Comparison of LLM Prompting Techniques

Moritz Lang[7430881] and Tim Schäfer[7395643]

University of Cologne

Abstract. Large Language Models (LLMs) have significant variability in translation performance based on prompt design. This study compares different prompting techniques, including zero-shot, few-shot, and advanced methods such as self-correction and response comparison, to improve translation between English and German. Through systematic evaluation across different levels of text complexity, we find that structured prompts significantly improve accuracy, with reasoning-based techniques performing best. While translation quality was generally higher from German to English, text complexity had minimal impact. Our results highlight the importance of optimized prompting for machine translation and provide insights for improving translation tasks for smaller LLMs.

Keywords: Prompting Techniques · MLFlow · Large Language Models · Machine Translation

1 Introduction

Large language models (LLMs) have revolutionized the field of natural language processing by demonstrating remarkable capabilities in a variety of tasks [5]. However, their performance is highly sensitive to the phrasing of the prompt, leading to significant variations in output quality [4]. This variability poses a particular challenge when attempting to optimize results, as prompt designs that are effective for one model do not necessarily translate to another. The present study aims to address this issue by comparing different prompting techniques specifically designed to improve the efficiency of smaller models when translating between English and German. By systematically evaluating a range of approaches - from basic zero-shot methods to more advanced techniques such as self-correction and response comparison - we aim to identify the most effective methods for achieving high-quality translations under controlled conditions. Our methodological framework includes a structured experimental design using standardized data sets, consistent translation directions, and a variety of evaluation metrics. This comprehensive approach not only highlights the strengths and limitations of each prompting technique, but also provides insights into the interplay between prompt structure, model architecture, and translation performance. Ultimately, this research not only deepens our understanding of prompt design in translation contexts, but also lays a foundation for future research in this area. By systematically evaluating a variety of prompting techniques, we have demonstrated that even small adjustments in prompt design can lead to significant improvements in both translation accuracy and fluency. This study provides practical examples for designing more effective prompts for translation tasks with LLMs. In addition, our work clarifies how prompt design, model architecture, and language complexity interact. This insight sets the stage for additional future research to further improve these techniques and apply them to a wide range of natural language processing tasks. The following section describes the methodology we use throughout the research.

2 Methodology

To explore the influence of various prompting techniques on machine translation tasks, this study evaluates their effectiveness across multiple models and text complexities. Our methodology follows a structured approach, including data selection, test setup, and result analysis to ensure reproducibility and systematic evaluation.

2.1 Translation Data

The evaluation is conducted using a dataset composed of texts with varying levels of linguistic complexity to assess how different prompts on different text complexities influence translation performance. We focus on bidirectional translation between German and English, testing each prompting technique in both directions. The dataset (see Figure 7) includes:

- **Simple Language:** Short sentences with minimal syntactic complexity and straightforward vocabulary.
- **Newspaper Article (General Vocabulary):** News articles with standard sentence structures and commonly used vocabulary.
- **Newspaper Article (Specialized Vocabulary):** News articles containing domain-specific terms from areas such as economics, law, and politics.
- **Popular Science Text:** Semi-technical texts aimed at a general audience, often featuring explanatory phrasing.
- **Scientific Text:** Highly specialized academic writing with complex terminology, passive voice, and intricate sentence structures.

By evaluating prompts in these categories for both German-to-English and English-to-German translation, we gain insight into how prompting techniques perform across different levels of linguistic complexity and directionality. In addition, we gain insight into how different levels of text complexity affect the ability of large language models to produce accurate translations.

2.2 Experimental Setup

To ensure consistency and comparability, all experiments adhere to the following controlled conditions:

Translation Directions: Each text is translated from German to English and English to German, ensuring that each prompting technique is tested in both directions. This allows for a comprehensive evaluation of how language directionality affects prompt effectiveness and model performance.

Models: This study uses four different translation models, each of which varies in size and capabilities. These models were deliberately chosen as small, open-source LLMs to balance efficiency with accessibility, making them suitable for research applications. A detailed breakdown of the models and their differences can be found in section 4 Models.

Inference Settings: The model parameters for all models are set to temperature = 0.1, top-k = 40, and top-p = 0.9, with a defined token limit to maintain consistency in output length across translations. Temperature controls the randomness of model outputs, with lower values making results more deterministic. Top-k sampling restricts token selection to the k most likely choices at each step, reducing randomness. Top-p sampling dynamically selects tokens based on cumulative probability mass, allowing for more flexible yet controlled output generation [12]. The chosen values are based on the recommended default settings [1].

Text-Prompt Composition: Each text complexity category is tested with each prompting technique (described in Section 5) to systematically evaluate their performance at different complexity levels.

Pipeline and Reproducibility: The experimental workflow follows a structured process, including data loading, prompt generation, model inference, and metrics calculation (further explained in Section 6 Metrics). Each pipeline run consists of testing all prompting techniques across all text complexities to ensure comprehensive evaluation and reproducibility.

2.3 Evaluation and Analysis

Translation outputs are analyzed using automated evaluation methods. The metrics section provides more details, but key aspects include:

Automated Metrics: BLEU, ROUGE, and BERTScore are automatically calculated to assess translation accuracy and fluency. Comparative plots are generated to visualize model performance across different prompting techniques and text complexities.

Systematic Prompt Optimization: The stored model outputs allow for an iterative approach to prompt refinement. By analyzing performance trends, systematic improvements are made to improve translation quality in subsequent experiments.

Comparative Analysis: Results are examined across models, text complexities, and prompting techniques to identify systematic trends and potential prompt optimizations.

This methodology provides a structured approach to understanding the effectiveness of prompting techniques in translation tasks while ensuring a thorough and consistent evaluation across diverse text types and translation directions. How this methodology is implemented is covered in the following section.

3 Pipeline

To systematically evaluate the effectiveness of different prompting techniques on the translation performance of small LLMs, we designed a structured pipeline.

The process, illustrated in Figure 1, ensures that each model is tested under controlled conditions while varying both prompting techniques and text complexity levels. By iterating through these factors systematically, we enable a direct comparison of how different prompting techniques influence translation accuracy.

We conduct the evaluation in a fully automated workflow, logging all results using MLflow to ensure experiment reproducibility and structured performance tracking. The pipeline is divided into several stages, each described further in the following subsections with explanations of their role in the evaluation process.

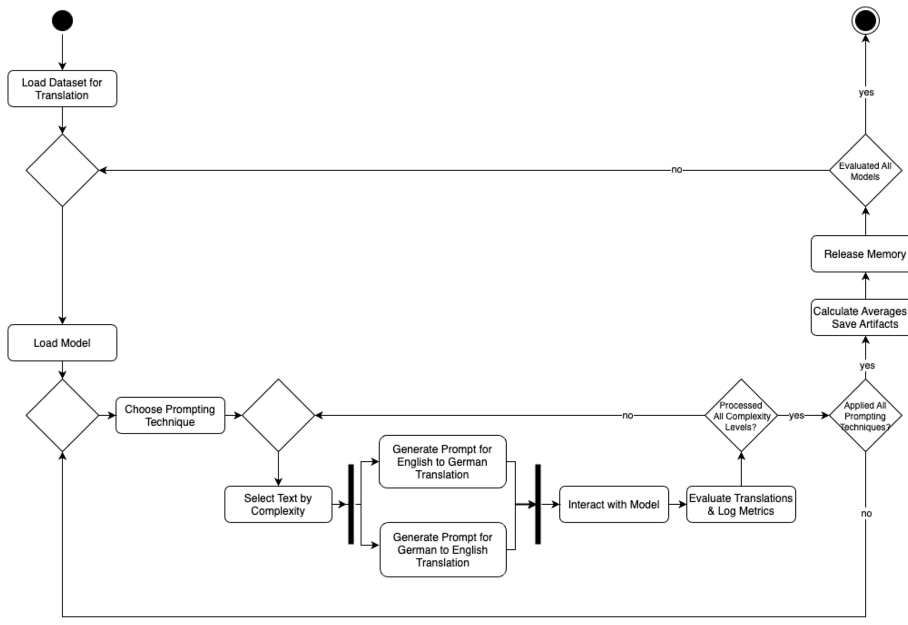

Load Dataset for Translation
no
yes
Evaluated All Models
Release Memory
Calculate Averages & Save Artifacts
yes
Load Model
Choose Prompting Technique
no
Processed All Complexity Levels?
yes
Applied All Prompting Techniques?
Generate Prompt for English to German Translation
Select Text by Complexity
Interact with Model
Evaluate Translations & Log Metrics
no
Generate Prompt for German to English Translation

Fig. 1. Pipeline Flow Chart

Dataset Loading: We begin by loading a dataset containing English and German text samples, categorized by complexity levels. All prompting techniques are tested using the same dataset, maintaining consistency in evaluation. After preparing the dataset, we initialize a selected LLM, ensuring that each model is assessed under identical conditions. By maintaining a controlled setup, we facilitate fair comparisons across different models and prompting techniques.

