# Motivation and Significance

This paper addresses a crucial bottleneck in Multi-Vector Retrieval (MVR) and Retrieval-Augmented Generation (RAG): **query decomposition**. Traditional approaches either decompose queries at the token level (e.g., **ColBERT**), where finer-grained matching leads to semantic misalignment, or rely on manually crafted prompts for decompositions. By analyzing the case where “Hong Kong” is mistakenly decomposed into “Hong” and “Kong”, the paper illustrates the limitations of existing methods: erroneous granularity and redundant sub-queries (e.g., “type”) can place irrelevant documents or images to higher ranks, thereby directly undermining downstream generation quality. The core idea of the paper is to automatically generate sub-queries from a query consisting of tokens, such approach is naturally well-suited for Multi-hop QA tasks. However, this idea has two major challenges: (1) since sub-queries are discretely generated with a LLM, the search process is non-differentiable, as sub-queries cannot propagate gradients from the downstream generation feedback and (2) evaluating candidate sub-queries requires training the downstream model, which is too expensive. Based on these challenges, the paper proposes **Performance-Oriented Query Decomposer for Multi-vector retrieval** (POQD): sub-queries are generated by query decomposer, while an LLM-based optimizer searches and iteratively refines prompts. Besides to evaluate the candidate prompt with its induced sub-queries, a training algorithm is proposed to alternatively refine the prompt while only training the model. This method is theoretically grounded and demonstrates substantial improvements over existing approaches across Image-QA and Document QA in both retrieval performance and end-to-end QA accuracy.

# Methodology

**POQD**

Input: Training query set , generator and retriever’s parameter and initial prompt

Output: new prompt and decomposed sub-queries

**Step 1**

**Abstract**: *Prompt Optimizer generates prefix to construct candidate prompt .*

The Prompt Optimizer generates a prompt prefix (e.g., *Design a query decomposition framework that ……*) based on two meta-prompts, which comprise question description and other instructions and current LS (*Solution-Score Pairs, few-shot and corresponding scores as an optimization problem description*).

Then prefix is concatenated with a fixed template (meta-prompts and input query ) to form the complete prompt for Query Decomposer. These are provided to the optimizer with LS altogether. As iterations progress, LS is continuously appended the new LS pair, enabling the optimizer to learn which tend to yield lower loss.

**Step 2**

**Abstract**: *Using candidate to decompose , and then re-evaluate*

The Query Decomposer generates sub-queries for each query in under . And then these sub-queries are used for MVR to compute , finally appending in LS for use in the next iteration of Step 1. The above two steps alternate in a loop: Step 1 generates new candidate prompts, Step 2 evaluate quality.

**Early Stop**

The loop will stop when , or when the loop has repeated times.

**End-to-End Algorithm**

While POQD framework optimizes the decomposition prompt under fix parameter , end-to-end algorithm embeds POQD into a larger alternating optimization circle. First using previous method to find a better prompt, then training parameter for several epochs under this prompt and so forth. Finally it fixes prompt and trains parameter to convergence. *In this section, only generator’s parameter is trained because updating retriever modals requires rebuilding indexes and re-embedding corpus.*

**Step 1**

Initialize one random , and recall POQD, finally get and its optimized sub-queries. When and are equal, terminating the loop.

**Step 2**

Parameter is trained for only steps using the Step 1’s result (a little warmup).

**Step 3**

Using to replace , and return to Step 1. When the prompt can no longer be improved, fix the final prompt and train to convergence, taking as the final model.

In addition, some assumptions like PL condition and L-smoothness provides solid evidences. Briefly, it could conclude that when is too small, queries are almost always decomposed, leading to redundancy, when it is too large, queries are rarely decomposed, resulting in insufficient coverage. ensures the optimizer does not excessively rewrite, maintaining semantic consistency. And theoretical analysis shows that excessive leads noise, degrading accuracy.

# Pros and Cons

The strengths of this work are as follows: First the motivation is clear and compelling. The “Hong Kong” to “Kong” intuitively illustrates the limitations of token-level matching and underscores the necessity of query decomposition at the phrase level. Second, the methodology is systematic and operational, embedding non-differential prompt search into training process while incorporating hallucination filtering and alternating optimization to balance performance and training cost. Third, the theoretical analysis provides formal convergence guarantees, not just purely empirical validation. Fourth, the entire experiments are comprehensive, involving multiple datasets and models, and demonstrating consistent improvements in retrieval accuracy and downstream QA performance, with less inference cost. Finally, the paper emphasizes reproducibility, offering ablation studies and code links in the appendix, facilitating reproduction and improvement in engineering implementation.

