Multi-view-guided Passage Reranking with Large Language Models

Anonymous ACL submission

Abstract

Recent advances in large language models (LLMs) have shown impressive performance in *passage reranking* tasks. Despite their success, LLM-based methods still face challenges in *efficiency* and *sensitivity to external biases*. (i) Existing models rely mostly on autoregressive generation and sliding window strategies to rank passages, which incurs heavy computational overhead as the number of passages increases. (ii) External biases, such as positional or semantic bias, hinder the model's ability to accurately represent passages and the input-order sensitivity. To address these limitations, we introduce a novel passage reranking model, called ***M**ulti-**V**iew-guided **P**assage Reranking (**MVP**)*, MVP is a non-generative LLM-based reranking method that encodes query–passage information into diverse view embeddings without being influenced by external biases. For each view, it combines query-aware passage embeddings to produce a distinct anchor vector, used to directly compute relevance scores in *a single decoding step*. Besides, it employs an orthogonal loss to make the views more distinctive. Extensive experiments demonstrate that MVP, with just 220M parameters, matches the performance of much larger 7B-scale fine-tuned models while achieving a 100× reduction in inference latency. Notably, the 3B-parameter variant of MVP achieves state-of-the-art performance on both in-domain and out-of-domain benchmarks.

1 Introduction

Passage reranking aims to assign fine-grained relevance scores to candidate passages – typically retrieved by a first-stage retriever (Robertson et al., 1994; Karpukhin et al., 2020) – by harnessing the language understanding capabilities of large language models (LLMs), in both zero-shot and fine-tuned settings. Recent studies (Sun et al., 2023; Liang et al., 2023) formulate a prompt that consists of a query and candidate passages and generate an ordered list of passage identifiers in a zero-shot setting. Subsequent work has fine-tuned open-source LLMs by distilling knowledge from the teacher model (Pradeep et al., 2023a,b), achieving competitive performance.

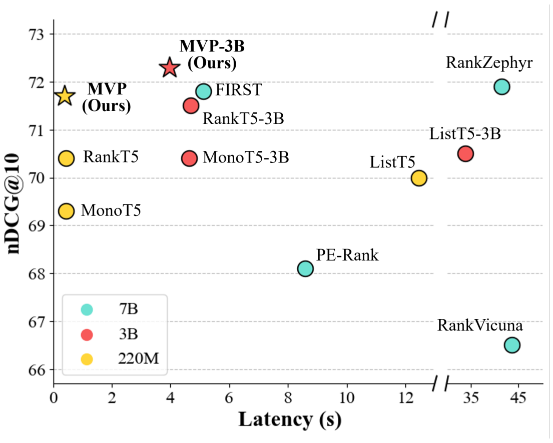

Figure 1: Comparison of latency and nDCG@10 across various reranking models. Latency refers to the time required to rerank for a single query and nDCG@10 is averaged over DL19 and DL20.

Despite their success, LLM-based reranking methods still face challenges in *efficiency* and *sensitivity to input order*. Specifically, we address two key issues for designing an efficient and effective LLM-based reranker.

(i) *How do we perform reranking without incurring unnecessary inference?* Efficient reranking hinges on two key aspects: **global ranking** (evaluating all candidates at once) and **single pass decoding** (performing reranking with a single decoding step). First, existing methods (Pradeep et al., 2023a,b) are unable to include all candidate passages in a single prompt due to input length limitations, leading to rely on sliding-window algorithms, as illustrated in Figure 2(a). Next, generative rerankers employ autoregressive decoding, generating one passage identifier at a time, which leads to substantial computational overhead in Figure 2(b).

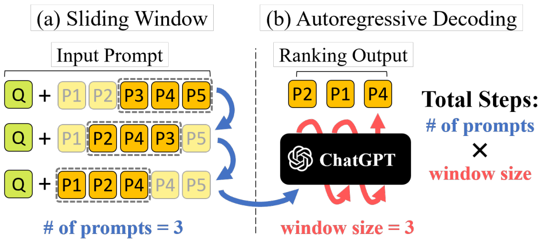

Figure 2: Inference pipeline of a generative listwise reranker. The total number of inferences is determined by the product of (i) the number of prompts and (ii) the window size required to evaluate all candidate passages.

(ii) *How do we represent query-passage explicitly without introducing bias?* While LLMs show strong zero-shot reranking performance, the unbiased modeling of query-passage relationships remains an underexplored challenge due to common biases. First, **positional bias** emerges in long-context prompts, a problem known as *lost-in-the-middle* (Liu et al., 2024a). Second, **semantic bias** arises when natural language tokens (e.g., "A", "1") are used as passage identifiers. These identifiers may encode unintended priors, potentially biasing reranking—as observed in multiple-choice settings (Zheng et al., 2024).

To this end, we propose a novel listwise reranking model, ***M**ulti-**V**iew-guided **P**assage reranking (**MVP**)*. It consists of two key components under the Fusion-in-Decoder (FiD) architecture (Izacard and Grave, 2020).

Multi-View Encoding. Each query–passage pair is encoded into learnable soft prompts in the FiD architecture. To eliminate positional bias, soft prompts are inserted at the same fixed positions across all passages. For each passage, distinct positional embeddings are used to separate relevant views. Since these prompts are not tied to any pretrained vocabulary, they allow for unbiased modeling of query-passage relationships. The encoder produces view-specific embeddings, called relevance vectors, which are then passed to the decoder to compute the final relevance scores.