Translation Task Setup: We select a prompting technique from a predefined set of techniques (see Section 5). Next, we choose a text sample based on its complexity level, ensuring that models are tested on a spectrum of linguistic challenges, from simple sentences to more complex structures. Using this information, we generate translation prompts for both directions:

- English-to-German
- German-to-English

The prompt structure is adapted based on the chosen technique to maintain consistency in evaluation.

Model Interaction and Logging: The model processes the generated prompts and outputs translated text. We log the results, including the original text, the prompt used, the generated translation, and the corresponding evaluation metrics in MLflow for tracking and later analysis (see Section 6 for details).

Iterative Processing Across Conditions: To ensure a comprehensive evaluation, we iterate through all text complexity levels for the selected prompting technique. Once completed, we move on to the next prompting technique, repeating the process.After all prompting techniques have been tested for a particular model, we release memory resources before loading the next LLM, and the full cycle begins again. This ensures that every model undergoes identical testing procedures, making direct comparisons possible.

Final Aggregation and Cleanup: After evaluating all models under all experimental conditions, we aggregate performance metrics and use them to generate plots that visualize technique comparison, text complexity comparison, and model performance comparison. These plots are stored as artifacts in MLflow, enabling further analysis and comparison. These results allow for a systematic comparison of models, prompting techniques, and complexity levels.

This pipeline provides a structured framework for assessing the impact of different prompting techniques on translation outputs, ensuring consistency and comparability. By systematically applying different prompting techniques and testing across multiple text complexity levels, we gain insights into how these factors influence model performance. The following section describes the four models used in the experiments, detailing their architectures and different traits.

4 Models

This section describes the four language models used in this study. The models were chosen because they are open-source and can be run locally, making them more accessible compared to larger proprietary models like GPT-4 [2]. This makes them a practical choice for research, as they do not require proprietary access or extensive computational resources while still offering a range of capabilities for comparison. All models are provided in the GGUF file format, a format designed for efficient storage and optimized inference, allowing them to run on local hardware without requiring extensive computational resources [10]. The following Table 1 summarizes their main attributes, followed by a detailed description of each model.

Table 1. Comparison of the Language Models used in the Experiment.

Model Name	Parameters	Provider	Model Size	Release Date
Gemma-2B-IT	2B	Google DeepMind	1.50 GB	Feb, 2024
Llama-3.1-8B-Instruct	8B	Meta	4.92 GB	July, 2024
Llama-3.2-3B-Instruct-Q8_0	3B	Meta	3.42 GB	Sept, 2024
Aya-23-35B	35B	Cohere For AI	10.19 GB	May, 2024

Gemma-2B-IT is a 2-billion-parameter decoder-only Transformer model developed by Google. It is part of the Gemma family of lightweight, open-source models built from the same research and technology used to create the Gemini models [16]. It is the smallest model used in our test runs, with a model size of 1.50 GB and released in February 2024 [11]. The model is instruction-tuned, making it well-suited for a variety of text generation tasks, including question answering, summarization, and reasoning. Its relatively small size allows for deployment in environments with limited resources, such as laptops [16].

Decoder-only Transformer models, such as Gemma-2B-IT, consist solely of the decoder component of the Transformer architecture. Unlike encoder-decoder models, which process input using both an encoder and a decoder, decoder-only models generate text in an autoregressive manner by predicting each token sequentially based on prior context. This design allows them to excel at text generation tasks while maintaining computational efficiency [21].

Llama-3.1-8B-Instruct is an 8-billion-parameter decoder-only Transformer model developed by Meta. Released in July 2024, it is part of the Llama 3.1 collection of multilingual large language models, which includes models of 8B, 70B, and 405B parameters. The Llama 3.1 instruction-tuned text-only models are optimized for multilingual dialogue use cases and outperformed many available open-source and closed chat models on common industry benchmarks at the time. With a model size of 4.92 GB, Llama-3.1-8B-Instruct is designed to balance computational efficiency with robust performance, making it suitable for a variety of applications, including text generation and conversational AI. It supports multiple languages, including English, German, French, Italian, Portuguese, Hindi, Spanish, and Thai, making it a strong choice for multilingual tasks [18].

Llama-3.2-3B-Instruct-Q8_0 is a 3-billion-parameter model from Meta's Llama 3.2 series, released in September 2024. Like Llama-3.1-8B-Instruct, it is optimized for multilingual dialogue and supports multiple languages. The model is designed to be computationally efficient while maintaining strong instruction-following capabilities. With a model size of 3.42 GB, it provides a balance between performance and resource constraints, making it suitable for various applications, including summarization and agentic retrieval tasks. The Llama-3.2 series builds on the advancements of Llama-3.1, focusing on improved contextual understanding and inference efficiency [19].

Aya-23-35B is a 35-billion-parameter auto-regressive language model developed by Cohere For AI [7]. Released in May 2024 [6], it employs an optimized transformer architecture and has been fine-tuned to follow human instructions. The model is particularly optimized for multilinguality, supporting 23 languages, including Arabic, Chinese (simplified & traditional), Czech, Dutch, English, French, German, Greek, Hebrew, Hindi, Indonesian, Italian, Japanese, Korean, Persian, Polish, Portuguese, Romanian, Russian, Spanish, Turkish, Ukrainian, and Vietnamese. With a context length of 8,192 tokens, Aya-23-35B is designed for complex multi-turn conversations and general-purpose text generation. Its substantial parameter count and extensive language support make it suitable for a wide range of applications, from detailed content creation to intricate reasoning tasks. While it requires more computational resources compared to smaller models, its performance across various domains provides valuable insights into the capabilities of large-scale multilingual models [7].

By using models of different sizes and architectures, we ensure a broad evaluation of prompting techniques rather than model performance. The next sections will delve into the prompting techniques used for this research.

5 Prompting Techniques

The techniques we are comparing can generally be divided into zero-shot techniques, few-shot techniques and advanced techniques. The zero-shot techniques provide as few words, information, and instructions to the model as possible. In general, we implement one technique as a baseline against which to compare the effectiveness of the other techniques. Then we systematically change only one part of it at a time to maintain comparability. Our final prompting techniques are as follows: eight zero-shot variants, four few-shot variants and one *AllTogether* variant, where we combined several techniques. In addition, with the advanced techniques, we test how the large language model reacts when given the chance to correct its own response with the *SelfCorrect* technique and if it can choose the best version out of several of its responses with the *Comparison* technique. An overview of all the techniques and their descriptions are listed in Table 2. The exact implementations of the prompting techniques are listed in the appendix in Table 13.

