RAP: A Metric for Balancing Repetition and Performance in Open-Source Large Language Models

Donghao HUANG[1,2], Thanh-Son NGUYEN[3], Fiona LIAUSVIA[3], Zhaoxia WANG[1]
[1]School of Computing and Information Systems, Singapore Management University, Singapore
[2]Research and Development, Mastercard, Singapore
[3]Institute of High Performance Computing (IHPC),
Agency for Science, Technology and Research (A*STAR), Singapore

Abstract

Large Language Models (LLMs) have significantly advanced natural language processing, but content repetition in open-source LLMs remains a critical challenge that adversely affects user experience. The repetition penalty parameter (RPP) aims to mitigate this issue by preventing repeated content generation, but excessive use of RPP can compromise the overall quality. In this paper, we propose Repetition-Aware Performance (RAP), a novel evaluation metric that quantifies and integrates repetition penalty into the assessment of model performance, enabling tuning of RPP. We evaluate our approach using twelve open-source LLMs, ranging from 2 billion to 70 billion parameters, tested on question answering and machine translation tasks across three datasets with varying prompting techniques. Experimental results show that RAP effectively tunes RPP, helping to identify a trade-off value that significantly reduces repetition while minimizing performance loss. The code and the dataset of generated text can be accessed at `https://github.com/inflaton/rap`.

1 Introduction

The rapid advancement of Large Language Models (LLMs) has transformed natural language processing, enabling remarkable text generation and comprehension capabilities. However, open-source LLMs often generate repetitive content, leading to extensive empty lines or recurring sentences, which undermines text quality and fluency. This is particularly problematic in tasks requiring coherent and contextually relevant responses, such as conversational chatbots. To address this, the *repetition penalty parameter* (RPP) is employed during the sampling process to reduce redundant outputs by penalizing previously seen tokens (Keskar et al., 2019). While effective, excessive application of RPP can result in incomplete or fragmented responses, ultimately diminishing user satisfaction.

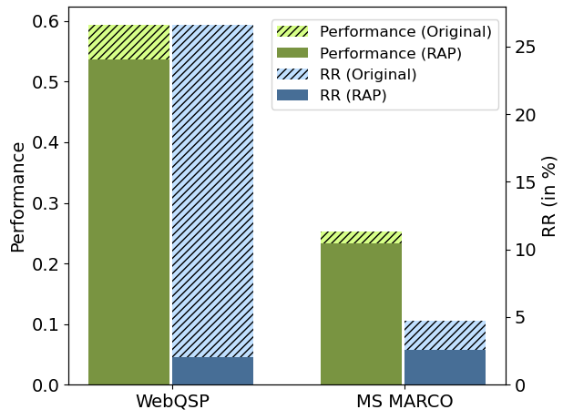


Figure 1: Best performance (F1 for WebQSP, BERT-F1 for MS MARCO) when tuning RPP using original evaluation metrics (Original) and our proposed repetition-aware performance metric (RAP). RAP helps find RPP values that minimally reduce performance while significantly lowering repetition ratio (RR). Result is from Llama-3-8B (RAG-Generic Prompt).

This paper explores the relationship between RPP and its effect on generated text. While parameters like *temperature* and *top-k sampling* are optimized using methods like random or grid search, these fail for tuning RPP since traditional metrics, such as precision and BLEU scores (Papineni et al., 2002), overlook repetition. To address this, we propose a metric called **R**epetition-**A**ware **P**erformance (RAP), which quantifies repetition and penalizes it in the model's performance evaluation, allowing for the tuning of RPP. Figure 1 illustrates tuning RPP for Llama-3-8B using the original evaluation metrics (Original) and our proposed metric. For the WebQSP dataset (Yih et al., 2016), repetition ratio (RR) is reduced by 93.1% but the performance only drops by 3.7%. For the MS MARCO dataset (Huang et al., 2024), RR drops by 73.9% with only 3.7% of performance drop. Tuning RPP using RAP can significantly reduce the RR with minimal performance loss. More examples can be found in Appendix (A.4).

To compute RAP, we introduce an efficient algorithm called ReDA– Repetition Detection Algorithm–designed to identify all repeating patterns, including non-word character (NWC) and text repetitions, along with their corresponding repetition lengths. Additionally, we define the repetition ratio (RR), which measures the proportion of repeated content relative to the total text length. RR is applied to penalize the model's performance using a cubic penalty formula, as detailed in Section 3. Figure 2 illustrates computing RAP with a running example. Although the accuracy is 100%, the content contains many repeated elements, resulting in a low RAP score of 24.63%.

We conduct experiments on three datasets encompassing question answering (including knowledge base question answering–KBQA– and open-ended QA) and machine translation tasks. We evaluate twelve open-source LLMs ranging from 2 to 70 billion parameters. We also examine the effect of repetition suppression for different prompting strategies including Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) and non-RAG, combining with generic prompt and chat template.

Our study reveals that increasing RPP typically reduces repetition but often results in performance degradation, highlighting the importance of tuning RPP. The effects of RPP differ across various models, datasets, and prompting techniques. Additionally, we find that using chat templates is effective in reducing repetition for QA task. Our experimental results demonstrate that RAP can effectively tune RPP, helping to identify a trade-off value that significantly reduces repetition while minimizing performance loss. These findings offer insights into optimizing RPP for performance and repetition suppression in open-source LLMs.

This paper presents three key contributions. First, we propose RAP, a novel evaluation metric based on the repetition ratio that quantifies and integrates repetition issues into model performance, enabling the tuning of hyperparameters such as RPP. We also develop an efficient Repetition Detection Algorithm (ReDA) algorithm for detecting both NWC and text repetition, addressing the key challenge in open-source LLMs. Second, we conduct comprehensive experiments on twelve open-source LLMs ranging from 2 to 70 billion parameters, across two tasks–QA and MT–using three datasets and multiple prompting techniques, including RAG and non-RAG methods, with the use of Generic Prompt and Chat Template. These experiments provide valuable insights into reducing repetition while maintaining performance and demonstrate the effectiveness of tuning RPP using RAP. Third, we publish all source code and data generated during these experiments at GitHub[1], which includes a significant amount of text produced by open-source LLMs. This dataset is a valuable resource for future research in developing advanced repetition detection methods, assessing LLM capabilities in entity extraction, and evaluating performance in question answering and machine translation tasks.

2 Related Work

Text repetition in generative models. The repetition problem has been a long-standing concern in generative models (Holtzman et al., 2019; Welleck et al., 2020). Fu et al. (2020) show that excessive word repetition occurs when transformer models predict the same subsequent word with high probability. Factors contributing to this issue include self-reinforcing behavior (Xu et al., 2022) and a concept known as *Distribution Collapse*, which results in decreased output diversity over time (Emrullah Ildiz et al., 2024). Various strategies introduce variation in the generated text by considering a subset of the most probable tokens at each step, namely: greedy (Klein et al., 2017), top-k (Fan et al., 2018), temperature (Ficler and Goldberg, 2017; Caccia et al., 2020), and nucleus sampling (Holtzman et al., 2019). Additionally, managing training data and applying techniques like length penalties and rebalanced encoding have been suggested to alleviate this issue (Wu et al., 2016; Klein et al., 2017; Fu et al., 2020). Moreover, models like CTRL (Keskar et al., 2019) use control codes, including the repetition penalty parameter (RPP), to guide generation and reduce unwanted repetitions; however, determining the optimal RPP value remains challenging, as excessive settings can degrade output quality. This paper explores the impact of RPP on repetition and task performance, introducing RAP as a metric to optimize RPP for better model outcomes.

LLMs for Question Answering. Advancements in LLMs have greatly improved QA tasks performance. Radford et al. (2019) show the transfer learning ability of GPT-2 and GPT-3 (Brown et al., 2020), which adapts to new QA tasks with minimal examples. RAG is widely adopted for QA tasks (Lazaridou et al., 2022; Muludi et al., 2024),

[1]`https://github.com/inflaton/rap`

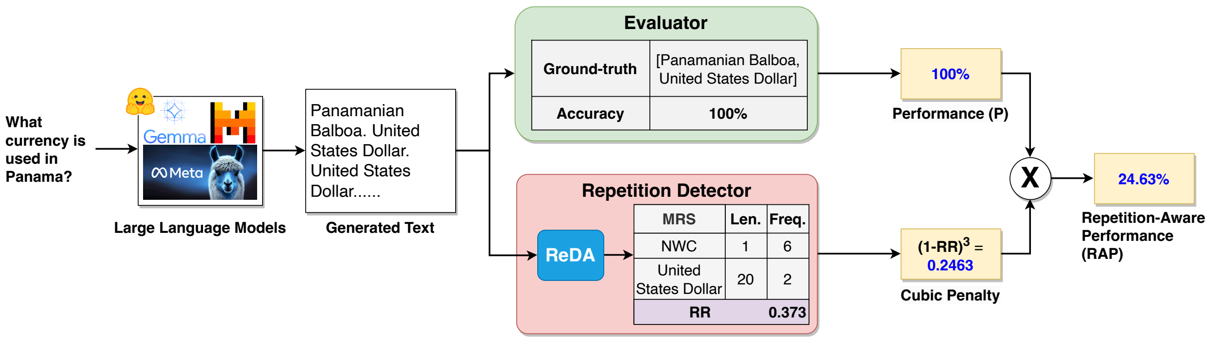


Figure 2: Overall Methodology: RAP integrates the repetition penalty in the evaluation score. In the example, the generated text contains non-word characters (NWC) repetition and text repetition. After penalization, the accuracy is reduced from 100% to 24.63%. ReDA stands for Repetition Detection Algorithm.

providing information that aids in generating answers (Alawwad et al., 2024). The synergy between LLMs and knowledge graphs (Pan et al., 2024) has led to increased adoption in LLMs for KBQA tasks (Zhang et al., 2022). Another method uses LLMs to convert questions into logical forms, which are executed to get answers (Chen et al., 2021; Li et al., 2023; Xiong et al., 2024; Wang and Qin, 2024). Others augment LLMs with relevant facts retrieved from a KG (Baek et al., 2023; Jiang et al., 2023). In this paper, we use QA tasks to study the impact of RPP on repetition and the performance of open-source LLMs across two tasks, using both RAG and non-RAG strategies.

