set of performance metrics. Moreover, the generative aspect of LLMs complicates quantitative assessment. Traditional evaluation metrics, such as F1, BLEU [27] or ROUGE [28], which measure string overlap, are somewhat effective in conventional settings where the output is a direct extraction from provided context. However, in the context of generative models, the spectrum of valid outputs expands dramatically, rendering these metrics less effective. Additionally, the quality of an LLM's output is not solely determined by quantitative content accuracy but also by the qualitative format and coherence of the generated text, adding another layer of complexity to the evaluation process.

Among the challenges connected to the format and coherence of the generated response, we can identify several key problems that frequently arise in the generation of LLM responses:

1. **Hallucination** - Generating responses which are linguistically sound but are factually incorrect, misleading or entirely fictional.

2. **Incorrect output formatting** - Generating responses in an unexpected format. This can greatly lower the usability of LLMs in downstream tasks.

3. **Wrong tone** - Generating responses in a tone that is not appropriate for the desired

use case.

4. **Overly toxic or overly cautious responses** - Generating responses which are inappropriate or refusing the generate a response for fear of being inappropriate.

5. **Repetitiveness** - Generating responses that repeat words or phrases, or that rephrase the same piece of information many times without saying anything new.

In assessing the performance of LLMs, several approaches can be employed, each with its own set of advantages and limitations. Specifically, the following methodologies are commonly utilized in the field:

1. **Human evaluation:** This involves the assessment of LLM outputs by human evaluators, who can provide nuanced judgments on the quality and relevance of responses. Although this approach is often considered the gold standard for its accuracy in capturing human-like understanding, it is typically expensive and challenging to scale, making it less feasible for large datasets or continuous evaluation.

2. **LLM based evaluation:** In this method, other LLMs, such as the state-of-the-art GPT-4, are used to evaluate the outputs of the target LLM. This approach can reduce costs compared to human evaluation and offers a degree of scalability, but it may still incur significant expenses and introduces the risk of propagating biases or errors inherent in the evaluating LLMs.

3. **Automated testing:** This approach leverages predefined tests and benchmarks to assess LLM performance. Automated testing is highly scalable and provides consistent, objective metrics for evaluation. However, it is not without challenges, particularly in capturing the subtleties of language understanding and generation that human evaluators or other LLMs might notice. As this thesis focuses on automated testing, the subsequent section will delve into the specifics of these tests and discuss the inherent challenges in greater detail.

2.5.1 Automated Testing

Automated testing is historically the most used method for evaluating machine learning models. For QA tasks specifically, measures of string overlap, such as exact match and F1 are employed, mostly on answers formulated through context extractions. Many traditional QA datasets are constructed to be used in this setting. This poses a problem for generative QA which can produce many variations of the correct answer, making it difficult to encapsulate the correct answer within the answer labels.

One of the ways question answering evaluation can be adapted for generative models is using multiple-choice datasets. Instead of open-ended questions the model gets asked a question with a set number of choices and is expected to output the letter of the correct answer. Assuming the model is at least good enough to understand the task and follow the given instruction, it becomes easier to conduct generative evaluation. However, differences in implementation of this method are still possible.

In general LLMs work by ingesting a prompt and generating a probability distribution over the vocabulary. The next token is then chosen according to this distribution, for example by choosing the most probable next token. This token is then fed back into the model and used to generate the next token and so on.

The two most common approaches used in evaluation are the intrinsic and extrinsic evaluation, both of which are described in continuation.

Intrinsic Evaluation

Intrinsic evaluation relies on loglikelihood of different answer options. For each answer option, the probabilities of its tokens are gathered in one score and then compared to each other. As LMs actually produce a probability distribution over their vocabulary, this method directly evaluates this distribution, hence it is referred to in this thesis as intrinsic evaluation.

The procedure for calculating the score can be formally written down as follows:

1. For every output option j:
    (a) Set $m_j = 0$.
    (b) For every word w_i in the data point j:
        i. Feed w_i's preceding context, which after the first few words will be the

sequence $w_1, w_2, \ldots, w_{i-1}$, into the language model as input.

   ii. Let p be the probability that the language model assigns to w_i (the correct next word).

   iii. Add $-\log(p)$ to m_j.

  (c) For normalized accuracy divide m_j by the number of tokens N, otherwise the score is m_j.

2. The final answer is the answer with the highest m_j. If this is the correct answer the model gets a point, if not no point is given.

This process is visualized in figure 2.4. There are also two versions of this evaluation method:

Cloze prompting: The input prompt is formatted only with the question and one of the answer options: (QUESTION, ANSWER). This is repeated for each answer option, and the probabilities for each question-answer pair are compared. This means that the answers are not directly compared except through the final probability. Normalization is recommended, as different answer options can have different lengths.

Multiple choice prompting: The input prompt is formatted with the question, all of the possible answers and the one of the answer options: (INPUT, OPTIONS, ANSWER LETTER). This is repeated for each option, and again the probabilities are compared. This version of prompting compares the answer options to each other more directly, as each prompt is aware of the other answer options. Normalization is not required as the length of all of the answers is the same, which is just the letter of the answer.

Pros: The intrinsic quality of the model is measured, there is no confounding factor of the format of the model response. This also allows the evaluation to be fully automated, and requires no model specific pre- or post-processing.

Cons: Access to the probability distribution over the vocabulary is required for this approach, which is not always the case. Furthermore, it is further away from the production setting of a question-answering system.

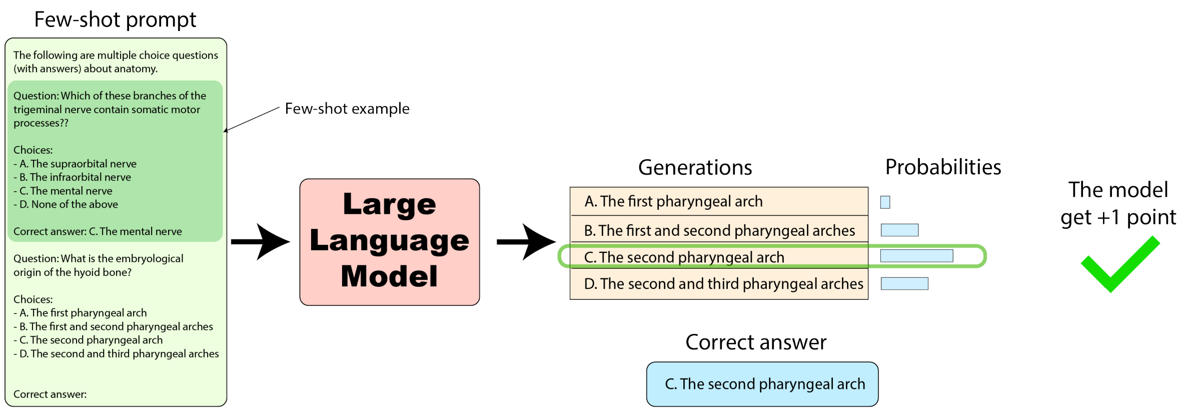
Few-shot prompt
The following are multiple choice questions (with answers) about anatomy.
Question: Which of these branches of the trigeminal nerve contain somatic motor processes??
Choices:
- A. The supraorbital nerve
- B. The infraorbital nerve
- C. The mental nerve
- D. None of the above
Correct answer: C. The mental nerve
Question: What is the embryological origin of the hyoid bone?
Choices:
- A. The first pharyngeal arch
- B. The first and second pharyngeal arches
- C. The second pharyngeal arch
- D. The second and third pharyngeal arches
Correct answer:
Few-shot example
Large Language Model
Generations
Probabilities
A. The first pharyngeal arch
B. The first and second pharyngeal arches
C. The second pharyngeal arch
D. The second and third pharyngeal arches
The model get +1 point
Correct answer
C. The second pharyngeal arch

Figure 2.4: Visualization of the intrinsic evaluation procedure. Image source: [29]

Extrinsic Evaluation

This approach to evaluating LLMs is based on feeding the prompt into the model and letting the model generate the response. This response is then evaluated compared to the correct answer. This process is visualized in figure 2.5. If the model response corresponds to the correct answer, the model gets a point, and if it doesn't, the model gets no points.

