MNLP - Homework 2 - Text Cleaning

Group Name: AttentionSeekers

Jędrzej Miczke
miczkejedrzej@gmail.com
Clemens Kubach
kubach@campus.tu-berlin.de

1 Introduction

The aim of this project is to develop and comprehensively evaluate different approaches for denoising the output of Optical Character Recognition (OCR) systems. As such systems, we leverage smaller Large Language Models (LLMs) and evaluate the results using both traditional OCR metrics (Neudecker et al., 2021) and LLM-based assessment ("LLM as a judge").

2 Methodology

Our approach consists of two parts: denoising the OCR-detected text to a cleaned version of it and subsequent evaluation, including comparative analysis.

2.1 Denoising

2.1.1 Models

We perform the OCR denoising task using the two following causal language models:

- `meta-llama/Llama-3.2-1B-Instruct`
- `google/gemma-3-1b-it`

We use models with a similar parameter count - around 1 billion - to reduce the influence of a bias due to the model size. The models are leveraged in various configurations, namely baseline, advanced, and fine-tuned.

Baseline Setting In the baseline approach, the prompt used for the denoising task, see appendix A.2.1, includes an introduction of the task, instructions on what should be done and what not, and a disclaimer requesting that no additional text be output.

Advanced Setting The advanced setting employs in-context learning, a technique to adapt LLM to specific downstream tasks. Our implementation uses 5-shot learning with the `[user, assistant, model]` template. We explicitly provide the error description, the corrupted text, and the corrected text. Additionally, we give the LLMs clear instructions on the most common OCR mistakes to explicitly guide their attention. The errors frequency is derived from the dataset itself and, while it can be understood as data leakage, it is intentional to test how explicit knowledge of underlying common errors affects model. For detailed implementation, see appendix A.2.2.

Fine-tuned Setting The fine tuning of the two models is performed using the LLaMA-Factory library. To this end, we have written functions that configure the appropriate training configurations and all other necessary data and configurations for the selected pretrained models, Gemma and Llama. This includes configurations for model export and registration of a custom hand-crafted fine-tuning dataset (see section 2.1.2) in the LLaMA Factory “dataset-info.json”. Both pretrained LLMs undergo supervised fine-tuning for 3 epochs using the "LoRA" method and the Llama model undergo a full training of all parameters additionally. Because the task is computationally demanding, we utilize the Leonardo HPC using one custom NVIDIA Ampere A100 GPU with 64GB. We encountered some bugs in the process see(A.3)

2.1.2 Fine-tuning Dataset

Our fine-tuning dataset is based on the work of (Langlais, 2024), which, to our knowledge, constitutes the largest publicly available collection of English OCR and post-correction text pairs. We analyze the error distribution in a randomly sampled portion of the corpus (approximately 6 million words), and use this distribution, normalized with uniform noise, to perturb two 19th-century English novels. This approach allows us to create data that reflects a plausible OCR error distribution and a vocabulary and style similar to our target dataset

2.1.3 Text Chunking

Although the given dataset is already divided into chunks, the respective texts can be very long. In some preliminary experiments, we felt that the models lost focus on the task when dealing with longer texts. We therefore decided to apply the denoising model sentence by sentence To do this, the character string “. ” was used for division and then reassembled after denoising.

2.2 Evaluation

2.2.1 Evaluation Metrics

Classic metrics We employ standard metrics to evaluate the quality of the denoised OCR text, including the character error rate (CER), the word error rate (WER), the precision of the unigram, the recall of the unigram, and the F1 score of the unigram.

LLM as a Judge In addition, we utilize the Gemini 2.5 Flash model as an automated evaluator to assess the denoising outputs. The model is prompted to evaluate the outputs with respect to four criteria: Faithfulness, Fluency, Completeness, and No Hallucinations. Based on these assessments, the model provides an overall ranking of denoised outputs.

We perform a series of **pairwise comparisons** between different LLMs within each configuration (baseline, advanced, fine-tuning),to assess the corresponding architecture's fit for the task and subsequently in other configurations see appendix A.5

To enhance model performance and better align its evaluations with human judgment, we incorporate few-shot learning. appendix A.5.

2.2.2 Human Evaluation

We perform a human evaluation on a subset of the denoising model outputs. This is done to assess the alignment between the LLM judge's evaluations and human judgment. To measure Judge LLM and human rater agreement, we use Cohen's kappa coefficient.