Anchor-Guided Decoding. Our method adopts a non-generative design that leverages anchor vectors for listwise relevance scoring across all candidates within a single decoding step. This approach operates independently from a language modeling (LM) head. During decoding, MVP aggregates view-specific relevance embeddings from all candidate passages using cross-attention in the decoder to produce anchor vectors. This design directly computes similarity-based scores, aligning both training and inference with the ranking objective while substantially improving efficiency.

As illustrated in Figure 1, MVP-3B achieves state-of-the-art performance on in-domain benchmarks (DL19 and DL20). Notably, our 220M-parameter model matches the nDCG@10 of 7B-scale listwise rerankers while reducing inference latency by up to 100×. These results highlight the efficiency and scalability of our reranking approach, demonstrating that high-quality reranking can be achieved without the computational overhead of large scale generative models.

Our contributions are summarized as follows:

- **Efficient Listwise Reranker**: We propose a novel non-generative reranking method named **MVP**, which enables global ranking in a single step.
- **Robustness to External Biases**: Our embedding-based architecture is robust to positional and semantic biases, enabling flexible adaptation to diverse passage input scenarios.
- **Extensive Experiments**: MVP achieves state-of-the-art performance on both in-domain and out-domain benchmarks.

2 Related Work

2.1 Rerranking with LLMs

Recent work has explored leveraging the natural language understanding capabilities of large language models (LLMs) for passage reranking (Zhu et al., 2023; Liang et al., 2023). Depending on the prompting strategy, methods can be categorized into pointwise and listwise approaches. Pointwise rerankers estimate the relevance between a query and a single passage. For example, Some pointwise approaches (Nogueira et al., 2019, 2020; Zhuang et al., 2024) compute relevance scores using the logits of relevance-related tokens such as “Yes” or “No”. Other approaches (Sachan et al., 2022; Zhuang et al., 2023b; Cho et al., 2023) estimate the relevance of a document based on the probability of generating the corresponding query sequence. In contrast, (Xian et al., 2023) demonstrated that listwise reranking methods outperform pointwise approaches by comparing candidate passages at once. Building on this, RankGPT (Sun et al., 2023) employed GPT-4 (OpenAI, 2023) to achieve state-of-the-art zero-shot reranking performance, and later work distilled knowledge into open-source LLMs (Pradeep et al., 2023a,b). While intuitive, this generation-based approach, which introduces

Model	Global Ranking	Single Pass Decoding	Bias Mitigation	
			Positional	Semantic
ListT5	✗	✗	✓	✗
RankZephyr	✗	✗	✓	✗
PE-Rank	✓	✗	✗	✗
FIRST	✗	✓	✗	✗
MVP	✓	✓	✓	✓

Table 1: Comparison of generative LLM rerankers and MVP with respect to bias mitigation, global ranking capability, and generation target.

inefficiencies and hinders alignment with the goals of ranking.

2.2 Generative Reranking with LLMs

To address the limitations discussed in Section 1, various generative reranking methods have been proposed. To mitigate the positional bias, ListT5 (Yoon et al., 2024) leverages the FiD architecture, while RankZephyr (Pradeep et al., 2023b) addresses the issue by shuffling input order and varying the number of passages during training. PE-Rank (Liu et al., 2024b) compresses each passage into a single token, allowing global ranking, but suffers from information loss during compression and projection. FIRST (Reddy et al., 2024) improves efficiency via single-pass decoding using the logits of the first generated token, yet supports neither global ranking nor effective bias mitigation.

While prior work has tackled individual aspects of generation-based reranking, no method has simultaneously achieved (i) mitigation of various biases, (ii) global ranking capability, and (iii) single-pass decoding. A comparison with existing methods is presented in Table 1.

3 Proposed Method

In this section, we propose a novel passage reranking model, ***M**ulti-**V**iew-guided **P**assage reranking (**MVP**)*. It is based on the FiD architecture. As shown in Figure 3, A query-passage pair is encoded into multiple relevance vectors, each capturing a unique relevance signal from a different view (Section 3.1). The decoder generates anchor vectors for each view, which score passages via dot product with their relevance vectors (Section 3.2). Finally, we train the model solely with a ranking objective with an orthogonality regularization term to ensure that anchor vectors remain distinct (Section 3.3).

3.1 Multi-View Encoding

To employ a query-aware passage embedding that summarizes the entire context, we encode each query-passage pair through a set of learnable soft prompts. Given a query q and a set of n candidate passages $[c_1, c_2, \ldots, c_n]$, we construct a distinct input prompt x_i by prepending m view tokens $\langle \mathrm{v}_1 \rangle, \langle \mathrm{v}_2 \rangle, \ldots, \langle \mathrm{v}_m \rangle$ to the query and the i-th passage. The relative positions of these view tokens are fixed across all passages, ensuring that each $\langle \mathrm{v}_k \rangle$ consistently appears at the same location, regardless of the query and passage content. Meanwhile, each view token in x_i is assigned a unique positional embedding, enabling the model to distinguish between views and capture diverse aspects of the query–passage relationship.

$$x_i = \langle \mathrm{v}_1 \rangle \cdots \langle \mathrm{v}_m \rangle \mid \text{Query: } q \mid \text{Context: } c_i \quad (1)$$