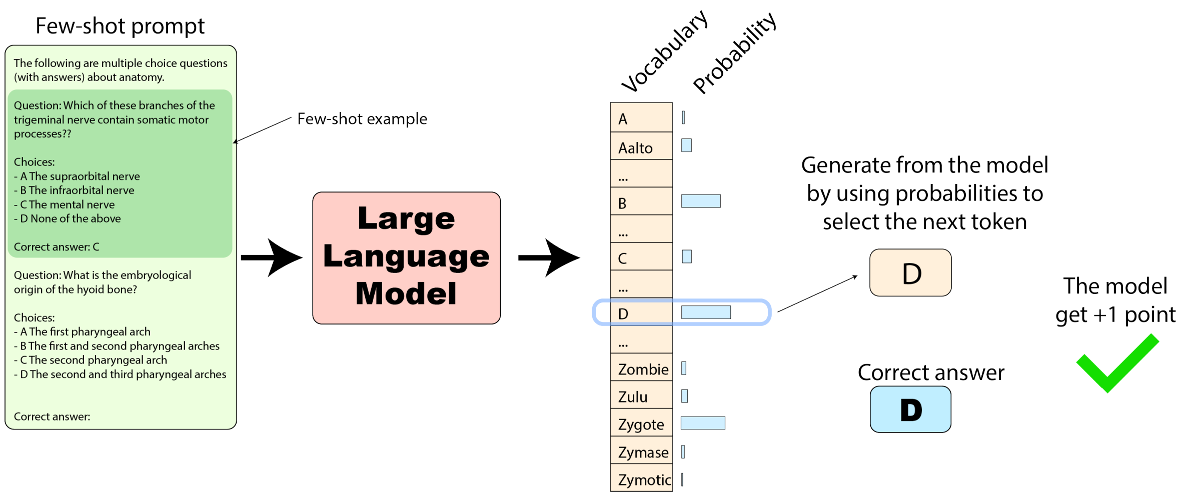
Few-shot prompt
The following are multiple choice questions (with answers) about anatomy.
Question: Which of these branches of the trigeminal nerve contain somatic motor processes??
Choices:
- A The supraorbital nerve
- B The infraorbital nerve
- C The mental nerve
- D None of the above
Correct answer: C
Question: What is the embryological origin of the hyoid bone?
Choices:
- A The first pharyngeal arch
- B The first and second pharyngeal arches
- C The second pharyngeal arch
- D The second and third pharyngeal arches
Correct answer:
Few-shot example
Large Language Model
Vocabulary
Probability
A
Aalto
...
B
...
C
...
D
...
Zombie
Zulu
Zygote
Zymase
Zymotic
Generate from the model by using probabilities to select the next token
D
The model get +1 point
Correct answer
D

Figure 2.5: Visualization of the extrinsic evaluation procedure. Image source: [29]

Pros: The approach doesn't require the model to provide the probability distribution over the vocabulary, but can be treated as a black box instead. Furthermore, this approach is good for evaluating the LLM as a question answering system, as it is more straightforward and closer to how it would be used in a production setting. If there are n answer choices it is also n times faster, as every prompt is processed only once, instead of n times as is the case in intrinsic evaluation.

Cons: The resulting score can be confounded by the format of the model response.

For example, if the model in figure 2.5 had produced a slightly higher probability for the word "Zygote", instead of the letter "D", the output would have been the word "Zygote". This answer would be considered incorrect and the model would not get a point for it. This is problematic because if we are trying to gauge the knowledge capacity of the model, and not its instruction following capabilities, this score would not represent the real situation. To mitigate this problem, model specific prompt engineering and answer post-processing can be conducted, but this can be time consuming, and lowers the extent of automatization of the test method.

Data Contamination

Model developers often don't disclose the data they used to pretrain or fine-tune a model. Even when a model is supposedly open-source, while the weights of the model are freely available, the training procedure and data used are most often not disclosed. Previous work has shown that many widely used LLMs have been trained on copyrighted books and personal information [30]. While many model developers try to prevent the model from memorizing or at least regurgitating memorized texts, novel attacks are developed all the time which result in extracting vast amounts of training data [31], including sensitive information.

The most widely used testing sets for evaluating LLMs are available on the Internet in plain text form with both the question and the solution. With the rise of models using crawled data for training and closed API models using requests sent to it as training data, data contamination has become widespread [32]. Combined with the aforementioned proven ability of LLMs to memorize huge portions of text, this poses a problem as it is considered a data leak for testing.

While some models go to lengths to prevent data contamination [7], many models don't provide any information on their training data or procedure, even when they are open-source. This significantly lowers the reliability of the most widely used benchmarks.

One of the recently proposed solutions to this problem is presented in [33]. The method is built on a straightforward hypothesis: an unobserved example is more likely to have a few outlier words with low probabilities under the LLM, while a recognized example is less inclined to contain words with such reduced probabilities. The method doesn't require any insight into the training data and is already beginning to be used in leaderboards such as the OpenLLMLeaderboard [34].

2.6 Models

This section goes into more detail about three open-source models relevant to this thesis: Llama-2, Platypus and LeoLM, while table 2.1 presents some of the state-of-the-art models.

Table 2.1: Overview of state-of-the-art models.

Model	Parameters	Creator	Source	Architecture
GPT-4	1760B	OpenAI	Closed	Decoder-only
GPT-3.5	175B	OpenAI	Closed	Decoder-only
Llama-2	70B	Meta	Open	Decoder-only
Mixtral	46.7B	MistralAI	Open	Decoder-only
PaLM	540B	Google AI	Closed	Decoder-only
BERT	340M	Google AI	Open	Encoder-only
T5	3B	Google AI	Open	Encoder-decoder

Llama-2

Llama-2 is a series of open-access large language models introduced by Meta in [1]. It represents an evolved iteration of its predecessor, LLaMA. Enhancements include a more robust data cleaning process, an expanded mix of publicly available pretraining data, resulting in a 40% increase in size, an extended context length, and the integration of grouped-query attention to optimize inference efficiency. It is available in three variants: 7B, 13B, and 70B parameters. Table 2.2 presents an overview of the models specifications.