2.2.3 Evaluation Dataset

We applied and evaluated the denoising LLMs on the given dataset derived from the XIX-century novel *The Vampyre* by John Polidori. The dataset consists of clean text, as the original book content divided into variable-length chunks aligned with chapters and subsections, and OCR text, the corresponding noisy OCR chunks. The statistics of the dataset can be found in appendix A.1.

3 Experiments

We execute the denoising task using both pre-trained instruction-tuned models, Gemma3-1B and Llama3.2-1B, under the baseline and fine-tuned setting, as defined earlier (section 2.1.1).

For fine-tuning, we adopt **LoRA** (Low-Rank Adaptation) (Hu et al., 2021) for both models. It is a parameter-efficient technique that reduces the computational cost of aligning large language models (LLMs) by introducing low-rank updates instead of fine-tuning all parameters. Additionally, the Llama model is also fully fine-tuned on all parameters and is later evaluated. The decreasing train loss for fine-tuning of the Gemma model is shown exemplarily in fig. 1. We evaluate the performance of all the models with respect to the Non-LLM metrics and compare subset of the pairs with the LLM as judge, due to computational caveats see(A.5.4)

4 Results

4.1 Classic Metrics

Classic Metric analysis underscores the superior performance of the fine-tuned models, especially the `google/gemma-3-1b-it`."see figures (2,4,3). and detailed analysis in A.6

4.2 LLM-as-Judge Comparative Analysis

In each configuration, according to the provided instructions and evaluation criteria, `google/gemma-3-1b-it` consistently outperforms `meta-llama/Llama-3.2-1B-Instruct`. The benefit of fine-tuning is clear (see Section A.5) and fine-tuned `google/gemma-3-1b-it` emerges as the best model overall .

4.3 LLMs alignment

We compare the LLMs alignment with both: metrics such as ROUGE-1, ROUGE-2, and ROUGE-L and humanly annoted by us examples. The scores did not reveal a strong alignment; however, in both cases it was better than random chance. (see A.8). The Judge's decisions were more closely aligned with the ROUGE metrics (see tables Table 4,3).

References

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Pierre-Carl Langlais. 2024. Post-ocr text correction with modern nlp. Hugging Face Blog. Accessed: 2025-06-22.

Yinhong Liu, Han Zhou, Zhijiang Guo, Ehsan Shareghi, Ivan Vulić, Anna Korhonen, and Nigel Collier. 2025. Aligning with human judgement: The role of pairwise preference in large language model evaluators. *Preprint*, arXiv:2403.16950.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics.

Clemens Neudecker, Constantin Baierer, Mike Gerber, Cristian Clausner, Apostolos Antonacopoulos, and Stefan Pletschacher. 2021. A survey of ocr evaluation tools and metrics. https://dl.acm.org/doi/10.1145/3476887.3476888. Accessed: 2025-06-22.

Lin Shi, Chiyu Ma, Wenhua Liang, Xingjian Diao, Weicheng Ma, and Soroush Vosoughi. 2025. Judging the judges: A systematic study of position bias in llm-as-a-judge. *Preprint*, arXiv:2406.07791.

A Appendix

A.1 Dataset Details

The dataset comprises 47 variable length text-chunks preserving the original formating of the novel: "The Vampyre". The statistical analysis reveals significant variability of the lengths of the chunks ($\mu = 226$ words, $\sigma = 193$ words), with a minimum length of 2 words and a maximum of 843 words, together 10838 words. The aforementioned significant length variability allows us to evaluate the models sensitivity to the context length. The shorter chunks limit the models contextual information (or in the extreme cases don't provide any context)

A.2 Denoising Approaches Details

A.2.1 Baseline Settings

The following Prompt is used: "Clean and denoise the noisy text you will receive. It is determined from an optical character recognition (OCR) system. Denoise the text word by word or character by character to existing words. Do not change grammar, word order or meaning of the text. VERY IMPORTANT: Output ONLY the denoised version of the text. Do NOT output anything else."

A.2.2 Advanced Setting

As basic instruction prompt, we are using the prompt defined for the baseline setting A.2.1. The examples in the five-shot learning have the following structure:

few-shot learning

- Noisy OCR input (e.g., *"Th1s 1s a samp1e"*)
- Explanation of errors (e.g., *"substitution ('1' for 'l')"*)
- Corrected output (e.g., *"This is a sample"*)

To create the few-shot examples, we use the most frequent error types identified in our target dataset — The Vampyre — as described in Section A.2.2. We then apply each error type to a set of arbitrarily constructed simple sentences, inserting one error per sentence (e.g., substitution of '1' for 'l').

explicit error occurrence information The explicit information about the mistakes had the following structure