The FiD encoder processes constructed input x_i to obtain hidden states H_i. From these hidden states, we extract the vectors corresponding to each special token $\langle \mathrm{v}_k \rangle$, denoted as e_{ik}, representing distinct views of query-passage relevance:

$$H_i = \mathrm{FiD}_{\mathrm{encoder}}(x_i), \quad H_i \in \mathbb{R}^{L \times d} \quad (2)$$

$$e_{ik} = H_i[\langle \mathrm{v}_k \rangle] \quad \text{for } k = 1, \ldots, m \quad (3)$$

Consequently, each candidate passage c_i is compressed into a set of m relevance vectors, $e_{i1}, e_{i2}, \ldots, e_{im}$. The integration of the FiD architecture and position-controlled soft prompts effectively eliminates both positional and semantic biases, enabling robust and view-specific encoding of query–passage interactions.

3.2 Anchor-Guided Decoding

To minimize the computational overhead of sequential generation, MVP adopts anchor guided decoding. Specifically, MVP generates multiple anchor vectors by applying cross-attention over all candidate relevance vectors in the decoder, enabling single-pass inference and global ranking without autoregressive decoding.

Each anchor, derived from the relevance vectors corresponding to each view, represents a distinct perspective of relevance. Specifically, given n candidate passages and their k-th view relevance vectors $e_{1k}, \ldots, e_{nk}$, we construct a matrix $E_k \in \mathbb{R}^{n \times d}$ as the key-value input to the decoder. E_k is then transformed into an anchor vector a_k via cross-attention.

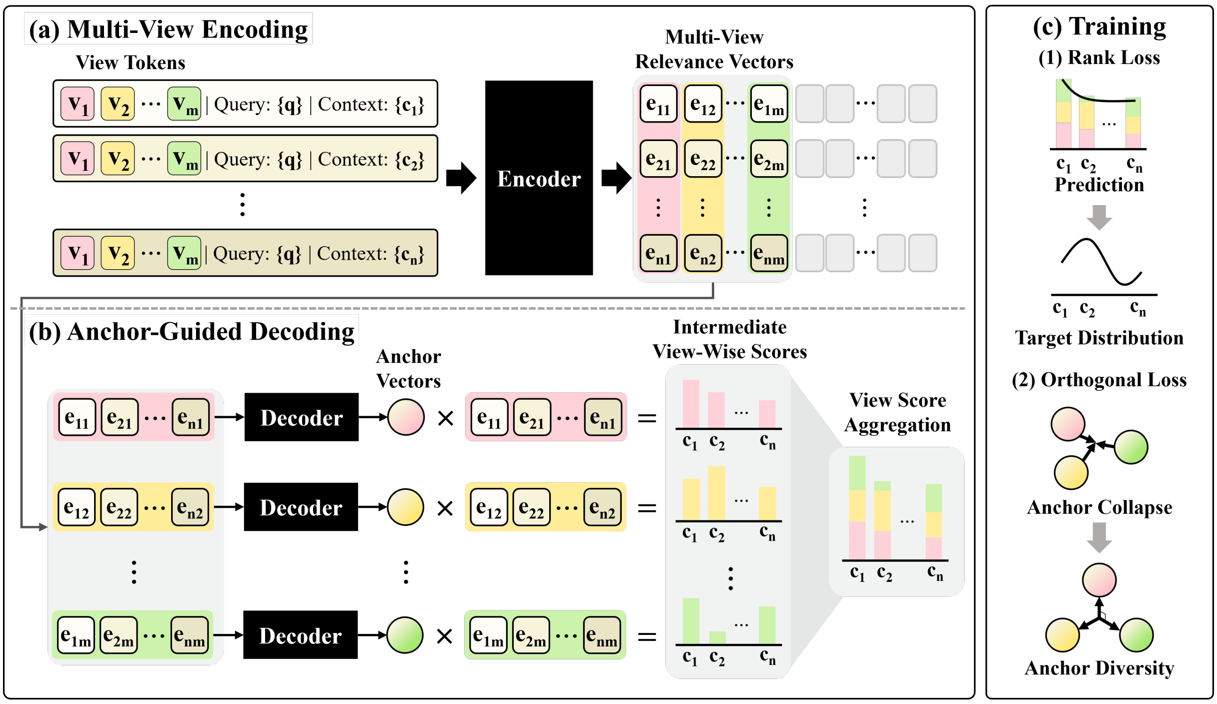
(a) Multi-View Encoding
View Tokens
V_1 V_2 ··· V_m | Query: {q} | Context: {c_1}
V_1 V_2 ··· V_m | Query: {q} | Context: {c_2}
V_1 V_2 ··· V_m | Query: {q} | Context: {c_n}
Encoder
Multi-View Relevance Vectors
e_{11} e_{12} ··· e_{1m}
e_{21} e_{22} ··· e_{2m}
e_{n1} e_{n2} ··· e_{nm}
(b) Anchor-Guided Decoding
Decoder
Anchor Vectors
e_{11} e_{21} ··· e_{n1}
e_{12} e_{22} ··· e_{n2}
e_{1m} e_{2m} ··· e_{nm}
×
=
Intermediate View-Wise Scores
c_1 c_2 c_n
View Score Aggregation
(c) Training
(1) Rank Loss
Prediction
Target Distribution
(2) Orthogonal Loss
Anchor Collapse
Anchor Diversity

