Ext2Gen: Alignment through Unified Extraction and Generation for Robust Retrieval-Augmented Generation

Anonymous ACL submission

Abstract

Retrieval-augmented generation (RAG) enhances LLMs by integrating external knowledge, but generation remains fragile due to the uncertain placement of relevant chunks and retrieval-induced information overload, leading to hallucinations. We propose **Ext2Gen**, a novel extract-then-generate model that enhances RAG robustness by first extracting query-relevant sentences before generating answers. To optimize this model, we employ preference alignment through pairwise feedback learning, enabling the model to generate robust answers regardless of variations in retrieval results. Extensive experiments demonstrate that Ext2Gen effectively identifies query-relevant sentences with high precision and recall, leading to highly reliable answers. Furthermore, deploying our model in a RAG environment reveals that it not only boosts the performance of the base LLM but also synergizes with advanced retrieval strategies like query expansion. The dataset and model will be available upon acceptance. A portion of the dataset is available beforehand at `https://bit.ly/4b2gSzc`.

1 Introduction

Retrieval-augmented generation (RAG) has proven its effectiveness in reducing hallucinations in large language models (LLMs), when their knowledge is incomplete, outdated, or lacks sufficient detail to accurately address specific queries (Gao et al., 2023b; Fan et al., 2024). A critical aspect of RAG is the "retrieval" process, which involves identifying and selecting relevant text chunks. The quality of these retrieved chunks plays a pivotal role in the overall performance of RAG, as they form the basis for generating factual and contextually relevant answers aligned with the query intent (Asai et al., 2024; Wang et al., 2023; Zhang et al., 2024).

In this regard, most recent works have primarily focused on improving retrieval accuracy to increase the likelihood of relevant chunks being included in the Top-k search results, such as query expansion (Wang et al., 2023; Zhang et al., 2024), re-ranking (Reddy et al., 2024; Hwang et al., 2024), and self-critique (Asai et al., 2024; Li et al., 2024). These methods work by expanding contextual information to the query, re-scoring retrieved chunks to prioritize relevance, and validating the chunks against the query to ensure consistency.

Despite advancements in retrieval accuracy, bottlenecks still persist in the "generation" process, due to *uncertain placement*, where relevant chunks appear unpredictably within retrieved results, and *information overload*, where irrelevant chunks are included with varying degrees. These issues result in hallucinations even when retrieved results contain correct information, as generation models struggle to utilize relevant content due to forgetting by the "lost in the middle" phenomenon (Liu et al., 2024a) and distraction caused by "information overload" from irrelevant chunks (Cuconasu et al., 2024). These challenges are particularly severe in RAG, where smaller models, more susceptible to noise-related vulnerabilities, are adopted in generation (see Section 4.1.1).

In this paper, we go beyond accurate retrieval to emphasize *robust generation* that remains resilient to forgetting and distraction by the two challenges. Our key idea for enhancing robustness is an **extract-then-generate** approach named **Ext2Gen**, where the model first extracts query-relevant sentences from the retrieved chunks and then refine the information to generate a precise answer. The extraction step here serves as a *chain-of-thought (CoT)* process (Wei et al., 2022; Chu et al., 2023), where the model provides the evidence first before generating the final answer. However, solely relying on `Ext2Gen` via prompt engineering is insufficient to ensure satisfactory robustness, since the model is not explicitly guided to pinpoint the exact positions of relevant chunks and to filter out the noisy information.

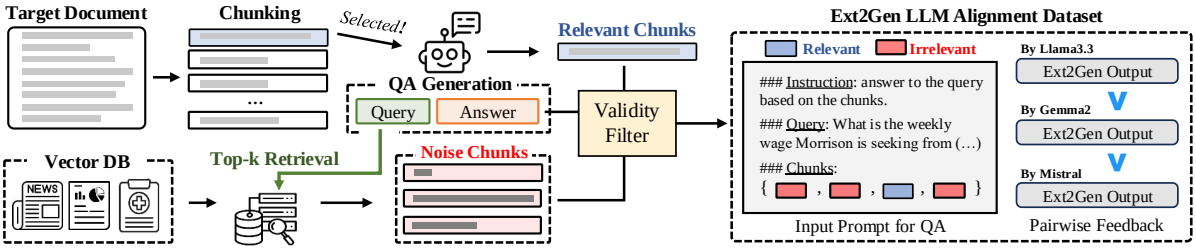

Figure 1: Overview of preference alignment through Ext2Gen.

We frame these challenges as an *alignment* problem (Zhou et al., 2024; Wang et al., 2024b), where a discrepancy exists between the model's desired capability and actual behavior. Ideally, the model should accurately identify query-relevant chunks regardless of their position and noise in the input, but in practice, it is often distracted by their placement and the unavoidable noise from the retrieval step. To bridge this gap, we enhance our Ext2Gen approach via *preference alignment* (Rafailov et al., 2024; Guan et al., 2024; Ethayarajh et al., 2024), introducing explicit training signals that guide the generation model. This process for Ext2Gen entails a sophisticated construction of an alignment training dataset, incorporating pairwise comparison feedback, as illustrated in Figure 1.

Specifically, we construct a large-scale dataset that mimics *real-world* retrieval conditions, where retrieved results include both relevant and irrelevant information. Firstly, we generate question and answer pairs using LLMs from multi-domain source datasets, including HotPotQA (wiki), MS-MARCO (web search), PubMed (medical), CNNDM (news), and GovReport (report). Secondly, for each query, we collect "relevant chunks" that contain the correct answer, along with multiple "irrelevant chunks" filtered from the chunk set obtained by a retrieval strategy. To simulate realistic RAG input, chunks are mixed with up to 25 sampled irrelevant chunks and an unpredictably placed relevant chunk. This design mirrors critical challenges in the generation step of RAG, namely uncertain placement, as well as information overload.

For effective preference alignment, the construction of *high-quality* feedback data is crucial, as it provides explicit signals that guide the model toward robust generation. By reinforcing preferred (chosen) outputs over rejected ones, the model reduces forgetting and minimizes distraction from noisy retrieval results, ensuring more reliable answers. To achieve this, we collect output completions using eight popular LLMs,[1] applying the Ext2Gen pipeline. We then construct pairwise feedback by evaluating these outputs with four popular QA metrics, including Accuracy (Acc), LLM-based evaluation (LLMEval), ROUGE-L, and BERTScore. This feedback offers direct supervision for preference alignment methods, such as DPO (Rafailov et al., 2024) and KTO (Ethayarajh et al., 2024).

Our main contributions are: (1) We introduce Ext2Gen, a novel extract-then-generate model that mitigates hallucination by explicitly separating evidence extraction from generation. (2) We construct the first preference alignment dataset tailored for RAG, enabling models to learn to prioritize relevant information while effectively filtering out noise. (3) Our experiments reveal several findings: Ext2Gen significantly enhances generation robustness, reducing hallucinations caused by uncertain placement and information overload; it achieves strong alignment, demonstrating that balancing inclusion- and similarity-based feedback leads to better Pareto-optimal performance; and it synergizes with advanced retrieval strategies, resulting in superior performance for RAG.

2 Related Work

Retrieval in RAG is an essential process to fetch the most relevant text chunks to ground the responses to the given query. Two traditional approaches are employed for retrieval: *sparse* retrieval, which relies on lexical-based methods such as BM25 (Robertson et al., 2009), and *dense* retrieval, which uses text embeddings from both queries and text chunks (Zhao et al., 2024). With the recent advance in RAG, significant efforts have been made to maximize retrieval performance. These include techniques: *query expansion* enriches the original query with semantically related terms to improve recall using LLMs (Gao et al., 2023a; Wang et al., 2023; Zhang et al., 2024; Rashid et al., 2024); *re-ranking* refines the initial retrieval results using more sophisticated models,

[1] Eight LLMs, varying in performance levels, are selected to ensure diverse response quality, enabling the construction of varied pairwise feedback for alignment tuning.

often leveraging cross-encoders for better relevance estimation (Reddy et al., 2024; Hwang et al., 2024; Yu et al., 2024b); and *self-critique* iteratively verifies retrieved content for factual consistency (Asai et al., 2024; He et al., 2024; Ye et al., 2024) and can integrate web search for up-to-date information (Yan et al., 2024).

Despite improving retrieval accuracy, hallucinations during generation necessitate complementary research (Laban et al., 2024; Islam et al., 2024).

Generation in RAG is the crucial process of producing responses grounded in retrieved content. However, hallucinations still persist due to the inability of the LLM on noisy and overloaded information (Cuconasu et al., 2024). In particular, Laban et al. (2024) evaluated 50 RAG systems on the "Summary of a Haystack" benchmark, revealing that robust generation remains an open challenge even with high retrieval accuracy. To the best of our knowledge, efforts to enhance the robustness of generation models against uncertain placement and information overload are limited.

A few recent works highlight additional challenges. Jain et al. (2024) integrated retrieval into generation, eliminating their separation for improved performance. Islam et al. (2024) enhanced reasoning capabilities using Mixture-of-Experts models. Xu et al. (2024a) reduced inference costs by compressing retrieved chunks into summaries.

Preference Alignment is essential for bridging the gap between human intent and the outputs generated by LLMs (Wang et al., 2024b; Guan et al., 2024). In this process, preference optimization plays a crucial role by guiding LLMs to prioritize human-preferred responses. There are several techniques, including PPO (Schulman et al., 2017), DPO (Rafailov et al., 2024), and KTO (Ethayarajh et al., 2024) . These methods have proven effective in aligning LLMs with human preferences, particularly in reducing hallucinations, harmful outputs, and biased content (Wang et al., 2024b).

