User-Driven and Adaptive Text Generation Using Prompting

Lena Oberkircher
2568971
Saarland University
s8leober@stud.uni-saarland.de

Abstract

As LLMs become more prevalent in everyday life, humans of more and more vast linguistic backgrounds and cognitive abilities become users of them. In any case, it is important that LLMs adjust their own language to the user in order to communicate effectively and efficiently. This work explores the inherent capabilities of LLMs to adjust to the user through prompting techniques. Through simple interaction with the user and explicit instruction, state-of-the-art (SOTA) LLMs are able to effortlessly adapt the difficulty and vocabulary of their generated responses, as shown by manual and automatic evaluation of multiple chat interactions. This work also discusses and questions the necessity of finetuning in purely Question-Answering based applications of LLMs, highlighting their advanced capabilities as question-answerers and communicators regardless of the user's language capabilities.

1 Introduction

Language used by humans who interact with large-language models (LLMs) is as diverse as humans themselves. No two people use the same style of language, even if they are native speakers of the same language. Our language use is shaped by our outside factors, our upbringing, education, linguistic background and surroundings, and also, inherently and internally, by our cognitive abilities.

Among this diversity we also find individuals with language-related limitations and disabilities, such as aphasia, autism spectrum disorder or a range of cognitive and developmental disabilities that affect how language is processed and produced. Others may be second-language learners, who do not or not yet have a full grasp of the language. For all of these individuals and for any user, LLMs can offer great opportunities if used correctly. With the use of LLMs, information is readily available, and the understanding can be deepened in a conversation-like manner, similarly to a conversation with a teacher. Questions the user may have can be answered on the spot, doubts explained, and relationships between concepts explored. Especially users who, as previously explained, may have issues comprehending or processing language or communicating themselves, have gained a tool to help them learn and explore with the rise of LLMs. Of course, LLMs are not equivalent to human teachers: they are prone to hallucinations (compare [1] for a survey on hallucinations) and may thus teach wrong information, and they do not have a perfect understanding of human language; for example, LLMs are not capable of understanding sarcasm at a human level[4].

However, in order for this information to be useful, it needs to be communicated in a way that can be processed and understood by the user without problems. This entails that useful LLMs need to have the ability to adjust the language they produce to the user. In a conversation with an expert of a field, LLMs can use complex language and concepts not known by the majority of the population. To the average user, less domain-specific language should be used and difficult concepts explained. Users with below average understanding of the language require simpler language, including shortening sentences, simplifying grammatical structures and reducing the number of difficult words.

This work explores the inherent capabilities of LLMs to simplify or complicate their language, adjusting it to the user through simple prompting approaches. It questions whether finetuning is necessary for state-of-the-art (SOTA) models to effectively communicate with users and discusses the advantages and disadvantages of finetuning compared to prompting.

2 Related Work

2.1 Zero- and Few-Shot Prompting

2.2 Chain-of-Thought Prompting & Zero-Shot Reasoning

[3]

However, chain-of-thought prompting is not necessary for a simple text generation task as in this project. LLMs already excel at this, and demanding full chain-of-thought reasoning just to simply or complicate text would simply be too resource-intensive for little reward. As Kojima et al.[2] show, zero-shot reasoning can be as effective as chain-of-thought reasoning, by simply instructing a model to "think step by step".

2.3 Readability Scores

3 Data & Preprocessing

To finetune a model for the task at hand, data of a specific format is required. Ideally, this data would come in the form of a Question Answering (QA) dataset with multiple different responses of varying difficulties for each question. The more fine-grained the distinction between difficulties, the better. However, those datasets are rare, and for this work, I was unable to find a dataset fitting these requirements. Instead, I only discovered a handful of datasets with minimal distinction between difficulties. For this project, I used

1. *onestop_qa*[1]: a dataset containing 567 texts at three difficulty levels - elementary, intermediate and advanced.

2. *CEFR-Sentence-Level-Annotations*[2]: a dataset with 10,004 short sentences; difficulties are divided into 6 levels (1-6), which correspond to the CEFR levels (A1-C2), meaning the dataset it targeted at applications for or concerning second language learners. Two annotators separately graded each sentence. The average between the two annotations was taken and rounded down to an integer.