Figure 3: The overall framework of **MVP**. (a) A query-passage pair is encoded into multiple relevance vectors, where each vector represents a distinct view. (b) For every view, an anchor vector is generated, and the view-wise relevance score is computed based on its similarity to the corresponding relevance vector. The final score is obtained by aggregating scores across all views. (c) The model is trained with a ranking loss to match the target distribution and an orthogonality loss to encourage diversity among anchor vectors.

$$E_k = [e_{1k}; e_{2k}; \dots; e_{nk}] \in \mathbb{R}^{n\times d} \tag{4}$$

$$a_k = \mathrm{FiD}_{\mathrm{decoder}}([\mathrm{BOS}], E_k) \in \mathbb{R}^{1\times d} \tag{5}$$

The relevance score from each view is computed by measuring similarity between a relevance vector e_{ik} and its anchor a_k, and the final score s_i is obtained by averaging across all m views:

$$s_i = \frac{1}{m}\sum_{k=1}^{m}\langle a_k, e_{ik}\rangle \tag{6}$$

By utilizing multiple anchors, the model effectively evaluates candidate passages from diverse semantic views, enabling efficient and accurate scoring without the need for ranking list generation. Importantly, this direct scoring mechanism removes the need to compute token-level logits, enabling both training and inference to rely solely on relevance-based objectives.

3.3 Training

Training MVP involves optimizing two complementary objectives that jointly enhance ranking accuracy and representational diversity. The first is a ranking objective that enables the model to learn the relevance ordering of candidate passages (Section 3.3.1). The second is an orthogonality objective that encourages each anchor to capture a distinct perspective on relevance (Section 3.3.2).

3.3.1 Ranking Loss

As the ranking objective to train MVP, we adopt the ListNet loss (Cao et al., 2007), which encourages the predicted ranking scores to align with the ground-truth relevance order. Given n candidate passages and their ground-truth ranks $r_i \in [1, 2, \dots, n]$ (with $r_i = 1$ indicating the most relevant), each rank is converted into a relevance score using a reciprocal transformation, *i.e.*, $y_i = 1/r_i$.

We then apply a temperature-scaled softmax to both ground-truth scores y_i and predicted scores s_i to obtain probability distributions, where τ is a temperature hyperparameter:

$$P(y_i) = \frac{\exp(y_i/\tau)}{\sum_{j=1}^{n}\exp(y_j/\tau)} \tag{7}$$

$$P(s_i) = \frac{\exp(s_i/\tau)}{\sum_{j=1}^{n}\exp(s_j/\tau)} \tag{8}$$

The listwise ranking objective used to approximate the predicted probability for the i-th passage is defined as follows:

$$\mathcal{L}_{\text{Rank}} = -\sum_{i=1}^{n} P(y_i) \log P(s_i) \tag{9}$$

3.3.2 Orthogonal Loss

Since MVP computes the relevance score for each passage by leveraging multiple anchor vectors, each anchor has to capture a distinct and complementary view of the query-passage relationship. To this end, we introduce an orthogonal regularization loss that promotes diversity among anchor vectors, inspired by the Orthogonal Projection Loss (OPL) (Ranasinghe et al., 2021), which encourages orthogonality in feature representations. The loss is defined as:

$$\mathcal{L}_{\text{Orthogonal}} = \sum_{k=1}^{m} \sum_{\substack{l=1 \\ l \neq k}}^{m} [\mathbf{a}_k, \mathbf{a}_l]^2 \tag{10}$$

$$[\mathbf{x}_i, \mathbf{x}_j] = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\|_2 \cdot \|\mathbf{x}_j\|_2} \tag{11}$$

Here, $[\,\cdot\,,\,\cdot\,]$ denotes the cosine similarity operator, and $\|\cdot\|_2$ represents the L2 norm.

This regularization encourages anchor vectors to remain in distinct directions, guiding the encoding stage to capture diverse semantic views across tokens. The final training loss combines the primary ranking loss and the orthogonality regularization term:

$$\mathcal{L} = \mathcal{L}_{\text{Rank}} + \mathcal{L}_{\text{Orthogonal}} \tag{12}$$

4 Experiments

In this section, we first describe the training and evaluation setup for MVP. We then present four main results: (i) overall ranking performance, (ii) efficiency against various reranking models, (iii) robustness to initial order, and (iv) ablation studies on key architectural components. All results for MVP are based on the T5-base model unless otherwise specified as 3B.

4.1 Experimental Setup

Datasets. We evaluated in-domain performance on the TREC-DL19, DL20 (Craswell et al., 2020, 2021a), and assessed zero-shot out-of-domain performance on the BEIR (Thakur et al., 2021) benchmark, which is designed to evaluate the generalization ability of ranking models. Although BEIR comprises eight diverse datasets, we followed prior work (Sun et al., 2023) and conducted evaluations on eight datasets with relatively fewer queries. We employed BM25 as the first-stage retrieval model and measured reranking performance using Normalized Discounted Cumulative Gain at rank 10 (nDCG@10). Note that while we use five passages per query during training, we rerank all candidate passages using single-pass decoding to measure inference time.

Implementation Detail. To train MVP, we utilized the Rank-DistiLLM (Schlatt et al., 2025) dataset, which is constructed from the MS MARCO Passage Ranking dataset (Nguyen et al., 2016) using 10,000 queries. For each query, the top 100 candidate passages were first retrieved using ColBERTv2 (Santhanam et al., 2022), and then re-ranking these passages with the RankZephyr (Pradeep et al., 2023b). To construct a more diverse training set, we sampled 5 candidate passages 100 times per query, resulting in approximately 1 million instances.