Unlike traditional alignment approaches focused on general preferences, our research explores the use of alignment techniques to address RAG-specific alignment in generation.

3 Alignment with Ext2Gen

3.1 Overview

To achieve the desired robustness in RAG generation models, direct model training is essential, as prompt engineering with LLMs proves insufficient even with the sophisticated prompt (Song et al., 2025; Liu et al., 2024b). To this end, we explicitly teach LLMs to extract key sentences from a given set of chunks, encouraging a CoT-based reasoning process where the model first identifies grounding sentences for the query before generating the final answer. At a high level, the process follows the three steps: *data generation* and *feedback collection* outlined in Figure 1, followed by alignment through `Ext2Gen` feedback via *preference optimization*, as summarized below.

Step 1: Data Generation. We begin with a collection of `<Question,Answer>` pairs sourced from multiple domains, where each QA pair is aligned with a set of mixed text chunks containing both relevant and irrelevant context.

Then, we filter the collected data based on two perspectives: "answer validity," ensuring the answer in QA is derivable from the relevant chunk; and "chunk validity," confirming that none of the noisy chunks can infer the answer.

Step 2: Feedback Collection. We collect a diverse set of possible output completions, achieved by prompting eight LLMs with the `Ext2Gen` prompt (in Table 17) along with the filtered query and mixed chunks. The outputs are then validated for format compliance, ensuring they include both extractive sentences and the final answer.

To assign ratings of each output completion, we apply four popular QA metrics. Based on these ratings, we build a set of pairwise feedback for each query by comparing multiple completions.

Step 3: Preference Optimization. We train LLMs using preference optimization, leveraging pairwise feedback to minimize the alignment gap in generation for RAG. We investigate the effectiveness of several optimization techniques, including supervised fine-tuning (SFT) and DPO.

3.2 Dataset Generation

QA Generation. The diversity of source domain is crucial for comprehensive QA pairs, helping the model learn diverse aspects. Therefore, we generate 4K `<Question,Answer>` pairs from each of the five domain-diverse datasets, including HotPotQA (Wiki) (Yang et al., 2018), MS-MARCO (Web Search) (Craswell et al., 2021), PubMed (Medical) (Cohan et al., 2018), CNNDM (News) (Nallapati et al., 2016), and GovReport (Report) (Huang et al.,

2021). For QA generation, we first divide all documents from each dataset (in the "train" split) into text chunks of 256 words to ensure sufficient context. We then randomly sample 4K chunks from each dataset and generate QA pairs using GPT-4o.[2] In particular, we define five query types to cover various short- and long-form answers, namely *fact*-based, *instruction*-based, *explanation*, *opinion*, and *binary* queries. The query examples and the prompt used are in Tables 7 and 18 in the Appendix.

Chunk Collection. To simulate generation in RAG, we align the query in QAs with the set of retrieved chunks. We treat the selected chunk in QA generation as the "relevant" one. For irrelevant ones, we first store all text chunks in a vector database (we use ChromaDB) and perform dense retrieval using the "multilingual-e5-large-instruct" model (Wang et al., 2024a), retrieving the Top-50 text chunks for each query. The retrieved chunks are considered "noisy" after filtering out those identical to the relevant chunks. This step yield 4K sets per dataset, each consisting of a QA pair aligned with chunks, totaling 20K sets over five sources.

Data Filtering. LLM-based QA generation can introduce hallucinations, leading to undesirable biases into the dataset (Das et al., 2024; Li et al., 2023). To mitigate this potential issue, we inspect QA pairs and their associated chunks, as hallucinations may produce answers unsupported by relevant chunks. Moreover, beyond hallucinations, chunks considered "noisy" may contain information that supports correct answers despite not being labeled as relevant. Hence, to prevent their adverse effects on alignment, we design an additional filtering step that refines our initial QA pairs and chunks. In this step, we use Llama3.3-70b-instruct as a filtering model to reduce GPT-4o's self-bias, since using the same LLM for multiple tasks could introduce bias toward its own outputs (Xu et al., 2024b).

Specifically, we prompt the Llama3.3 model with the validity check prompt in Table 19 of the Appendix to assess: *"Answer Validation,"* where the answer is evaluated to ensure that it is fully derived from the relevant chunk — if not, the QA pair is filtered out as incorrectly generated (i.e., hallucination); and *"Chunk Validation,"* where each noisy chunk is checked to confirm that the answer cannot be derived from it — if it can, the chunk is removed from the noisy set (i.e., incorrect labels). We use the same prompt for both checks, as they perform the same task of verifying whether a given chunk can support the answer, regardless of its label. This process yields 18K QA pairs with clearly labeled chunks as either "relevant" or "irrelevant," with the answers serving as "true" references.

Input Consolidation. The query and its corresponding chunks in the 18K subset will form the input to mimic the generation step of RAG. To better reflect realistic input, we combine the relevant chunk with up to 25 chunks, sampled uniformly from the irrelevant set, and apply random shuffling to the combined chunks, forming a chunk list. This chunk list simulates a chunk set retrieved with varying Top-k values in RAG, reflecting the unpredictable placement of the relevant chunk and varying levels of irrelevant ones in the input. Therefore, the final input prompt for answer generation follows the `Ext2Gen` prompt in Table 17, using the query and the processed chunk list.

3.3 Feedback Collection

Output Generation. With the realistic input for RAG, we collect multiple `Ext2Gen` output completions from eight LLMs with varying performance levels, by prompting the input to them. We specifically utilize four Llama series models, varying in both version and model size, i.e., Llama3.1-8b/405b-instruct, Llama3.2-3b-instruct, and Llama3.3-70b-instruct, along with four other well-known LLMs, i.e., Mistral-nemo-12b-instruct, Gemma2-27b-it, Wizardlm2-8x22b, and GPT-4o-mini. This is crucial for effective alignment, as it allows us to gather responses of varying quality, facilitating the creation of diverse pairwise feedback even for the same input (Song et al., 2024; Chaudhari et al., 2024; Song et al., 2025). Hence, these LLMs generate rich output completions with varying quality, producing 192K input–response pairs for alignment with `Ext2Gen`.

Output Compliance. To align with the expected `Ext2Gen` output, we normalize LLMs' output completions by removing unintended ones, such as those containing only extracted sentences, direct answers to the query without sentences extracted, or outputs that follow an incorrect format. In alignment, this process helps the model maintain the consistent completion format as:

[2]For HotPotQA and MS-MARCO, well-curated QA pairs already exist. So, we sample QA pairs from each dataset. In MS-MARCO, we consider "description" queries as long-form and others as short-form, sampling 2K pairs for each. In total, 8K queries are sampled from the two datasets.

LLMs	Llama3.2-3b-inst.	Llama3.1-8b-inst.	Nemo-12b-inst.	Gemma2-27b-it	Wizardlm-2-8x22b	GPT-4o-mini	Llama3.3-70b-inst.	Llama3.1-405b-inst.
Chosen	7.9%	9.6%	11.0%	11.0%	12.7%	14.4%	16.5%	16.9%
Rejected	22.7%	17.5%	15.5%	15.1%	11.9%	6.7%	5.0%	5.6%

Table 1: Distribution of "chosen" and "rejected" output completions for eight LLMs, with Rule 2 applied for pairwise comparison. The models are sorted in ascending order from left to right based on MMLU and OpenLLM benchmark scores (see the details in Table 8). That is, stronger LLMs positioned further to the right.

Ext2Gen Output Completion

Extracted Sentences:
- sentence 1.
- sentence 2.

Answer: generated answer.

Feedback Composition. We configure pairwise feedback for preference alignment by contrasting the correctness of multiple output completions. We use four metrics to evaluate the correctness of the output from two perspectives: Accuracy (`Acc`) and `LLMEval` for assessing the "inclusion" of the true answer in the generated one[3], and `ROUGE-L` and `BERTScore` for measuring lexical and semantic "similarity" between the true and generated one. The details of the metrics are in Appendix B.1.

Based on the rated score, we design two rules for determining "chosen" and "rejected" output completions: one considers only inclusion metrics, which are more important than similarity metrics in RAG (Yu et al., 2024a), while the other considers all metrics, prioritizing them accordingly, as:

Rule 1: Inclusion-only. This considers only binary inclusion metrics (`Acc` and `LLMEval`), where 1 indicates the generated answer includes the true answer, and 0 indicates it does not. An output is considered "chosen" if either metric equals 1. The condition holds for any chosen output i:

$$\text{Acc}_i + \text{LLMEval}_i \geq 1, \tag{1}$$

indicating at least one of the inclusion metrics confirms the presence of the true answer, where $\{\text{metric}\}_\text{i}$ denotes the metric score for the output i. Then, for any chosen output i, another output j is considered a "rejected" output if:

$$\text{Acc}_i + \text{LLMEval}_i > \text{Acc}_j + \text{LLMEval}_j, \tag{2}$$

ensuring that the chosen one has a stronger inclusion signal than the rejected one.