Table 2.2: Overview of the Llama 2 model specifications.

Model size	Pretraining tokens	Context length
7B, 13B, 70B	2 trillion	4096

Training corpus: The training corpus of Llama-2 models includes CommonCrawl, c4, GitHub code, Wikipedia, Books and ArXiv papers, amounting to around 2 trillion tokens.

Language limitation: Table 2.3 presents a summary of the languages utilized in the pretraining process. It is evident that the training corpus primarily comprises of English, with the second category labeled as "unknown" encompassing content such as code. Moreover, all performance and safety evaluations documented upon the model's release are exclusively in English. This focus underscores the predominance of English within the LLM's development and assessment stages.

Language	Percent	Language	Percent
en	89.70%	uk	0.07%
unknown	8.38%	ko	0.06%
de	0.17%	ca	0.04%
fr	0.16%	sr	0.04%
sv	0.15%	id	0.03%
zh	0.13%	cs	0.03%
es	0.13%	fi	0.03%
ru	0.13%	hu	0.03%
nl	0.12%	no	0.03%
it	0.11%	ro	0.03%
ja	0.10%	bg	0.02%
pl	0.09%	da	0.02%
pt	0.09%	sl	0.01%
vi	0.08%	hr	0.01%

Table 2.3: Language distribution in pretraining data with percentage >= 0.005%. Source: [1]

Platypus

The Platypus model, introduced in [7], is an open-source model based on Llama-2 13B and 70B models that achieved the top position on HuggingFace's Open LLM Leaderboard [34] at the time of its release. The main contributions from the paper include:

1. **Open-Platypus Dataset:** Platypus relies on the Open-Platypus dataset, a carefully curated subset of open datasets released to the public.

2. **Fine-Tuning and Merging LoRA Modules:** Platypus uses LoRA modules for fine-tuning and merging, preserving pretraining priors while incorporating domain-specific knowledge.

3. **Test Data Leak Detection:** Platypus undergoes rigorous testing to detect and address potential test data leaks and contamination, ensuring reliable performance metrics.

Platypus demonstrates strong quantitative performance across various metrics and model sizes, despite using less fine-tuning data and computational resources compared to other models. For example, a 13B Platypus model was trained on a single A100 GPU using 25k questions in just 5 hours. This is the reason the Platypus model is used as a significant inspiration for the fine-tuning part of the thesis.

LeoLM

LeoLM (Linguistically Enhanced Open Language Model) [9] is a suite of German-language foundation language models based on Llama-2. The training procedure for LeoLM consists of stage 2 pretraining and fine-tuning. Stage 2 pretraining refers to continued training of the full Llama-2 model on a German text corpus of 65 billion tokens. They also take care in hyperparameter choice to minimize the risk of forgetting already existing knowledge. For fine-tuning they take partial inspiration from the Platypus model, mentioned in section **??** and translate the OpenPlatypus dataset into German, along with an assortment of other high quality datasets.

2.7 Datasets

This section covers datasets used for fine-tuning Llama-2 and evaluating the fine-tuned models.

ARC

The ARC (Advanced Reasoning Challenge) dataset, presented in [35], consists of natural, grade-school science questions. It is partitioned into an Easy and Challenge Set, where the Challenge Set contains only questions answered incorrectly by both a retrieval-based algorithm and a word co-occurence algorithm. For the purposes of this thesis only the Challenge Set is used. The questions are multiple choice with one correct answer. An example question can be found in 2.6

MMLU

The MMLU (Massive Multitask Language Understanding) dataset, introduced in [36], covers 57 tasks including elementary mathematics, US history, computer science, and

Which property of a mineral can be determined just by looking at it?
(A) luster
(B) mass
(C) weight
(D) hardness

Figure 2.6: Example from the ARC Challenge task.

more. The questions are multiple choice with one correct answer. An example question can be found in figure 2.7.

One of the reasons that the government discourages and regulates monopolies is that
(A) producer surplus is lost and consumer surplus is gained.
(B) monopoly prices ensure productive efficiency but cost society allocative efficiency.
(C) monopoly firms do not engage in significant research and development.
(D) consumer surplus is lost with higher prices and lower levels of output.

Figure 2.7: Example from the Microeconomics MMLU task.

HellaSwag

The HellaSwag dataset, introduced in [37], comprises of sentences along with multiple potential follow-up sentences for each. The objective is to select the most probable follow-up sentence. An example question can be found in 2.8

A woman is outside with a bucket and a dog. The dog is running around trying to avoid a bath. She…
(0) rinses the bucket off with soap and blow dry the dog's head.
(1) uses a hose to keep it from getting soapy.
(2) gets the dog wet, then it runs away again.
(3) gets into a bath tub with the dog.

Figure 2.8: Example from the HellaSwag task.

OpenPlatypus

The OpenPlatypus dataset, introduced in [7], is a small-scale dataset of curated subselection of public text datasets. The dataset is focused on improving LLMs' STEM and logic knowledge. It consists of mainly human-designed questions with only 10% of the questions being generated by an LLM.

An important aspect of the dataset is the contamination check which aims to prevent benchmark test questions from being part of the training data. As presented in 2.5.1, data contamination is a common problem in LLM training. To manage potential leaks, they crafted heuristics for manually filtering questions from Open-Platypus with over 80% cosine embedding similarity to benchmark questions.

INSTRUCTION:
Erika, who is 14 years old, flips a fair coin whose sides are labeled 10 and 20, and then she adds the number on the top of the flipped coin to the number she rolls on a standard die. What is the probability that the sum equals her age in years? Express your answer as a common fraction.
OUTPUT:
The only way for the sum to be a 14 is for her coin flip to be a 10 and for her roll to be a 4. This can only occur in $\frac{1}{2} \cdot \frac{1}{6} = \frac{1}{12}$.

Figure 2.9: An example of an instruction and output pair from the OpenPlatypus dataset.

HotpotQA

HotpotQA is a question answering dataset, presented in [38]. It consits of 113k Wikipedia-based question-answer pairs. The key features of the dataset are: (1) the questions require finding and reasoning over multiple supporting documents to answer; (2) the questions are diverse and not constrained to any pre-existing knowledge bases or knowledge schemas (3) the dataset also provides sentence-level supporting facts required for reasoning.

QUESTION:
Which magazine was started first Arthur's Magazine or First for Women?
CONTEXT:
First for Women is a woman's magazine published by Bauer Media Group in the USA. The magazine was started in 1989. It is based in Englewood Cliffs, New Jersey. In 2011 the circulation of the magazine was 1,310,696 copies.
Arthur's Magazine (1844–1846) was an American literary periodical published in Philadelphia in the 19th century. Edited by T.S. Arthur, it featured work by Edgar A. Poe, J.H. Ingraham, Sarah Josepha Hale, Thomas G. Spear, and others. In May 1846 it was merged into Ödey's Lady's Book.
ANSWER:
Arthur's Magazine

Figure 2.10: An example of a question with context from the HotpotQA dataset.