We adopted T5-base and T5-3B (Raffel et al., 2020) as our backbone models. For optimization, we applied DeepSpeed Stage 2. For T5-base, we used a batch size of 16, gradient accumulation steps set to 2, a learning rate of 1e-4, and a linear scheduler with a warm-up ratio of 5%. For T5-3B, we used a batch size of 2, gradient accumulation steps of 16, and a learning rate of 1e-5. The maximum input sequence length was fixed at 256 tokens for both models. Training was conducted for a single epoch, taking approximately 5 hours on 2 × NVIDIA RTX 3090 GPUs for T5-base, and 40 hours on 2 × NVIDIA A6000 GPUs for T5-3B. We use $m = 4$ special tokens to represent the relevance views, implemented using the T5 tokenizer's predefined tokens `<extra_id_0>` to `<extra_id_3>`, and set $\tau = 0.8$ to control the sharpness in the ListNet loss. For validation, we use the TREC-DL21 dataset (Craswell et al., 2021b) with nDCG@10 as the validation metric.

4.2 Ranking Performance

We compare MVP against seven reranking models built on the T5 architecture. Specifically, pointwise models are **MonoT5** (Nogueira et al., 2020) and **RankT5** (Zhuang et al., 2023a)). The listwise model is **ListT5** (Yoon et al., 2024). For 7B-scale rerankers, we employ **RankVicuna** (Pradeep et al., 2023a), **RankZephyr** (Pradeep et al., 2023b), **FIRST** (Reddy et al., 2024), and **PE-Rank** (Liu et al., 2024b). Note that MVP is based on 220M

Model	DL19	DL20	Covid	NFCorpus	Signal	News	Robust04	SciFact	Touche	DBPedia	BEIR Avg.
MonoT5 (220M)	71.5	67.0	78.3	35.7	32.0	48.0	53.4	73.1	29.6	42.8	49.1
RankT5 (220M)	72.4	68.3	77.7	35.1	30.8	45.4	54.3	73.5	37.1	43.7	49.7
ListT5 (220M)	71.8	68.1	78.3	35.6	**33.5**	48.5	52.1	74.1	33.4	43.7	49.9
MVP (220M)	**74.3**	**69.2**	**80.2**	**36.0**	32.7	**49.1**	**55.1**	**75.0**	**39.1**	**43.8**	**51.4**
MonoT5 (3B)	71.8	68.9	79.8	37.3	32.2	48.3	58.5	76.3	32.5	44.8	51.2
RankT5 (3B)	72.5	70.4	81.7	37.4	31.9	49.5	58.3	**77.1**	**38.8**	45.0	52.5
ListT5 (3B)	71.8	69.1	**84.7**	**37.7**	33.8	**53.2**	57.8	77.0	33.6	46.2	53.0
FIRST (7B)	72.4	**71.1**	82.4	36.3	34.0	52.4	54.6	75.0	38.0	46.3	52.6
PE-Rank (7B)	70.8	65.4	77.8	34.8	32.0	52.3	48.7	70.2	34.2	40.6	49.0
RankVicuna (7B)	66.5	66.4	79.5	32.5	33.3	45.0	47.0	68.8	32.9	44.5	48.1
RankZephyr (7B)	73.1	70.8	83.2	36.3	31.5	52.5	54.3	74.9	32.4	44.5	51.4
MVP (3B)	**73.5**	**71.1**	83.1	37.6	**34.2**	51.2	**60.5**	76.4	37.2	**46.6**	**53.3**

Table 2: Results (nDCG@10) of reranking top-100 passages on TREC and BEIR benchmarks. The initial candidate passages are retrieved using BM25. The best-performing model in each section is highlighted in **bold**, and the second-best is marked with underline.

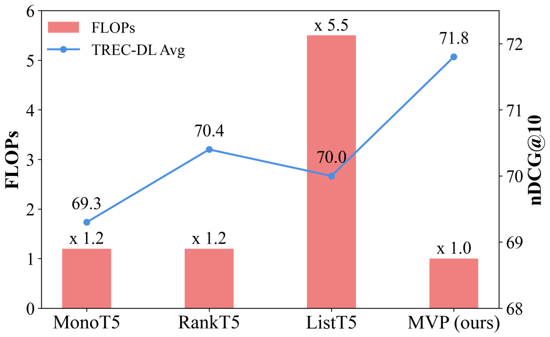

Figure 4: Real-time FLOPs comparison of the models. The reported performance is averaged over DL19 and DL20.

and 3B base models.

Table 2 reports the overall result. When evaluated at the T5-base scale, MVP outperforms other baselines of the same model size across most datasets. On the BEIR benchmark, MVP achieves an average nDCG@10 score of 51.4, surpassing MonoT5, RankT5, and ListT5 by 2.3, 1.7, and 1.5 points. This performance is also comparable to that of RankZephyr (7B), a much larger model. On the TREC-DL19 and DL20 datasets, MVP also exceeds RankT5 by 1.9 and 0.9 points, respectively.