Rule 2: Inclusion → Similarity. This rule considers both inclusion and similarity metrics, giving higher priority to `Acc` and `LLMEval` over `ROUGE-L` and `BERTScore`. The basic criteria for determining chosen and rejected outputs are the same as in Rule 1, defined in Eqs. (1) and (2). However, for rejected outputs, in addition to the condition in Eq. (2), we introduce an additional criterion to generate more chosen-rejected feedback pairs when two outputs, i and j, have identical inclusion scores, i.e., $\text{Acc}_i = \text{Acc}_j$ and $\text{LLMEval}_i = \text{LLMEval}_j$. Specifically, even if output j has the same inclusion score as the chosen output i, it is considered "rejected" if:

$$\begin{aligned} &\text{ROUGE-L}_i + \text{BERTScore}_i \\ &> \text{ROUGE-L}_j + \text{BERTScore}_j + \epsilon. \end{aligned} \tag{3}$$

This guarantees that outputs are preferred not only for including the true answer but also for exhibiting higher lexical and semantic similarity to it. The ϵ is set to 0.30 for the chosen one to have a sufficiently higher similarity score than the rejected one.

Finally, we obtain 120K feedback pairs using Rule 1 while 150K using Rule 2 (Statistics in Appendix H). Table 1 shows the proportion of each LLM's outputs judged as chosen or rejected in the 150K feedback. While stronger (right) LLMs are more frequently chosen, not all of their outputs are preferred, and weaker (left) LLMs often produce better completions. Thus, **our feedback composition promotes diverse LLM feedback in alignment training**, enabling models to learn from a wide range of quality variations rather than relying on a single LLM's outputs. The distribution under Rule 1 follows a consistent trend, as the 150K pairs form a superset that includes all 120K pairs.

3.4 Preference Optimization

Using pairwise feedback, we directly train the generation model[4] to prefer the chosen output over the rejected one. The chosen output is more robust to uncertain placement and information overload in the input prompt. Here, we mainly consider seven possible setups based on SFT and DPO:

[3]We use only the answer parsed from `Ext2Gen` output completions to compute all metric scores.

[4]We mainly use Llama3.2-3b-instruct and Llama3.1-8b-instruct as generation models for alignment tuning.

Backbone	Llama3.1-8b-instruct					Llama3.2-3b-instruct				
Metric	Acc	LLMEval	ROUGE	BERT	Avg.	Acc	LLMEval	ROUGE	BERT	Avg.
Ideal	0.439	0.918	0.339	0.881	0.644	0.446	0.877	0.310	0.876	0.627
Default	0.341	0.733	0.212	0.859	0.536	0.286	0.595	0.162	0.849	0.473
SFT-Best	0.363	0.763	0.282	0.871	0.570	0.295	0.649	0.226	0.861	0.508
Ext2Gen-R1	**0.481**	**0.889**	0.212	0.860	0.610	**0.401**	**0.773**	0.179	0.854	0.552
Ext2Gen-R2	0.463	0.860	**0.370**	**0.885**	**0.644**	0.390	0.750	**0.228**	**0.860**	**0.557**

Table 2: Evaluation results of five methods using Llama3.1-8b and Llama3.2-3b as the backbone, where ROUGE and BERT refer to ROUGE-L and BERTScore, respectively.

• **SFT-Best**: We first identify the best Ext2Gen output for each query from the eight LLMs, selecting the one with the highest average score across four QA metrics. This output is then used as the unique reference completion for SFT.

• **SFT-{Metric}**: Similar to SFT-Best, but the best output is selected based on a single metric rather than the average of all four. This setup includes four more SFT variants: SFT-Acc, SFT-LLMEval, SFT-ROUGE, and SFT-BERT.

• **Ext2Gen-{Rule}**: Unlike the SFT variants, which rely on a single reference, we leverage *multiple* pairwise feedback instances as optimization signals, even for the same query. We optimize our model using DPO based on the two feedback composition rules separately, resulting in two models: Ext2Gen-R1 and -R2. In addition to DPO, other alignment tuning methods can be applied. We compare DPO with KTO (Ethayarajh et al., 2024) and SimPO (Meng et al., 2024) in Appendix E.2.

4 Evaluation

This section presents two assessments: (i) Section 4.1: Robustness against uncertain placement and information overload in generation; and (ii) Section 4.2: Deployment to a real RAG environment.

We primarily compare our two main models (Ext2Gen-R1/R2) with other variants trained with or without SFT. For both SFT and DPO, we fine-tune Llama3.2-3b-instruct and Llama3.1-8b-instruct using QLoRA (Dettmers et al., 2024) on four NVIDIA H100 GPUs. For evaluation metrics, we employ the four metrics, Acc, LLMEval (using GPT-4o), ROUGE-L, and BERTScore, to assess the correctness of the generated answer against the true answer. Further details on the metrics and setup are provided in Appendices B.1 and C.

4.1 Robustness Evaluation

Test Dataset. We construct the test set using the same pipeline as the Ext2Gen training set to assess robustness in QA generation, but with the "test split" of the five source datasets. Since only the input is required, the process in Figure 1 runs only up to the input consolidation step for the test split. This results in a total of 1K QA pairs, with 200 QA pairs generated from each of the five source datasets. Note that in the Ext2Gen input prompt, each query is paired with a list of chunks containing both relevant chunks and up to 25 irrelevant ones, i.e., noisy chunks. The positions of the relevant chunks within the list are randomly assigned.

4.1.1 Main Results

Table 2 summarizes the generation performance of five models for the test set. Default (base model) refers to the results obtained using the Ext2Gen prompt in Table 17 without preference alignment, neither SFT nor DPO is applied, while Ideal represents those obtained with Default when only relevant chunks are provided as the chunk list.

Firstly, **the base model, Default, is highly vulnerable to uncertain placement and information overload**, with a significant performance drop, especially in smaller models like Llama3.2-3b. This issue becomes more critical in scenarios favoring compact models like RAG.

Secondly, **alignment with Ext2Gen significantly improves generation scores across all QA metrics.** Particularly, the use of constructed pairwise feedback yields much greater improvements, as evidenced by the performance gains of the Ext2Gen series over SFT-Best. This enhancement is attributed to increased robustness (see Section 4.1.2 for a detailed analysis).

Lastly, Ext2Gen-R2 reveals that **balancing inclusion and similarity metrics leads to better Pareto alignment**, resulting in the best model based on the average score. With Llama3.1-8b, Ext2Gen-R2 exhibits strong robustness, achieving performance on par with Ideal even when up to 25 irrelevant chunks were added to the input. Meanwhile, Ext2Gen-R1 surpasses Ext2Gen-R2 in inclusion metrics but falls behind in similarity metrics.

See the qualitative analysis in Appendix G.

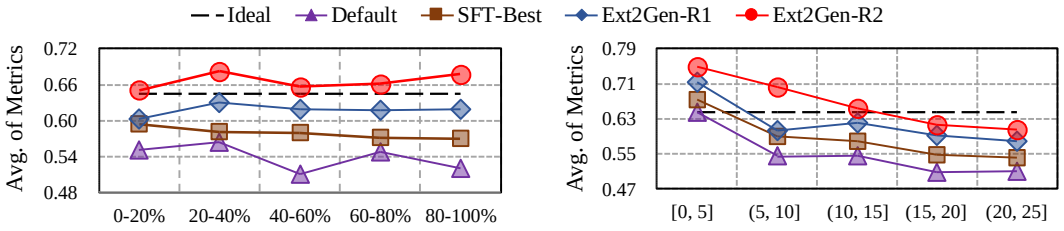

Figure 2: Robustness to **(left)** relevant chunk position (moving down as it shifts right) and **(right)** the number of added irrelevant chunks (increasing noise to the right). Results are based on the Llama3.1-8b-instruct backbone.

Backbone	Llama3.1-8b		Llama3.2-3b	
Method	Pre.	Rec.	Pre.	Rec.
`Default`	0.43	0.76	0.30	0.68
`SFT-Best`	0.50	0.75	0.41	0.69
`Ext2Gen-R1`	0.46	0.91	0.36	0.86
`Ext2Gen-R2`	0.62	0.81	0.42	0.82

Table 3: Precision (Pre.) and recall (Rec.) of the extracted sentences in output by four models.

4.1.2 Robustness to Uncertain Placement and Information Overload

Figure 2 details the results from Table 2, showing how the average score (Avg.) of four metrics varies with the relevant chunk's position (left) and the addition of irrelevant ones (right) in the input prompt. Note that `Ideal` maintains a constant score, unaffected by uncertain placement or information overload, as only relevant chunks are provided.

For uncertain placement, `Ext2Gen-R2` consistently outperforms all other methods across all relevant chunk positions, even surpassing `Ideal`, indicating its strong adaptability to positional shifts. `SFT-Best` and `Ext2Gen-R1` also exhibit improvements over `Default`, but their robustness is less pronounced. Similarly, for information overload, `Ext2Gen-R2` demonstrates superior resistance to the degradation caused by added irrelevant chunks, maintaining a significantly higher performance compared to other baselines.