German Translations

The German translations of the ARC, MMLU, HellaSwag and OpenPlatypus datasets were done by the LAION team[1]. They used GPT-3.5-turbo to translate the original prompts and manually fixed the broken examples. Using GPT-3.5-turbo ensures that the context between prompts and responses remains intact and that complex instructions are accurately translated.

The translation of HotpotQA was done using Google TranslateAPI. This option was chosen because of the availability of this resource, although it turned out GPT-3.5-turbo would have been a better option. The issues encountered are discussed in more detail in 3.3.3.

[1]https://laion.ai/

3 Experiments and Results

This chapter begins with an overview of prior research on language capability transfer, followed by a brief examination of feasibility of German language transfer and an overview of the experimental framework. Then, it delves into the detailed description and presentation of results for the two primary approaches investigated in this thesis: the in-context learning approach and the fine-tuning approach.

3.1 Previous Works

Even though the majority of LLMs are predominately trained and tested in English, many of them can comprehend input and generate output in other languages to some extent as well. This ability is greatly dependant on the amount of training data from that language and the script of the language (where the Latin and Cyrillic scripts are usually better represented).

Previous efforts in transferring language capability have mainly focused on three approaches:

1. **Vocabulary extension:** Extending the model vocabulary with tokens of the target language to enhance the encoding expressiveness of that language.
2. **Further pretraining:** Continuing the pretraining stage of an LLM to enhance language capabilities in the target language. Usually the scale this is done on is in the order of billions of tokens which is much less then the trillions used for the original pretraining.
3. **Instruction tuning:** Performing only instruction tuning in the target language to enhance instruction-following capabilities in that language.

Both vocabulary extension and further pretraining are very resource intensive. The focus of this thesis will be on less resource intensive approaches, namely in-context learning and instruction tuning in the target language.

3.2 German Capability Transfer Feasibility

Since this thesis mainly focuses on enhancing LLM's German capabilities using the Llama-2 model, the focus of the language capabilities transfer exploration is the transfer to the German language. As mentioned in 2.6, while Llama-2 has been trained on an overwhelmingly English corpus, the second most represented language in the training data is German. This means that if language transfer is possible, German is the best candidate.

One of the first aspects to consider when trying to transfer capabilities of a languge model to another language is the vocabulary of the pretrained model. Since the vocabulary is fixed at the beginning of the pretraining process, if it doesn't allow meaningful tokenization it would be expected for an LLM to not work well in that language, at least without a vocabulary extension. However, since German uses the same script as English and is in the same linguistic group it is reasonable to expect that the vocabulary of Llama-2 will be sufficiently compatible with German.

A good indicator of vocabulary compatibility is the number of tokens a word gets split into. A well fitted vocabulary is good at recognizing meaningful word subparts.

```
English: ['_The', '_curious', '_f', 'eline', '_obser', 'ves', '_the', '_ser', 'ene', '_landscape', '.’]
German:  ['_Die', '_ne', 'ug', 'ier', 'ige', '_Kat', 'ze', '_be', 'ob', 'acht', 'et', '_die', '_ru', 'h', 'ige', '_Land', 'schaft', '.’]
Greek:   ['_', 'Η', '_', 'π', 'ε', 'ρ', 'ί', 'ε', 'ρ', 'γ', 'η', '_', 'γ', 'ά', 'τ', 'α', '_', 'π', 'α', 'ρ', 'α', 'τ', 'η', 'ρ', 'ε’,
           'ί', '_', 'τ', 'ο', '_', 'γ', 'α', 'λ', 'ή', 'ν', 'ι', 'ο', '_', 'τ', 'ο', 'π', 'ί', 'ο', '.']
```

Figure 3.1: Example of Llama-2 tokenization of the same sentence: "The curious feline observes the serene landscape." in English, German and Greek.

As demonstrated in the example 3.1, the Llama-2 tokenizer manages to identify meaningful word subcomponents within both English and German sentences. However, for Greek, the segmentation into individual letters indicates a poor fit with the existing vocabulary. Since expanding the vocabulary necessitates substantial volumes of data to integrate new tokens into an established LLM, such expansion is out of the scope of this thesis and is also likely unnecessary anyway.

3.3 Experimental Setup

This section goes over the specifics of the setup used in both in-context learning and fine-tuning experiments.

3.3.1 Hardware

All experiments conducted in this thesis are done on a Google Cloud Compute Engine using 8 NVIDIA L4 GPUs, with 24GB of RAM each.

3.3.2 vLLM

To speed up inference during evaluation, the vLLM library[1] is used. This library incorporates the PagedAttention algorithm to tackle the issue of memory waste in the key-value cache (KV cache) of existing systems. Such waste, often caused by fragmentation and duplication, restricts batch sizes. The PagedAttention algorithm, drawing inspiration from virtual memory and paging methods in operating systems, enables vLLM to minimize waste in KV cache memory significantly. Moreover, it facilitates the flexible allocation and sharing of KV cache both within and across different requests. Consequently, vLLM enhances the throughput of popular LLMs by 2 to 4 times.

3.3.3 Data Preprocessing

The datasets used in the training and evaluation are presented in section 2.7. As already mentioned, most datasets were already freely available on the Internet so they are used in their original form with minimal pre-processing.

The pre-processing procedure used was as follows:

1. If there are multiple categories of data in a dataset (such as the subjects in MMLU) bin the samples from the same category together.

2. Randomly sample an equal sample from each bin, such that the total amount of samples is 500 (for intrinsic evaluation) and 1000 (for extrinsic evaluation).

[1]https://docs.vllm.ai/

The reason for using subsamples of the original datasets is that some of the original datasets are huge in size (HotpotQA for example has over 100k samples) and would take an enormous amount of time to process. This is both infeasible and unnecessary [39].

For HotpotQA no publicly available German translation was found. Therefore a translated version of the dataset is created using Google Translate API. Preliminary tests were conducted in which it was assured that named entities got proper translation into the German language. For example, the book/movie series "The Hunger Games" is correctly translated to "Die Tribute von Panem" as this is the correct translation in German, even though it is not a direct translation.

It is important to note however, that while this method was employed because of the availability of the resource, in hindsight it would have been better to use chatgpt-3.5-turbo API to translate the dataset. This is because the consistency between the entities in the question, context and answer is better preserved by correctly prompting an LLM. Figure 3.2 presents an example of a broken translation. While some of these examples were fixed by hand, it is certain that many of them have remained broken. However, as the most likely outcome is that none of the models will correctly answer the broken samples, the score is skewed but evaluation can still be valuable.

QUESTION:
Wie heißt das Kampflied der Universität, deren Hauptcampus in Lawrence, Kansas liegt und deren Zweigcampusse im Großraum Kansas City liegen?
CONTEXT:
Kansas Song (We're From Kansas) ist ein Kampflied der University of Kansas...
ANSWER:
Kansas-Lied

Figure 3.2: An example of broken translation. While in the context, the name of the song is kept in the original English language, in the answer the name of the song is translated to German. This is because the answer was translated separately from the context so the translator model didn't have the necessary context cues to infer that the answer is actually an entity.

3.3.4 Base Model Choice

The base model used in this thesis is Llama-2 13B model by Meta, described in more detail in section 2.6. Llama-2 models are some of the strongest open-source models available today, so they are a good base for further improvements. The 13B version was chosen over the 7B because the available hardware allowed for a larger model, while the 70B model was too large.