For zero-shot Techniques, we start with our above mentioned baseline prompt, which is "Translate this text into German. Only return the translation: (*TextToTranslate*)" and respectively for translating into English. There is no scientific based reason behind this specific choice of words, it is just as simple and straightforward as possible. For the *JSONOutput* technique, we use a paper about instructing the model to deliver the output in a given format. The paper uses another fine-tuned LLM that adapts the prompt as an intermediary and tells the final model what to include in the output [14]. We adapt this technique to tell the model what structure the output should have, and the widely used and simple JSON format is perfect for this. For the *HighlightOnly* technique, we spotted an interesting formatting of prompts in the paper [3]. Here the authors printed a keyword in bold that was important for the output. The paper itself is not about this topic, but we adopted this formatting for this technique and put * around the word **only**. The reason we chose this word was that the models often returned text that had nothing to do with the translation, and this was a test to see if it could reduce that. For the *TranslationTextFirst* technique, we used the results of our tests. There we noticed that the model often only continues with the last part of the prompt, in our case the text to be translated. From this observation, we thought that if we reversed the order of the translation text and the instruction part of the prompt, the model would continue/start with the translation, since this is now the last part. For the *ConcretePromptEnding* technique, we stumbled upon the existence of a default end token for the GPT models [8]. We adapted it a bit to <|endofprompt|> and added it at the end of the prompt, with surprisingly good results. In the *ZeroShotAllTogether* technique, we have combined all of the other zero-shot techniques above. This is more of an experiment as it does not measure the performance of a single technique, but several of them combined. It was a spontaneous idea that came up while researching different prompting techniques. This technique also serves as a new baseline for the next two more complex zero-shot techniques. For the *Instruction* technique, where we combine the *HighlightOnly*, *TranslationTextFirst*, and *ConcretePromptEnding* techniques and add two short instructions to the prompt. Similar to the *TranslationTextFirst* technique, we observed our baseline results and added instructions to explicitly reduce the amount of irrelevant text generated by the model. The final zero-shot technique we use is the *Persona* technique, where the LLM is assigned a specific role to guide the linguistic style and focus of the output. This technique is explored in the paper [24], and we use the speaker-specific variant. Basically, the model is told that it is an expert in translating, born and living in Germany, raised bilingually with English and German, and studied English for 45 years.

For the few-shot Techniques, our general approach was similar to the zero-shot techniques. We start with a baseline few-shot technique that adds two examples in the middle of the prompt. However, they are customized so that the language into which the model is to be translated always comes last. We adapted this technique from the [17] paper, which is also the basis for the following techniques. As with the *TranslationTextFirst* technique before, we have also reversed the order of the instruction and the translation text to prevent the model from continuing with the text instead of translating it. In the next technique, *WithExampleToken*, we wrap the examples in a <start> and <end> token, similar to the paper [13]. The rest of the prompt is unchanged. For the *ContinueAfterExamples* technique, we add another example at the end of the prompt, but leave the translation blank. The idea behind this technique is that it also guides the output and prevents the model from generating unrelated text. Finally, we designed another *AllTogether* technique, this time consisting of the *TranslationTextFirst*, *Persona*, *FewShot*, *HighlightOnly*, and *ConcreteEndPrompt* techniques. Like the previous *ZeroShotAllTogether* technique, this is more of an experiment to see how a combination of techniques performs.

The Advanced Techniques that we have implemented consist of more than one model interaction. We take advantage of the model's ability to remember context. The first advanced technique is the *SelfCorrect* technique, which is a variation based on the Chain of Verification prompting technique from the paper [9]. The difference from the technique in the paper is that we remove the two steps of Plan Verification and Execute Verification and directly instruct the model to generate a final verified response. The second advanced technique is the *Comparison* technique, where the model generates 2 different translations of the original reference text and then decides which one is better. This technique is based on the Self-Consistency technique presented in the paper [22]. The original self-consistency technique instructs the model to generate different answers to a question and chooses the answer by majority vote. Since the translations are too different, it is not possible to just take the one that appears most often. Therefore, we instruct the model to choose itself which translation to take as the final answer.

Token limits and the context window are two important hyperparameters to consider when prompting large language models. The token limit stops the model's response when the limit is reached. It is important to set the limit so that the model has enough space for the entire translation. If it is set too low, the metrics will be severely affected because parts of the translation may be missing. If it is set too high, there should be no concern if the model is decent. Decent means that the model stops when the translation is finished. If the model does not stop, some metrics will decrease and the time to generate the answer will increase. With many iterations this can lead to a large loss of time. The context window determines how many tokens the model can remember. This hyperparameter is especially important for techniques with more than one model interaction. If the context window is set too small, the model responses will be cut off at the end, affecting all metrics. In the next section we describe and explain the metrics, as well as our reasons for selecting them.

Table 2. Descriptions of Prompting Techniques.

Prompting Techniques	**Description**
ZeroShotTechnique	our baseline prompt
ZeroShotTechniqueJSONOutput	explicitly giving the model the format it should use
ZeroShotTechniqueHighlightOnly	put important instructive words in **
ZeroShotTechniqueTranslationTextFirst	translation text is put in front of the instructive prompt
ZeroShotTechniqueConcretePromptEnding	putting a special end-token at the end `<|endofprompt|>`
ZeroShotTechniqueAllTogether	combining the four previous techniques into one prompt
InstructionTechnique	give the model explicit instructions
PersonaTechnique	give the model a relevant role, for example a translator
FewShotTechnique	provide a few examples relevant for the task
FewShotTechniqueTranslationTextFirst	translation text is put in front of the instructive prompt
FewShotTechniqueWithExampleToken	use a token to mark the examples `<example1><endExample1>`
FewShotTechniqueContinueAfterExamples	provides two examples to illustrate the expected output
AllTogetherTechnique	combination of text first, persona, fewshot, highlighting and endtoken techniques
SelfCorrectTechnique	the model is able to correct its first response once
ComparisonTechnique	the model gives two responses and is then asked to choose the better one

6 Metrics

BLEU (Bilingual Evaluation Understudy) compares a reference text with a machine translation and measures how many n-grams of the machine translation appear in the reference text. Using a modified n-gram precision, it ensures that only as many equal n-grams are counted as there are total n-grams in the reference text [20]. The BLEU score is calculated using the following formula:

$$\text{BLEU} = \text{BP} \times \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{1}$$

where:

- p_n is the precision for n-grams (the fraction of n-grams from the machine translation that appear in the reference text),
- w_n is the weight for each n-gram level (usually uniform),
- N is the highest n-gram level considered (e.g., 1-gram, 2-gram, etc.),
- BP is the brevity penalty, which is given by:

$$\text{BP} = \begin{cases} 1, & \text{if } c \geq r \\ \exp(1 - \frac{r}{c}), & \text{if } c < r \end{cases} \tag{2}$$

where:

- r is the length of the reference text,
- c is the length of the machine translation.

To prevent very short translations, it adds a multiplicative brevity penalty factor that is 1 if the length of the machine translation is similar to or longer than the length of the reference text, and $e^{(1-r/c)}$ if the machine translation is shorter. The n-gram comparison already penalizes long machine translations inherently. The final score is bounded between 0 and 1, with higher values indicating a better score. BLUE does not take into account synonyms or semantics in general and penalizes different word orders, no matter how correct they may be [20].

ROUGE has different possible variations, but we focus on ROUGE1, ROUGE2 and ROUGEL. ROUGE1 focuses on unigrams and reflects in the score how many words of the reference text appear in the machine translation. This metric does not consider the order of words or the structure of the sentence [15]. The ROUGE1 score is given by:

$$\text{ROUGE-1} = \frac{\sum_{w \in \text{Words}} \text{count}_{\text{match}}(w)}{\sum_{w \in \text{Words}} \text{count}_{\text{ref}}(w)} \tag{3}$$

ROUGE2 measures the overlap of bigrams and therefore does evaluate the order of words and structure of the sentence [15]. The ROUGE2 score is given by [15]:

$$\text{ROUGE-2} = \frac{\sum_{\text{bigram } w \in \text{Bigrams}} \text{count}_{\text{match}}(w)}{\sum_{\text{bigram } w \in \text{Bigrams}} \text{count}_{\text{ref}}(w)} \tag{4}$$

ROUGEL measures the longest common sequence of words that appear in the reference text and the machine translation [15]. The ROUGEL score is calculated as:

$$\text{ROUGE-L} = \frac{LCS(\text{MT}, \text{Ref})}{\text{Length of Ref}} \tag{5}$$

ROUGE-L rewards machine translations that keep the same order of words as the reference text, even if there are extra words in between [15].