4.1.3 Additional Analysis

Quality of Extracted Sentences. The quality of extracted sentences is crucial for grounding the generated answer. In Table 3, we evaluate *precision* for accuracy in selecting sentences from relevant chunks and *recall* for how much they cover the relevant ones (see Appendix B.2 for details). `Ext2Gen-R1` achieves the highest recall by favoring longer extractions due to the lack of lexical similarity in alignment, leading to lower precision. In contrast, `Ext2Gen-R2` considers all QA metrics, balancing precision and recall, enhancing robustness in Section 4.1.2. Thus, **robustness improvements depend on balancing sentence extraction precision and recall.**

Method	Sentence Number	Answer	Latency
`Default`	4.81 (115)	46	6.66
`Ext2Gen-R1`	5.10 (127)	59	7.52
`Ext2Gen-R2`	3.26 (77)	43	5.34

Table 4: Statistics of the Ext2Gen output on average: the number of extracted sentences (with their word count in parentheses), the word count of the answer, and the query processing latency (seconds per query). The input prompt length is 2,161 words on average.

Method	Acc	LLMEval	ROUGE	BERT
`SFT-Best`	0.363	0.763	0.282	0.871
`SFT-Acc`	**0.376**	0.763	0.282	0.871
`SFT-LLMEval`	0.360	**0.777**	0.220	0.861
`SFT-ROUGE`	0.368	0.748	**0.284**	0.869
`SFT-BERT`	0.357	0.744	0.280	**0.873**

Table 5: Comparison of SFT variants for alignment.

Output and Latency. Table 4 presents the statistics of the generated outputs from the `Ext2Gen` series compared to `Default`, along with their latency when using a single NVIDIA H100 with a batch size of 1. **Incorporating lexical similarity (ROUGE) in Ext2Gen-R2 improves latency**, resulting in even faster inference than `Default` due to its more concise extracted sentences and answers. In contrast, `Ext2Gen-R1` produces more extracted sentences and longer answers, as it considers only inclusion metrics (`ACC` and `LLMEval`).

SFT Variants. Table 5 compares `Ext2Gen-R2` with other SFT variants that rely on a single QA metric. **Focusing on a specific metric can imposes an alignment tax, degrading performance on other QA metrics.** Each variant excels in its target metric but at the cost of others. For instance, `SFT-LLMEval` achieves the highest LLMEval score but the lowest ROUGE-L score, while `SFT-ROUGE` maximizes the ROUGE-L score at the expense of the LLMEval score.

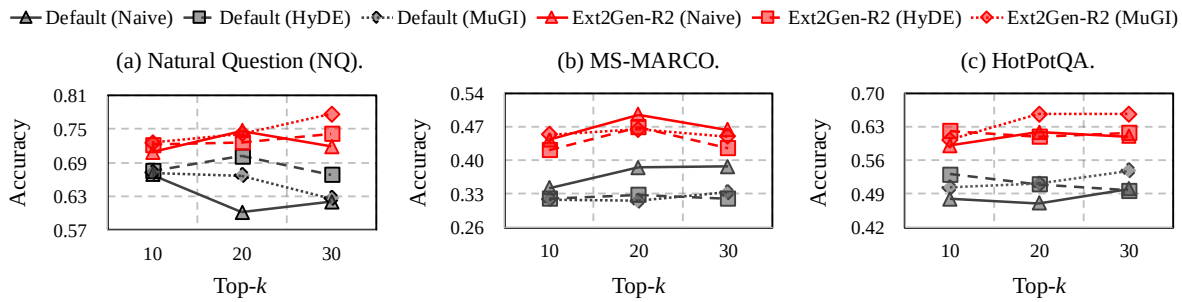

Figure 3: Accuracy (Acc) of Ext2Gen using the Llama3.1-8b backbone in a RAG environment, where three different retrieval approaches are applied: a naive dense retrieval (Naive) and its enhanced versions with two query expansion methods, namely HyDE (Gao et al., 2023a) and MuGI (Zhang et al., 2024).

Other Experiments. Additional experimental results are provided in Appendix E, offering an in-depth analysis of Ext2Gen, including (1) the impact of feedback size on alignment, (2) comparison with alternative alignment methods like KTO and SimPO, (3) an Ext2Gen model without sentence extraction in generation, (4) comparison with SFT using gold references, and (5) its application to another backbone (Qwen2.5-3b-instruct).

4.2 Deployment to RAG

Test Dataset. We deploy Ext2Gen-R2 in a real RAG environment, retrieving text chunks online from a large corpus in a vector database and prompting LLMs with the target query and the Top-k retrieved chunks.We sample 200 query-answer pairs from each of the three RAG datasets, Natural Questions (NQ), MS-MARCO, and HotpotQA, totaling 600 pairs. For the search corpus, we follow the BEIR benchmark (Thakur et al., 2021), using 2.7M and 5M text chunks for NQ and HotpotQA, respectively, and adopt the official MS-MARCO setup (Bajaj et al., 2016) with 88M chunks.

Retrieval. Before generation, we retrieve the Top-k text chunks for each query, varying k in $\{10, 20, 30\}$. To verify the generalization performance of our model, we apply three retrieval methods – Naive: a naive dense retrieval using the multilingual-e5-large-instruct model (Wang et al., 2024a) and two advanced retrieval combined with query expansion methods, namely HyDE (Gao et al., 2023a) and MuGI (Zhang et al., 2024). The retrieved chunks are added to the Ext2Gen prompt to extract key sentences and generate the answer.

4.2.1 Main Results

Figure 3 compares answer accuracy across three benchmark datasets for two model categories: one using three retrieval approaches with the canonical Llama3.1-8b-instruct backbone, Default, and the other with our Ext2Gen-R2 backbone. The trend is consistent across other metrics. Please see Appendix F for details.

A key observation is that increasing Top-k improves retrieval recall by including more relevant chunks, yet the Default model shows accuracy drops in NQ and only marginal gains in HotPotQA. This suggests that Default struggles with information overload and sensitivity to chunk position, limiting its ability to utilize additional evidence.

In contrast, **Ext2Gen-R2 demonstrates strong robustness, effectively closing the alignment gap**, where human expectations demand consistency despite retrieval-induced overload and uncertain placement, which standard models fail to handle. By better integrating retrieved content, Ext2Gen-R2 achieves significant performance gains in real RAG environments.

Moreover, while Ext2Gen-R2 performs well with naive dense retrieval, **advanced query expansion methods (HyDE, MuGI) exhibit even stronger synergy with Ext2Gen-R2**, achieving the highest accuracy on NQ and HotPotQA. This underscores how the combination of enhanced retrieval quality and Ext2Gen not only improves answer accuracy but also strengthens the model's ability to effectively utilize retrieved information, ultimately enhancing RAG effectiveness.

5 Conclusion

We introduce an extract-then-generate model, we call Ext2Gen, designed for robust RAG, mitigating retrieval-induced overload and uncertainty in chunk placement. By leveraging preference-aligned pairwise feedback, it effectively balances precision and recall in sentence extraction, ensuring reliable answer generation. Extensive evaluations across diverse datasets and real-world RAG scenarios establish Ext2Gen as a powerful and synergistic generation model for robust RAG.

Limitation. While Ext2Gen enhances robustness in RAG, it has several limitations.

Firstly, Ext2Gen relies on retrieved chunks for extraction and generation in RAG. When no relevant chunks are available in database, the model struggles to produce accurate responses. To address this, the Ext2Gen prompt in Table 17 is designed to return "no answer" when no relevant information is available. Furthermore, this issue is not unique to our model but is a fundamental challenge for all models used in RAG systems. Handling cases with no relevant information remains a research problem, requiring further exploration to improve robustness and reliability in RAG.

Secondly, compared to standard RAG, Ext2Gen introduces an additional extraction step before generation, increasing processing time. This added latency may become significant when handling long documents or deploying the model in real-time applications. However, providing explicit evidence for generated answers is essential for enhancing explainability, which is a crucial factor for real-world applications. Moreover, as shown in Appendix E.3, removing the extraction step in Ext2Gen slightly reduces performance but significantly improves latency while still outperforming baseline models, demonstrating a practical trade-off between efficiency and robustness.

Our research serves as an initial step toward improving the robustness of generation in RAG systems and is expected to inspire many future studies addressing these challenges.

Ethics Statement. Our research focuses on aligning LLMs through a unified extraction and generation framework (Ext2Gen) to enhance robustness in RAG. Since our work primarily involves model training on publicly available datasets and does not include the collection of sensitive or personally identifiable data, it does not pose direct ethical concerns related to privacy or data security.

Scientific Artifacts. The QA pairs and output completions were generated using various LLMs to enhance the diversity of model outputs. For open-source models, we utilized publicly available checkpoints from Hugging Face to ensure transparency and reproducibility. For the proprietary model, GPT-4o, we accessed OpenAI's paid API services. A detailed overview of the models is provided in Table 6 in the Appendix.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *ICLR*.

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*.

Shreyas Chaudhari, Pranjal Aggarwal, Vishvak Murahari, Tanmay Rajpurohit, Ashwin Kalyan, Karthik Narasimhan, Ameet Deshpande, and Bruno Castro da Silva. 2024. Rlhf deciphered: A critical analysis of reinforcement learning from human feedback for llms. *arXiv preprint arXiv:2404.08555*.

Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Tao He, Haotian Wang, Weihua Peng, Ming Liu, Bing Qin, and Ting Liu. 2023. A survey of chain of thought reasoning: Advances, frontiers and future. *arXiv preprint arXiv:2309.15402*.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *NAACL*.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Jimmy Lin. 2021. MS-MARCO: Benchmarking ranking models in the large-data regime. In *SIGIR*.

Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. 2024. The power of noise: Redefining retrieval for rag systems. In *SIGIR*.