3.3.5 Evaluation Frameworks

As explained in section 2.5 there are generally two approaches to evaluating an LLM. While loglikelihood based evaluation is measuring the intrinsic quality of the model, the topic of this thesis is using LLMs as question answering systems. The goal of the evaluations conducted in this thesis is assessing both the intrinsic quality of the models in German and the quality of the model in the extrinsic setting of a German question answering system. Therefore, both evaluation methods were employed.

LM Evaluation Harness

Intrinsic model evaluation relies on the LM Evaluation Harness [40], accessible through a public code repository. This repository also serves as the foundation for one of the most popular LLM leaderboards, the Open LLM Leaderboard [34] hosted on HuggingFace. Task configurations for evaluating both English and German models remain unaltered, with dataset paths being the only parameters adjusted to accommodate subsampled versions. The metric used in this evaluation approach is the intrinsic evaluation method, described in more detail in section 2.5.1. For the ARC and HellaSwag tasks, normalized cloze prompting is used, while MMLU uses multiple choice prompting.

Custom Evaluation Framework

The custom evaluation framework generates model outputs, followed by a postprocessing stage to obtain the final answer. In initial testing, a compilation of phrases that typically precede model answers was created. The full list of these phrases is provided in the appendix A.2. The postprocessing procedure is outlined as follows:

1. Retrieve the model output.

2. Eliminate leading and trailing whitespace.

3. Determine the presence of a common answer prefix within the model's response.

   (a) For tasks involving multiple choice formats (i.e. ARC, HellaSwag, and MMLU): If a prefix is identified, extract the first non-whitespace character following the prefix. Otherwise, extract solely the initial non-whitespace character from the output.

(b) For extractive question-answering tasks (i.e. HotpotQA): If a prefix is detected, extract the text following the prefix up to two consecutive newline characters (or up to the end of the string if none are present). If absent, extract the text from the beginning of the output up to two consecutive newline characters.

This approach was devised after initial tests and attempts to ensure the accurate extraction of model responses across various task formats. While it is not exhaustive in its approach and both false positives and false negatives are possible, it presents a setting that would be feasible in production. The metrics used in this evaluation setting were quasi-exact match for multiple choice questions and both quasi-exact match and F1 for open answer questions. These metrics are described in more detail in the appendix A.2.

3.4 In-context Learning Approach

In-context learning is a more straightforward and resource-efficient approach compared to fine-tuning. This section explores the experiments conducted within the domain of in-context learning, beginning with preliminary tests that lay the groundwork for a more in-depth analysis of the in-context learning results.

3.4.1 Preliminary Tests

As mentioned in 2.4, in-context learning can be considered a subclass of prompt learning. Table 3.1 gives an overview of the prompts used for the experiments, while table 3.1 presents their results.

As depicted in Table 3.2, both the Zero-shot CoT and the Base prompt in English exhibit inferior performance compared to random guessing, particularly noticeable in HellaSwag, where model generations were so unrelated to the task that no score was generated. This discrepancy may stem from the inherent similarity between the HellaSwag task of sentence continuation and the LLM pretraining objective, where introducing a CoT prompt confused the model. A similar scenario unfolded with the English prompt,

Table 3.1: Overview of prompts used in preliminary tests. The answer prefix refers to the string "Antwort:".

Prompt name	**Description**
Base prompt	Task description in German followed by the question, choices (each choice is presented in a new line) or context and the answer prefix.
Zero-shot CoT	The same as Base prompt, with the first line of the answer being "Denken wir Schritt für Schritt." (*engl. Let's think step by step.*).
Base prompt - EN	The same as Base prompt with the task description and keywords written in English.
Random baseline	Since all multiple choices datasets have 4 answer choices, the random guess would achieve 25% accuracy. HotpotQA questions are open ended, so there is no random baseline.
Base prompt - DE - one line	The same as Base prompt but with all the choices being presented in one line.
Llama-2 chat prompt	The same as Base prompt but uses the recommended Llama-2 prompt style provided in A.1 with no answer prefix.
Few-shot (3-shot)	No task description is provided, only three examples of the task with the correct answers before the actual task.

resulting in answers that mixed English and German, failing to yield coherent outputs. However, HotpotQA stands as an exception, albeit with a relatively low score.

In comparing the Base prompt in German, where all options in the multiple-choice question are presented in the same line, with the prompt in which every option is in a new line, we observe an average 7% decrease in the score, signifying a notable difference. This variation serves as a good demonstration of the delicate nature of LLM prompting strategies.

The best overall strategy proved to be few-shot prompting with the exception of HellaSwag where the best result was achieved using the recommended Llama-2 chat prompt. This again underscores the importance of testing different prompts for each specific task.

Another notable distinction among the various prompting strategies is the consistency of answer formats. The most consistent responses were obtained through the few-shot prompting approach, as the model typically generated correct answers at the beginning of the output, using the same format as the few-shot prompts and didn't require additional postprocessing. Conversely, prompts that performed below a random baseline typically lacked a standardized structure in their responses, consequently lowering their scores. The Llama-2 chat prompt tended to render the model excessively verbose, resulting in answers that, while often accurate and containing some common prefixes, frequently veered off-topic unnecessarily.

Table 3.2: Preliminary results. The metric for ARC_de, MMLU_de, HellaSwag_de is quasi-exact match, and for HotpotQA_de both quasi-exact match and F1 are displayed.

	ARC_de	MMLU_de	HellaSwag_de	HotpotQA_de
Zero-Shot CoT	22.75	12.0	x	1.0 | 11.05
Base prompt - EN	24.55	13.1	23.9	10.6 | 24.34
Random baseline	25.0	25.0	25.0	N/A
Base prompt - DE - one line	43.88	34.9	23.2	N/A
Llama-2 chat prompt	46.53	37.4	**44.1**	6.0 | 20.79
Base prompt - DE	48.67	41.8	32.6	10.0 | 30.0
Few-shot (3-shot)	**59.11**	**46.0**	36.1	**30.3 | 43.84**

3.4.2 Results

As demonstrated in section 3.4.1, few-shot prompting proved to be the overall best method for question-answering, in both the accuracy and consistency of the answer format. Therefore, a more in-depth search of the hyperparameter space is conducted.

Table 3.3: In-context learning results. The metric for ARC_de, MMLU_de, HellaSwag_de is quasi-exact match, and for HotpotQA_de both quasi-exact match and F1 are displayed.

	0-shot	1-shot	2-shot	3-shot	4-shot	5-shot
ARC_de	52.86	56.03	54.83	**59.11**	57.82	58.76
MMLU_de	44.1	43.9	44.1	**46.0**	45.4	44.8
Hellaswag_de	25.6	26.4	29.6	32.6	**40.2**	35.0
HotpotQA_de	x	25.9 | 41.36	29.9 | 43.61	30.3 | 43.84	**30.8 | 44.60**	29.8 | 43.0

Table 3.3 presents the results using 0- to 5-shot prompts for each test set. As can be seen, the best results were obtained using 3- or 4-shot prompts. For 0-shot HotpotQA_de no score was generated as the outputs were completely unrelated to the task. The results suggest that while adding the examples in the prompt increases the score, largely through unifying the model output format, there is a limit on how much a score can be improved using this technique.