LLMs for Machine Translation (MT). Large Language Models (LLMs) have made significant strides in machine translation (Zhang et al., 2023; Moslem et al., 2023). Models such as BERT (Devlin et al., 2018), GPT-3 (Brown et al., 2020), and T5 (Raffel et al., 2020) have demonstrated exceptional performance in zero-shot and few-shot translation scenarios (Liu, 2020). A recent comprehensive study evaluated LLMs across zero-shot, few-shot, and fine-tuned paradigms, highlighting trade-offs between model size, translation accuracy, and processing speed (Huang and Wang, 2025). In this paper, we build on their fine-tuned open-source LLMs and curated dataset to investigate the impact of RPP on repetition and overall performance.

3 The Proposed Method

The core concept of RAP is to incorporate a penalty for the model's performance based on the degree of repetition in the generated response. To achieve this, we propose an efficient algorithm, ReDA, for detecting and quantifying repetition in text. We introduce the notion of the *repetition ratio* (RR), which measures the proportion of repeated content. Additionally, we define *repetition-aware performance* (RAP), a metric that integrates repetition penalties into standard performance evaluations, enabling tuning of the repetition penalty parameter. Given a text sequence generated by an LLM, we first detect *non-word characters* (NWC) repetition, that includes white spaces and non-alphanumeric characters such as punctuation and symbols. A sequence has NWC repetition if it contains at least k consecutive identical NWC. After carefully checking the outputs of various LLMs, we choose $k = 5$ to ignore cases where NWC are used for formatting the output. We then remove these repeated NWC and detect *text repetition*.

3.1 Definitions

Repetition in LLM-generated content can occur not only in sequences of words separated by spaces but also within a single, super-long "word" that contains no spaces. Both of these issues are equally important. Therefore, we extend the definition of text repetition from Fu et al. (2020) to cover not only the repetition of word sequences but also the repetition of characters within no-space content. We denote an LLM's generated sequence as $S = [w_1, w_2, \cdots, w_{|S|}]$ where w_i is the i^{th} character in S and $|S|$ is the length of the sequence in terms of characters. We denote $S_{p:q} = [w_p, w_{p+1}, \cdots, w_q]$, where $1 \leq p < q \leq |S|$, as a continuous subsequence of S.

Definition 1 (Equal Subsequences) *Subsequences $S_{a:b}$ and $S_{c:d}$ are said to be equal if $b - a = d - c$ and $w_{a+i} = w_{c+i}$, for $\forall i \in [0, b - a]$.*

Definition 2 (Repeated Subsequence) *Given a sequence $S = [w_1, w_2, \cdots, w_{|S|}]$, a subsequence*

$S_{p:q}$, $1 \leq p < q \leq \frac{|S|+p-1}{2}$ *is a repeated subsequence if it equals to its consecutive subsequence,* $S_{q+1:2q-p+1}$.

Definition 3 (Maximal Repeated Subsequence) *A subsequence $S_{p:q}$ is said to be a maximal repeated subsequence if $\nexists c, d$ such that $S_{c:d}$ is a repeated subsequence, where $c \leq p < q \leq d$ and $d - c > q - p$.*

A *maximal repeated subsequence* (MRS) in a sequence refers to the longest subsequence that occurs repeatedly without interruption, representing the largest repeating unit. There may be multiple MRSes within a text, each representing a distinct repeated pattern. The repetition length of an MRS is defined as the length of the repeated content, excluding the first occurrence of the pattern. The repetition length of the entire sequence is calculated as the sum of the repetition lengths of all the MRSes within the sequence.

Definition 4 (Text Repetition Detection Problem) *Given a sequence S, identify all maximal repeated subsequences (MRSes) within S and determine their corresponding repetition lengths.*

Definition 5 (Repetition Ratio – RR) *The repetition ratio (RR) of a sequence S is defined as the proportion of repeated content relative to the length of the sequence:*

$$RR(S) = \frac{R(S)}{|S|} \tag{1}$$

where $R(S)$ is the repetition length of the entire sequence S.

In Figure 2, the repeated content includes NWC repetition (repeated dots), and text repetition ("United States Dollar"). The RR computed using Equation 1 is 0.373, i.e., about 37% of the text is repeated content.

Repetition Penalty Parameter (RPP). RPP is applied during token sampling to address repetition in generated text (Keskar et al., 2019). Specifically, the probability of predicting the i^{th} token is computed as:

$$p_i = \frac{\exp(x_i/(T \cdot I(i \in g)))}{\sum_j \exp(x_j/T \cdot I(j \in g)))} \tag{2}$$

where $I(c) = \theta$ if c is True else 1, T is the temperature value, g is the list of generated tokens, and θ is the RPP (Keskar et al., 2019). RPP adjusts the probabilities of previously generated tokens, penalizing them to reduce their likelihood of being selected again. The extent of this adjustment is governed by the RPP value: higher RPP values impose stronger penalties, effectively suppressing repetitive outputs. However, this comes with a trade-off, as overly penalizing tokens can potentially impact the coherence and fluency of the generated text. We now present the repetition detection algorithm and RAP (repetition-aware performance), which facilitate tuning RPP to achieve an optimal balance.

3.2 Repetition Detection Algorithm (ReDA)

ReDA adopts *regular expressions* to detect both NWC and text repetitions. ReDA can detect repetitions within single no-space content, an issue often overlooked (Fu et al., 2020). The regular expression for detecting NWC repetition is as follows:

$$[\backslash s \backslash W]\{5,\} \tag{3}$$

which matches any sequence of 5 or more consecutive characters where each character is either a whitespace or a NWC. The regular expression for identifying text repetition is:

$$(?P<r>.\{5\}.*?)(?:[\backslash s\backslash W]*(?P=r)+ \tag{4}$$

which matches a sequence of at least 5 alphanumeric (captured in the group r), followed by one or more repetitions of the same sequence, potentially separated by whitespace or non-word characters. ReDA first detects and records all NWC repetitions, removes them, and then identifies text repetitions. For text repetition, it identifies all MRSes and their frequencies. Finally, ReDA returns the repetition lengths of NWC and text repetition. ReDA is detailed in Algorithm 1.

3.3 Repetition-Aware Performance (RAP)

After measuring the repeated length of each LLM response, we compute the RR for the entire dataset based on Equation 1:

$$\text{RR}_D = \frac{\sum_{S \in D} R(S)}{\sum_{S \in D} |S|} \tag{5}$$

where RR_D is the repetition ratio for the entire dataset D, and each $S \in D$ is a sequence (LLM response) in D. The RAP score of the entire dataset is computed as follows:

$$RAP = P \times F(RR) \tag{6}$$

Algorithm 1 Repetition Detection (ReDA)

```
1:  Input: $G_{text}$                                          ▷ Input text
2:  Output: $RL_{NWC}$, $RL_{text}$, $RL_{total}$   ▷ Repetition length
    of NWC, text, and total, respectively
3:  $P_{NWC}$ = re.compile(r"([\s\W]{5,})")           ▷ Regex
    pattern for detecting NWC repetition
4:  $P_{text}$ = re.compile(r"(?P<r>.{5}.*?)
5:                   (?:[\s\W]*(?P=r)+")               ▷ Regex
    pattern for detecting text repetition
6:                    ▷ STEP 1. Detecting NWC repetition
7:  $G'_{text}$ = $P_{NWC}$.sub("\t ", $G_{text}$)    ▷ Remove NWC
    repetition
8:  $RL_{NWC}$ = len($G_{text}$) - len($G'_{text}$)   ▷ Compute NWC
    repetition
9:                   ▷ STEP 2. Detecting text repetition
10: $RL_{text}$ = 0
11: matches = $P_{text}$.finditer($G'_{text}$)
12: for each match in matches do
13:     (start, end) = match.span()
14:     $RL_{text}$ += end - start - len(match.group(1))   ▷
    Aggregate text repetition lengths
15: end for
16: $RL_{total}$ = $RL_{NWC}$ + $RL_{text}$
17: return $RL_{NWC}$, $RL_{text}$, $RL_{total}$
```

where P can be any evaluation metric such as F1, BERT-F1, or COMET, and $F(RR)$ is the penalty function that calculates the penalty volume based on RR. Unless otherwise specified, the results in this paper are generated using cubic penalty function, i.e., $F(RR) = (1 - RR)^3$. If no repetition is found in the generated text, $RAP = P$; otherwise, $RAP < P$, meaning the performance is penalized for having repetition. RAP enables tuning RPP to achieve the balance between model performance and repetition, allowing for high performance while minimizing repetitive content.