BERTScore basically consists of three key steps, as described in this paper [23]. The first step is to generate contextualized word embeddings for the reference text and the machine translation. For each embedded token from the machine translation, the cosine similarity to the most similar embedded token from the reference text is computed. The average similarity of all these differences is then used to compute precision, recall, and F1 score.

The BERTScore primarily captures the semantic similarity between the reference text and the machine translation. This is important because it rewards similar but not exact word choices, rather than penalizing synonyms the same as unrelated words. The precision score is computed as:

$$\text{Precision} = \frac{1}{|\text{MT}|} \sum_{t \in \text{MT}} \max_{r \in \text{Ref}} \text{sim}(t, r) \tag{6}$$

where $\text{sim}(t, r)$ represents the cosine similarity between the embedded tokens t (from the machine translation) and r (from the reference text).

The recall score is computed as:

$$\text{Recall} = \frac{1}{|\text{Ref}|} \sum_{r \in \text{Ref}} \max_{t \in \text{MT}} \text{sim}(t, r) \tag{7}$$

Finally, the F1 score is the harmonic mean of precision and recall:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{8}$$

BLEU and ROUGE were both chosen as pre-implemented metrics by MLFlow and the BERTScore as an additional more recent metric. BLEU and ROUGE1 are 1-gram metrics, meaning they tokenize the LLM response and compare how many single words match with the reference text. This is very helpful to get a basic understanding of how close the response was regarding the choice of words. Adding to this, we selected ROUGE2 and ROUGEL, which check for 2-grams and the longest common sequence between the response and the reference text. These two metrics therefore measure how close the response was regarding the order of the words. Lastly, the BERTScore measure the cosine similarity between the words chosen in the response and the machine translation. This provides us with an idea about the semantic similarity between the response and the reference text. The next sections presents our results using these exact metrics.

7 Results

In this section, we present our MLflow-generated artifacts, analyzing the overall performance of various prompting techniques, the differences between the two language directions, the impact of text complexity, and the effectiveness of the models themselves.

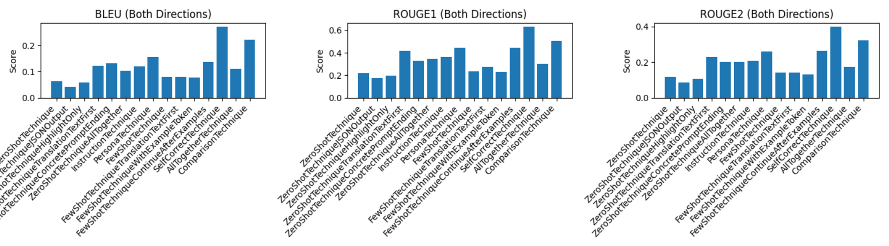


Fig. 2. Prompting Technique Comparison

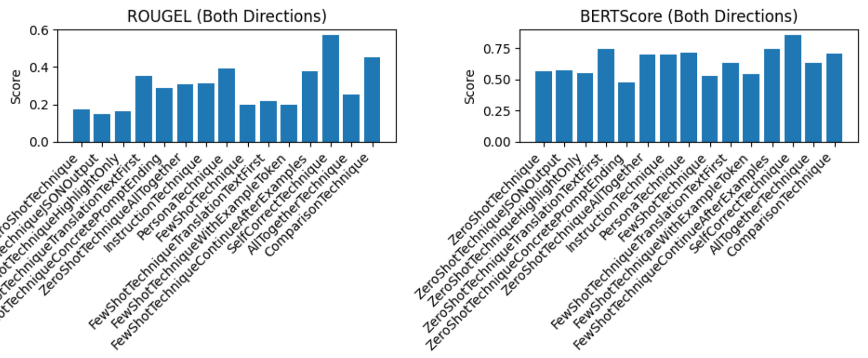


Fig. 3. Prompting Technique Comparison

7.1 Comparison of Prompting Techniques

Our evaluation reveals several interesting insights about the effectiveness of the different prompting techniques, that can be seen in Fig. 2 and Fig. 3. Even though the basic zero-shot techniques work quite well, a few small changes greatly increase their effectiveness, showing that even small adjustments can have a big impact. The few-shot techniques, on the other hand, seem to perform not significantly better than the basic zero-shot techniques. Only one few-shot technique performs decently well, suggesting that adding examples does not guarantee an improved performance. Furthermore, techniques based on reasoning and self reflection like the *Comparison* technique and the *SelfCorrect* technique achieve the best results across all metrics. Finally, our results show that combining different prompting techniques does not always lead to better performance, highlighting the need to carefully design prompts rather than simply making them more complex.

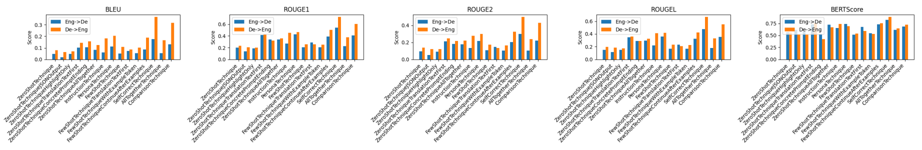


Fig. 4. Language Direction Comparison

7.2 Comparison of Translation Direction

Another noticeable difference is in the translation direction (Fig. 4). In general, the performance is more accurate from German to English, which is not really surprising since the models were trained mostly on English text. The biggest gap appears with the *SelfCorrect* technique. Moreover, the more the metric changes from exact word comparison to semantic comparison, the smaller these differences become. This suggests that even though the models seem to produce more precise words in English, the meaning behind the German equivalents they choose is similar. Especially in the BERTScore metric it is very obvious that there is no significant difference between the performance of the directions, which means that the models still convey the meaning correctly, regardless of the translation direction. For a larger version of this figure, see Appendix Fig. 12.

7.3 Comparison of Text Complexities

Text complexity seems to have only a minimal effect on translation quality (Fig. 5). However, an unexpected observation is that the scientific text performs marginally but still noticeably better. Pop science texts and simple texts seem to perform about the same. From the general news text to the science text, the performance seems to increase gradually. Another interesting observation is that the consistency of the score depends on the language direction. For English to German, all techniques perform fairly equally, while for German to English, the scores for the techniques differ drastically. There is a line chart for each metric, and they are included in the Appendix.