- **Example**:'The most frequent mistakes found in the data include':
  - `'t'` often misread as `'l'`
  - `'h'` often misread as `'b'`

explicit error analysis For determining the most common mistakes committed by the OCR system, we conduct an analysis of our target data, *The Vampyre*. Our approach consists of the following steps:

1. We employ a hand-crafted algorithm to align the clean text words with the corrupted OCR words based on the Levenshtein distance. This alignment also handles merged and over-segmented words, addressing cases where the OCR mistakenly subtracted or inserted spaces.
2. For each pair of aligned words, we use the Python library `Levenshtein` to obtain the edit operations between the original and OCR text.
3. We aggregate the edit operation data across all words in the dataset.
4. Finally, we apply the KMeans clustering algorithm to identify which types of error are frequent and typical for OCR, and which could be considered infrequent noise.

A.3 fine-tune-bugs

Since some bugs, like device mappings and handling the absence of a processor config in the Gemma model, occurred in LLaMA-Factory when using our training configurations and environment, we had to fix the issues and contributed it to an own fork[1] of the LLaMA-Factory repository.

A.4 Fine-tuning Dataset

For the fine-tuning, as mentioned in the main part, we develop our own dataset. We utilize the Post-OCR-Correction dataset (https://huggingface.co/datasets/PleIAs/Post-OCR-Correction), which serves as a large source of OCR and post-OCR corrected text pairs. However, we do not use it explicitly for the following reasons:

1. The post-OCR outputs are fairly clean, representing state-of-the-art corrections derived from very noisy OCR readings. However, they still contain various errors and are therefore not ideal for fine-tuning.
2. We aim to match, at least approximately, the style and vocabulary of our target denoising text, "The Vampyre", which the dataset does not provide, as it consists mainly of excerpts from American newspapers from different years.

Instead, we approximate the real OCR system error distribution with the help of this huge Post-OCR-Correction dataset.

A.4.1 Implementation of OCR Distribution Approximation

The pipeline for this algorithm proceeds as follows:

1. We randomly sample approximately 2% of the dataset, amounting to around 6 million words, divided into several hundred training examples.
2. Within each training example, we divided the `clean_text` into chunks of approximately 200 characters. Approximately because we strive to designate meaningful borders, dots signifying the sentence ends, or if not found, at least spaces.
3. For each `clean_text` chunks we iterate over the `ocr_corrupted` text using a sliding window of the same length as corresponding clean chunk with the stride of 5 characters to identify the best-matching text chunk corresponding to given `clean_text` .
4. To reduce computational complexity, we limit the matching process to a window of 1000 characters centered around the end of the previous found match, avoiding full-text search and ensuring, most of the times, possibility to match the corresponding clean chunk.
5. As an additional safety precaution, in case of the spurious alignment, if the best-matching `OCR_text` chunk exhibits poor similarity to the corresponding `clean_text` chunk, based on the exact matching of chunks (`difflib.SequenceMatcher.ratio() method`), we discard these chunks.
6. For aligned pairs, we extract symbol-level edit operations with the help of `difflib.SequenceMatcher`: substitutions, insertions, and deletions.
7. We aggregate these operations across all data, normalize them by category, and cluster the results using the KMeans algorithm to identify frequent OCR errors.

A.4.2 Perturbation Function Implementation

Based on the approximated error distribution, we implement a stochastic function to generate synthetic OCR-like artifacts in the text. This function includes:

- **Empirical error probabilities** for character-level substitutions, insertions, and deletions collected as mentioned in the previous subsection.
- **Uniform normalization noise** to prevent overfitting to the Post-OCR-Correction corpus and to support generalization.
- **Visual pattern errors**, capturing common OCR confusions involving multiple characters (e.g., `vv` $\rightarrow$ `w`).
- **Random non-alphanumeric noise** to mimic OCR sensor artifacts, encouraging the model to treat such symbols as spurious.

A.4.3 Fine-Tuning Dataset Construction

We apply the above perturbation functions to two 19th-century English novels: *Frankenstein* by Mary Shelley and *The Castle of Otranto* by Horace Walpole. Texts are split into chunks at meaningful boundaries (e.g., sentence ends), with chunk lengths similar to those used in the evaluation set from *The Vampyre*.

The clean and perturbed pairs from both novels

[1]https://github.com/ClemensKubach/LLaMA-Factory.git in branch "no-processor-fallback"

are then merged, resulting in the dataset that we use for the fine-tuning.