4 Experiments

4.1 Experimental settings

Datasets. We conduct the experiments using three datasets including WebQSP (Yih et al., 2016) and MS MARCO (Bajaj et al., 2016; Huang et al., 2024) for question answering (QA), and MAC (Huang and Wang, 2025) for Chinese-English machine translation (MT). The datasets contain 1,008, 500, and 1,133 testing questions, respectively. To perform RAG on WebQSP, we crawled relevant articles from Wikipedia, and retrieve 8 most relevant document chunks when answering each question for RAG prompting. We also adopt Gemini-1.0-Pro to evaluate for the knowledge base question answering task (KBQA) where the ground-truth answers contains lists of entities instead of natural language text. More details of the dataset and LLM-based KBQA evaluation are presented in the Appendix (A.1).

Open-Source LLMs. This study evaluates a diverse set of open-source large language models (LLMs) for question answering (QA) and machine translation (MT) tasks. For QA, we assess nine models: Llama-2 (7B, 13B, 70B), Llama-3 (8B, 70B), Gemma-1.1 (2B, 7B), Phi-3-mini (3.8B), and Mistral-7B. For MT, we evaluate three models: Phi-3.5-mini (3.8B), Mistral-7B, and Llama-3.1-70B.

Table 1 provides an overview of these models, including their respective companies, model names, Hugging Face identifiers, and the tasks they were evaluated on. This selection spans a broad range of model sizes and architectures, representing the state of the art in open-source LLM development. These models are widely used in commercial applications under their respective licenses.

The diversity of this selection enables a robust comparison across model families and sizes, offering insights into the relative strengths of different open-source LLMs in QA and MT tasks.

Automating QA and MT Tasks with LLMs. We developed a Python script specifically designed to evaluate the performance of LLMs on QA and MT tasks. The script leveraged the open-source HuggingFace Transformers library (Wolf et al., 2020) and was executed across all models and datasets. For QA tasks, an RPP range of 1.0 to 1.3 was applied, while for MT tasks, an RPP range of 1.0 to 1.1 was used, both with increments of 0.02. To ensure consistency and reproducibility across experiments, the temperature was set to 0 and `top_p` was fixed at 0.95.

Prompt Templates: To investigate LLMs' behaviors under different prompting strategies, we implemented three QA prompt templates: (1) *RAG - Generic Prompt* (RAG-GP): a uniform basic template for all models; (2) *RAG - Chat Template* (RAG-CT): LangChain's default template tailored to each LLM's chat format, maintaining the same core content; and (3) *Non-RAG*: simulates a single-turn conversation in the LLM's chat format. For MT, we used two templates: (1) *MT - Generic Prompt* (MT-GP): a basic template applied uniformly, and (2) *MT - Chat Template* (MT-CT): a template designed for each LLM's chat format. Prompt templates are shown in the Appendix (A.6).

Evaluation Metrics: For WebQSP, we use *precision*, *recall*, and *F1* scores. For MS MARCO, we employ BERTScore (Zhang et al., 2019), using deberta-xlarge-mnli model to ensure optimal

Table 1: Overview of Large Language Models and Their Specifications by Task

Company	Model Name	Hugging Face Model ID	Task
Meta	Llama-2-7B	meta-llama/Llama-2-7b-chat	QA
	Llama-2-13B	meta-llama/Llama-2-13b-chat	QA
	Llama-2-70B	meta-llama/Llama-2-70b-chat	QA
	Llama-3-8B	meta-llama/Meta-Llama-3-8B-Instruct	QA
	Llama-3-70B	meta-llama/Meta-Llama-3-70B-Instruct	QA
	Llama-3.1-70B	shenzhi-wang/Llama3.1-70B-Chinese-Chat	MT
Microsoft	Phi-3-mini	microsoft/Phi-3-mini-128k-instruct	QA
	Phi-3.5-mini	microsoft/Phi-3.5-mini-instruct	MT
Google	Gemma-1.1-2B	google/gemma-1.1-2b-it	QA
	Gemma-1.1-7B	google/gemma-1.1-7b-it	QA
Mistral AI	Mistral-7B-v0.2	mistralai/Mistral-7B-Instruct-v0.2	QA
	Mistral-7B-v0.3	shenzhi-wang/Mistral-7B-v0.3-Chinese-Chat	MT

performance, as it provides the highest correlation with human evaluations. Results are reported as the average F1 score across the entire dataset, referred to as *BERT-F1*. For MAC, we employ the Crosslingual Optimized Metric for Evaluation of Translation (*COMET*), a neural-based evaluation metric that leverages contextual embeddings from pretrained language models to assess translation quality (Rei et al., 2022). COMET has demonstrated strong correlation with human judgments, making it a reliable metric for translation evaluation. For repetition evaluation, we report the repetition ratio (Equation 5). We also report the RAP scores, applied on the corresponding evaluation metrics (Section 3.3).

4.2 Impact of RPP on Repetition and Performance

Figures 3 illustrates the impact of RPP on the repetition and performance of different open-source LLMs on the three datasets, using different prompting strategies. The results show that increasing RPP generally reduces repetition across strategies and datasets but often leads to a decline in performance. The trending and severity vary by model; for instance, while Phi-3-mini's performance on MS MARCO-RAG-GP dropped significantly with RPP above 1.14, Llama-3-8B showed improved performance with higher RPP values under the same conditions. Therefore, selecting an optimal RPP value is crucial to balance repetition reduction with minimal impact on performance.

The datasets also show different behaviors. WebQSP generally exhibits higher repetition levels (e.g., at RPP = 1.0) than MS MARCO, especially with the RAG-GP strategy. This indicates that the nature of the dataset and task also influence repetition tendencies. In MAC dataset, despite a smaller RPP range tested, repetition still decreases as RPP increases. For both QA datasets, the Chat Template strategy consistently reduces repetition, showing lower RR values than the Generic Prompt approach. This suggests that the Chat Template effectively minimizes repetition in QA tasks. However, it surprisingly does not offer the same benefit for MT task, where increasing RPP effectively reduces repetition.

Table 2: Comparison of model performance across different datasets, LLMs, and prompting strategies.

Dataset	LLM	RAG-GP	RAG-CT	Non-RAG
WebQSP	Gemma-1.1-2B	0.397	0.361	0.175
	Phi-3-mini	0.527	0.575	0.595
	Gemma-1.1-7B	0.109	0.302	0.291
	Mistral-7B-v0.2	**0.609**	**0.604**	0.648
	Llama-2-7B	0.589	0.590	0.548
	Llama-3-8B	0.572	0.558	0.668
	Llama-2-13B	0.595	0.588	0.627
	Llama-2-70B	0.577	0.596	0.657
	Llama-3-70B	0.517	0.537	**0.710**
MS MARCO	Gemma-1.1-2B	0.272	**0.277**	**0.250**
	Phi-3-mini	0.263	0.261	0.246
	Gemma-1.1-7B	0.149	0.276	0.249
	Mistral-7B-v0.2	0.256	0.256	0.239
	Llama-2-7B	0.266	0.257	0.243
	Llama-3-8B	0.252	0.265	0.239
	Llama-2-13B	0.265	0.258	0.246
	Llama-2-70B	0.270	0.258	0.248
	Llama-3-70B	**0.275**	0.264	0.238
	LLM	**MT-GP**	**MT-CT**	
MAC	Phi-3.5-mini	0.707	0.702	
	Mistral-7B-v0.3	0.726	0.728	
	Llama3.1-70B	**0.743**	**0.750**	