Because these datasets utilize different gradings for the difficulty levels, the first step was to unify the difficulty levels to a common scale. One large issue is that while *onestop_qa* distinguishes between only three levels, the standard CEFR scale, and therefore also *CEFR-Sentence-Level-Annotations*, makes a more fine-grained differentiation at 6 levels. There are many ways to unify such different scales, but the simplest is to agree on a three-way split between elementary, intermediate and advanced, and allow for a more fine-grained distinction of all three levels into 2 sublevels each. This also aligns with the CEFR scale, which first distinguishes between level A for beginners, level B for intermediate and level C for advanced learners, and then subdivides each of these levels into two (A1, A2 etc.). Thus, the final dataset ended up being divided into these levels:

1. *elementary-1*, containing *onestop_qa*'s elementary level and *CEFR-Sentence-Level-Annotations*'s A1 level (level 1 in the dataset)

2. *elementary-2*, containing*CEFR-Sentence-Level-Annotations*'s A2 level (level 2 in the dataset)

3. *intermediate-1*, containing *onestop_qa*'s intermediate level and *CEFR-Sentence-Level-Annotations*'s B1 level (level 3 in the dataset)

4. *intermediate-2*, containing *CEFR-Sentence-Level-Annotations*'s B2 level (level 4 in the dataset)

5. *advanced-1*, containing *onestop_qa*'s advanced level and *CEFR-Sentence-Level-Annotations*'s C1 level (level 5 in the dataset)

6. *advanced-2*, containing *CEFR-Sentence-Level-Annotations*'s C2 level (level 6 in the dataset)

It is of course debatable whether it is sensible to combine CEFR data, which is targeted at second-language learners, with first language, native data at different levels. Much of texts targeted at learners is artificial and unnatural in structure, as it requires to adhere to a strict selection of vocabulary and grammar. But due to the data scarcity and the small size of the *onestop_qa* dataset, for this project it may be beneficial to increase the data size significantly, even if at the risk of losing some of the fluidity of the language.

After structuring and preprocessing the data, it was necessary to transform it into a QA-like

[1] iastate/onestop_english

[2] edesaras/CEFR-Sentence-Level-Annotations

dataset, as the goal was to finetune a model to respond to prompts. In order to achieve this, Openai's *gpt-4o-mini* was employed to generate possible prompts for each text. The model was prompted with the system prompt

```
"You are a helpful assistant
    aiding in the generation of
    a Q&A dataset. When given a
    text, your task is to
    generate a prompt that the
    text could have been a
    response to."
```

followed by the prompt

```
Given the following text, please
    generate a prompt that it
    could be a response for:
    <text>
```

Text generation was conducted through the `client.chat.completions.create` function. With the resulting dataframe, a model can be fine-tuned by formatting the training data's prompts into the form `[DIFFICULTY: <difficulty>] <prompt>`.

4 Methods

For the task of adapting text generation, a chat interface allowing the user to chat with the model and give feedback was built. At default, the model is prompted to respond to user prompts at an intermediate level:

```
system_prompt = "You are a
    helpful assistant generating
    texts at an intermediate
    language level."
```

At each of the iterations, the user is given the following options:

1. Through typing `'exit'` or `'quit'`, the user can end the chat.

2. Through typing one of the feedback options (`'too simple / easy'`, `'too difficult / advanced'` or `'just right'`), the user can give feedback about the previously generated response. This will prompt the model to 1) regenerate the previously generated response and 2) adjust the language used in the following conversation. Typing `'just right'` will not affect the language used in the future, but will just regenerate the previous response. It was added to make the system feel more balanced, allowing for positive feedback.

3. Through typing `'update'`, the user has the possibility to paste or write a self-written text to the model. Given this text, the model is then prompted to 1) regenerate the previous response in a similar style as the text and 2) adjust its future language use to the style of the text. The user can pass as many of these texts as desired.

4. The default is a simple prompt, which will be answered by the model. The style of the response will depend on the previous chat history.

Before the chat with the model begins, the user is asked to choose between one of the three prompting methods: zero-shot, few-shot and zero-shot reasoning prompting.

4.1 Zero-Shot Prompting

When choosing zero-shot prompting as a technique, the model will receive simple instructions. For example, when the user gives the feedback `'too simple'`, the model will simply be prompted with

```
"The previous output was 'too
    simple'. Please  regenerate
    the response with 'more
    advanced' language."
```

The model thus does not receive any further instruction on how to regenerate a prompt or how to adjust its language.