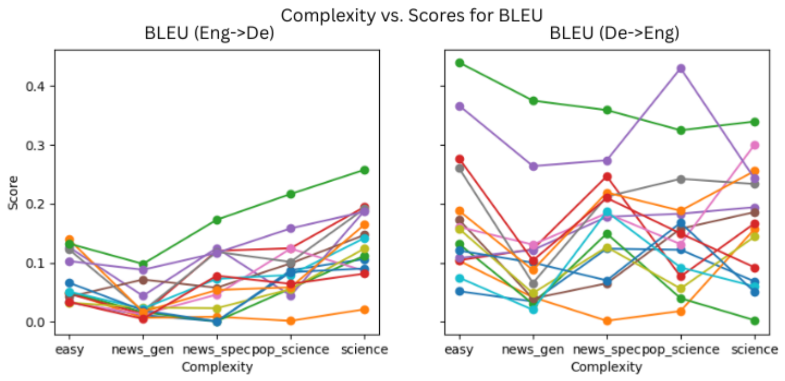


Fig. 5. Text Complexity Comparison with BLEU

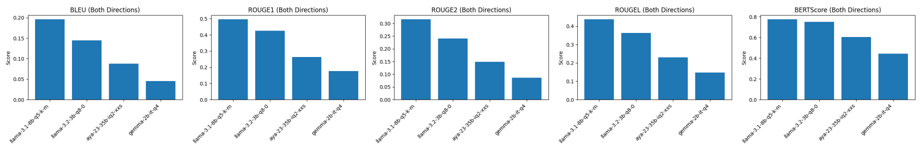


Fig. 6. Model Ranking

7.4 Comparison of Model Performance

For the models, there is a very clear hierarchy regarding their translation performance, which is consistent across all metrics (Fig. 6). The key observation here is that the model recency seems to have the greatest impact on translation performance. Notably, recent developments in LLMs and parameter sizes seem to have had a positive impact, as the newer models consistently outperformed the older ones. This improvement is clearly visible during a quality check of the model outputs, as the smallest model returned nothing in 80% of its responses, and the largest model repeated sentences or words very often until the maximum number of tokens was reached. The smallest model, gemma-2b-it, was released in February 2024 [11], and the largest model, aya-23-35B, was released in May 2024 [6]. Both Llama models were released months later, with Llama-3.1-8B-Instruct in July 2024 [18] and Llama-3.2-3B-Instruct-Q8_0 in September 2024 [19]. However, the number of parameters is also important. This is most noticeable when actually instructing the model to generate tokens. In addition, if the models have a similar architecture, the number of parameters seems to matter, as can be seen between the two Llama models. We discuss these results in the following section.

8 Discussion

In this section we are discussing and evaluating our results. Generally we can definitely conclude that prompting techniques have a significant impact on the quality of the output of LLMs regarding machine translation tasks.

8.1 Insights

Our evaluation of prompting techniques provides several key insights into their effectiveness and limitations.

The analysis of prompting techniques highlights the significant impact that even minor modifications can have on performance. While basic zero-shot techniques produce decent results, small modifications significantly improve their effectiveness. Surprisingly, few-shot techniques do not consistently outperform zero-shot methods, with only one few-shot approach showing comparative results. This suggests that the mere addition of examples does not guarantee improved performance.

The techniques that use reasoning and self-reflection, such as the *Comparison* and *SelfCorrection* techniques, achieve the highest scores across all metrics. This suggests that structured reasoning and iterative refinement play a critical role in improving translation quality. In particular, the *SelfCorrection* technique shows a very high performance, suggesting that allowing models to check and adjust their own output leads to better results. In contrast, more straightforward approaches do not achieve similar levels of performance, highlighting the importance of incorporating deeper reasoning mechanisms.

Interestingly, combining different prompting techniques does not necessarily lead to better results. This highlights the need to carefully design prompts rather than to simply increase their complexity. Overly complicated prompts can introduce noise rather than precision, reducing the effectiveness of the approach. Furthermore, it suggests that strategic prompt design is more important than only combining different prompting techniques. This emerges as a key consideration in optimizing translation performance.

8.2 Challenges

During the course of this project, we encountered several challenges, both technological and content-related, that influenced our approach and progress.

Technical Challenges started with getting the pipeline to run on Windows. Our biggest technical challenge was ensuring that our pipeline would work properly on Windows environments. Since many machine learning tools and frameworks are optimized for Unix-based systems, adapting them for Windows required time-consuming debugging. Specifically, a file from the llama-cpp-python package was always built with an x32 Windows architecture on our x64 architecture. This took forever to fix. Also, the pipeline failed even after all files had the correct architecture. Somehow the import of mlflow has to come after the import of llama-cpp-python, which caused us a lot of headaches until we figured it out.

In addition, the extremely long pipeline execution time proved to be a major limitation. Specifically, the largest model, aya-23-35B, with its 35,000,000,000 parameters and high number of prompts, 180 to be exact, took about 11 hours to run. With such a long timeframe, we had no choice but to let the pipeline run through the night and wait for the results the next morning.

Content challenges began with identifying prompting techniques based on existing scientific papers, which requires extensive literature research. It is easy to find a large number of scientifically presented prompting techniques, but most of them are very advanced and complex, or not practically applicable to our translation task.

Initially, there was no clear benchmark for the translation scores achieved, which made it difficult to interpret the translation performance. After running some tests and talking to the other two groups, we gained an understanding of how to evaluate the different scores of the different prompting techniques. Another challenge was to refine the prompts in a structured and systematic way. Due to the black box nature, there is no deterministic relationship between the prompt and the output, especially given the variety of factors that influence the translation quality. Systematically adapting prompts then required a lot of experimentation and iteration to determine the most effective techniques. The biggest influence we found was controlling the output, i.e. effectively instructing the model to output only the translation and not any additional unrelated text.

9 Conclusion

In conclusion, our results highlight the impact of prompt design, translation direction, text complexity and model architecture in machine translation. Well-structured prompts, especially reasoning-based techniques such as self-correction and comparison methods, significantly improve translation quality. While few-shot prompting shows potential, its inconsistent benefits and the effort required to create examples make it less efficient in practice. Importantly, we found that combining multiple prompting techniques does not necessarily yield better results, suggesting that concisely designed prompts are often more effective. This aligns with our general principle: include as much as necessary, but as little as possible. Because modern AI chatbots offer a chat-like interface and features that allow the AI model to store and remember context, it is very easy to let the model correct itself to improve performance. The translation performance is generally stronger from German to English, probably because the models are trained with mostly English textual training data. However, the semantic meaning is largely preserved in both directions, even if the exact word choice is different. While text complexity shows minimal impact on overall translation quality, scientific texts demonstrate marginally better performance, possibly due to their structured nature and consistent terminology. Model characteristics emerge as the most influential factor, with two key findings: First, newer models consistently outperform their predecessors, suggesting significant improvements in architecture and training methodologies. Second, while the number of parameters correlates with improved performance, this effect is less pronounced than the impact of recent architectural advances.

Looking ahead, future research could focus on context aware prompting techniques, given that context awareness is a fundamental feature in modern AI chatbots. This can significantly improve the performance of machine translations as well as maintain continuity in conversations, understand nuanced user input, and change responses based on previous interactions. However, refining prompting techniques to make even better use of context remains an open challenge.

References

1. Abetlen, A.: llama-cpp-python: Python Bindings for llama.cpp (2023), https://llama-cpp-python.readthedocs.io/en/latest/api-reference/, accessed: 2025-02-14
2. Alassan, M.S.Y., Espejel, J.L., Bouhandi, M., Dahhane, W., Ettifouri, E.H.: Comparison of open-source and proprietary llms for machine reading comprehension: A practical analysis for industrial applications (2024), https://arxiv.org/abs/2406.13713
3. Bhargava, A., Witkowski, C., Looi, S.Z., Thomson, M.: What's the magic word? a control theory of llm prompting. arXiv preprint arXiv:2310.04444 (2023)
4. Bhargava, A., Witkowski, C., Looi, S.Z., Thomson, M.: What's the magic word? a control theory of llm prompting (2024), https://arxiv.org/abs/2310.04444
5. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners (2020), https://arxiv.org/abs/2005.14165
6. Cohere: Aya 23: Open-source large language model. https://cohere.com/blog/aya23 (2024), accessed: 2025-02-14
7. Cohere for AI: Aya 23 35b model. https://huggingface.co/CohereForAI/aya-23-35B (2024), accessed: 2025-02-14
8. daveshapautomator: Advantages of using default end token? (<|endoftext|>), https://community.openai.com/t/advantages-of-using-default-end-token-endoftext/4635, accessed: 2025-02-14
9. Dhuliawala, S., Komeili, M., Xu, J., Raileanu, R., Li, X., Celikyilmaz, A., Weston, J.: Chain-of-verification reduces hallucination in large language models. arXiv preprint arXiv:2309.11495 (2023)
10. GGML Organization: Gguf file format documentation. https://github.com/ggml-org/ggml/blob/master/docs/gguf.md (2024), accessed: 2025-02-14
11. Google AI: Gemma: Google's open models. https://blog.google/technology/developers/gemma-open-models (2024), accessed: 2025-02-14
12. Holtzman, A., Buys, J., Du, L., Forbes, M., Choi, Y.: The curious case of neural text degeneration (2020), https://arxiv.org/abs/1904.09751
13. Kumar, S., Talukdar, P.: Reordering examples helps during priming-based few-shot learning. arXiv preprint arXiv:2106.01751 (2021)
14. Li, Z., Peng, B., He, P., Galley, M., Gao, J., Yan, X.: Guiding large language models via directional stimulus prompting. Advances in Neural Information Processing Systems **36** (2024)
15. Lin, C.Y.: Rouge: A package for automatic evaluation of summaries. In: Text summarization branches out. pp. 74–81 (2004)
16. LM Studio AI: Gemma 2b it gguf model. https://huggingface.co/lmstudio-ai/gemma-2b-it-GGUF (2024), accessed: 2025-02-14
17. Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., et al.: Language models are few-shot learners. arXiv preprint arXiv:2005.14165 **1** (2020)
18. Meta AI: Llama 3.1 8b instruct model. https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct (2024), accessed: 2025-02-14