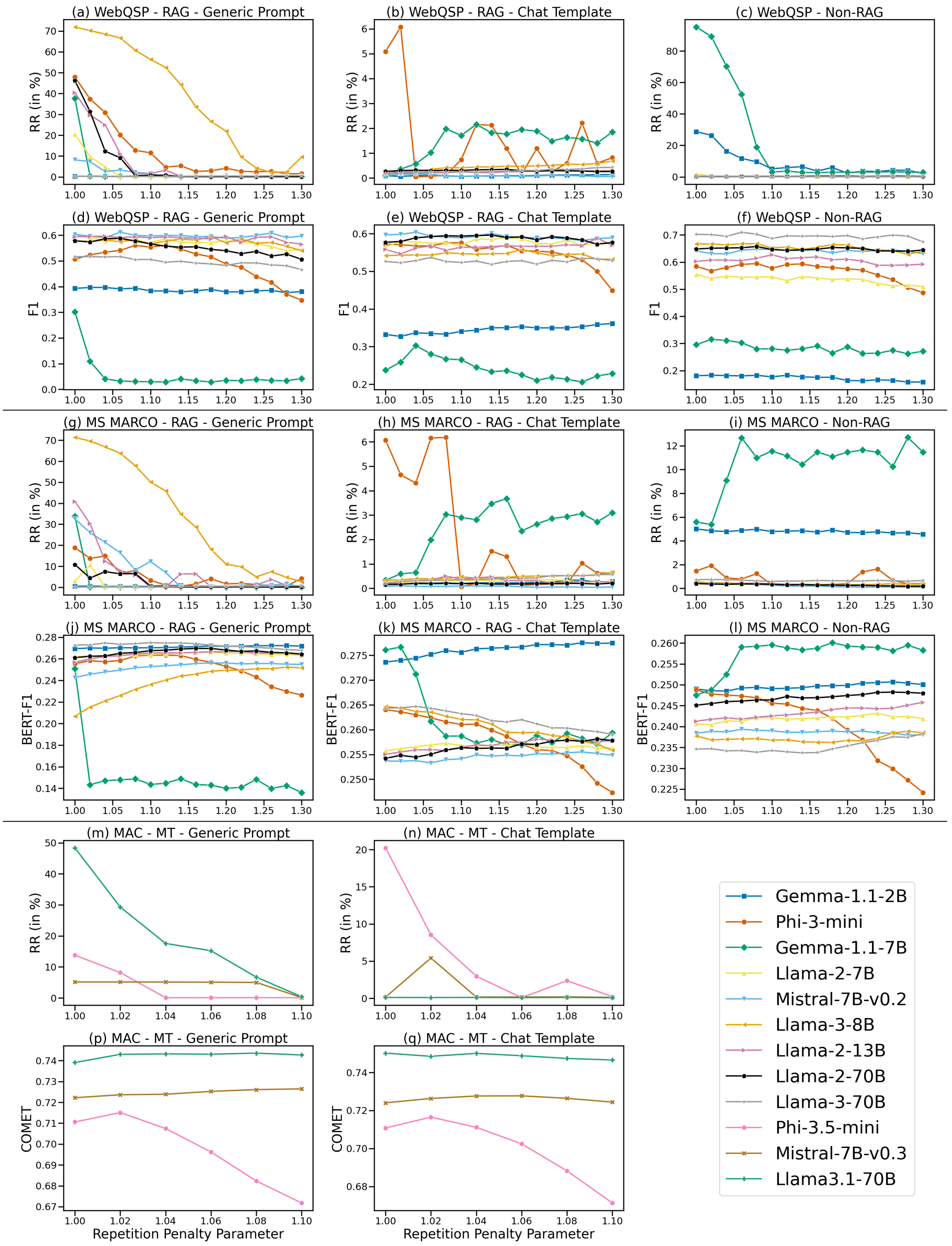


Figure 3: Illustrating the impact of RPP on RR and performance across models, datasets, and prompting strategies.

4.3 Optimizing the Trade-Off Between Performance and Repetition

This section demonstrates the use of RAP in tuning RPP by comparing model performance and repetition when using conventional metrics (Original Performance) versus RAP. Figure 4 shows the relationship between RPP, performance, and repetition for Phi-3-mini on WebQSP and MS MARCO, and Phi-3.5-mini on MAC. Blue lines represent original

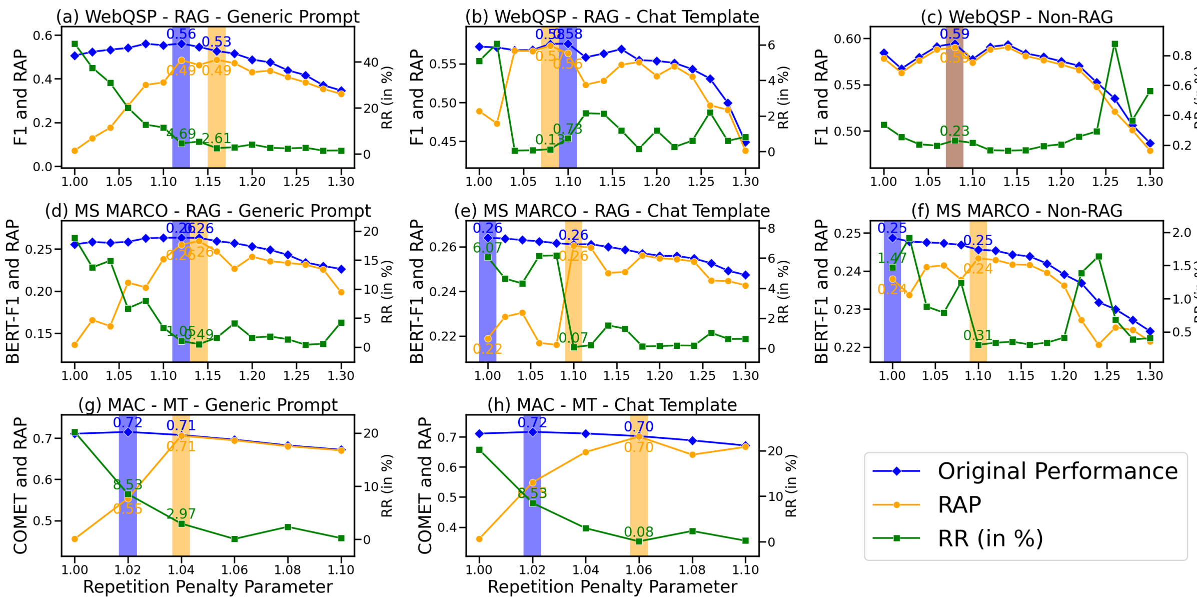


Figure 4: Illustrating original performance, repetition-aware performance (RAP), and repetition ratio (RR) while varying RPP. Results are shown for Phi-3-mini in QA (Subplots (a) to (f)) and Phi-3.5-mini in machine translation (Subplots (g) and (h)). The blue shaded region highlights the RPP for optimal original performance, while the orange region indicates the RPP for the best RAP; overlapping regions are marked in brown to show alignment.

performance (F1 for WebQSP, BERT-F1 for MS MARCO, and COMET for MAC), orange lines show RAP scores, and green lines (right y-axis) indicate RR percentages. The blue shaded area marks the RPP with the best performance based on conventional metrics, the orange marks the best RAP, and brown indicates overlap between the two.

In most cases (except Figure 4c), the optimal RPP differs when tuning for original performance versus RAP. For instance, in Figure 4e, tuning with BERT-F1 gives the best RPP at 1.0 with a performance of 0.264, but the repetition ratio is high at 6.07%. With RAP, the best RPP shifts to 1.1, where performance drops by 1.1% (drop 0.003), but the repetition ratio significantly decreases by 98.8% (drop 6%). This demonstrates that tuning RPP with RAP helps balance performance with reduced repetition. Another example with different visualization is illustrated in Figure 1. These results highlight the importance of tuning RPP to balance performance and repetition reduction, which can be achieved using RAP. The optimal RPP varies by task, dataset, and prompting strategy.

4.4 Comparing the RAP-Tuned Performance of All Models

Table 2 shows the performance of the open-source LLMs after tuning RPP using RAP. On **WebQSP**, Mistral-7B leads for both RAG-GP and RAG-CT, scoring 0.609 and 0.604, respectively. However, Llama-3-70B outperforms all models in the Non-RAG prompt with a high F1 score of 0.710. For **MS MARCO**, Gemma-1.1-2B excels in both RAG-CT and Non-RAG, with scores of 0.277 and 0.250, while Llama-3-70B takes the top spot in RAG-GP with 0.275. In the MT task (**MAC**), Llama3.1-70B dominates both the generic prompt and chat template categories, scoring 0.743 and 0.750. These results show that larger models like Llama-3-70B and Mistral-7B consistently achieve top performance across datasets and prompting strategies. However, mid-sized models like Gemma-1.1-2B perform competitively on specific tasks, such as MS MARCO, highlighting that model choice should be task-dependent and not solely based on model size.

4.5 How Severe can Repetition be?

In our experiments, the length of text repeated can account for up to 97.85% of the overall text output, while the proportion of responses with repetition across the entire dataset can reach 41.27%. Figure 5 shows the top-3 highest and lowest RR for MS MARCO. Colored squares above each point indicate the percentage of repeated responses, with blue, yellow, and red squares marking values below 10%, between 10-20%, and above 20%, respectively. Notably, while Gemma-1-1.7B (RAG-GP) has only 3.8% of responses with repetition, the

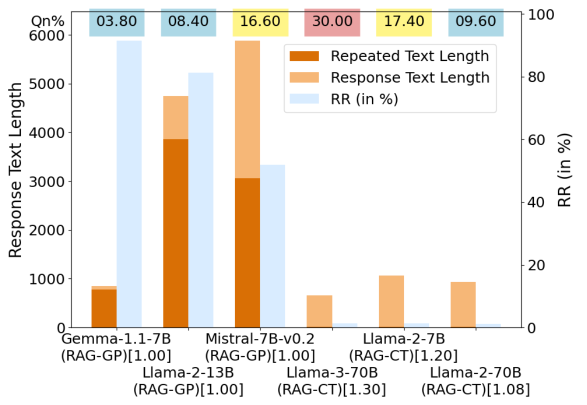


Figure 5: Severity of repetition. Colored squares above indicate the percentage of repetitive responses. X-label contains model, prompting method, and tuned RPP.

repeated text case is very severe as it makes up 91.53% of the overall answer length. In contrast, 30% of Llama-3-70B (RAG-CT)'s responses contains repetition, but the repeated text only constitutes 1.29% of the overall length. The length of repeated content can also be substantial. For instance, Llama-2-13B generated repeated content that averaged around 4,000 characters.