A.5 LLM as Judge - Gemini Flash 2.5

A.5.1 Motivation

We select the Gemini 2.5 Flash model for our evaluation setup because it is a large LLM in comparison to our 1B-parameter denoising models and it is free to use at high rate limits.

A.5.2 Evaluation criteria

We request the model to evaluate the denoising performance of the LLMs based on the following four criteria, with their meaning explicitly explained to the model in the exact same formulation as below:

1. **Faithfulness** — Does the `denoised_text` text preserve the meaning of `clean_text`?

2. **Fluency** — Is the text grammatically correct and naturally flowing?

3. **Completeness** — Is any information missing compared to `ocr_text`?

4. **No Hallucinations** — Are there added or altered details that were not present in `ocr_text` or `clean_text` ?

These criteria are suited to our task and often appear among the most common metrics for the evaluation of various text-generating LLM tasks (Maynez et al., 2020).

A.5.3 Pairwise Comparison Setting

We chose the pairwise evaluation setting because it is widely regarded to exhibit the highest correlation with human preferences (Liu et al., 2025).

We configure the prompt so that the LLM judge:

1. **Evaluates** the two outputs for each criterion, assigning:
   - **1 point** to the better output,
   - **0 points** if the outputs do not differ for that criterion.

2. **Tallies** the points for each output after scoring all criteria.

3. **Declares the winner** based on the output with the higher total score.

4. **Resolves ties** by applying the following priority order as a **tiebreaker**:
   1. **Fluency**
   2. **Faithfulness**
   3. **No Hallucinations**
   4. **Completeness**

5. If all criteria are scored **0**, the judge is instructed to **randomly select** the better output.

6. To improve alignment with human evaluation and guide the model's decision-making, we introduce the **few-shot learning** by incorporation of evaluated synthetic example pairs. Specifically, for each of the four criteria, we design one example where the two denoised outputs differ only with respect to that criterion. To avoid positional bias, we choose the first denoised output as superior in two examples and the second denoised output in the remaining two.

To address the well-known issue of positional bias in LLM-as-a-judge evaluations(Shi et al., 2025), we randomly shuffle the order of denoised outputs for each prompt. Although this strategy does not eliminate positional bias, it reduces its impact in pairwise comparisons by distributing the bias symmetrically across both models.

A.5.4 Shortcomings of the Pairwise Comparison Setting

Despite its strong alignment with human judgment, the pairwise evaluation setting has notable drawbacks. As the number of models or configurations increases, the number of required comparisons grows quadratically—$\mathcal{O}(n^2)$—leading to a potentially intractable evaluation process.

To keep the evaluation feasible, we limited our experiments to comparisons within each of the prompt settings i.e: the baseline prompt, in-context learning, and fine-tuning. We first evaluate the models within each of these categories and only then compare the best-performing ones to identify a "winner".

However, it is important to emphasize that a notion of winner must be interpreted with caution in this case. Due to the inherent randomness and non-determinism of LLM outputs, there is no guarantee of transitivity across evaluations.(Liu et al., 2025)

A.6 Results classic metrics analysis

A.6.1 Unigram-Based Metrics

As shown in the results, the fine-tuned models clearly outperform the other configurations with respect to unigram-based metrics: *precision*, *recall*, and *F1 score*. They achieve higher average scores and display substantially lower variability compared to the other configurations (see Table 1) "see figures (2,4,3).

A.6.2 Edit-Based Metrics: CER and WER

In contrast, when examining edit-based metrics—namely, the *character error rate* (CER) and *word error rate* (WER), which quantify the number of edit operations per character and word respectively—the fine-tuned `gemma` model shows extreme variability (see Table 2) and unusually high error values. This behavior is highly undesirable in practical applications.

Closer inspection reveals that this outlier behavior is caused by a single instance in which the model generates an excessively long and hallucinated output, inflating both CER and WER dramatically. It displays the dangers of the hallucinations in the LLMs and it's deplorable effects on the task related performance.

A.6.3 In-Context Learning Impact

Interestingly, the in-context learning configuration did not lead to improvements in classic metrics. In fact, it appears to have a detrimental effect on overall performance—especially in the case of the `gemma` model—suggesting that in-context learning may not be well-suited for this task or data setting, or may have not been adapted properly to the task.

A.7 LLM-as-Judge Evaluation Results

As mentioned previously in Appendix A.5, we employed the Gemini 2.5 Flash LLM, providing it with a consistently formatted and clearly structured prompt. This prompt included detailed instructions for the evaluation task, along with few-shot examples to guide proper assessment.