Debarati Das, Karin De Langis, Anna Martin-Boyle, Jaehyung Kim, Minhwa Lee, Zae Myung Kim, Shirley Anugrah Hayati, Risako Owan, Bin Hu, Ritik Parkar, et al. 2024. Under the surface: Tracking the artifactuality of llm-generated data. *arXiv preprint arXiv:2401.14698*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. QLoRA: Efficient finetuning of quantized llms. *NeurIPS*.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. KTO: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*.

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *SIGKDD*.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023a. Precise zero-shot dense retrieval without relevance labels. In *ACL*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023b. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Melody Y Guan, Manas Joglekar, Eric Wallace, Saachi Jain, Boaz Barak, Alec Heylar, Rachel Dias, Andrea Vallone, Hongyu Ren, Jason Wei, et al. 2024. Deliberative alignment: Reasoning enables safer language models. *arXiv preprint arXiv:2412.16339*.

Bolei He, Nuo Chen, Xinran He, Lingyong Yan, Zhenkai Wei, Jinchang Luo, and Zhen-Hua Ling. 2024. Retrieving, rethinking and revising: The chain-of-verification can improve retrieval augmented generation. In *EMNLP*.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. 2021. Efficient attentions for long document summarization. In *NAACL*.

Taeho Hwang, Soyeong Jeong, Sukmin Cho, SeungYoon Han, and Jong C Park. 2024. DSLR: Document refinement with sentence-level re-ranking and reconstruction to enhance retrieval-augmented generation. *arXiv preprint arXiv:2407.03627*.

Shayekh Islam, Md Asib Rahman, KSM Tozammel Hossain, Enamul Hoque, Shafiq Joty, and Md Rizwan Parvez. 2024. Open-RAG: Enhanced retrieval augmented reasoning with open-source large language models. In *EMNLP*.

Palak Jain, Livio Baldini Soares, and Tom Kwiatkowski. 2024. From rag to riches: Retrieval interlaced with sequence generation. In *EMNLP*.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. In *EMNLP*.

Philippe Laban, Alexander Richard Fabbri, Caiming Xiong, and Chien-Sheng Wu. 2024. Summary of a haystack: A challenge to long-context llms and rag systems. In *EMNLP*.

Zhuohang Li, Jiaxin Zhang, Chao Yan, Kamalika Das, Sricharan Kumar, Murat Kantarcioglu, and Bradley Malin. 2024. Do you know what you are talking about? characterizing query-knowledge relevance for reliable retrieval augmented generation. In *EMNLP*.

Zhuoyan Li, Hangxiao Zhu, Zhuoran Lu, and Ming Yin. 2023. Synthetic data generation with large language models for text classification: Potential and limitations. In *EMNLP*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Zixuan Liu, Xiaolin Sun, and Zizhan Zheng. 2024b. Enhancing llm safety via constrained direct preference optimization. *arXiv preprint arXiv:2403.02475*.

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. SimPO: Simple preference optimization with a reference-free reward. *arXiv preprint arXiv:2405.14734*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *SIGNLL*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *NeurIPS*.

Muhammad Shihab Rashid, Jannat Ara Meem, Yue Dong, and Vagelis Hristidis. 2024. Progressive query expansion for retrieval over cost-constrained data sources. *arXiv preprint arXiv:2406.07136*.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *SIGKDD*.

Revanth Reddy, Jaehyeok Doo, Yifei Xu, Arafat Sultan, Deevya Swain, Avi Sil, and Heng Ji. 2024. FIRST: Faster improved listwise reranking with single token decoding. In *EMNLP*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Feifan Song, Bowen Yu, Minghao Li, Haiyang Yu, Fei Huang, Yongbin Li, and Houfeng Wang. 2024. Preference ranking optimization for human alignment. In *AAAI*.

Hwanjun Song, Igor Shalyminov, Hang Su, Siffi Singh, Kaisheng Yao, and Saab Mansour. 2023. Enhancing abstractiveness of summarization models through calibrated distillation. In *EMNLP*.

Hwanjun Song, Taewon Yun, Yuho Lee, Gihun Lee, Jason Cai, and Hang Su. 2025. Learning to summarize from llm-generated feedback. In *NAACL*.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *NeurIPS*.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024a. Multilingual e5 text embeddings: A technical report. *arXiv preprint arXiv:2402.05672*.

Liang Wang, Nan Yang, and Furu Wei. 2023. Query2Doc: Query expansion with large language models. In *EMNLP*.

Zhichao Wang, Bin Bi, Shiva Kumar Pentyala, Kiran Ramnath, Sougata Chaudhuri, Shubham Mehrotra, Xiang-Bo Mao, Sitaram Asur, et al. 2024b. A comprehensive survey of llm alignment techniques: RLHF, RLAIF, PPO, DPO and more. *arXiv preprint arXiv:2407.16216*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *NuerIPS*.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2024a. RECOMP: Improving retrieval-augmented lms with compression and selective augmentation. In *ICLR*.

Wenda Xu, Guanglei Zhu, Xuandong Zhao, Liangming Pan, Lei Li, and William Wang. 2024b. Pride and prejudice: Llm amplifies self-bias in self-refinement. In *ACL*.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*.

Fuda Ye, Shuangyin Li, Yongqi Zhang, and Lei Chen. 2024. Rˆ 2ag: Incorporating retrieval information into retrieval augmented generation. In *EMNLP*.

Hao Yu, Aoran Gan, Kai Zhang, Shiwei Tong, Qi Liu, and Zhaofeng Liu. 2024a. Evaluation of retrieval-augmented generation: A survey. In *BigData*.

Yue Yu, Wei Ping, Zihan Liu, Boxin Wang, Jiaxuan You, Chao Zhang, Mohammad Shoeybi, and Bryan Catanzaro. 2024b. RankRAG: Unifying context ranking with retrieval-augmented generation in llms. In *NeurIPS*.

Le Zhang, Yihong Wu, Qian Yang, and Jian-Yun Nie. 2024. Exploring the best practices of query expansion with large language models. In *EMNLP*.

Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. 2024. Dense text retrieval based on pretrained language models: A survey. *ACM Transactions on Information Systems*, 42(4):1–60.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2024. Lima: Less is more for alignment. *NeurIPS*.

Model Name	Checkpoints
Llama3.2-3b-inst	meta-llama/Llama-3.2-3B-Instruct
Llama3.1-8b-inst	meta-llama/Llama-3.1-8B-Instruct
Nemo-12b-inst	nvidia/Mistral-NeMo-12B-Instruct
Gemma2-27b-inst	google/gemma-2-27b-it
Wizardlm2-8x22b	alpindale/WizardLM-2-8x22B
GPT-4o-mini	gpt-4o-mini-2024-07-18 (OpenAI)
Llama3.3-70b-inst	meta-llama/Llama-3.3-70B-Instruct
Llama3.1-405b-inst	meta-llama/Llama-3.1-405B-Instruct

Table 6: **Checkpoints** for the eight LLMs. For open-source models, we use publicly available checkpoints from Hugging Face, whereas for proprietary models, we access them through OpenAI's paid API services.

A Query Category

We define five query types to cover various short- and long-form answers, namely *fact*-based, *instruction*-based, *explanation*, *opinion*, and *binary* queries. This enables the model to grasp the characteristics of diverse queries, allowing it to better identify key sentences and generate more accurate responses. We generate `<query, answer>` pairs using GPT-4o by using the prmopt in Table 18. THe description and example of each query type is presented in Table 7.

B Evaluation Metric

B.1 Four QA Metrics

To assess the correctness of the generated answers, we employ four widely used evaluation metrics: two inclusion-based metrics, `Acc` and `LLMEval`, and two similarity-based metrics, `ROUGE-L` and `BERTScore`. Please refer to (Yu et al., 2024a) for the detailed evaluation of RAG systems.

Accuracy (Acc) evaluates whether the true answer is contained within the predicted answer. Unlike exact match (EM), this metric checks for the inclusion of the reference answer in the generated response, making it more flexible for assessing correctness in RAG systems.

LLMEval assesses whether the true answer is contained within the predicted answer, like `Acc`, but leverages LLMs to judge correctness beyond exact lexical matching, enabling a more flexible and context-aware evaluation. We prompt GPT-4o with the evaluation prompt in Table 20.

ROUGE measures lexical similarity between the true and predicted answers. Specifically, we use the F1-score of `ROUGE-L`, which evaluates the longest common subsequence to capture both precision and recall, assessing key phrase and word order overlap between the true and predicted answers.

BERTScore measures semantic similarity between the true and predicted answers using token-level cosine similarity from contextual embeddings, capturing nuanced meaning beyond lexical overlap. We use the F1-score of `BERTScore`.

B.2 Precision and Recall

The quality of extracted sentences is crucial for grounding the generated answer. Therefore, we define two metrics, which can evaluate the performance in sentence extraction.

Let $\mathcal{C}_N = \{C_1, ..., C_{k|}\}$ denote the set of retrieved Top-k chunks for a query Q, where $\mathcal{C}_R \subseteq \mathcal{C}_N$ is the set of relevant chunks within the retrieved set. Then, suppose that $\mathcal{S} = \{S_1, ..., S_{|\mathcal{S}|}\}$ be the set of extracted sentences by running `Ext2Gen` for the same query.

To evaluate the extracted sentences, we define *extraction precision* as the proportion of extracted sentences that originate from the relevant chunks:

$$\text{Precision} = \frac{|\{S_i \mid \text{match}(S_i) \in \mathcal{C}_R\}|}{|\mathcal{S}|}$$

where $\texttt{match}(\cdot)$ is a function that returns the chunk in $\mathcal{C}_N$ that is lexically closest to the target sentence.