19. Meta AI: Llama 3.2 3b instruct model. `https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct` (2024), accessed: 2025-02-14
20. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting of the Association for Computational Linguistics. pp. 311–318 (2002)
21. Roberts, J.: How powerful are decoder-only transformer neural models? In: 2024 International Joint Conference on Neural Networks (IJCNN). p. 1–8. IEEE (Jun 2024). `https://doi.org/10.1109/ijcnn60899.2024.10651286`, `http://dx.doi.org/10.1109/IJCNN60899.2024.10651286`
22. Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., Zhou, D.: Self-consistency improves chain of thought reasoning in language models. arXiv preprint arXiv:2203.11171 (2022)
23. Zhang, T., Kishore, V., Wu, F., Weinberger, K.Q., Artzi, Y.: Bertscore: Evaluating text generation with bert. arXiv preprint arXiv:1904.09675 (2019)
24. Zheng, M., Pei, J., Logeswaran, L., Lee, M., Jurgens, D.: When" a helpful assistant" is not really helpful: Personas in system prompts do not improve performances of large language models. In: Findings of the Association for Computational Linguistics: EMNLP 2024. pp. 15126–15154 (2024)

A Appendix

The code used for the experiments and analysis presented in this paper is publicly available.
You can access the repository at:
`https://github.com/MoritzderProgrammierer/comparison-of-LLM-prompting-techniques`

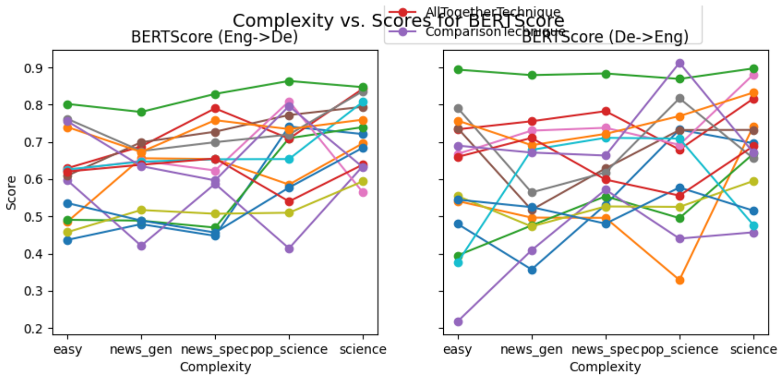


Fig. 8. Complexity Overview (BERTScore).

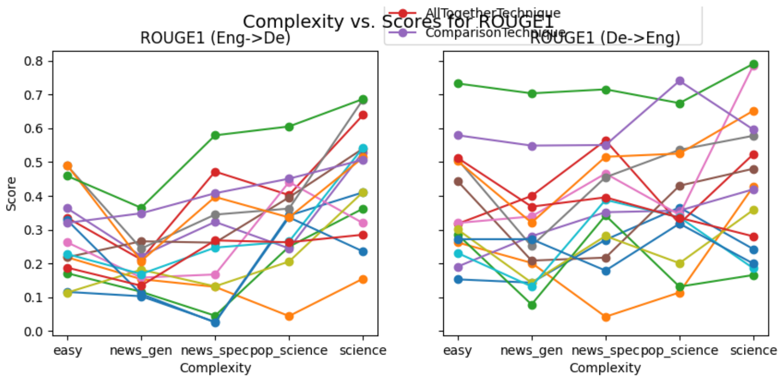


Fig. 9. Complexity Overview (ROUGE1).

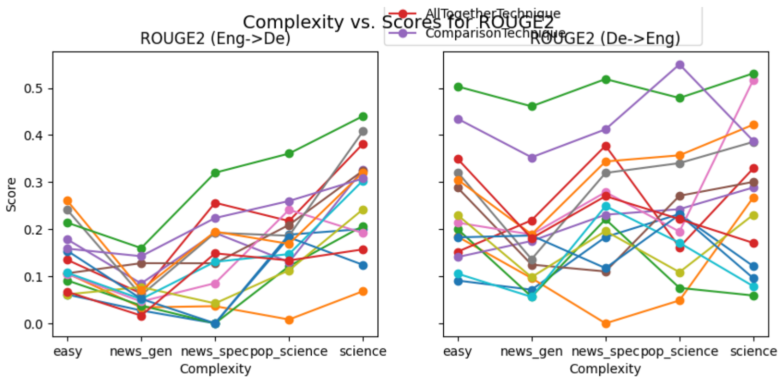


Fig. 10. Complexity Overview (ROUGE2).