A.7.1 Baseline Setting: Architectures Comparison

In the baseline setting, the Gemma model outperformed the simple LLAMA model in the majority of cases, particularly on the criteria of *faithfulness*, *completeness*, and *no hallucinations*. However, LLAMA demonstrated superior performance in terms of *fluency* in most instances.(see figure 9) the figure represents how many times each model wins in each category, the same for subsequent figures in this sections.

This outcome suggests that Gemma aligns more closely with task requirements by avoiding artifacts and hallucinations, even if its denoising capabilities might be slightly weaker.

A.7.2 Advanced Setting: Architectures Comparison

In the advanced setting, Gemma again emerges as the overall winner. Notably, it demonstrates a marked advantage in *fluency*, while the differences in other criteria diminish compared to the baseline scenario. Nonetheless, Gemma consistently outperforms LLAMA across the evaluation. (see datails 10)

A.7.3 Fine-Tuning Architectures Comparison

Both the fine-tuned Gemma and fine-tuned LLAMA models perform well according to the classic metrics, making this a competitive comparison. Ultimately, Gemma is again judged to be the overall winner. However, LLAMA is favored in terms of *fluency* in most examples. Despite a relatively balanced distribution of wins across individual criteria, Gemma's remains superior (see figure 5

A.7.4 Fine-Tuning Configuration Comparison

To investigate the impact of fine-tuning configurations, we compare the fully fine-tuned LLAMA model with its LoRA fine-tuned counterpart, which also performed strongly on classic metrics (especially with regards to WER and CER which were not as variable for the Lora configuration).

Interestingly, the LoRA version outperforms the standard fine-tuned LLAMA across all criteria in the majority of cases. This notable difference may be attributed to better generalization or increased training stability offered by LoRA fine-tuning. (see figure 7)

A.7.5 Fine-Tuning Architectures Comparison Revisited

Given the superior performance of the LoRA fine-tuned LLAMA over the standard fine-tuned version, we re-evaluate the architecture comparison using the LoRA variant against the fine-tuned Gemma model. In this setting, Gemma once again clearly outperforms its counterpart, winning across all evaluation criteria in the majority of examples. (see figure 6

A.7.6 Advanced vs. Simple Configuration

Prompted by the unexpectedly better performance of the simple Gemma configuration over the in-context version on classic metrics, we evaluate whether the LLM judge aligns with this observation.

Overall, the simple configuration is preferred, although the advantage is not consistent across all criteria. The advanced setting performs particularly well in terms of *fluency* but tends to fall short in other areas, indicating a potential vulnerability to generating artifacts. The judge's preference for the simple configuration could be influenced by the defined tie-breaking policy, which prioritizes *fluency* when scores are otherwise comparable. (see 8)

A.7.7 Unbeaten Models Comparison: Fine-Tuned Gemma vs. Simple Gemma

Since both the simple and fine-tuned Gemma models have each won two out of two previous comparisons (in terms of the *overall_win* criterion), we perform a direct comparison between them.

The fine-tuned Gemma model clearly outperforms the simple version in *fluency*, and also secures wins in *faithfulness* and *completeness*. The simple model, however, performs better on the *no hallucinations* criterion, winning in more examples in that category.

This comparison confirms the added value of fine-tuning, while also highlighting the potential dangers of hallucinations due to overfitting.

A.7.8 Key Conclusions from Judge LLM

- **Gemma consistently outperforms LLAMA** across all evaluated configurations, particularly excelling in reducing hallucinations and preserving faithfulness and completeness.

- **Fine-tuning—both full and LoRA-based—significantly enhances performance**. Notably, the LoRA fine-tuned variant surpasses the fully fine-tuned model, possibly due to improved generalization and training stability. In contrast, full fine-tuning may be more susceptible to overfitting (see Figure 1).

- **Simpler configurations sometimes outperform more complex configurations** , indicating that increased prompt complexity does not always produce a better result. It could be caused be the difficulty in predicting the best prompt tailoring due to LLMs randomness.

- **The fine-tuned Gemma model emerges as the strongest overall**, winning all direct comparisons: against the baseline Gemma, the fully fine-tuned LLAMA, and the LoRA fine-tuned LLAMA. However, it is important to note that pairwise comparison results are not necessarily transitive, that is, performance superiority is not guaranteed across untested pairs.

A.8 LLM Alignment with Evaluation Criteria

we limit our human annotation effort to a small subset, due to many evaluation criteria(4). We choose seven text chunks from *The Vampyre*, specifically keys `["0", "5", "36", "37", "43", "47"]`. These chunks were selected based on their relatively short length to allow the human annotator to effectively assess each chunk in its entirety.