We also compare the 3B variant of MVP with large-scale (3B-7B) LLM-based re-ranking models. MVP-3B achieves nDCG@10 scores of 73.5 on DL19, 71.1 on DL20, and 53.3 on the BEIR average, outperforming all other models at the 3B and 7B scales. These results suggest that the architectural advantages of MVP generalize well to larger model configurations.

4.3 Efficiency

The key strength of MVP lies in its ability to represent each query-passage pair with multiple relevance vectors and to perform anchor-guided decoding, achieving both high effectiveness and significantly improved efficiency. To empirically validate efficiency, we report both floating-point operations (FLOPs) and latency. All experiments are conducted on a 24GB NVIDIA RTX 3090 GPU.

FLOPs. To assess the computational efficiency of each model, we measured FLOPs using DeepSpeed's FLOPs Profiler[1]. The evaluation was conducted on 43 queries from the DL19 dataset. For each query, we measured the FLOPs required to rerank the top 10 passages from BM25-Top100 candidates. The input sequence length was set to 256 tokens. For ease of comparison, we normalized MVP's FLOPs to 1.0[2] with the relative FLOPs of other models computed accordingly. As illustrated in Figure 4, MVP achieves the lowest computational cost among all models while outperforming them in ranking quality. Compared to the previous listwise model ListT5, it reduces FLOPs by approximately 82%. Notably, MVP also consumes fewer operations than pointwise models such as MonoT5 and RankT5, despite delivering stronger reranking performance.

Latency. We also measure the latency required to rerank the top-10 passages from BM25-Top100 candidate passages. Latency is defined as the average time per query, measured in seconds. Our experiments are conducted on DL19 and DL20, along

[1] We use DeepSpeed's FLOPs profiler for measurement: https://github.com/microsoft/DeepSpeed

[2] The exact FLOPs value is 197,983,445,625,792.

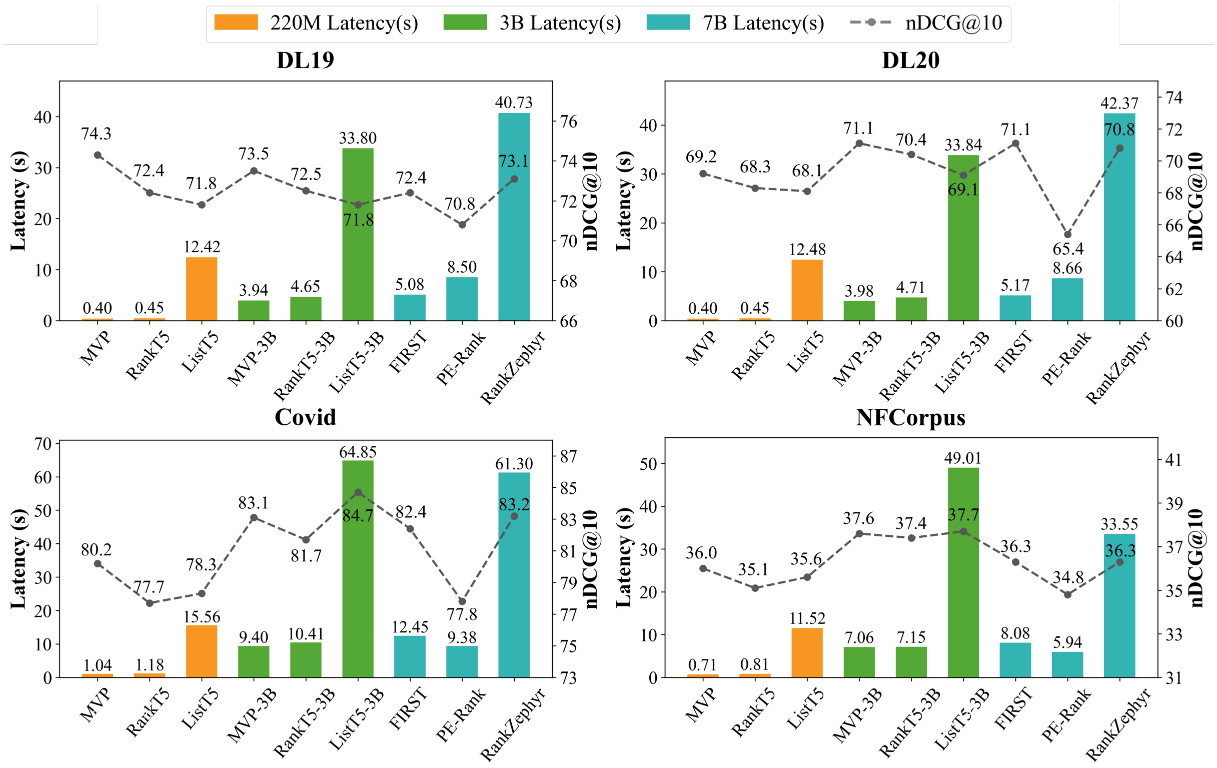

Figure 5: Ranking performance (nDCG@10) for the reranker's latency (s). Latency indicates the average time required to rerank a single query.

with two datasets from the BEIR benchmark: Covid and NFCorpus. For fair comparison, all vLLM acceleration features are disabled, ensuring that the latency reflects the raw inference time of each model.

The results in Figure 5 show that MVP achieves faster inference than existing listwise models across all datasets, even surpassing the pointwise model RankT5. Specifically on DL19, it achieves 100× faster than RankZephyr and 12.7× faster than FIRST, while maintaining comparable ranking performance. At the larger scale (3B-7B), MVP-3B offers a favorable trade-off between latency and ranking performance. Notably, even compared to FIRST, a well-balanced 7B model, MVP-3B achieves both faster inference and better reranking accuracy.