Similarly, we define *extraction recall* as the proportion of relevant chunks in $\mathcal{C}_R$ successfully extracted by running `Ext2Gen`.

$$\text{Recall} = \frac{|\{\text{match}(S_i) \mid \text{match}(S_i) \in \mathcal{C}_R\}|}{|\mathcal{C}_R|},$$

Note that all redundant sentences and chunks should be removed during pre-processing to ensure accurate computation of precision and recall.

C Training Configuration

For preference alignment, we explore four approaches. The specifics of each configuration are outlined below.

Supervised Fine-tuning (SFT). We fine-tune the model using QLoRA (Dettmers et al., 2024) and DeepSpeed (Stage-2) (Rasley et al., 2020) on a cluster of four NVIDIA H100 GPUs. The fine-tuning process runs for 9,000 steps with AdamW as the optimizer, employing a batch size of 32, an initial learning rate of 5e-4, and a weight decay of 0.05. To maintain consistency across different SFT approaches, we use the same training setup for all

Query Type	Description	Example
Fact	A query that asks for specific details like dates, names, locations, etc., and provide a concise factual answer.	What were some of the reasons Premier League clubs were hesitant to sign Ravel Morrison?
Instruction	A query asking how to perform an action, and provide a concise step-by-step guide or instruction.	How can Ravel Morrison improve his chances of securing a contract with a Premier League club?
Explanation	A query asking for a brief definition or explanation of a term or concept, and provide a clear explanation.	What does it mean when a football club is "paralysed by fear" in the context of signing a player?
Opinion	A query that seeks advice or a recommendation based on the document content, and provide a brief opinion or recommendation.	Should a Premier League club take the risk of signing Ravel Morrison?
Yes/No	A yes/no query based on the document chunk and answer it with "Yes" or "No."	Has Ravel Morrison been made available on a free transfer by West Ham?

Table 7: **Descriptions for the five query types**, with examples generated from the same chunk in CNN/DM.

LLMs	Llama3.2-3b-inst.	Llama3.1-8b-inst.	Nemo-12b-inst.	Gemma2-27b-it	Wizardlm-2-8x22b	GPT-4o-mini	Llama3.3-70b-inst.	Llama3.1-405b-inst.
MMLU	54.5	66.7	68.0	75.2	-	82.0	86.0	88.6
OpenLLM	24.1	28.2	-	32.3	33.0	-	-	36.8

Table 8: **MMLU and OpenLLM benchmark** scores for the eight LLMs used in `Ext2Gen` output generation.

SFT variants (SFT-{Best, Acc, LLMEval, ROUGE, BERT}) regardless of how the best output completion is determined. The primary distinction in SFT lies in how the model is conditioned: it is trained on the input paired with a single reference output, which is chosen based on a specific selection criterion. For example, this reference may correspond to the Ext2Gen output that achieves the highest average score across four evaluation metrics.

Direct Preference Optimization (DPO). We fine-tune the model using DPO (Rafailov et al., 2024). Since the model has already undergone instruction tuning, we directly apply DPO for further optimization. Similar to SFT, we utilize QLoRA and DeepSpeed (Stage-2) to train the model on a four NVIDIA H100 GPU setup. The training process spans 9,000 steps, employing AdamW as the optimizer with a batch size of 32, an initial learning rate of 5e-6, and a weight decay of 0.05.

Others. For further analysis on feedback size i(n Table 10) and comparisons with other optimization techniques, including KTO and SimPO (in Table 11), we follow the exact same setup as DPO.

D MMLU and OpenLLM Benchmark

We present the MMLU[5] and OpenLLM[6] benchmark scores of the eight LLMs used to generate `Ext2Gen` output completions in Table 8.

E Further Analysis of Ext2Gen

Additional experimental results are presented, offering an in-depth analysis of Ext2Gen, including (1) the impact of feedback size on alignment, (2) comparison with alternative alignment methods like KTO and SimPO, (3) an `Ext2Gen` model without sentence extraction in generation, (4) comparison with SFT using gold references, and (5) its application to another backbone (Qwen2.5-3b-instruct).

E.1 Impact of Feedback Size

Table 10 summarizes the performance of `Ext2Gen`-R2 in LLM alignment via DPO across varying feedback sizes. The model's robustness increases with more feedback, as reflected in the rising metric scores from 0K to 150K feedback. However, the most significant performance gain occurs within the first 30K feedback, suggesting that if high-quality alignment feedback can be curated, a smaller amount may suffice.

E.2 Ext2Gen with KTO and SimPO

With the rapid advancement of preference optimization, several techniques beyond DPO (Rafailov et al., 2024) have been proposed recently. Unlike DPO, kahneman-tversky optimization (KTO)

[5] https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard

[6] https://paperswithcode.com/sota/multi-task-language-understanding-on-mmlu

Method	Generation Quality					Number of Words		Latency
Method	Acc	LLMEval	ROUGE	BERT	Avg.	Sent.	Answer	Latency
`Default`	0.341	0.733	0.212	0.859	0.536	115	46	6.66
wo. Extraction	0.314	0.602	0.206	0.856	0.495	-	86	3.00
`Ext2Gen-R2`	0.463	0.860	0.370	0.885	0.644	77	43	5.34
wo. Extraction	0.447	0.850	0.361	0.880	0.634	-	50	1.74

Table 9: Evaluation results for `Default` and `Ext2Gen-R2`, **with and without the sentence extraction step** in the input prompt (as specified in Table 17), before generating the final answer. The results are based on the Llama3-8b-Instruct backbone and the test dataset introduced in Section 4.1.

Size	Acc	LLMEval	ROUGE	BERT	Avg.
0K	0.341	0.733	0.212	0.859	0.536
30K	0.435	0.815	0.348	0.883	0.620
60K	0.448	0.822	0.363	0.884	0.629
90K	0.445	0.821	0.352	0.882	0.625
120K	**0.467**	0.847	0.354	0.882	0.637
150K	0.463	**0.859**	**0.370**	**0.885**	**0.644**

Table 10: Impact of the **feedback size** for `Ext2Gen-R2`.

Size	Acc	LLMEval	ROUGE	BERT
`Default`	0.34	0.73	0.21	0.86
`DPO`	0.46 (+0.12)	0.86 (+0.13)	0.37 (+0.16)	0.86 (+0.00)
`KTO`	0.44 (+0.10)	0.85 (+0.12)	0.35 (+0.14)	0.85 (-0.01)
`SimPO`	0.32 (-0.02)	0.74 (+0.02)	0.34 (+0.13)	0.88 (+0.02)

Table 11: Comparison of **DPO**, **KTO**, and **SimPO** for using `Ext2Gen` on the Llama3.1-8b backbone.

(Ethayarajh et al., 2024) does not require paired preference data. Instead, it relies on binary feedback indicating whether an output is desirable ("1") or not ("0"). For KTO, we convert our pairwise dataset to the binary-wise dataset.

On the other hand, simple preference optimization (SimPO) (Meng et al., 2024) does not require a reference model. Instead, it utilizes the average log probability of a sequence as an implicit reward. This approach enhances alignment between the reward function and generation metrics, leading to improved computational and memory efficiency.

Table 11 compares the performance of `Ext2Gen` trained with three different optimization methods using the Llama3.1-8b-instruct backbone. Overall, all methods improve QA performance, with DPO yielding the highest gains in `Acc`, `LLMEval`, and `ROUGE-L`. This is why we chose DPO as the primary optimization method in our main experiments.

More specifically, KTO performs on par with DPO despite not utilizing paired chosen and rejected feedback, as it simply assigns them binary labels. This suggests that SimPO could be more efficient than DPO, as it simplifies the construction of the feedback set. In contrast, it appears that SimPO does not perform well than others.

E.3 Ext2Gen without Sentence Extraction

We demonstrate the trade-off in latency by comparing `Ext2Gen` with and without its extraction step during generation. Table 9 summarizes the answer quality of four models along with their latency.

Explicitly extracting evidence during generation leads to more accurate answers compared to omitting the extraction step. Specifically, both `Default` and `Ext2Gen-R2` exhibit a performance drop when the extraction step is removed. In terms of latency, models with sentence extraction incur additional inference time overhead. However, note that this does not undermine our contributions.

Firstly, it is evident that the preference alignment we introduce remains effective in both cases of `Ext2Gen`, showing improvements over `Default` in terms of both accuracy and latency; specifically, compare to the `Default` family, `Ext2Gen` without extraction achieves significantly higher answer accuracy while ensuring much faster inference. Secondly, providing evidence for the answer is a crucial process that enhances the explainability of LLMs' responses, making it essential for real-world application. Therefore, if latency is a critical factor, opting for `Ext2Gen` without extraction is preferable. Conversely, if evidence retrieval and higher accuracy are the priorities, `Ext2Gen` with extraction is the better choice.