However, there are still some limitations. The framework relies heavily on powerful LLMs for decomposition and optimization (GPT-4 and DeepSeek-v3), which may limit its usage in resource-constrained or on-premise deployment scenarios due to cost and stability issues. What’s more, theoretical guarantees rest on strong assumption like PL condition and L-smoothness, whose validity under different optimizers or data distributions requires further empirical support. In addition, experiments are primarily conducted on QA/RAG tasks, leaving its generalization to long-document retrieval, complex knowledge bases, or cross-modal settings unverified. Despite hallucination filtering, the stability of the LLM-based decomposition process remains a concern. The method is also sensitive to hyperparameters, requiring careful tuning. Lastly, the evaluation of retriever is limited, the performance advantages may partly depend on specific retriever choices, needing more comparisons.

# Future Direction

The core contribution of POQD is introducing a query decomposition pipeline to address the structural limitations of traditional token-level matching. Based on this, POQD integrates a Prompt Optimizer and a Query Decomposer to jointly tackle this challenge. But optimization is mainly used on the query side. On the document side, evidence is still represented as whole embeddings or fixed segments, without comparable decomposition. This design leads to two flaws: (1) in long documents, noise or irrelevant content may obscure key signals, such that even well-decomposed queries fail to align with the most critical fragments. (2) text-image matching often operates across granularities, yet fixed-granularity evidence cannot align effectively with decomposed queries. *So decomposing not only queries but also documents/images and to jointly learn cross-side alignment is an interesting direction, let’s say idea as Bilateral Collaborative Decomposition (BCD).*

The core idea of BCD is, in query side, following POQD, complex queries are decomposed into semantic sub-queries. In document side, documents and images are also decomposed into semantic segments.

1. Text: sentences, triplets, entity descriptions
2. Image: object detection boxes, OCR text, local patches

BCD framework should use an alignment layer to align the granularity between query and document, which learns a differentiable alignment matrix , replacing MaxSim matching to enable one-to-many or many to many alignments. There should also has an end-to-end collaborative optimization, during training, query decomposition and document decomposition are alternately updated, with the alignment layer serving as an intermediary to ensure gradual adaptation on both sides, *decomposition-alignment-retrieval-generation*. Because of its differentiable feature, an advanced end-to-end algorithm could also be considered in this idea. Generally speaking, this updates from *decompose one side, keep the other coarse* to *decompose both sides, align both sides* at the same time, enabling complex queries to be mapped to structured evidence within complex documents.

This idea has several advantages, first more fine-grained semantic alignment, BCD enables structured sub-queries to doc sub-unit matching, improving retrieval interpretability. Second, mitigating MaxSim weakness, since MaxSim is vulnerable to the “Hong Kong to Kong black patch” phenomenon. While bilateral decomposition + learnable alignment avoids single-pillar support, allowing multiple pieces of evidence to jointly support queries. Third, generalizing to cross-modal tasks, queries can be text while documents can be images, tables, or knowledge graphs, as long as decomposition methods are defined, BCD should naturally extend to cross-modal retrieval and QA tasks. Fourth, end-to-end trainability, both collaborative decomposition and alignment are within the gradient calculation, enabling joint optimization with downstream QA/RAG tasks.

Of course this idea faces several potential challenges. In academic aspect, query decomposition is mature, but document decomposition is more complex (textual boundaries, insufficient semantic nature of image patches), potentially generating noise.

And MaxSim has concise upper bound properties, while learnable alignment introduces factors like softmax and redundancy penalties mechanism, making convergence more complicated. Not to mention that decomposition across different modalities (text phrases v.s. image region v.s. table cells) may have mismatched granularities, making it challenging to ensure reasonable alignment. Computational cost also needs to concern, bilateral decomposition expands both queries and documents into multiple sub-units, doubling storage and computational. Poor document decomposition quality may actually increase noise. Joint optimization of decomposition and alignment may lead to gradient oscillations.