3.5 Fine-tuning Approach

The fine-tuning process, although more resource-demanding than in-context learning, is far less demanding than further pretrainig or building a foundational model. Therefore it holds promise for enhancing model performance and adaptability in a resource-efficient manner. This section describes the fine-tuning experiments done in the scope of this thesis.

3.5.1 Previous Work

The experimental setup of the fine-tuning approach takes inspiration from the following works:

LIMA: Less Is More for Alignment paper: The authors of [6] fine-tune the LLaMa language model with only 1000 carefully curated prompts and achieve remarkably strong performance. They argue that almost all knowledge in large language models is learned during pretraining and only limited instruction tuning is necessary to teach models to produce high quality output.

LLaMA Beyond English: An Empirical Study on Language Capability Transfer paper: The paper [8] conducts extensive investigation of language transfer abilities based on LLaMa. They demonstrate that state-of-the-art transfers can be achieved using less than 1% of the pretraining data, both in terms of knowledge alignment and response quality.

Platypus: Quick, Cheap, and Powerful Refinement of LLMs paper: The paper [7] is described in more detail in section 2.6.

Various community GitHub repositories: Various GitHub repositories such as: Cabrita[2], an attempt at fine-tuning Llama-2 7B for Portuguese and Zicklein[3], an attempt at doing the same in German.

In prior research, there is evidence of the potential of LLMs to align effectively with limited data, thereby enhancing model performance across diverse tasks. While much of this research has primarily focused on English tasks, this thesis aims to investigate whether comparable performance enhancements can be attained through language fine-tuning techniques.

[2]https://github.com/22-hours/cabrita
[3]https://github.com/avocardio/Zickleinzicklein-a-german-finetuned-instruction-llama-

3.5.2 Experimental Setup

Table 3.4: Overview of fine-tuning hyperparameters.

Hyperparameter	Value
learning rate	4e-4
batch size	64
warmup steps	100
epochs	1
lr scheduler	cosine
lora alpha	16
lora rank	16
lora dropout	0.05
lora target modules	gate_proj, up_proj, down_proj
group by length	True
train on inputs	True

Table 3.4 gives an overview of the hyperparameters used for fine-tuning of the final model. They are largely inspired by the hyperparameters used in the Platypus paper [7], with a notable exception being that while they didn't train on inputs, preliminary tests showed that better convergence for German fine-tuning is achieved by also training on inputs. Additionally, the inputs were grouped by length as this proved beneficial, accelerating the training process without inducing any unusual loss curves (which was the reason why the Platypus paper didn't use this hyperparameter). Figure 3.3 presents the prompt used for formatting the fine-tuning instructions. The fine-tuning was done using the LoRA PEFT method, described in 2.3.1.

```
Nachfolgend finden Sie eine Anweisung, die eine Aufgabe beschreibt.
Schreiben Sie eine Antwort, die die Anfrage angemessen vervollständigt.

### Anweisung:
{instruction}

### Antwort:
{answer}
```

Figure 3.3: Fine-tuning prompt.

Figure 3.4 presents the training loss. There is a sharp initial fall in loss, which is typical for instruction-tuning, followed by a slow but steady decline. This usually corresponds to the model essentially learning the instruction prompt format from the first few hundred examples.

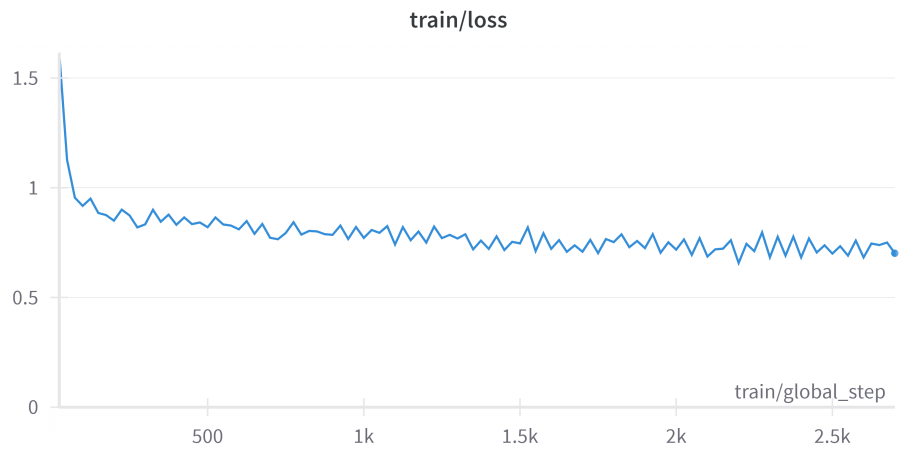

Figure 3.4: Fine-tuning loss

The fine-tuning was done in around 5 hours using the 8 NVIDIA L4 GPUs.

3.5.3 Results

The model is evaluated in two parts: using intrinsic and extrinsic evaluation metrics.

Intrinsic Evaluation Results

Table 3.5 presents the results of the intrinsic evaluation of the fine-tuned model, in both German and English. The results are also compared to the baseline Llama-2 13B model and the LeoLM model. The fine-tuned model scores better than the baseline but is outperformed by the LeoLM model, which is not suprising as LeoLM has been trained with significantly more data. The biggest difference can be observed with the HellaSwag task, which is natively the task which is the most similar to next token prediction. A big improvement on this task might mean that the model does understand the intrinsic structure of the German language a lot more.

Regarding the English part of the assessment, as expected some degradation in results is present. Since the fine-tuned model was only fine-tuned in German, even though its weights are changed much less than LeoLM, it experiences more English performance degradation. This underscores the speed at which the model can forget and shows the importance of joint learning that helps prevent English degradation.

Table 3.5: Fine tuning results - intrinsic evaluation

		Average	ARC	HellaSwag	MMLU
DE	Llama 2 13B	49.87	47.70	55.40	47.20
	Fine-tuned 13B	51.00	**48.80**	57.20	47.00
	LeoLM 13B	**54.73**	48.60	**65.80**	**49.80**
EN	Llama 2 13B	**62.00**	**59.80**	**74.00**	**52.20**
	Fine-tuned 13B	58.86	52.60	71.60	52.40
	LeoLM	60.80	57.60	73.20	51.60

Extrinsic Evaluation Results

Table 3.6 presents the results of extrinsic evaluation. The setting used for evaluation was 3-shot prompting, which showed promising results in preliminary tests of the in-context learning experiments 3.4.1. The extrinsic results paint a starkly different picture to the intrinsic results. The fine-tuned model scores worse than both the original Llama-2 chat model and the LeoLM on all tasks except HotpotQA_de where it is better than LeoLM. LeoLM on the other hand scores better then the original Llama-2 chat, however the margins are very small, and is considerably worse on the HotpotQA_de task.

Table 3.6: Fine tuning results - extrinsic evaluation using 3-shot prompting.