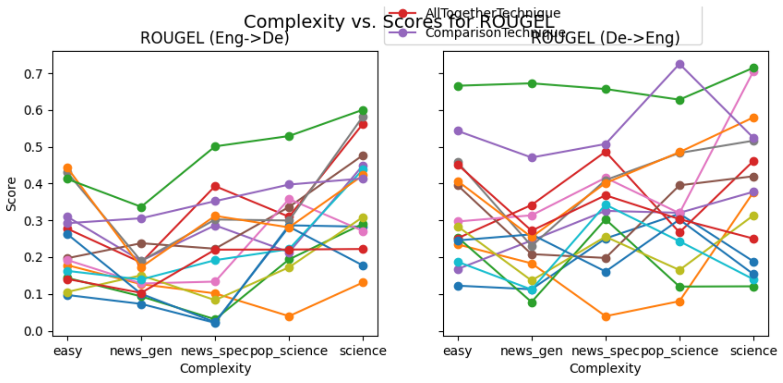
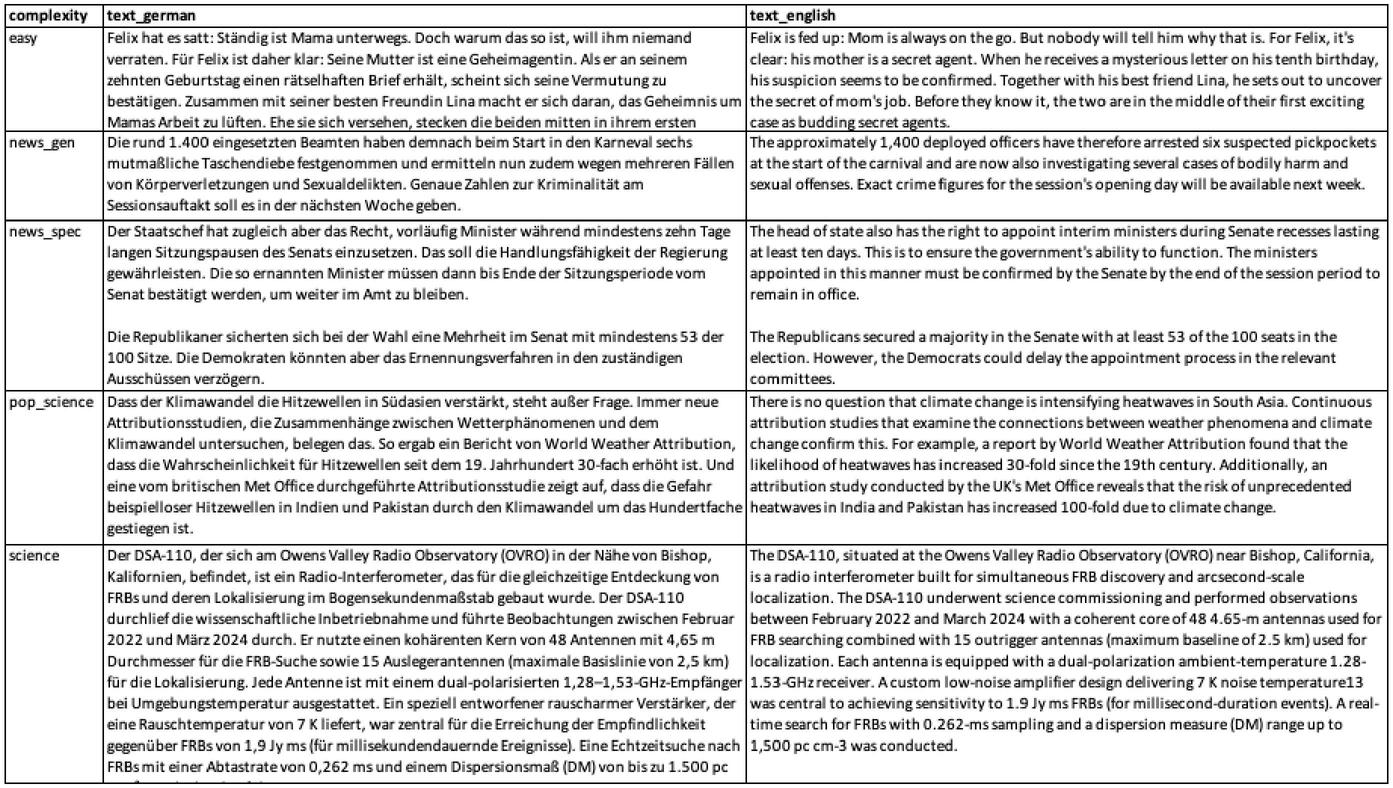


Fig. 11. Complexity Overview (ROUGEL).



complexity	text_german	text_english
easy	Felix hat es satt: Ständig ist Mama unterwegs. Doch warum das so ist, will ihm niemand verraten. Für Felix ist daher klar: Seine Mutter ist eine Geheimagentin. Als er an seinem zehnten Geburtstag einen rätselhaften Brief erhält, scheint sich seine Vermutung zu bestätigen. Zusammen mit seiner besten Freundin Lina macht er sich daran, das Geheimnis um Mamas Arbeit zu lüften. Ehe sie sich versehen, stecken die beiden mitten in ihrem ersten	Felix is fed up: Mom is always on the go. But nobody will tell him why that is. For Felix, it's clear: his mother is a secret agent. When he receives a mysterious letter on his tenth birthday, his suspicion seems to be confirmed. Together with his best friend Lina, he sets out to uncover the secret of mom's job. Before they know it, the two are in the middle of their first exciting case as budding secret agents.
news_gen	Die rund 1.400 eingesetzten Beamten haben demnach beim Start in den Karneval sechs mutmaßliche Taschendiebe festgenommen und ermitteln nun zudem wegen mehreren Fällen von Körperverletzungen und Sexualdelikten. Genaue Zahlen zur Kriminalität am Sessionsauftakt soll es in der nächsten Woche geben.	The approximately 1,400 deployed officers have therefore arrested six suspected pickpockets at the start of the carnival and are now also investigating several cases of bodily harm and sexual offenses. Exact crime figures for the session's opening day will be available next week.
news_spec	Der Staatschef hat zugleich aber das Recht, vorläufig Minister während mindestens zehn Tage langen Sitzungspausen des Senats einzusetzen. Das soll die Handlungsfähigkeit der Regierung gewährleisten. Die so ernannten Minister müssen dann bis Ende der Sitzungsperiode vom Senat bestätigt werden, um weiter im Amt zu bleiben. Die Republikaner sicherten sich bei der Wahl eine Mehrheit im Senat mit mindestens 53 der 100 Sitze. Die Demokraten könnten aber das Ernennungsverfahren in den zuständigen Ausschüssen verzögern.	The head of state also has the right to appoint interim ministers during Senate recesses lasting at least ten days. This is to ensure the government's ability to function. The ministers appointed in this manner must be confirmed by the Senate by the end of the session period to remain in office. The Republicans secured a majority in the Senate with at least 53 of the 100 seats in the election. However, the Democrats could delay the appointment process in the relevant committees.
pop_science	Dass der Klimawandel die Hitzewellen in Südasien verstärkt, steht außer Frage. Immer neue Attributionsstudien, die Zusammenhänge zwischen Wetterphänomenen und dem Klimawandel untersuchen, belegen das. So ergab ein Bericht von World Weather Attribution, dass die Wahrscheinlichkeit für Hitzewellen seit dem 19. Jahrhundert 30-fach erhöht ist. Und eine vom britischen Met Office durchgeführte Attributionsstudie zeigt auf, dass die Gefahr beispielloser Hitzewellen in Indien und Pakistan durch den Klimawandel um das Hundertfache gestiegen ist.	There is no question that climate change is intensifying heatwaves in South Asia. Continuous attribution studies that examine the connections between weather phenomena and climate change confirm this. For example, a report by World Weather Attribution found that the likelihood of heatwaves has increased 30-fold since the 19th century. Additionally, an attribution study conducted by the UK's Met Office reveals that the risk of unprecedented heatwaves in India and Pakistan has increased 100-fold due to climate change.
science	Der DSA-110, der sich am Owens Valley Radio Observatory (OVRO) in der Nähe von Bishop, Kalifornien, befindet, ist ein Radio-Interferometer, das für die gleichzeitige Entdeckung von FRBs und deren Lokalisierung im Bogensekundenmaßstab gebaut wurde. Der DSA-110 durchlief die wissenschaftliche Inbetriebnahme und führte Beobachtungen zwischen Februar 2022 und März 2024 durch. Er nutzte einen kohärenten Kern von 48 Antennen mit 4,65 m Durchmesser für die FRB-Suche sowie 15 Auslegerantennen (maximale Basislinie von 2,5 km) für die Lokalisierung. Jede Antenne ist mit einem dual-polarisierten 1,28–1,53-GHz-Empfänger bei Umgebungstemperatur ausgestattet. Ein speziell entworfener rauscharmer Verstärker, der eine Rauschtemperatur von 7 K liefert, war zentral für die Erreichung der Empfindlichkeit gegenüber FRBs von 1,9 Jy ms (für millisekundendauernde Ereignisse). Eine Echtzeitsuche nach FRBs mit einer Abtastrate von 0,262 ms und einem Dispersionsmaß (DM) von bis zu 1.500 pc	The DSA-110, situated at the Owens Valley Radio Observatory (OVRO) near Bishop, California, is a radio interferometer built for simultaneous FRB discovery and arcsecond-scale localization. The DSA-110 underwent science commissioning and performed observations between February 2022 and March 2024 with a coherent core of 48 4.65-m antennas used for FRB searching combined with 15 outrigger antennas (maximum baseline of 2.5 km) used for localization. Each antenna is equipped with a dual-polarization ambient-temperature 1.28-1.53-GHz receiver. A custom low-noise amplifier design delivering 7 K noise temperature13 was central to achieving sensitivity to 1.9 Jy ms FRBs (for millisecond-duration events). A real-time search for FRBs with 0.262-ms sampling and a dispersion measure (DM) range up to 1,500 pc cm-3 was conducted.