In summary, the FLOPs and latency results confirm that MVP is both efficient and effective for real-time reranking. The scoring strategy of MVP enables simultaneous evaluation of all candidates without repeated decoding, eliminating redundancy and supporting strong ranking performance.

4.4 Robustness to Initial Order

Most listwise reranking models are sensitive to the initial passage order due to two main factors: (1) positional bias, and (2) semantic bias introduced by using natural language tokens (e.g., "1", "2") as passage identifiers. To evaluate whether our model effectively mitigates these biases, we conduct experiments on the DL19, DL20, and News datasets.

We consider two passage reordering settings: (1) Shuf., the same set of candidate passages is randomly shuffled using three different seeds, and the results are reported as averaged values; and (2) Rev., where the input order is simply reversed. The results are summarized in Table 3.

This robustness indicates that MVP effectively mitigates positional and semantic biases, which can be attributed to two key design choices: (1) each passage is encoded independently, resulting in shared positional information across all passages; and (2) instead of using unique identifier tokens for each passage, MVP encodes all passages using the same view token, eliminating semantic bias.

4.5 Ablation Study

To evaluate the impact of key architectural components on model performance, we design several model variants and perform ablation studies.

Initial Order	DL19	DL20	News	Average
MVP				
BM25	74.3	69.2	49.1	64.2
Shuf. BM25	74.3	69.2	49.1	64.2 (±0.0)
Rev. BM25	74.3	69.2	49.1	64.2 (±0.0)
ListT5 (*ts*: m=5, r=2)				
BM25	71.8	68.1	48.5	62.8
Shuf. BM25	71.2	68.2	48.6	62.7 (–0.1)
Rev. BM25	71.2	67.8	48.5	62.5 (–0.3)
RankZephyr (*sw*: w=20, s=10)				
BM25	73.1	70.8	52.5	65.5
Shuf. BM25	73.1	70.7	51.3	65.0 (–0.4)
Rev. BM25	72.1	71.5	51.8	65.1 (–0.3)
FIRST (*sw*: w=20, s=10)				
BM25	72.4	71.1	52.4	65.3
Shuf. BM25	70.0	69.4	47.3	62.2 (–3.1)
Rev. BM25	67.5	68.3	42.4	59.4 (–5.9)
PE-Rank (*sw*: w=20, s=10)				
BM25	70.8	65.4	52.3	62.8
Shuf. BM25	66.0	58.5	46.8	57.1 (–5.7)
Rev. BM25	67.5	59.1	46.5	57.7 (–5.1)

Table 3: nDCG@10 scores under different initial input orders. Values in parentheses indicate change relative to the BM25 ranking order. *ts* denotes tournament sort and *sw* denotes sliding window. For each sorting algorithm, the basic operating unit ($m \rightarrow r$), window size (w), and stride (s) are set according to prior work.

Model	DL19	DL20	BEIR Avg.
MVP	**74.3**	**69.2**	**51.4**
w/o $\mathcal{L}_{\text{Orthogonal}}$	73.6	66.7	50.7
w/o Multi-view Encoding	73.8	68.8	50.9

Table 4: nDCG@10 for MVP and its ablations on different training strategies. See Table 10 for full BEIR results.

4.5.1 Training Strategies

To investigate the impact of each component in MVP, we perform ablation experiments by removing two key design elements: (i) orthogonality regularization among anchors and (ii) the use of multi-view encoding. The results are reported in Table 4.

w/o Orthogonality. Removing the orthogonality regularization among anchor vectors consistently degrades performance across datasets. This suggests that, in the absence of this constraint, different anchors tend to collapse into similar directions within the embedding space, leading to redundant rather than complementary representations. A detailed analysis of anchor vector similarities is provided in Appendix C.1.

w/o Multiple Token. Using a single special token to represent relevance results in a 0.4–0.5 point drop in performance on average. This degradation is attributed to the limited capacity of a single token to capture both query and passage information, leading to a loss of discriminative features.

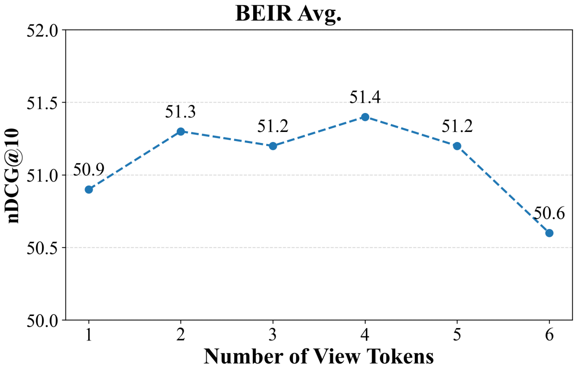

Figure 6: Average nDCG@10 on BEIR with respect to the number of view tokens.

4.5.2 Number of View Tokens

To analyze the impact of the number of view tokens on model performance, we varied the number of relevance tokens from 1 to 6 and evaluated the average performance across BEIR datasets. As illustrated in Figure 6, incorporating multiple views leads to improved performance up to a certain point, beyond which performance begins to degrade. This suggests that, while orthogonality regularization encourages representational diversity, an excessive number of view tokens may introduce less informative signals, thereby degrading ranking performance.