E.4 SFT with Gold Reference

We define the `Ext2Gen` output completions obtained by Llama3.3-70b-instruct when given only relevant chunks as the chunk list. This definition is reasonable because it uses a stronger model instead of our main 3B/8B models, ensuring better reasoning and generation. Additionally, only relevant chunks are provided, eliminating noise and reducing the risk of hallucination. As a result, the output closely approximates the gold standard, making it a strong reference for evaluation. We denote the model trained using SFT with the gold reference as

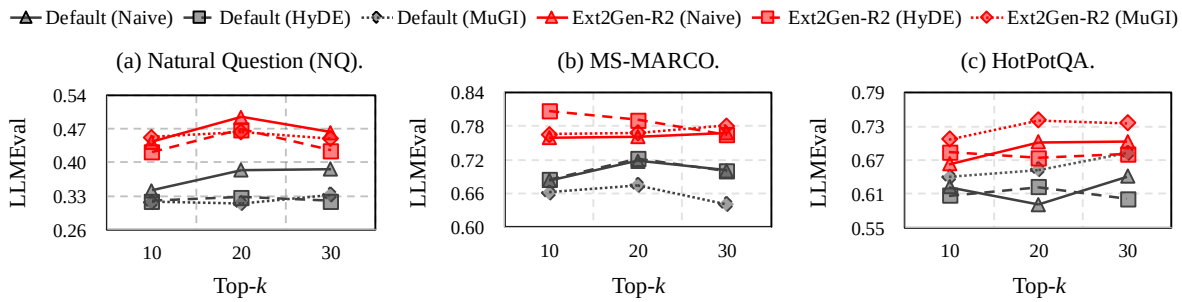

Figure 4: `LLMEval` scores of `Ext2Gen` using the Llama3.1-8b backbone in a RAG environment, where three different retrieval approaches are applied: a naive dense retrieval (`Naive`) and its enhanced versions with two query expansion methods, namely `HyDE` (Gao et al., 2023a) and `MuGI` (Zhang et al., 2024).

Backbone	Llama3.1-8b-instruct					Llama3.2-3b-instruct				
Metric	Acc	LLMEval	ROUGE	BERT	Avg.	Acc	LLMEval	ROUGE	BERT	Avg.
`Default`	0.341	0.733	0.212	0.859	0.536	0.286	0.595	0.162	0.849	0.473
SFT-GT	0.339	0.689	0.253	0.866	0.537	0.266	0.591	0.226	0.861	0.486
`SFT-Best`	0.363	0.763	0.282	0.871	0.570	0.295	0.649	0.226	0.861	0.508
`Ext2Gen-R2`	0.463	0.860	**0.370**	**0.885**	**0.644**	0.390	0.750	**0.228**	**0.860**	**0.557**

Table 12: **Comparison with SFT-GT** trained based on Llama3.1-8b and Llama3.2-3b as the backbone, where ROUGE and BERT refer to ROUGE-L and BERTScore, respectively.

Backbone	Qwen2.5-3b-instruct			
Metric	Acc	LLMEval	ROUGE	BERT
`Ideal`	0.417	0.877	0.271	0.871
`Default`	0.258	0.516	0.141	0.843
`SFT-Best`	0.305	0.609	0.217	0.859
`Ext2Gen-R2`	0.318	0.649	0.267	0.851

Table 13: Evaluation results of five methods using **Qwen2.5-3b** as the backbone, where ROUGE and BERT refer to ROUGE-L and BERTScore, respectively.

`SFT-GT` and compare it with `Default`, `SFT-Best`, and `Ext2Gen-R2` in Table 12.

Even though the gold reference is used to supervise the model, there is no improvement in the "Avg" score, compared to `Default`. Even worse, its performance is significantly worse than that of `SFT-Best`. This demonstrates that leveraging gold learning as the supervisory signal is too ideal for the model to effectively learn. On the other hand, using the best output provided by other LLMs in the same noisy setup (`SFT-Best`), where both relevant and irrelevant chunks are mixed, is a more plausible solution. This approach aligns with knowledge distillation, specifically sequence-level knowledge distillation (Kim and Rush, 2016; Song et al., 2023).

E.5 Ext2Gen with Qwen2.5-3b

We validate the generalization of our alignment pipeline by training Qwen2.5-3b-instruct as the backbone, instead of our two primary models, Llama3.2-3b-instruct and Llama3.1-8b-instruct. Table 13 summarizes the QA accuracies using four different metrics across four approaches: `Ideal`, `Default`, `SFT-Best`, and `Ext2Gen-R2`, all of which are trained with the Qwen backbone.

In general, `Ext2Gen-R2` consistently outperforms other baseline methods. Notably, its performance gain over `Default` (the base model) is significant for the two key metrics, `Acc` and `LLMEval`.

F Details in Deployment to RAG

Since we present only the results for the `Acc` metric in Section 4.2.1, we provide additional results using another metric here. Note that we observe consistent results across the four QA metrics; however, `ROUGE-L` and `BERTScore` exhibit smaller performance variations than others. So, we focus on analyzing `LLMEval` as the key metric.

Figure 4 compares the `LLMEval` scores over `Default` and `Ext2Gen-R2` incorporated with three different retrieval strategies, including `Naive`, `HyDE`, and `MuGI`. Similar to the results with `Acc`, `Ext2Gen-R2` enhances the answer quality in terms of `LLMEval` for all datasets and, additionally, makes more synergies with advanceds retrieval methods on MS-MARCO and HotPotQA.

G Qualitative Comparison

We present an example from MS-MARCO for qualitative analysis, with its input (query and chunk list) shown in Table 15 and the generated `Ext2Gen` outputs from four models displayed in Table 16. For

the output, we present the four QA metric scores along with its output completions.

As highlighted in our main results in Section 4.1.1, only Ext2Gen-R2 generates accurate and concise responses with a correct answer to the query, as evidenced by its ACC and LLMEval scores of 1.0, as well as its higher ROUGE-L and BERTScore compared to other methods. Interestingly, Ext2Gen-R1 achieves an ACC score of 1.0 but a LLMEval score of 0.0, as it includes excessive irrelevant information in its answer. This clearly demonstrates that incorporating both inclusion and similarity metrics is crucial for achieving preference alignment with Ext2Gen.

Meanwhile, the other methods, including Default and SFT-Best, fail to pinpoint the relevant information from the noisy chunk list, resulting in incorrect answers of 150,000–500,000 instead of the correct one, 150–300,000. This qualitative analysis complements our experimental results and analysis, providing a clearer understanding of the relationship between Ext2Gen-R2's performance and the evaluation metrics.

H Data Statistics

Table 14 presents the statistics of the Ext2Gen training dataset (150K feedback for Ext2Gen-R2). To ensure dataset balance across types, we processed the data so that each source contains the same number of feedback samples. For HotPotQA and MS-MARCO using original QA and chunks, the input prompt length is shorter than that of the three other datasets, i.e., CNN/DM, PubMed, and GovReport, we created for a more challenging scenario.

Dataset	Domain	# of Instances	Prompt Word count	Chosen Answer Word count	Rejected Answer Word count
HotPotQA	Wiki	25,000	746	74	76
MS-MARCO (Desc.)	Web Search	25,000	1,105	130	138
MS-MARCO (Non-desc.)	Web Search	25,000	1,097	157	166
CNN/DM	News	25,000	3,090	133	177
PubMed	Medical	25,000	3,875	151	157
GovReport	Report	25,000	3,563	177	184

Table 14: Statistics of the `Ext2Gen` training set. We prsent the number of pairwise feedback from each source data and the average number of words in its prompt, chosen, and rejected answers.