4.2 Few-Shot Prompting

When choosing few-shot prompting, the model receives some more guidance for reformulating its responses. For example, when the user gives the feedback `'too simple'`, the model will be prompted with

```
"The previous output was 'too
    simple'. Please regenerate
    the response with 'more
    advanced' language. Here
    are some examples of how
    to do this:

    Example 1:
    Response: The process of
```

```
    photosynthesis allows
    plants to convert sunlight
    into energy.
    Feedback: too difficult
    Adapted: Photosynthesis is
    how plants turn sunlight
    into energy.

    Example 2:
    Response: The capital of
    France is Paris, which is
    known for its art, culture,
    and history.
    Feedback: too simple
    Adapted: Paris, the capital
    of France, is renowned
    not only for its ...
    "
```

In total, 3 examples are passed. The model receives examples for feedback and updating loops. Normal response generation for standard prompts is still zero-shot, due to it being difficult to make it few-shot.

4.3 Zero-Shot Reasoning

The zero-shot reasoning loops are based on the work by Kojima et al.[2]. It differs from zero-shot prompting in one way. During the feedback and update step of the chat loop, the model receives the additional instruction `'Let's think about this step by step!'`, which, when the feedback is `'too simple'`, looks like this:

```
"The previous output was 'too
    simple'. Please  regenerate
    the response with 'more
    advanced' language.
    Let's think about this step
    by step!"
```

Issues with Finetuning One other goal of the project was to compare finetuning to the simple prompting approach. After generating the prompts and formatting the dataset into a workable training dataset, however, finetuning failed due to repeated `CUDA OOM` errors, even on an `A100` GPU on the Google Colab platform[3]. Therefore, finetuning was not conducted for this project. Had it been conducted, another chat interface similar to the prompting approach. This would have led to a combination of finetuning and prompting. It would differ from the pure prompting approach in the following ways:

1. The initial model would be instructed to respond at a level corresponding to `intermediate-1` of the training dataset.
2. After receiving the feedback `'too simple'`, the level would have been increased to `intermediate-2`, and all the way up to `advanced-2`.
3. Vice-versa, after receiving the feedback `'too difficult'`, the level would have been decreased to `beginner-2`, and all the way down to `beginner-1`.
4. Upon reaching the highest and lowest levels, prompting would take over: if level `beginner-1` was still deemed `'too difficult'`, the model would have been instructed to use even simpler language through a prompt.
5. Handling of user-written texts would have been handled purely through prompting. It would be ideal to add user-written text to the fine-tuning data, but this is of course unrealistic.

Despite having issues with finetuning and not being able to complete this part of the project, the finetuning code is available in the `code` folder of the project. I hope the efforts can be taken into consideration during grading.

4.4 Evaluation Pipeline

Upon initiating the chat, the user can enable 'evaluation mode', which is used to evaluate model outputs in the following ways.

The model `gpt-4o-mini` serves as an automatic evaluator for the generated responses. In two separate functions, the model is instructed to evaluate the generated responses compared to the previous response. One function instructs the model to rank two responses against each other in terms of difficulty. This is used after the user gives feedback to the model. The model is instructed to determine which of the texts uses more difficult language (by stating whether the second provided text uses more difficult, simpler or similar

[3]colab.research.google.com

language compared to the first text), and also provide a short explanation.

The second function is responsible for evaluating the updating of the model through user-written text input. For this, the function takes the previously generated response, the newly generated response and the user-written text. The model then decides whether the newly generated or the previous response is closer in language (not in content) to the user-written text, and provide a short explanation for its decision.

In both functions, two readability scores are automatically calculated for both the old and the new response (and the user-written text): Flesch Kincaid Grade Level (FKGL) and Flesch Reading Ease (FRE). FKGL approximates the reading grade level of a text and considers sentence length and complexity. A higher FKGL corresponds to a higher grade level and thus to a more difficult text. FRE evaluates a text on its readability based on sentence length and word complexity, and returns a score between 1 and 100, with 1 corresponding to lowest readability (highest difficulty) and 100 to highest readability (lowest difficulty). These scores are unbiased and provide an automatic, clear evaluation alongside the more biased evaluation of the GPT model.

5 Analysis & Results

To test the prompting approach, a series of prompting, feedback and updating prompts were conducted and the results analyzed. For each prompting approach, I tested three loops:

1. After the original prompt, pass the feedback `'too simple'` five times, leading to an increase in difficulty of the generated prompt 5 times.

2. After the original prompt, pass the feedback `'too difficult'` five times, leading to a decrease in difficulty of the generated prompt 5 times.