4.6 Ablation Study: the Effects of Different Penalty Functions

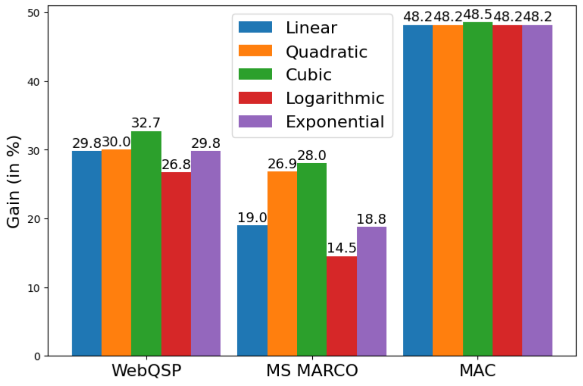


Figure 6: Gains across different penalty functions. *Cubic* penalty obtains the best gains across three datasets.

We now present an ablation study to evaluate the impact of different formulas for the penalty function $F(RR)$ on RAP (Equation 6). We consider five options including *Linear*, *Quadratic*, *Cubic*, *Logarithmic*, and *Exponential* where the penalty are $1-RR$, $(1-RR)^2$, $(1-RR)^3$, $\log_2(2-RR)$, and e^{-RR}, respectively. We compare the *Gain* achieved with each repetition penalty formula. For a model, *Gain* is defined as the difference between the reduction in repetition and the performance drop when tuning RPP using original performance versus RAP. It reflects how much repetition decreases (i.e., gain) relative to the performance loss when tuning RPP with RAP. The higher *Gain*, the better. Figure 6 shows the *Gain* scores for each formula across the three datasets. The *Gain* for a dataset is calculated by averaging the *Gain* values from all models and prompting strategies. The results show that the *Cubic* penalty consistently achieves the highest gains across all datasets (32.7% for WebQSP, 28.0% for MS MARCO, and 48.5% for MAC), offering the best balance between reducing repetition and maintaining performance. These findings recommend the *Cubic* penalty for the RAP metric, as it performs well across diverse datasets and tasks.

5 Conclusion

This research tackles the challenge of reducing repeated content generation while preserving overall performance in open-source LLMs. We propose the repetition-aware performance metric, RAP, which quantifies and integrates repetition penalty into model evaluation. Conventional metrics often overlook repetition, making it difficult to tune the repetition penalty parameter (RPP). In contrast, RAP enables effective tuning of RPP and can be easily applied to other repetition methods as well.

We evaluated RAP on twelve open-source LLMs using three datasets across QA and MT tasks, with different prompting strategies. Results show that higher RPP values generally reduce repetition but can significantly impact performance. This phenomenon emphasizes the need of tuning RPP. The experiments also demonstrate the capability of using RAP for identifying the RPP value that efficiently reduce repetition while minimizing performance loss. We also highlight the severity of the repetition issues in the generated content, calling for greater attention from the research community on this matter. Our experiments offer valuable insights into managing repetition in open-source LLMs, including a recommendation to use Chat Templates for RAG prompting. We will release the source code and all experimental data, offering a large dataset of LLM-generated content for future research on repetition issues and LLM capabilities in QA and MT tasks.

6 Limitations

This study has several limitations. First, while we conducted experiments with twelve open-source LLMs, our findings may not be generalizable to other open-source models or newer architectures due to variations in design and training data. This limitation suggests that further research could explore a wider array of models to validate the applicability of our findings.

Second, our current repetition detection method focuses solely on non-word character and text repetition. This may overlook other relevant forms of repetition, such as word salad repetition, which could further impact the quality of generated content. Addressing these additional types of repetition in future work could lead to a more comprehensive understanding of repetition issues in LLMs.

Third, this research examines two specific tasks: question answering and machine translation. While these tasks are significant, extending the study to include other applications, such as text summarization, could provide deeper insights into LLM behavior concerning text repetition across various NLP tasks. This expansion would help in understanding how different tasks may influence repetition patterns and the overall performance of LLMs.

Finally, this work concentrates on the repetition penalty parameter (RPP). However, other methods, such as sampling techniques, temperature adjustments, and context size modifications, can also be used to mitigate repetition issues in generative models. While RAP can be easily applied alongside these methods, further research evaluating RAP across different repetition control methods would enhance our understanding of how to manage repetition issues in LLMs more effectively.

References

Hessa Abdulrahman Alawwad, Areej Alhothali, Usman Naseem, Ali Alkhathlan, and Amani Jamal. 2024. Enhancing textbook question answering task with large language models and retrieval augmented generation. *arXiv preprint arXiv:2402.05128*.

Jinheon Baek, Alham Fikri Aji, and Amir Saffari. 2023. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. *Proceedings of the First Workshop on Matching From Unstructured and Structured Data (MATCHING 2023)*.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. 2020. Language gans falling short. In *International Conference on Learning Representations*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 4171–4186.

M Emrullah Ildiz, Yixiao Huang, Yingcong Li, Ankit Singh Rawat, and Samet Oymak. 2024. From self-attention to markov models: Unveiling the dynamics of generative transformers. *arXiv e-prints*, pages arXiv–2402.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104, Copenhagen, Denmark. Association for Computational Linguistics.

Zihao Fu, Wai Lam, Anthony Man-Cho So, and Bei Shi. 2020. A theoretical analysis of the repetition problem in text generation. In *AAAI Conference on Artificial Intelligence*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *ArXiv*, abs/1904.09751.

Donghao Huang, Zhenda Hu, and Zhaoxia Wang. 2024. Performance analysis of llama 2 among other llms. In *2024 IEEE Conference on Artificial Intelligence (CAI)*, pages 1081–1085.

Donghao Huang and Zhaoxia Wang. 2025. Optimizing chinese-to-english translation using large language models. In *2025 IEEE Symposium Series on Computational Intelligence (forthcoming)*. IEEE.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *ArXiv*, abs/1909.05858.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Angeliki Lazaridou, Elena Gribovskaya, Wojciech Stokowiec, and Nikolai Grigorev. 2022. Internet-augmented language models through few-shot prompting for open-domain question answering. *arXiv preprint arXiv:2203.05115*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhu Chen. 2023. Few-shot in-context learning on knowledge base question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6966–6980.

Y Liu. 2020. Multilingual denoising pre-training for neural machine translation. *arXiv preprint arXiv:2001.08210*.

Yasmin Moslem, Rejwanul Haque, John Kelleher, and Andy Way. 2023. Adaptive machine translation with large language models. In *Proceedings of the 24th Annual Conference of the European Association for Machine Translation*, pages 227–237.

Kurnia Muludi, Kaira Milani Fitria, Joko Triloka, et al. 2024. Retrieval-augmented generation approach: Document question answering using large language model. *International Journal of Advanced Computer Science & Applications*, 15(3).

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Ricardo Rei, José GC De Souza, Duarte Alves, Chrysoula Zerva, Ana C Farinha, Taisiya Glushkova, Alon Lavie, Luisa Coheur, and André FT Martins. 2022. Comet-22: Unbabel-ist 2022 submission for the metrics shared task. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 578–585.

Daniil Sorokin and Iryna Gurevych. 2018. Modeling semantics with gated graph neural networks for knowledge base question answering. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3306–3317. Association for Computational Linguistics.

Shouhui Wang and Biao Qin. 2024. No need for large-scale search: Exploring large language models in complex knowledge base question answering. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 12288–12299.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2020. Neural text generation with unlikelihood training. In *8th International Conference on Learning Representations, ICLR 2020*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *Preprint*, arXiv:1609.08144.

Guanming Xiong, Junwei Bao, and Wen Zhao. 2024. Interactive-kbqa: Multi-turn interactions for knowledge base question answering with large language models. *arXiv preprint arXiv:2402.15131*.

Jin Xu, Xiaojiang Liu, Jianhao Yan, Deng Cai, Huayang Li, and Jian Li. 2022. Learning to break the loop: Analyzing and mitigating repetitions for neural text generation. *Advances in Neural Information Processing Systems*, 35:3082–3095.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206.

Biao Zhang, Barry Haddow, and Alexandra Birch. 2023. Prompting large language model for machine translation: A case study. In *International Conference on Machine Learning*, pages 41092–41110. PMLR.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D Manning, and Jure Leskovec. 2022. Greaselm: Graph reasoning enhanced language models for question answering. *arXiv preprint arXiv:2201.08860*.