To keep the annotation process feasible, we focus only on comparisons within configuration categories (e.g., *baseline Gemma vs. baseline LLaMA*, *advanced Gemma vs. advanced LLaMA*, and *fine-tuned Gemma vs. fine-tuned LLaMA*).

To quantify the agreement between human judgments and those of the LLM-as-a-Judge, we compute Cohen's Kappa coefficient, which measures inter-annotator consistency. The detailed results are presented in Table 3.

The agreement scores are generally low. The alignment between the human annotator and the LLM ranges from *fair* (0.2–0.4) to *slight* (0.01–0.2), which may be attributed to several factors: the small sample size, unbalanced class distributions, potential ambiguity in Gemini 2.5's interpretation of instructions, or the limited context due to short chunk lengths. In particular, some examples, such as the chunk corresponding to the key `"0"`, which is simply the phrase *"THE VAMPYRE"*, may be too brief to support a meaningful evaluation across all criteria.

Furthermore, to evaluate how well the LLM judge aligns with traditional non-LLM metrics, we compare its decisions with the ROUGE-1, ROUFGE-2, and ROUGE-L scores. We choose the f1 score. Since our judge comparisons are binary, while the ROUGE scores are continuous, we first convert the ROUGE scores into binary variables. We analyze the same subset of comparisons as before—baseline Gemma vs. baseline LLaMA, advanced Gemma vs. advanced LLaMA, and fine-tuned Gemma vs. fine-tuned LLaMA—to maintain consistency with the human-LLM data subset, but this time we consider all keys.

To binarize the ROUGE scores, if the first model's ROUGE score is higher (in the given comparison pair), we assign a value of 1; if the second model's score is higher, we assign 2; otherwise, we assign 0. This approach mimics the approach of the LLM judge, assigns point to better performing model. Using these binary variables, we compute Cohen's kappa coefficient for each pair of (ROUGE metric, LLM criterion), resulting in a total of 15 scores (see Table 4).

The analysis reveals that all pairs fall within the slight agreement range (between 0.2 and 0.4). In particular, the **no halucinations** criterion exhibits the highest Cohen's kappa coefficients across all classical ROUGE metrics.

A.9 Figures

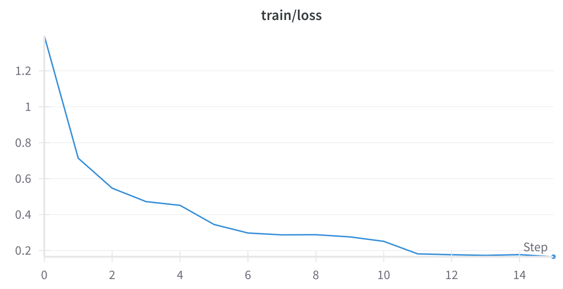

Figure 1: Train loss of the fine-tuning of the Gemma model.

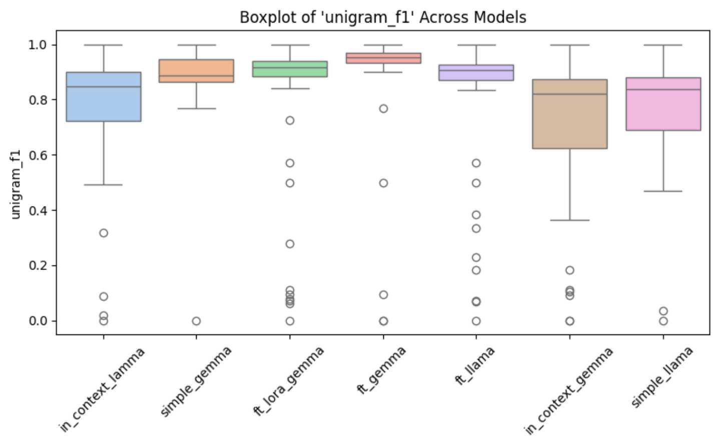

Figure 2: unigram f1 comparison.