3. After the original prompt, type `'update'` and pass 5 different, user-written texts, leading to an adjustment of the writing style 5 times.

For both update loops, both the model's judgment of increase or decrease of diffiulty as well as the change of FKGL and FRE scores are taken into consideration. For the update loop, both the model's judgment of text similarity as well as the similarity of FKGL and FRE scores between the old and new generated text and the user-written text are being considered. Prompts and generated texts are omitted due to limited space, but full tables can be found in the subfolder `results`.

5.1 Zero-Shot Prompting: Feedback Loops

Tables 1 and 2 show the results of both feedback loops for zero-shot prompting. In zero-shot prompting, instructing the model to reformulate a response with more difficult language is successful. For the first 3 rounds of the feedback loop, `gpt-4o-mini` judges the newly generated responses to be more difficult than the previously generated one. Afterwards, as the responses become increasingly more advanced and eventually reach a ceiling, the model judges the two responses as similar. Inspecting FKGL and FRE scores agrees with this judgment. In fact, for all five iterations, FKGL scores increase, however less so during the last iterations. FRE decreases steadily, indicating a continued decrease in readability and thus an increased difficulty.

When instructing the model to produce simpler language, it seems to successfully do so for almost all 5 iterations. For all but the 4th iteration, the newly generated responses are judged as simpler than the previous one by `gpt-4o-mini`. FKGL and FRE scores are inconclusive. While we can observe a general decrease of FKGL and a general increase of FRE scores, there are strong fluctuation that disallow conclusive statements.

Manual evaluation agrees with the model's judgment and the trend of readability scores. Instructing the model to generate more advanced texts leads to responses that are longer, more grammatically complex and contain more difficult vocabulary. Compare the start of the initial response

Fine-tuning an LLM, or Large Language Model, means adjusting a pre-trained model so it performs better on a specific task or set of tasks. Here's how it works: ...

to that of the response on the third iteration

Fine-tuning a Large Language Model (LLM) is a sophisticated procedural paradigm that involves the iterative recalibration of a model initially trained on a voluminous and diverse corpus of textual data, with the intent of enhancing its applicability and precision in specific, domain-

Iteration	Command	Model judgment	FKGL	FRE
Initial	-	-	9.9	54.32
1	more advanced	more advanced	13.5	34.05
2	more advanced	more advanced	18.5	10.43
3	more advanced	more advanced	19.2	8.61
4	more advanced	similar	19.6	1.97
5	more advanced	similar	20.7	-6.29

Table 1: Results of zero-shot prompting, feedback loop 1 - too simple. Prompt and texts omitted for space. Full results in gpt-4o-mini_zero-shot_ranking_scores_2025-04-12-11-58.xlsx

Iteration	Command	Model judgment	FKGL	FRE
Initial	-	-	10	53.92
1	simpler	simpler	9.1	61.87
2	simpler	simpler	8.8	62.48
3	simpler	simpler	8.8	62.58
4	simpler	similar	7.2	72.16
5	simpler	simpler	8.2	64.1

Table 2: Results of zero-shot prompting, feedback loop 2 - too difficult. Prompt and texts omitted for space. Full results in gpt-4o-mini_zero-shot_ranking_scores_2025-04-12-12-00.xlsx

centric tasks. This complex process can be elucidated through several key components: ...

Not only is the newly generated response much longer, but it contains advanced, domain-specific vocabulary and structurally complex sentences. Similarly, comparing the start of the initial response

Fine-tuning an LLM, or Large Language Model, means adjusting a pre-trained model to make it perform better on a specific task or set of data. Here's how it works: ...

to that of the response on the third iteration

Fine-tuning an LLM (Large Language Model) means helping a smart computer program do a specific job better. Here's how it works: ...

shows that after instructing the model to simplify its responses, sentences become shorter and contain less complex vocabulary and grammatical structures.

5.2 Few-Shot Prompting: Feedback Loops

Tables 3 and 4 show the feedback loop results for few-shot prompting. Again, we see a constant increase of difficulty in Table 3, both according to the GPT model's judgment and according to FKGL and FRE scores. In fact, these results can be deemed more successful than the equivalent results for zero-shot prompting. While the model does not indicate consistent successful rewriting with simpler language in Table 4, FKGL and FRE scores show a different picture. While FKGL scores are consistently decreasing, FRE scores are increasing, indicating success of few-shot prompting. And, in fact, according to the scores alone, few-shot prompting is more successful than zero-shot prompting.