A Appendix

A.1 Data Collection and Preprocessing

We evaluate our proposed metric, RAP, using two tasks of question answering and machine translation. For question answering, we use WebQSP and MS MARCO. The **WebQSP** dataset (Yih et al., 2016) is a widely used benchmark for KBQA task. We selected a subset of 1008 questions from test data, mainly following the criteria as for WebQSP-WD (Sorokin and Gurevych, 2018) that to keep questions with at least one acceptable answer in Wikidata. **MS MARCO** (Microsoft Machine Reading Comprehension) is a comprehensive dataset for reading comprehension and question answering, released by Microsoft[2]. For our experiments, we used the dataset created by (Huang et al., 2024). This dataset includes 100 queries, each with well-formed answers across five categories: LOCATION, NUMERIC, PERSON, DESCRIPTION, and ENTITY, totaling 500 queries. Each query is accompanied by 10 passages that provide context for the RAG-based question answering task. For machine translation task, we use the **MAC** (Manually Aligned Chinese-English) dataset curated by (Huang and Wang, 2025). This dataset comprises sentences from six Chinese novels and their corresponding English translations, spanning a diverse range of genres, including humor, martial arts, classics, war, romance, and science fiction. It contains a total of 5,661 entries, with 4,528 used for training and 1,133 reserved for testing.

Preparing Context-Data for WebQSP To evaluate open-source LLMs using various prompting strategies, including Retrieval-Augmented Generation (RAG), we crawled Wikipedia for relevant documents to address WebQSP questions. We developed a Python script to prepare the context data by retrieving up to 10 documents per query, which were then processed through text extraction, segmentation, and embedding. These document chunk embeddings were stored locally for vector search. For each question, we retrieved the 8 most relevant document chunks through similarity search to form the context for RAG prompting.

A.2 LLM-based KBQA Evaluation

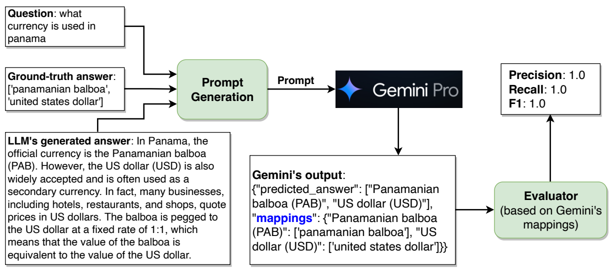
In the KBQA task, answers are typically lists of entities from the KG. For instance, the answer to "What language is spoken in Switzerland?" is "[Italian, German, French, Romansh]". While LLMs can be asked to generate entity lists, they do not always follow this format. Therefore, processing LLM-generated answers to extract entity lists for evaluation is crucial. Using Named Entity Recognition (NER) alone is not sufficient because answers may include non-entity information and differing entity wordings, e.g., "United States Dollar" versus "US dollar (USD)". To address this, we propose an LLM-based evaluation method where an LLM extracts and maps answer-items (entities) from the generated answer to the ground-truth entities. Figure 7 illustrates this workflow with an example generated by Llama-3-8B. We construct a prompt using a predefined template (Listing 1), including the question, ground-truth answer, LLM-generated answer, and a request for entity extraction and mapping. As shown in the figure, Gemini outputs a list of *predicted answers* and a *mapping* of predicted answers to ground-truth answers. A predicted answer can map to multiple ground-truth items or

[2]https://microsoft.github.io/msmarco/

```
You are evaluating a question answering task. The
↪  ground-truth answer is a list of
↪  answer-items.
First, extract the answer-items from the
↪  generated answer.
Then, map each generated answer-item to the
↪  ground-truth answer-item if possible. If no
↪  matching, use an empty list.
Output using the following format.
↪  {"predicted_answer": [list of answer-items
↪  extracted from the generated answer],
↪  "mappings": {"predicted_answer_1": [matched
↪  groundtruth answers], "predicted_answer_2":
↪  []}}

Question: <KBQA Question>
Ground-truth answer: <KBQA Ground-truth Answer>
Generated answer: <LLM-generated Answer>
```

Listing 1: Prompt template for KBQA evaluation using Gemini Pro. The actual content will be filled in the placeholders of *Question*, *Ground-truth answer*, and *Generated answer*.



Question: what currency is used in panama

Ground-truth answer: ['panamanian balboa', 'united states dollar']

LLM's generated answer: In Panama, the official currency is the Panamanian balboa (PAB). However, the US dollar (USD) is also widely accepted and is often used as a secondary currency. In fact, many businesses, including hotels, restaurants, and shops, quote prices in US dollars. The balboa is pegged to the US dollar at a fixed rate of 1:1, which means that the value of the balboa is equivalent to the value of the US dollar.

Gemini's output: {"predicted_answer": ["Panamanian balboa (PAB)", "US dollar (USD)"], "**mappings**": {"Panamanian balboa (PAB)": ['panamanian balboa'], "US dollar (USD)": ['united states dollar']}}

Precision: 1.0
Recall: 1.0
F1: 1.0

Figure 7: LLM-based KBQA evaluation workflow using prompts for precision, recall, and F1 scores with Gemini-1.0-Pro as evaluator.

none (i.e., incorrect answer).

The mapping is then used to compute evaluation metrics including Precision, Recall, and F1. The details are as follows. Given a question, we denote the ground-truth answer as $A = [A_1, A_2, \cdots, A_n]$ and the list of predicted answer-items as $\hat{A} = [\hat{A}_1, \hat{A}_2, \cdots, \hat{A}_m]$. The mapping is denoted as $M = \{\hat{A}_1 : M_1, \hat{A}_2 : M_2, \cdots, \hat{A}_m : M_m\}$ where M_i is a list of ground-truth answer-items matched with the corresponding predicted answer-item, i.e., $M_i \subseteq A$ $(i = 1, \cdots, m)$ and M_i is empty when the predicted answer-item is wrong. A predicted answer-item is called *correct* if it matches with at least one of the ground-truth answer-items. We define $c_G = |\text{set}(M_1 \cup M2 \cup \cdots \cup M_m)|$ as the number unique ground-truth answer-items that matched with at least one predicted answer-item and c_P as the number of correct predicted answer-items. The final number of correct answers is chosen as the smallest number between c_G and c_P. This is to deal with the cases where different predicted answer-items carry the same meaning (mapped to the same ground-truth answer-item). The precision and recall are then computed as $\frac{c}{|\hat{A}|}$ and $\frac{c}{|A|}$, respectively. F1 score is computed as the harmonic mean of precision and recall. We chose Gemini-1.0-Pro for this evaluation task because of its strong natural language processing capabilities and its availability during our experiments.

A.3 Tuning RPP for All Models and Datasets

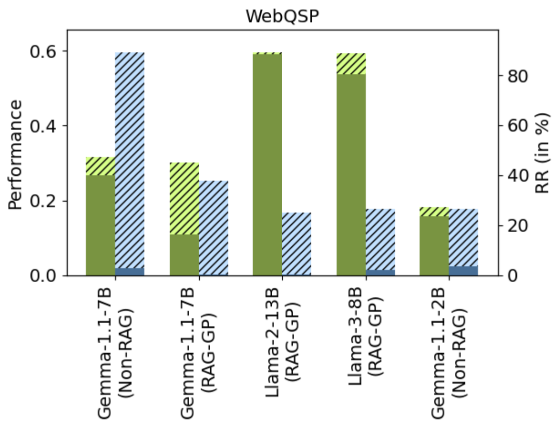


Figure 8: Top-5 model with highest RR reduction for WebQSP dataset

We visualize the best performance when tuning RPP using original evaluation metrics (Original) and our proposed repetition-aware performance metric (RAP). RAP helps find RPP values that minimally reduce performance while significantly lowering repetition ratio (RR). Results for all models across three datasets are shown in Figure 15 (WebQSP, MS MARCO) and 10 (MAC). Figure 8 and 9 show the top-5 highest RR reduction after tuning for WebQSP and MS MARCO dataset respectively. Green color bar indicates performance and blue color bar indicates RR. The hatched bar visualize the magnitude of reduction after tuning.