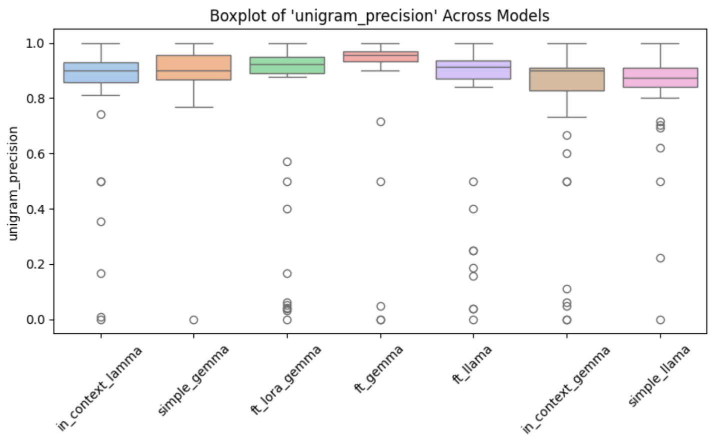

Figure 3: Unigram precision comparison.

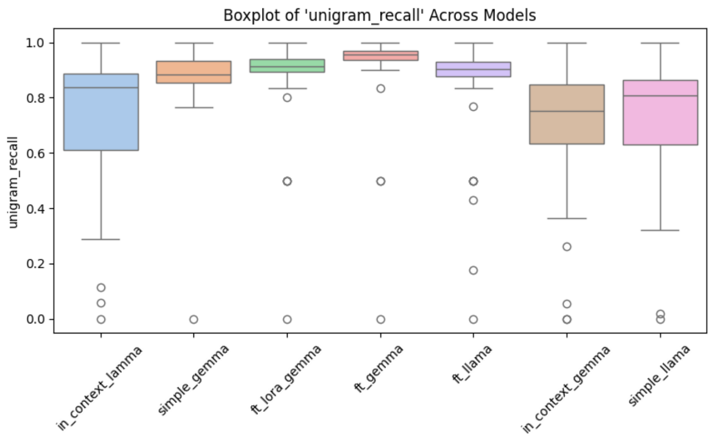

Figure 4: Unigram recall comparison.