Query	How many people died at chelmno?
Chunk List	["And none of you who said Just ******* google it or stop being lazy and google it do not get 10 points! I googled it for days!!! hours on end!!! Like i said before, please, i need the statistics of the deaths. Update 2: I need to know how many germans, how many americans, italinas... I need to know the statistics of deaths. Edit1: You people are not helping. Either you state what I know or you tell me to google it. I have googled it!!!", "there was no real count on how many were actually there but most estimates are between 450,000 and 500,000.", "The extermination camp at Chelmno demands special attention, because during the German occupation only a very few people in Poland ever knew of its existence and the hundreds of thousands of its victims. The village of Chelmno (district of Kolo) is situated 14 km.", "By the end of 1943 the Germans closed down the death camps built specifically to exterminate Jews. The death tolls for the camps are as follows: Treblinka, (750,000 Jews); Belzec, (550,000 Jews); Sobibor, (200,000 200000); Jews, (150,000 150000) jews And (lublin also Called, 50,000 50000). jews", "These vans were used until the completion of the first death camp, Chelmno, which began operations in late 1941. Nazi correspondence detailing the operation of gassing vans. Nazi testimony regarding gassing vans.", "The estimates for the Jadovno concentration camp generally offer a range of 10,000 Ž013 72,000 deaths at the camp over a period of 122 days (May to August 1941). Most commonly Jadovno victims were bound together in a line and the first few victims were murdered with rifle butts or other objects.", "189 people died in the Pentagon. Another 59 died in the plane. (not counting the five terrorists). 63 were injured.", "Tortured by the SS-men, they were forced to pull the dead bodies out of the gas-vans and bury them in mass graves. Whilst there is no idea of how many unsuccessful escape attempts there were, it is known that only four people managed to escape from the death camp at Chelmno.", "Industrialization - 500,000 deaths. Stalin embarked on a huge industrialization drive during his rule and many many people were punished for resisting. At least half a million and possibly a greater number of people were killed during his rule for opposing these policies. Holodomor - 3-7 million deaths In 1932-1933, the USSR was struck by a massive famine.", "By Frederick Taylor. The question of how many people died in the World War II Allied bombing of Dresden has been politically charged for decades. Now, a commission of historians has said the real total could be much lower than previously thought. But the debate likely won't go away.", "Major Soviet Paper Says 20 Million Died As Victims of Stalin. Published: February 4, 1989. MOSCOW, Feb. 3Ž014 A Soviet weekly newspaper today published the most detailed accounting of Stalin's victims yet presented to a mass audience here, indicating that about 20 million died in labor camps, forced collectivization, famine and executions.", "How Many People Died atŽ026. How Many People Died in the Korean War. The Korean War stared on 25th of June, 1950 and ended on 27th of July, 1953. This war is actually a battle between the Communists and the Capitalists.", "We know that Stalin killed 60 million, are the 27 million people who died in WW2 counted with them? How many of them were killed in the civil war? Update Cancel", "Wow many americans died in world war 2?, how many people died in world war 2?, how many jews died during the holocaust?, how many bombs were dropped in world war II?", "The reasons for choosing the village of Chelmno as a site for an extermination camp included Ž013 its location by the road connecting the town of Kolo The town was called Warthbruecken warthbrŎ0fccken during the, occupation an important regional, centre with A jewish population of approximately 2300. people", "Part II. The aim of the Chelmno camp was the extermination of the Jews from the Warthegau, the part of Poland which consisted of the 1939 province (voivodship) of Poznania, almost the whole province of Lodz, and a part of the province of Warsaw, inhabited altogether by 4,546,000 People (including 450,000 Jews).", "The estimated number of deaths is **150-300,000**, mainly Jews. Chelmno, also known as Kulmhof, was a small town roughly 50 miles from the city of Lodz. It was here that the first mass killings of Jews by gas took place as part of the Final Solution.. The murder process was set up by a Sonderkommando, under the command of Herbert Lange.", "LEAD: A Soviet weekly newspaper today published the most detailed accounting of Stalin's victims yet presented to a mass audience here, indicating that about 20 million died in labor camps, forced collectivization, famine and executions.", "How many were killed in the Cambodian genocide? The exact number of people who died (either from torture, execution or starvation is unknown), but estimates range from 200,000 to two million (out of a population of 8,000,00Ž0260 at the time).", "Update: I need to know how many germans, how many americans, italinas... I need to know the statistics of deaths. Edit1: You people are not helping. Either you state what I know or you tell me to google it. I have googled it!!!"],

Table 15: **[Input] A qualitative example** from MS-MARCO, where the relevant chunk in the chunk list is colored in blue.

Model	Output
Default	Extracted Sentences: - there was no real count on how many were actually there but most estimates are between 450,000 and 500,000. - The estimated number of deaths is 150-300,000, mainly Jews. Answer: The estimated number of deaths at Chelmno is between **150,000 to 500,000**, with the majority being Jewish victims. ### QA Metric Scores: {"ACC": 0.0, "ROUGE-L": 0.066, "BERTScore": 0.833, "LLMEval": 0.0}
SFT-Best	Extracted Sentences: - There was no real count on how many were actually there but most estimates are between 450,000 and 500,000. - The extermination camp at Chelmno demands special attention, because during the German occupation only a very few people in Poland ever knew of its existence and the hundreds of thousands of its victims. - By the end of 1943 the Germans closed down the death camps built specifically to exterminate Jews. The death tolls for the camps are as follows: Treblinka, (750,000 Jews); Belzec, (550,000 Jews); Sobibor, (200,000 200000); Jews, (150,000 150000) jews And (lublin also Called, 50,000 50000). jews - The estimates for the Jadovno concentration camp generally offer a range of 10,000 ž013 72,000 deaths at the camp over a period of 122 days (May to August 1941). - The estimated number of deaths is 150-300,000, mainly Jews. - The exact number of people who died (either from torture, execution or starvation is unknown), but estimates range from 200,000 to two million (out of a population of 8,000,00ž0260 at the time). - Answer: The estimated death toll at Chelmno concentration camp ranges from 450,000 to 500,000 people, with the majority of those being Jews, according to most estimates. Answer: The estimated number of deaths at Chelmno is between **150,000 to 500,000**, with the majority being Jewish victims. ### QA Metric Scores: {"ACC": 0.0, "ROUGE-L": 0.173, "BERTScore": 0.831, "LLMEval": 0.0}
Ext2Gen-R1	Extracted Sentences: - There was no real count on how many were actually there but most estimates are between 450,000 and 500,000. - The extermination camp at Chelmno demands special attention, because during the German occupation only a very few people in Poland ever knew of its existence and the hundreds of thousands of its victims. - The estimated number of deaths is 150-300,000, mainly Jews. - These vans were used until the completion of the first death camp, Chelmno, which began operations in late 1941. - Tortured by the SS-men, they were forced to pull the dead bodies out of the gas-vans and bury them in mass graves. Whilst there is no idea of how many unsuccessful escape attempts there were, it is known that only four people managed to escape from the death camp at Chelmno. Answer: The estimated number of deaths at Chelmno is **150-300,000**, mainly Jews, and there is no real count on how many were actually there but most estimates are between 450,000 and 500,000. However, it's worth noting that the exact number of victims is not known, but it is considered to be hundreds of thousands. ### QA Metric Scores: {"ACC": 1.0, "ROUGE-L": 0.098, "BERTScore": 0.829, "LLMEval": 0.0}
Ext2Gen-R2	Extracted Sentences: - The estimated number of deaths is 150-300,000, mainly Jews. Answer: The estimated number of deaths at Chelmno is **150-300,000**, mainly Jews. ### QA Metric Scores: {"ACC": 1.0, "ROUGE-L": 0.375, "BERTScore": 0.854, "LLMEval": 1.0}

Table 16: **[Output] A qualitative example** from MS-MARCO, where the key content of the predicted answers is highlighted in blue if correct and in red otherwise.

Ext2Gen Prompt
You are an expert assistant trained to extract essential sentences from document chunks and generate answers based on the extracted sentences. Your task is twofold: - Extraction: Identify sentences that contribute to constructing a precise and accurate response to the given query. - Generation: Formulate a concise and coherent answer based on the extracted sentences. ### Extraction Instruction: - A query will be provided for you to answer. - Extract only the sentences that contribute to forming an answer to the query. - Ensure that the extracted sentences are sufficient to derive a correct and complete answer. - If no relevant sentences are found in the provided chunks, return an empty list. ### Generation Instruction: - Use the extracted sentences to generate a well-formed answer to the query. - If no sentences are extracted, return "No Answer". ### Output Example: Extracted Sentences: - Sentence 1 - Sentence 2 Answer: Your Answer ### Query: {query} ### Chunk List: {chunk list} ### Output:

Table 17: **Base Prompt for Ext2Gen.** The prompt instructs the model to extract the essential sentences from document chunks and then to response to the query.

Data Generation: QA Generation Prompt

You are a Question and Answer generation system.
Your task is to create a relevant query and provide a corresponding answer based on the given document chunk.
The query should be concise, clear, and directly relevant to the content of the document chunk.
The answer must be concise, factually grounded by the chunk, and formatted as either a phrase or a single sentence, aligned with one of the following categories:
 	
1. Fact-based: Generate a query that asks for specific details like dates, names, locations, etc., and provide a concise factual answer.
2. Instruction-based: Generate a query asking how to perform an action, and provide a concise step-by-step guide or instruction.
3. Definition or Explanation: Generate a query asking for a brief definition or explanation of a term or concept, and provide a clear explanation.
4. Opinion: Formulate a query that seeks advice or a recommendation based on the document content, and provide a brief opinion or recommendation.
5. Yes/No: Create a yes/no question based on the document chunk and answer it with "Yes" or "No."
 	
Your output must include a single query and its corresponding answer in JSON format:
{
"query": "your query belong to the five categories",
"answer": "your answer"
}
 	
### Document Chunk: {target chunk} 	
 	
### JSON Output: 	

Table 18: **QA Generation Prompt.** The prompt instructs the model to generate one of five QA types, including fact-based, instruction-based, explanation, opinion, and binary QAs.

Filtering: Answer–Chunk Validity Check Prompt

You are responsible for evaluating whether the provided answer to the query can be derived from the given chunk.

Instructions:
1. Analyze the provided answer in response to the query, using the information available in the chunk.
2. If the answer can be fully derived from the chunk, respond with "Supported".
3. If the answer cannot be fully derived from the chunk, respond with "Not Supported".

Your output must be in JSON format. The output should be a dictionary whose a single key is "response".
{
"response": "Supported",
}

Query: {query}

Answer: {answer}

Chunk: {relevant chunk}

JSON Output:

Table 19: **Answer–Chunk Validity Check Prompt.** For "answer validity ," if the answer is not supported by the relevant chunk, the corresponding QA pair is removed. For "noisy chunk validity," if the answer is supported by a noisy chunk, that noisy chunk is removed from the noisy chunk set.

LLMEval Prompt

Your task is to evaluate the correctness of the predicted answer based on the true answer.

Instructions:
- Read the QUERY and then compare the ANSWER and the Predicted ANSWER.
- Check if the Predicted Answer includes the core content of the True Answer (True/False in text).

QUERY: {query}

TRUE ANSWER: {true answer}

Predicted ANSWER: {predicted answer}

Output Format: { "Correctness": "True or False" }

Output (Only JSON):

Table 20: **LLMEval Prompt.** This prompt is used to verify the faithfulness of the generated answer and the correctness of relevant and irrelevant chunks.