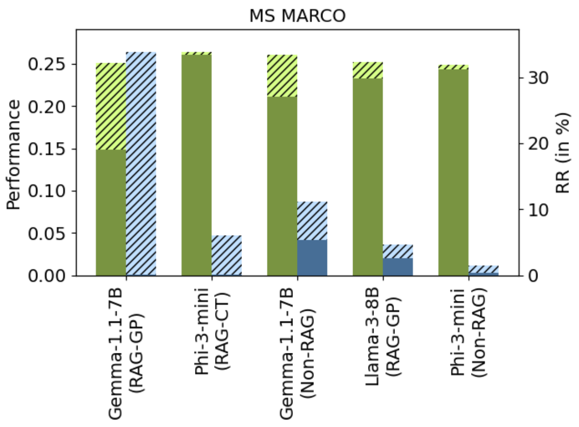


Figure 9: Top-5 model with highest RR reduction for MS MARCO dataset

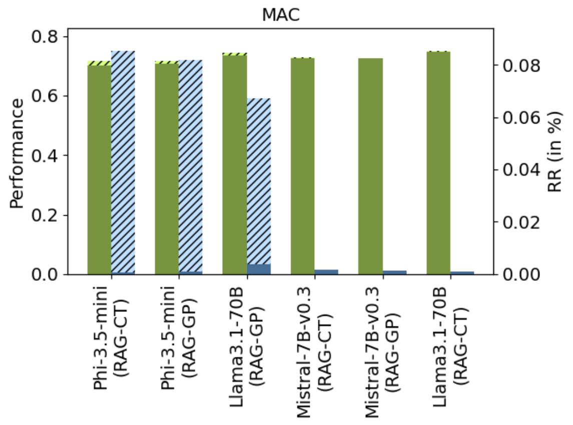


Figure 10: Tuning RPP of all models for MAC dataset

A.4 How Severe can Repetition be for Other Models and Datasets?

Figure 11 visualize the top-3 highest and lowest RR for WebQSP, while Figure 13, 14, and 12 visualize all models for WebQSP, MS MARCO, MAC respectively. The leftmost three models show the Top-3 highest RR, while the rightmost three display the lowest. The ratio of repeated text length to text length is also visualized. Colored squares above each point indicate the percentage of repeated questions, with blue, yellow, and red squares marking

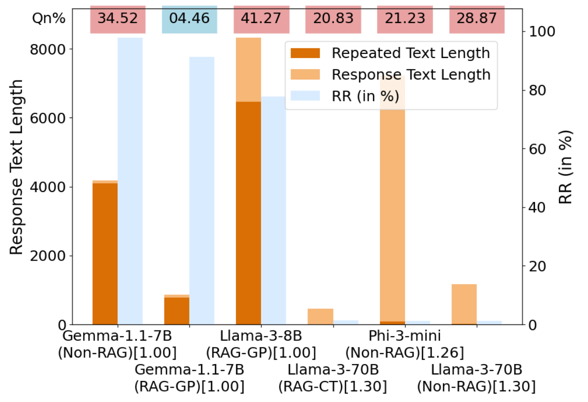


Figure 11: Top-3 highest and lowest RR for WebQSP dataset

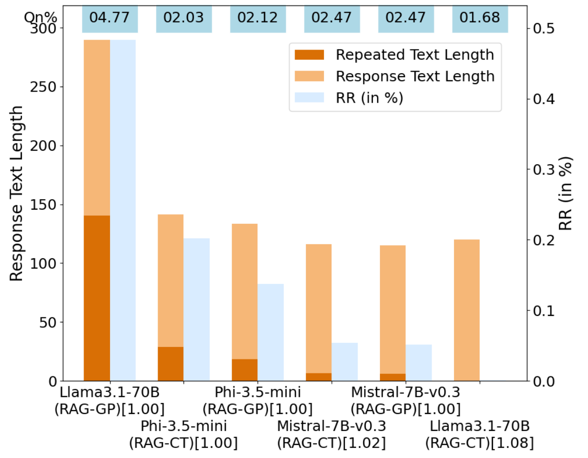


Figure 12: Percentage of repeated responses, repeated text length, and RR for MAC dataset

values below 10%, between 10-20%, and above 20% respectively. The x-label indicates the model, the prompting method, and the RP of the RR.

A.5 Evaluating the Quality of Gemini's Evaluation

To assess the effectiveness of our LLM-based KBQA evaluation method, we randomly selected 100 samples from the experiment results of the Llama-3-70B Non-RAG model, as this configuration achieved the best performance on WebQSP. A human evaluator was then asked to review and provide feedback on Gemini's evaluation of these samples (following the process described earlier). The agreement rate between the human evaluator and Gemini was 89.0%. While this indicates that the method is not flawless, it demonstrates a high level of reliability, making our LLM-based KBQA evaluation suitable for automatically assessing LLM responses in KBQA tasks.

We conducted a qualitative analysis of the proposed LLM-based KBQA evaluation using Gemini-1.0-Pro (Section A.2). Table 3 presents examples of outputs from Llama-3-70B for five WebQSP questions, along with the corresponding evaluations from Gemini.

In the first example, Gemini correctly extracts the answer item "Lawrence E. Roberts" and accurately maps the predicted answer to the ground-truth answer, resulting in a precise evaluation. A similar correct evaluation occurs in the second example.

However, Gemini occasionally fails to extract the correct information from the generated text. In the third question, for example, Gemini mistakenly identifies "Elizabeth Bowes-Lyon" as part of the predicted answer, despite the LLM response not implying this. This error leads to a precision score of only 0.5 for this answer, negatively affecting the LLM's overall performance evaluation. A similar issue occurs in the fifth question, where Gemini extracts too many keywords from the LLM response, yielding a very low final score, despite the fact that the predicted answer is correct when reading the response in context.

In summary, to further evaluate the quality of our LLM-based KBQA evaluation method, we randomly selected 100 samples from the Llama-3-70B Non-RAG model's results, as this configuration performed the best on WebQSP. A human evaluator reviewed Gemini's assessments, and the agreement rate was 89.0%. While not perfect, this indicates that our LLM-based KBQA evaluation method demonstrates a strong level of reliability and can be effectively utilized for automatically evaluating LLM responses in KBQA tasks.

A.6 Prompt Templates

In this section, we present the various prompt templates used for different tasks and configurations in our experiments. These templates guide the interaction between the models and the provided input, ensuring consistency across models and tasks. The templates cover both question answering (QA) and machine translation (MT) tasks, with specialized designs for models with and without retrieval-augmented generation (RAG) capabilities. Each template is structured with placeholders that dynamically incorporate the question, context, or input data. The following listings describe the prompt templates in detail.

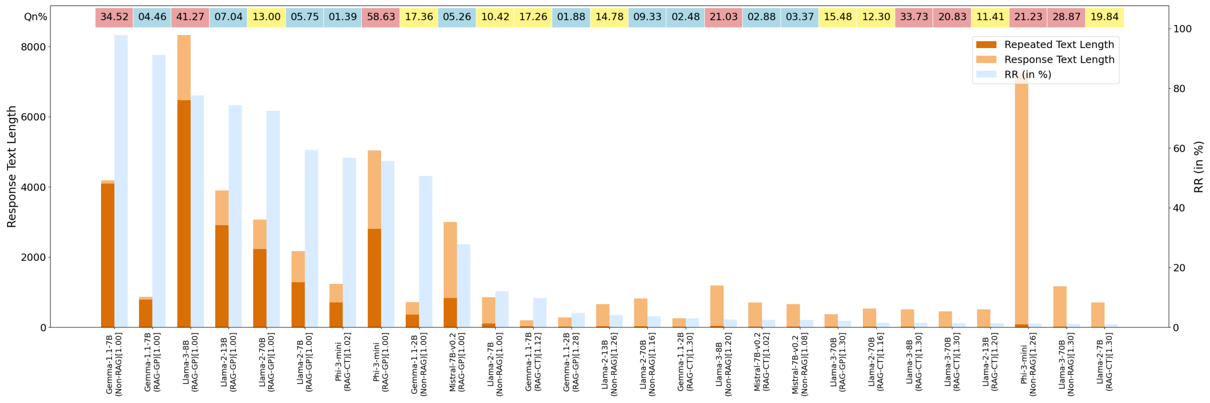


Figure 13: Percentage of repeated responses, repeated text length, and RR for WebQSP dataset

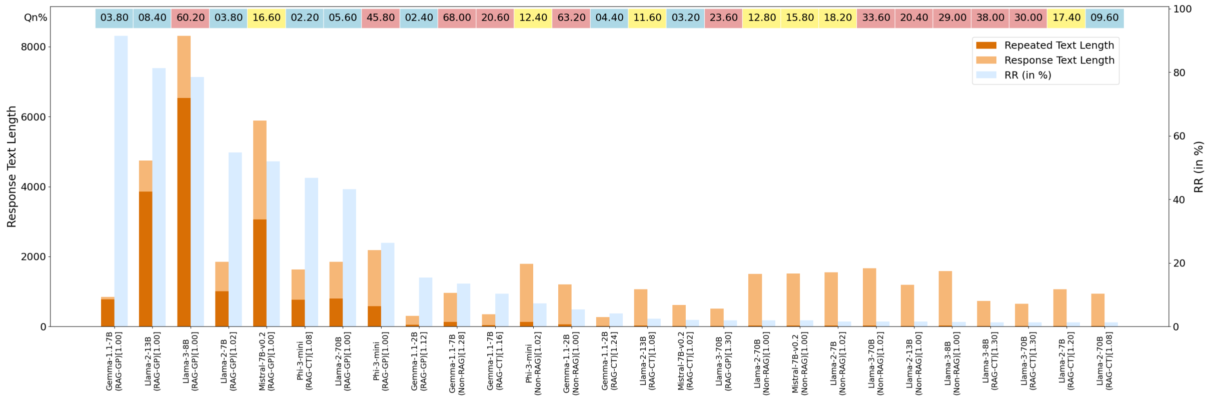


Figure 14: Percentage of repeated responses, repeated text length, and RR for MS MARCO dataset

```
Use the following pieces of context to answer the
↪  question at the end. If you don't know the
↪  answer, just say that you don't know, don't
↪  try to make up an answer.
{context}
Question: {question}
Helpful Answer:
```

Listing 2: RAG - Generic Prompt template for question answering. The placeholders *{question}* and *{context}* will be filled with the question and the retrieved data, respectively.

```
<|begin_of_text|>
<|start_header_id|>
system
<|end_header_id|>
You are a chatbot having a conversation with a
↪  human.
<|eot_id|>
<|start_header_id|>user
<|end_header_id|>
{question}
<|eot_id|>
<|start_header_id|>
Assistant
<|end_header_id|>
```

Listing 3: Non-RAG prompt template for question answering with Llama-3 models. The placeholder *{question}* will be filled with the question.