Fig. 7. Translation Data

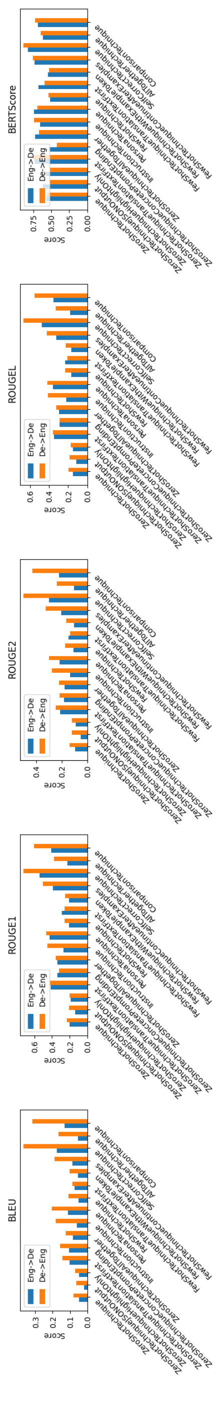
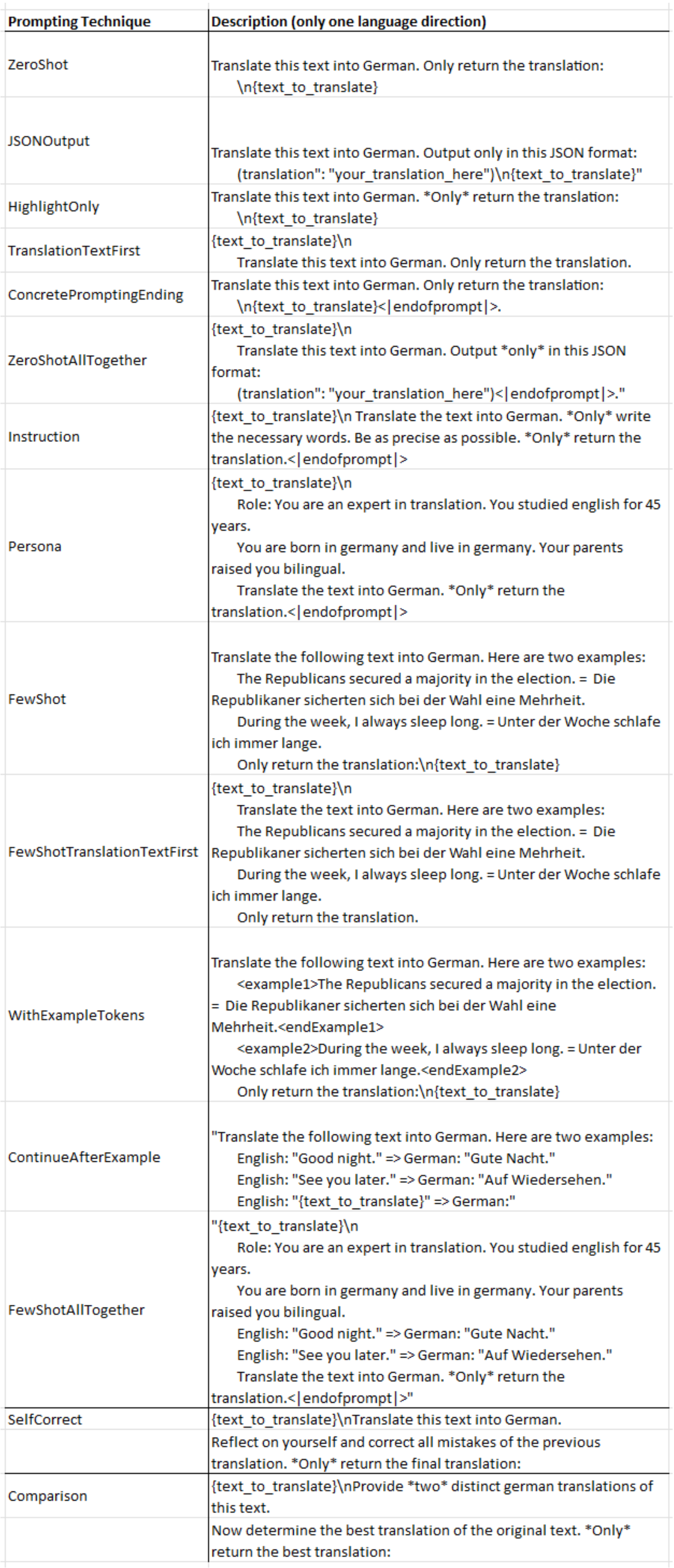


Fig. 12. Language Direction Comparison



Prompting Technique	Description (only one language direction)
ZeroShot	Translate this text into German. Only return the translation: \n{text_to_translate}
JSONOutput	Translate this text into German. Output only in this JSON format: (translation": "your_translation_here")\n{text_to_translate}"
HighlightOnly	Translate this text into German. *Only* return the translation: \n{text_to_translate}
TranslationTextFirst	{text_to_translate}\n Translate this text into German. Only return the translation.
ConcretePromptingEnding	Translate this text into German. Only return the translation: \n{text_to_translate}<|endofprompt|>.
ZeroShotAllTogether	{text_to_translate}\n Translate this text into German. Output *only* in this JSON format: (translation": "your_translation_here")<|endofprompt|>."
Instruction	{text_to_translate}\n Translate the text into German. *Only* write the necessary words. Be as precise as possible. *Only* return the translation.<|endofprompt|>
Persona	{text_to_translate}\n Role: You are an expert in translation. You studied english for 45 years. You are born in germany and live in germany. Your parents raised you bilingual. Translate the text into German. *Only* return the translation.<|endofprompt|>
FewShot	Translate the following text into German. Here are two examples: The Republicans secured a majority in the election. = Die Republikaner sicherten sich bei der Wahl eine Mehrheit. During the week, I always sleep long. = Unter der Woche schlafe ich immer lange. Only return the translation:\n{text_to_translate}
FewShotTranslationTextFirst	{text_to_translate}\n Translate the text into German. Here are two examples: The Republicans secured a majority in the election. = Die Republikaner sicherten sich bei der Wahl eine Mehrheit. During the week, I always sleep long. = Unter der Woche schlafe ich immer lange. Only return the translation.
WithExampleTokens	Translate the following text into German. Here are two examples: <example1>The Republicans secured a majority in the election. = Die Republikaner sicherten sich bei der Wahl eine Mehrheit.<endExample1> <example2>During the week, I always sleep long. = Unter der Woche schlafe ich immer lange.<endExample2> Only return the translation:\n{text_to_translate}
ContinueAfterExample	"Translate the following text into German. Here are two examples: English: "Good night." => German: "Gute Nacht." English: "See you later." => German: "Auf Wiedersehen." English: "{text_to_translate}" => German:"
FewShotAllTogether	"{text_to_translate}\n Role: You are an expert in translation. You studied english for 45 years. You are born in germany and live in germany. Your parents raised you bilingual. English: "Good night." => German: "Gute Nacht." English: "See you later." => German: "Auf Wiedersehen." Translate the text into German. *Only* return the translation.<|endofprompt|>"
SelfCorrect	{text_to_translate}\nTranslate this text into German.
	Reflect on yourself and correct all mistakes of the previous translation. *Only* return the final translation:
Comparison	{text_to_translate}\nProvide *two* distinct german translations of this text.
	Now determine the best translation of the original text. *Only* return the best translation:

Fig. 13. Implementation of our Prompting Techniques