Inspecting the responses manually again confirms these judgments. Instructing the model to produce more advanced responses is successful, for example compare the initial response

Fine-tuning an LLM, or large language model, means taking a pre-trained model that has learned general language patterns and making small adjustments to improve its performance on a specific task or dataset. This process helps the model understand and generate more relevant responses for particular types of questions or content.

to the response on the third iteration

... Fine-tuning an LLM represents the intricate art of optimizing a pre-trained linguistic model by engaging in a meticulous adjustment of its hyperparameters. This sophisticated endeavor necessitates the integration of a carefully curated dataset, one that embodies the specificities and subtleties of the intended application. Through this process, the model undergoes a transformative recalibration, enabling it to exhibit heightened responsiveness and precision in addressing domain-specific

Iteration	Command	Model judgment	FKGL	FRE
Initial	-	-	14.2	37.64
1	more advanced	more advanced	13.6	39.97
2	more advanced	more advanced	17.4	13.07
3	more advanced	more advanced	17.7	7.05
4	more advanced	more advanced	20.2	0.45
5	more advanced	more advanced	20.6	-0.57

Table 3: Results of few-shot prompting, feedback loop 1 - too simple. Prompt and texts omitted for space. Full results in gpt-4o-mini_few-shot_ranking_scores_2025-04-12-12-02.xlsx

Iteration	Command	Model judgment	FKGL	FRE
Initial	-	-	10.3	58.62
1	simpler	simpler	8.5	63.19
2	simpler	similar	8.4	63.7
3	simpler	similar	7.6	65.73
4	simpler	similar	7	67.25
5	simpler	similar	6.4	74.19

Table 4: Results of few-shot prompting, feedback loop 2 - too difficult. Prompt and texts omitted for space. Full results in gpt-4o-mini_few-shot_ranking_scores_2025-04-12-12-04.xlsx

queries. Consequently, the fine-tuned model transcends its initial generalized capabilities, emerging as a bespoke tool adept at navigating the complexities of specialized linguistic contexts, thus enhancing its effectiveness as a solution in applied scenarios.

which clearly shows that complexity has increased. However, an issue with few-shot prompting is that the responses decrease in complexity overall. The model seems to adapt to the few-shot examples too much, adapting their simplicity and thus overall behave completely differently from the zero-shot prompting approach.

5.3 Zero-Shot Reasoning: Feedback Loops

Zero-shot reasoning seems to be the least successful of all three approaches. Tables 5 and 6 show that the model struggles to consistently adapt its language to increased / decreased difficulty. Both FGKL and FRE scores and the GPT model's judgment fluctuate. The model seems to interpret the instruction 'Let's think step by step!' to mean that it is supposed to break down its response into various steps or an itemized list

*Of course! Let's make it even simpler step by step. 1. **What is an LLM?** An LLM is a computer program that can read and write language, like a person. It learns from a lot of text. 2. ...*

which is understandable and indicates that this kind of prompting is not suitable for simple text generation tasks.

5.4 User-Written Text Updates

The full results of the user-written text updates loops can be found in the files *gpt-4o-mini_zero-shot_updating_scores_2025-04-12-12-12.xlsx* for zero-shot prompting, *gpt-4o-mini_few-shot_updating_scores_2025-04-12-12-15.xlsx* for few-shot prompting and *gpt-4o-mini_reasoning_updating_scores_2025-04-12-12-18.xlsx* and zero-shot reasoning.

For zero-shot prompting, overall, after providing a user-written text, the model seems to successfully create language that is more similar to the provided text. For example,

- *Original response*: "Fine-tuning an LLM, or Large Language Model, is the process of taking a pre-trained model and training it further on a specific dataset. Here's what it means:

  1. **Pre-trained Model**: LLMs are initially trained on a large amount of text from various sources. This training helps them understand language patterns, grammar, facts, and some general knowledge.

  ...