	ARC_de	MMLU_de	HellaSwag_de	HotpotQA_de
Llama-2 chat	59.11	46.0	32.6	**30.3 | 43.84**
Fine-tuned 13B	55.68	44.1	32.2	24.8 | 35.44
LeoLM	**60.39**	**46.3**	**32.8**	19.0 | 27.88

The increase in performance in the LeoLM model is mostly expected, while the decrease in performance for the fine-tuned model is somewhat disappointing. A possible explanation for this behaviour may be that there is insufficient training data to actually improve German capabilities but there is enough data to disrupt the existing alignment. Interestingly, even with a lot more training, LeoLM manages to improve the score on three out of four tasks only by a slight margin.

3.6 Discussion

While the results of the fine-tuning approach are disappointing, they can give insights about the state of other languages in LLMs trained predominately in English.

As explained in section 2.3, the goal of instruction-tuning is improving zero-shot model performance. While the main evaluation presented in section 3.5.3 was done using few-shot prompting as it produced the best results in preliminary tests in section 3.4.1, a comparison of zero-shot model performance is given in table 3.7. The results point to the fact that the fine-tuned model did learn something, as it outperforms the other two models in a zero-shot setting in all tasks except HellaSwag_de. The HellaSwag_de achieves the best score on LeoLM, outperforming even the few-shot evaluation setting. This is analogous to the results for Llama-2 chat in the preliminary experiments in section 3.4.1 where the best results for HellaSwag_de were achieved using the recommended model prompt.

Table 3.7: Zero-shot model performance comparison.

	ARC_de	MMLU_de	HellaSwag_de	HotpotQA_de
Llama-2 chat	46.53	37.4	44.1	6.0 | 20.79
Fine-tuned	**47.73**	**39.4**	32.5	**27.7 | 39.58**
LeoLM	41.91	36.7	**60.0**	17.0 | 31.93

As was seen in experiments conducted in 3.5.3 intrinsic and extrinsic evaluation can give quite different results. Intrinsic LLM evaluation, which is the backbone of one of the most popular LLM leaderboards, the OpenLLMLeaderboard [34], is a good indicator of general model quality, but it tells only a small part of the story. Every LLM task can have nuances that are best discovered by evaluating the LLM directly on the specific task. While there is a lot of overlap between the question answering task and general LLM model evaluation as it usually gets framed as a question answering task, differences can prove vital in a production setting.

Finally, even the supposedly open-source models don't always disclose their training corpus, and even if they did and all information about LLM training was disclosed, it would still be impossible (at the current stage of LLM development and understanding) to clearly trace why an LLM produces one output over another. Therefore, while there are some general trends that can be observed, LLMs still require extensive experimenting to maximize their performance.

3.7 Future Work

As explained in 3.2, German is the second most prevalent language after English in the Llama-2 pretraining corpus and is also linguistically very similar to English. A possible direction for future work would be to use languages which are less present in the pretraining corpus and observe if they can achieve better increases in performance with small amounts of training data.

This thesis dedicated significant attention to the formatting of answers. The reason for this was that incorrectly formatted answers or answers with hard to parse outputs both lower the LLM score and are inconvenient for use in production settings. So far, no significant research was done in quantifying the correctness of answer formatting, even though this would prove beneficial when evaluating the quality of LLMs. This is another possible direction for future work.

Furthermore, while some prior research exists on mitigating forgetting in LLM fine-tuning, more investigation into preventing language-specific forgetting, particularly when fine-tuning with small high-quality datasets, presents a possible research topic.

Finally, this thesis was done using the 13B version of Llama-2. While observations on smaller models can often be extended to bigger ones, increasing model size sometimes results in unexpected changes of model behaviour. Additionally, the continual release of open-source LLM architectures presents an opportunity to explore language capabilities of models other than Llama-2.

4 Conclusion

The aim of this thesis was to explore the state-of-the-art of question answering systems, specifically based on LLMs. Special focus was given on systems based on a non-English language, namely German.

For assessing the quality of QA systems a custom evaluation framework for extrinsic QA performance was implemented spanning three closed-book and one open-book QA task. Intrinsic evaluation was used for assessing the raw model quality and understanding of German. For this, the LM Evaluation Harness [40] code was employed and evaluated on three tasks with a loglikelihood based metric.

Two main low-resource approaches were used for enhancing the German question answering capabilities of the chosen base model, Llama-2 chat: in-context learning and fine-tuning.

Prompt learning and its sub-technique, in-context learning, are techniques which enable the model to learn from context without updating the model weights. The experiments conducted in the scope of this approach revealed that the few-shot prompts, especially the 3- and 4-shot prompts produced the best results overall, both in terms of quality and output formatting consistency. However, there was a notable exception for one of the tasks, in which the recommended prompt achieved the best result, which reinstates the importance of testing different prompting techniques for a each specific task.

The fine-tuning approach was done using a small high quality dataset [7] that in English produced excellent results with a very short training time. The German version of this dataset was used. While the intrinsic evaluation revealed that the model managed to learn some structure of the German language, the extrinsic evaluation using few-shot in-context learning showed that the fine-tuned model was outperformed by the base model for the QA downstream task. Interestingly, while few-shot prompting was the best approach overall and was used for evaluating the fine-tuned model for this reason, a deeper analysis of results revealed that fine-tuning did manage to improve zero-shot and answer formatting capabilities of the model. The results were also compared to LeoLM, a model that underwent continued pretraining in German, alongside fine-tuning. This model outperformed both the fine-tuned and baseline model on both intrinsic and extrinsic few-shot evaluation.

The described experiments show that while some trends in LLM prompting exist, it is still important to test different prompting techniques and their effects on both answer quality and formatting. Additionally, while fine-tuning might not be feasible for language transfer in LLMs, it could still enhance zero-shot capabilities, even in non-English languages.

References

[1] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom, "Llama 2: Open foundation and fine-tuned chat models," 2023.

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

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

[4] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du,

B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel, "Palm: Scaling language modeling with pathways," 2022.

[5] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," 2023.

[6] C. Zhou, P. Liu, P. Xu, S. Iyer, J. Sun, Y. Mao, X. Ma, A. Efrat, P. Yu, L. Yu, S. Zhang, G. Ghosh, M. Lewis, L. Zettlemoyer, and O. Levy, "Lima: Less is more for alignment," 2023.

[7] A. N. Lee, C. J. Hunter, and N. Ruiz, "Platypus: Quick, cheap, and powerful refinement of llms," 2023.

[8] J. Zhao, Z. Zhang, L. Gao, Q. Zhang, T. Gui, and X. Huang, "Llama beyond english: An empirical study on language capability transfer," 2024.

[9] B. Plüster, "Leolm: Igniting german-language llm research," Sep 2023. [Online]. Available: https://laion.ai/blog/leo-lm/

[10] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," vol. 3, no. null, p. 1137–1155, mar 2003.

[11] A. Graves, "Generating sequences with recurrent neural networks," 2014.

[12] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," 2014.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[15] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," 2019.

[16] A. Radford and K. Narasimhan, "Improving language understanding by generative pre-training," 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:49313245

[17] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," 2023.

[18] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, "Finetuned language models are zero-shot learners," 2022.