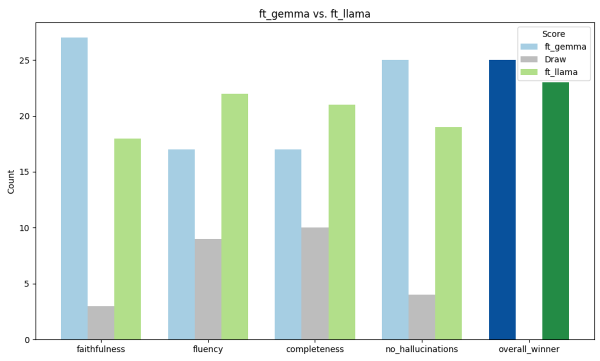

Figure 5: ft gemma vs ft llama

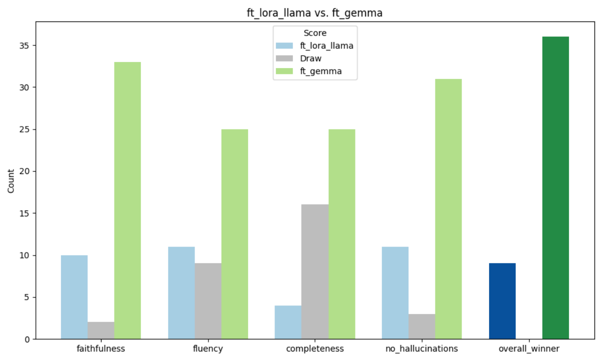

Figure 6: llama ft lora vs ft gemma

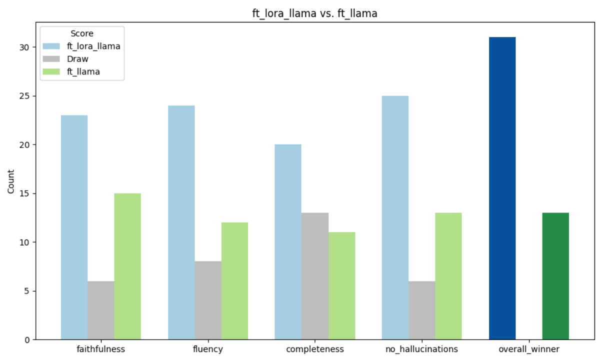

Figure 7: llama ft lora vs llama ft

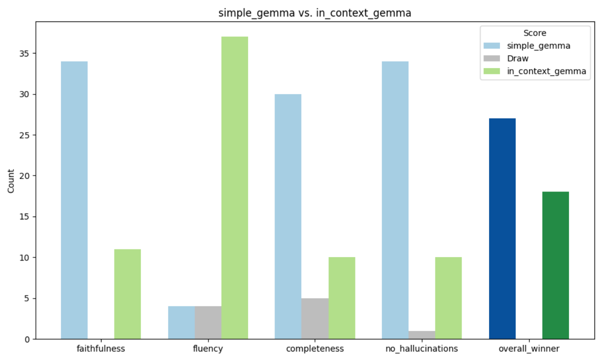

Figure 8: gemma baseline vs gemma advanced

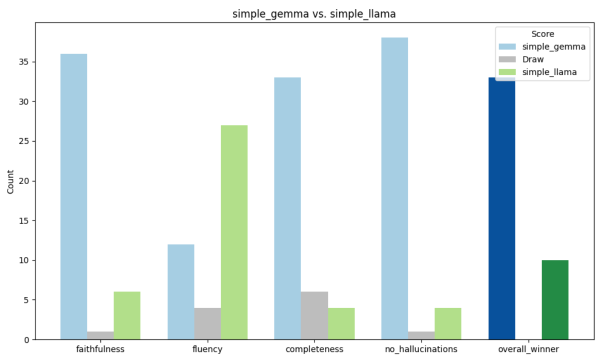

Figure 9: llama baseline vs Gemma baseline

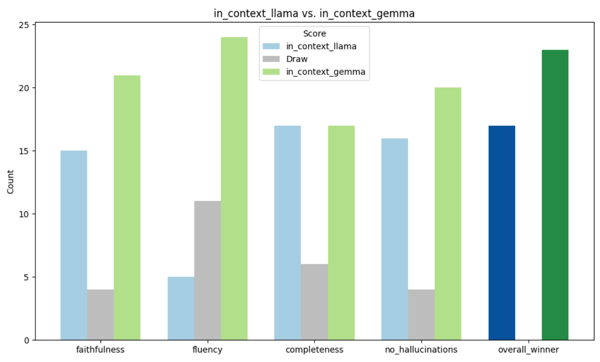

Figure 10: Advanced setting LLama vs advanced setting Gemma

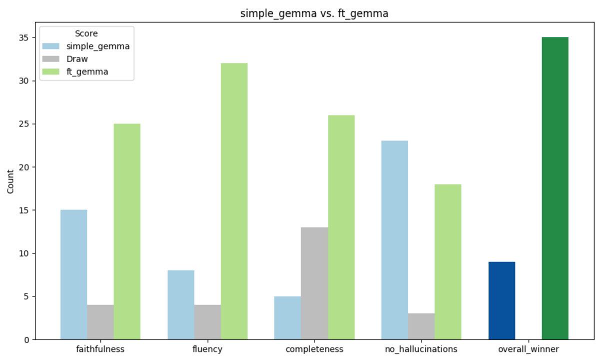

Figure 11: Simple Gemma vs ft Gemma

A.10 Tables

Table 1: Mean scores of evaluation metrics for each model (best values in **bold**)

Model	Precision	Recall	F1	CER	WER
in_context_llama	0.8243	0.7307	0.7662	0.5520	0.6425
simple_gemma	**0.8900**	0.8770	0.8834	0.0721	**0.2137**
ft_lora_llama	0.7764	0.8763	0.7836	1.9676	2.2968
ft_gemma	0.8828	**0.9151**	**0.8848**	50.4003	67.0821
ft_llama	0.7858	0.8409	0.7915	1.2489	1.4359
in_context_gemma	0.7825	0.6922	0.7065	0.9494	1.0414
simple_llama	0.8350	0.7363	0.7733	**0.2942**	0.4139

Table 2: Standard deviation of CER and WER for each model (lowest values in **bold**)

Model	CER Std.	WER Std.
in_context_llama	1.8857	1.7880
simple_gemma	**0.0600**	**0.1982**
ft_lora_llama	5.6642	6.3927
ft_gemma	342.4922	456.3354
ft_llama	4.1316	4.3828
in_context_gemma	2.6974	2.5789
simple_llama	0.2468	0.2706

Evaluation Criterion	Cohen's Kappa
Completeness	0.327
Faithfulness	0.333
Fluency	0.140
No Hallucinations	0.116
Overall Winner	0.045
Overall (average)	**0.274**

Table 3: Cohen's Kappa agreement scores between human annotations and LLM-as-a-Judge across different evaluation criteria.

Criterion	ROUGE-1	ROUGE-2	ROUGE-L
completeness	0.275	0.227	0.271
faithfulness	0.303	0.299	0.343
fluency	0.224	0.209	0.207
no_hallucinations	**0.344**	**0.309**	**0.369**
overall_winner	0.235	0.236	0.292

Table 4: Cohen's Kappa between human judgment criteria and ROUGE metrics.