  In summary, fine-tuning makes a large language model more effective for specific ap-

Iteration	Command	Model judgment	FKGL	FRE
Initial	-	-	11.3	50.57
1	more advanced	more advanced	14	32.94
2	more advanced	more advanced	16.8	14.8
3	more advanced	more advanced	18.5	5.02
4	more advanced	more advanced	18.6	4.71
5	more advanced	similar	17.1	13.88

Table 5: Results of zero-shot reasoning, feedback loop 1 - too simple. Prompt and texts omitted for space. Full results in gpt-4o-mini_reasoning_ranking_scores_2025-04-12-12-06.xlsx

Iteration	Command	Model judgment	FKGL	FRE
Initial	-	-	11.7	49.65
1	simpler	simpler	5.2	77.23
2	simpler	simpler	3.3	87.52
3	simpler	similar	4.5	79.06
4	simpler	simpler	3.3	87.52
5	simpler	similar	3.1	88.23

Table 6: Results of zero-shot reasoning, feedback loop 2 - too difficult. Prompt and texts omitted for space. Full results in gpt-4o-mini_reasoning_ranking_scores_2025-04-12-12-10.xlsx

plications by training it further on targeted data."

- *User-written text*: "I love going to the park. It's fun to walk around, look at the trees, and watch the birds fly. Sometimes I bring a snack and sit on a bench. I like hearing the kids play and seeing the dogs run. It always makes me feel happy and relaxed."

- *Newly generated response*: "Fine-tuning an LLM is like giving a big language machine a special training to learn even more. First, the model has already learned from lots of different texts, so it knows how to understand and use language pretty well. This is like how I learn to read and write by practicing with many books.

  ...

  In short, fine-tuning makes the language model even better at what we want it to do, just like how focused practice helps me improve in my favorite activities."

shows that clearly, the newly generated response has a more similar writing style to the user-written text compared to the original response. FRE and FKGL scores are inconclusive, which can be attributed to the fact that their calculation is dependent on sentence length, which the model seems to not be consistently capable of adjusting.

Results for zero-shot reasoning are similar to zero-shot prompting: the model seems capable of adapting its writing style to that of the user-written texts, but again, FKGL and FRE scores are inconclusive. It is questionable whether zero-shot reasoning has any effect on model performance at all, and since it seems to negatively impact the performance in the feedback loops, its usefullness is questionable.

Few-shot prompting performs worse than all of the other approaches, which is most likely caused by the few-shot examples. It seems like the model adapted the writing style of the few-shot examples so closely that it failed to adapt to the user-provided texts.

6 Discussion & Conclusion

The results of the prompting approaches show that SOTA LLMs have clear inherent capabilities of adjusting their language to the user. By allowing simple feedback and instructing the model to adjust to user-written texts, `gpt-4o-mini` is able to adjust the difficulty and writing style of the texts it produces. This project clearly shows that even with a relatively simple and straight-forward implementation of such prompting, a user-driven and adaptive text generation can be achieved. Results show that despite being the most simplistic approach, zero-shot prompting achieves the best or competing performance, both in feedback loops and in the update loop with user-written texts.

6.1 Is Finetuning Necessary?

For many downstream tasks, LLMs are finetuned on task-specific data to achieve better performance (resource). However, (resources) already show that prompting can be a powerful tool even for complex tasks that are unfamiliar to most LLMs.

7 On the use of ChatGPT

Parts of this project were supported by the use of OpenAI's ChatGPT[4]. In particular, the model was used to generate

- *examples for few-shot prompting*: the exact examples are displayed in section 4.2. To generate these examples, the layout of the zero-shot prompt, the goal of the task and a brief explanation of few-shot prompting were presented to the model.
- *user-written texts for analysis*: all of the texts that, during testing of the prompting approach, were given to the model as üser-written textsẗo test the update feature. To generate these texts, the model was prompted to create a list of texts of varying difficulties and writing styles, ranging from child-like language to the language of an adult expert with an expansive vocabulary.

All other parts of the project, including the coding and the writing of this report, were done without assistance of ChatGPT or other LLMs. For part of the work, I based my code on my work during the assignments during the semester.

References

[1] Lei Huang et al. "A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions". In: *ACM Transactions on Information Systems* 43.2 (2025), pp. 1–55.

[2] Takeshi Kojima et al. "Large language models are zero-shot reasoners". In: *Advances in neural information processing systems* 35 (2022), pp. 22199–22213.

[4] chatgpt.com

[3] Jason Wei et al. "Chain-of-thought prompting elicits reasoning in large language models". In: *Advances in neural information processing systems* 35 (2022), pp. 24824–24837.

[4] Yazhou Zhang et al. "SarcasmBench: Towards Evaluating Large Language Models on Sarcasm Understanding". In: *arXiv preprint arXiv:2408.11319* (2024).