[19] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds. Online: Association for Computational Linguistics, Aug. 2021, pp. 4582–4597. https://doi.org/10.18653/v1/2021.acl-long.353

[20] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp," 2019.

[21] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," 2021.

[22] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient fine-tuning of quantized llms," 2023.

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

[24] Q. Dong, L. Li, D. Dai, C. Zheng, Z. Wu, B. Chang, X. Sun, J. Xu, L. Li, and Z. Sui, "A survey on in-context learning," 2023.

[25] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," 2021.

[26] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," 2023.

[27] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, P. Isabelle, E. Charniak, and D. Lin, Eds. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, Jul. 2002, pp. 311–318. https://doi.org/10.3115/1073083.1073135

[28] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81. [Online]. Available: https://aclanthology.org/W04-1013

[29] C. Fourrier, N. Habib, J. Launay, and T. Wolf, "What's going on with the open llm leaderboard?" Jun 2023. [Online]. Available: https://huggingface.co/blog/evaluating-mmlu-leaderboard

[30] M. Mozes, X. He, B. Kleinberg, and L. D. Griffin, "Use of llms for illicit purposes: Threats, prevention measures, and vulnerabilities," 2023.

[31] M. Nasr, N. Carlini, J. Hayase, M. Jagielski, A. F. Cooper, D. Ippolito, C. A. Choquette-Choo, E. Wallace, F. Tramèr, and K. Lee, "Scalable extraction of training data from (production) language models," 2023.

[32] A. Jacovi, A. Caciularu, O. Goldman, and Y. Goldberg, "Stop uploading test data in plain text: Practical strategies for mitigating data contamination by evaluation benchmarks," 2023.

[33] W. Shi, A. Ajith, M. Xia, Y. Huang, D. Liu, T. Blevins, D. Chen, and L. Zettlemoyer, "Detecting pretraining data from large language models," 2023.

[34] E. Beeching, C. Fourrier, N. Habib, S. Han, N. Lambert, N. Rajani, O. Sanseviero, L. Tunstall, and T. Wolf, "Open llm leaderboard," https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard, 2023.

[35] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord, "Think you have solved question answering? try arc, the ai2 reasoning challenge," 2018.

[36] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring massive multitask language understanding," *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[37] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi, "Hellaswag: Can a machine really finish your sentence?" 2019.

[38] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning, "Hotpotqa: A dataset for diverse, explainable multi-hop question answering," 2018.

[39] Y. Perlitz, E. Bandel, A. Gera, O. Arviv, L. Ein-Dor, E. Shnarch, N. Slonim, M. Shmueli-Scheuer, and L. Choshen, "Efficient benchmarking (of language models)," 2023.

[40] L. Gao, J. Tow, B. Abbasi, S. Biderman, S. Black, A. DiPofi, C. Foster, L. Golding, J. Hsu, A. Le Noac'h, H. Li, K. McDonell, N. Muennighoff, C. Ociepa, J. Phang, L. Reynolds, H. Schoelkopf, A. Skowron, L. Sutawika, E. Tang, A. Thite, B. Wang, K. Wang, and A. Zou, "A framework for few-shot language model evaluation," 12 2023. https://doi.org/10.5281/zenodo.10256836

[41] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," 2016.

Abstract

Question Answering Systems Based on Large Language Models

Eva Jagodić

Large Language Models (LLMs) have revolutionized the field of Natural Language Processing (NLP). Unlike traditional machine learning systems, which are trained on a specific task and then applied exclusively to this task, LLMs demonstrate proficiency on a wide variety of tasks, both traditional and novel. Question answering (QA) is one of the fundamental NLP tasks that has experienced notable changes with the development of LLMs. This thesis explores the state-of-the-art of question answering systems based on LLMs and attempts to address one of the significant limitations of these models: the language limitation. Most LLMs are English-centered, in both their pretraining corpus and in their evaluation. Therefore, the focus of the thesis is on trying to adapt a powerful open-soruce model, Llama-2, to German. To achieve this, a German evaluation framework is constructed, and two low-resource enhancement techniques are explored, namely in-context learning and parameter efficient instruction tuning. The results reveal that while significant improvement can be achieved using simple few-shot prompting, fine-tuning proves to be insufficient for further improvements of the model's German QA capabilities.

Keywords: large language models; question answering; multilingual models; in-context learning; instruction-tuning

Sažetak

Sustavi pitanja i odgovora zasnovani na velikim jezičnim modelima

Eva Jagodić

Veliki jezični modeli označili su revoluciju u području obrade prirodnog jezika. Za razliku od tradicionalnih sustava strojnog učenja, koji su trenirani na jednom zadatku i potom korišteni isključivo na tom istom zadatku, veliki jezični modeli postižu dobre rezultate na velikom rasponu zadataka, kako tradicionalnih tako i potpuno novih. Zadatak pitanja i odgovora jedan je od osnovnih zadataka obrade prirodnog jezika koji je također doživio promjene razvojem velikih jezičnih modela. Ovaj rad istražuje najsuvremenije sustave pitanja i odgovora zasnovane na velikim jezičnim modelima i pokušava riješiti jedan od značajnih ograničenja tih modela: jezično ograničenje. Naime, većina modela usmjerena je na engleski jezik, i u korpusu za treniranje i u evaluaciji. Stoga je fokus ovog rada na prilagodbi modela Llama-2 na njemački jezik. U ovu svrhu konstruiran je okvir za evaluaciju na njemačkom i istražene su dvije tehnike poboljšanja s niskim zahtjevima na računalne resurse: učenje iz konteksta i fino podešavanje instrukcijama na parametarski učinkovit način. Rezultati ukazuju na to da dok se značajno poboljšanje može postići korištenjem jednostavnih *few-shot* instrukcija, fino podešavanje nedovoljno je za dodatna poboljšanja njemačkih sposobnosti.

Ključne riječi: veliki jezični modeli; sustavi pitanja i odgovora; višejezični modeli; učenje iz konteksta; fino podešavanje instrukcijama

Appendix A: The Code

A.1 Prompts

Llama-2 chat prompt

```
<s>[INST] <<SYS>>
{{ system_prompt }}
<</SYS>>

{{ user_message }} [/INST]
```

Fine-tuned model prompt

```
### Anweisung:
{instruction}

### Antwort:
{answer}
```

LeoLM prompt

```
<|im_start|>system
{system_message}<|im_end|>
<|im_start|>user
{prompt}<|im_end|>
<|im_start|>assistant
```

A.2 Metrics

Quasi-exact match The correctness condition is based on exact match after postprocessing. The exact postprocessing procedure is described in 3.3.5. The full list of common prefixes is given here:

```
list_of_common_prefixes = [
    "die beste fortsetzung des gegebenen kontextes ist:",
    "richtige Antwort:",
    "Antwort ist:",
    "Antwort:",
    "Antwort lautet:",
    "Antwort ist",
    "Antwort lautet"
    "richtige Antwort",
    "Antwort"
]
```

F1 Unlike exact match methods, the correctness condition for F1 is more nuanced, allowing for partial string overlap rather than being strictly all-or-nothing. This is done following the works of [41].