No.	Question	Ground-truth	LLM Response (Llama 3-70B)	Gemini's Evaluation
1	who is robin roberts father	[lawrence e. roberts]	Robin Roberts father is Lawrence E. Roberts, a pilot in the United States Air Force and a colonel. He passed away in 2004.	{**predicted_answer**: [Lawrence E. Roberts], **mappings**: {Lawrence E. Roberts: [lawrence e. roberts]}}
2	who does michael oher play for in 2010	[baltimore ravens]	In 2010, Michael Oher played for the Baltimore Ravens.	{**predicted_answer**: [Baltimore Ravens], **mappings**: {Baltimore Ravens: [baltimore ravens]}}
3	who was queen elizabeth ii mom	['queen elizabeth the queen mother']	Queen Elizabeth II's mother was Queen Elizabeth The Queen Mother, also known as Elizabeth Bowes-Lyon. She was the wife of King George VI and the mother of Queen Elizabeth II and Princess Margaret.	{"**predicted_answer**": ["Queen Elizabeth The Queen Mother", "Elizabeth Bowes-Lyon"], "**mappings**": {"Queen Elizabeth The Queen Mother": ['queen elizabeth the queen mother'], "Elizabeth Bowes-Lyon": []}}
4	where was rihanna born and raised	[saint michael parish]	Rihanna was born in Saint Michael, Barbados, and raised in Bridgetown, Barbados.	{**predicted_answer**: [Saint Michael, Bridgetown], **mappings**: {Saint Michael: [saint michael parish], Bridgetown: []}}
5	what country was king tut s tomb found	[egypt]	King Tuts tomb, also known as KV62, was discovered in Egypt. Specifically, it was found in the Valley of the Kings, a necropolis in Luxor, Egypt, on November 4, 1922, by British archaeologist Howard Carter.	{**predicted_answer**: [Egypt, Valley of the Kings, Luxor, Egypt, November 4, 1922, Howard Carter], **mappings**: {Egypt: [egypt], Valley of the Kings: [], Luxor: [], Egypt: [], November 4, 1922: [], Howard Carter: []}}

Table 3: Showing example of using Gemini-1.0-Pro for evaluating KBQA task. In many cases, Gemini can extract predicted answer-items and generate the mappings correctly, but sometimes it failed to extract the correct prediction, making the evaluation inaccurate. Nevertheless, a human evaluation of 100 samples showed a high agreement between human and Gemini of 89.0%. LLM Response is taken from Llama 3-70B, Non-RAG setting.

```
<|begin_of_text|>
<|start_header_id|>
system
<|end_header_id|>
You are a chatbot having a conversation with a
↪  human.
<|eot_id|>
<|start_header_id|>user
<|end_header_id|>
Use the following pieces of context to answer the
↪  question at the end. If you don't know the
↪  answer, just say that you don't know, don't
↪  try to make up an answer.
{context}
Question: {question}
<|eot_id|>
<|start_header_id|>
Assistant
<|end_header_id|>
```

Listing 4: RAG - Chat Template for Llama-3 models. The placeholders *{question}* and *{context}* will be filled with the question and retrieved data, respectively.

```
You will be given a Chinese sentence to
↪  translate. If it is an incomplete sentence,
↪  or if you are unsure about the meaning,
↪  simply copy the input text as your output. Do
↪  not output any additional sentences, such as
↪  explanations or reasoning.
Chinese: {input}
```

Listing 5: Machine Translation - Generic Prompt (MT-GP) for all models. The placeholder *{input}* will be filled with the Chinese sentence retrieved from the MAC testing set.

```
<|begin_of_text|>
<|start_header_id|>
system
<|end_header_id|>
You are a helpful assistant that translates
↪  Chinese to English.
<|eot_id|>
<|start_header_id|>user
<|end_header_id|>
You will be given a Chinese sentence to
↪  translate. If it is an incomplete sentence,
↪  or if you are unsure about the meaning,
↪  simply copy the input text as your output. Do
↪  not output any additional sentences, such as
↪  explanations or reasoning.
Chinese: {input}
<|eot_id|>
<|start_header_id|>
Assistant
<|end_header_id|>
```

Listing 6: Machine Translation - Chat Template (MT-CT) for Llama-3 models. The placeholder *{input}* will be filled with the Chinese sentence retrieved from the MAC testing set.

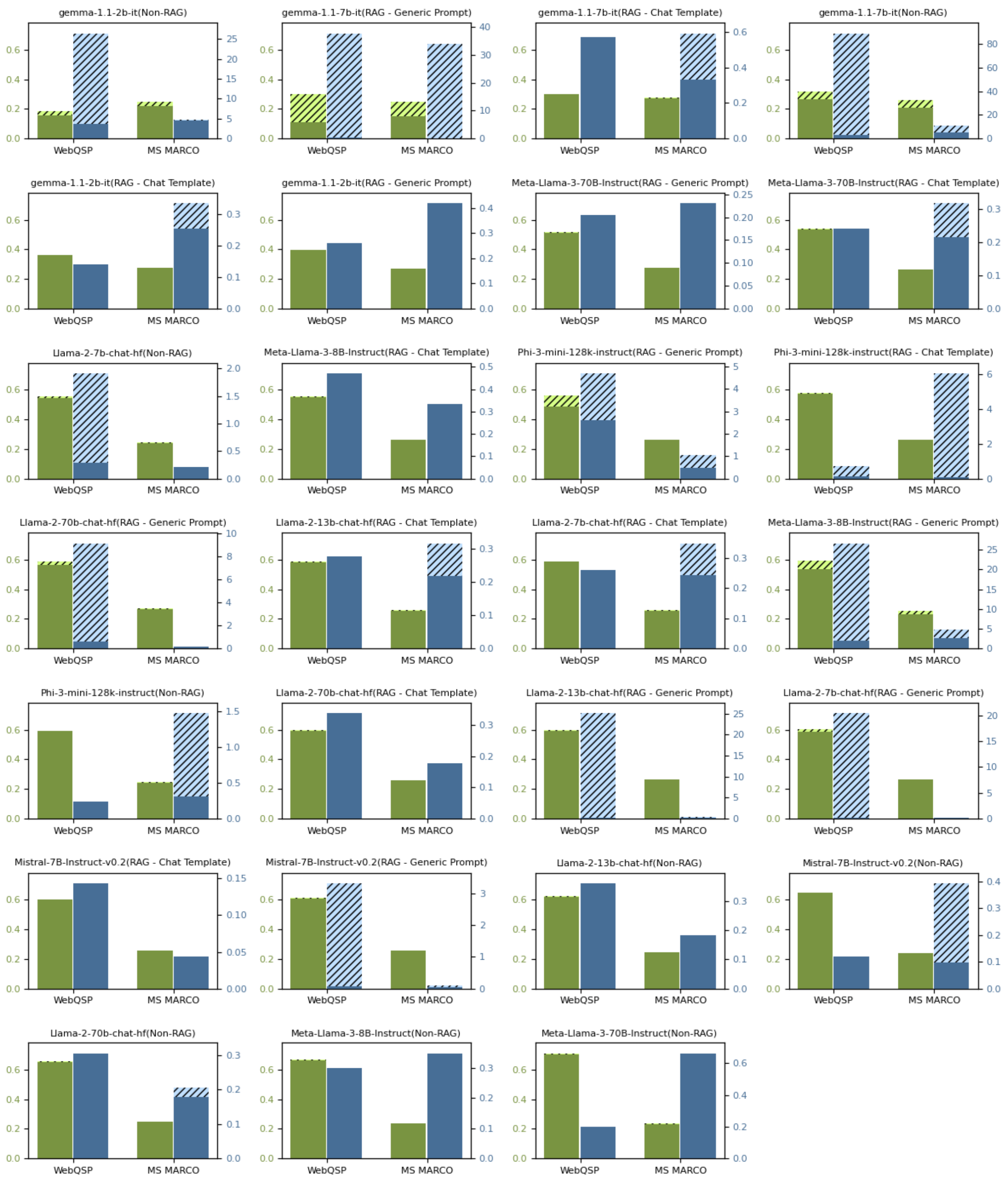


Figure 15: Tuning RPP of all models for WebQSP and MS MARCO dataset