5 Conclusion

We presented **MVP**, a novel passage reranking model that addresses key limitations of listwise LLM-based approaches, including high computational cost and sensitivity to input order. By leveraging multi-view encoding through soft prompts and multi-anchor guided decoding, MVP captures diverse relevance signals efficiently via compact context embeddings, enabling all candidate passages to be evaluated in a single pass—making it particularly well-suited for real-world retrieval scenarios. Experimental results show that MVP, with only 220M parameters, matches or surpasses the performance of 7B-scale models while reducing inference latency up to 100× . Moreover, its 3B variant achieves state-of-the-art results on both in-domain and out-of-domain benchmarks.

6 Limitations

While MVP employs a fixed number of views across all datasets—a simple and generally effective strategy—using fewer views in some cases can reduce redundancy and improve performance. In addition, MVP aggregates relevance scores by assigning equal weights to all views. Although this uniform aggregation is straightforward, it may overlook the fact that different queries can benefit more from certain views than others. Exploring dynamic view selection or learning query-specific view weights remains a promising direction for future work.

Ethics Statement

This work fully respects ACL's ethical guidelines. We have utilized scientific resources available for research under liberal licenses, and our use of these tools is consistent with their intended applications.

References

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, pages 129–136.

Sukmin Cho, Soyeong Jeong, Jeongyeon Seo, and Jong C. Park. 2023. Discrete prompt optimization via constrained generation for zero-shot re-ranker. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 960–971. Association for Computational Linguistics.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021a. Overview of the TREC 2020 deep learning track. *CoRR*, abs/2102.07662.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Jimmy Lin. 2021b. Overview of the TREC 2021 deep learning track. In *Proceedings of the Thirtieth Text REtrieval Conference, TREC 2021, online, November 15-19, 2021*, volume 500-335 of *NIST Special Publication*. National Institute of Standards and Technology (NIST).

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2019 deep learning track. *CoRR*, abs/2003.07820.

Sebastian Hofstätter, Jiecao Chen, Karthik Raman, and Hamed Zamani. 2023. Fid-light: Efficient and effective retrieval-augmented text generation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*, pages 1437–1447. ACM.

Gautier Izacard and Edouard Grave. 2020. Leveraging passage retrieval with generative models for open domain question answering. *CoRR*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel J. Orr, Lucia Zheng, Mert Yüksekgönül, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri S. Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. 2023. Holistic evaluation of language models. *Trans. Mach. Learn. Res.*, 2023.

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. *Trans. Assoc. Comput. Linguistics*, pages 157–173.

Qi Liu, Bo Wang, Nan Wang, and Jiaxin Mao. 2024b. Leveraging passage embeddings for efficient listwise reranking with large language models. *CoRR*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*, volume 1773 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with BERT. *CoRR*, abs/1910.14424.

Rodrigo Frassetto Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pretrained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, pages 708–718.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023a. Rankvicuna: Zero-shot listwise document reranking with open-source large language models. *CoRR*.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023b. Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze! *CoRR*, abs/2312.02724.

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. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Kanchana Ranasinghe, Muzammal Naseer, Munawar Hayat, Salman H. Khan, and Fahad Shahbaz Khan. 2021. Orthogonal projection loss. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 12313–12323. IEEE.

Revanth Gangi Reddy, JaeHyeok Doo, Yifei Xu, Md. Arafat Sultan, Deevya Swain, Avirup Sil, and Heng Ji. 2024. FIRST: faster improved listwise reranking with single token decoding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 8642–8652. Association for Computational Linguistics.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In *Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994*, volume 500-225 of *NIST Special Publication*, pages 109–126. National Institute of Standards and Technology (NIST).

Devendra Singh Sachan, Mike Lewis, Mandar Joshi, Armen Aghajanyan, Wen-tau Yih, Joelle Pineau, and Luke Zettlemoyer. 2022. Improving passage retrieval with zero-shot question generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 3781–3797. Association for Computational Linguistics.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. Colbertv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 3715–3734. Association for Computational Linguistics.

Ferdinand Schlatt, Maik Fröbe, Harrisen Scells, Shengyao Zhuang, Bevan Koopman, Guido Zuccon, Benno Stein, Martin Potthast, and Matthias Hagen. 2025. Rank-distillm: Closing the effectiveness gap between cross-encoders and llms for passage re-ranking. In *Advances in Information Retrieval - 47th European Conference on Information Retrieval, ECIR 2025, Lucca, Italy, April 6-10, 2025, Proceedings, Part III*, volume 15574 of *Lecture Notes in Computer Science*, pages 323–334. Springer.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is chatgpt good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 14918–14937.

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

Ruicheng Xian, Honglei Zhuang, Zhen Qin, Hamed Zamani, Jing Lu, Ji Ma, Kai Hui, Han Zhao, Xuanhui Wang, and Michael Bendersky. 2023. Learning list-level domain-invariant representations for ranking. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Soyoung Yoon, Eunbi Choi, Jiyeon Kim, Hyeongu Yun, Yireun Kim, and Seung-won Hwang. 2024. Listt5: Listwise reranking with fusion-in-decoder improves zero-shot retrieval. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 2287–2308.

Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. 2024. Large language models are not robust multiple choice selectors. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Zhicheng Dou, and Ji-Rong Wen. 2023. Large language models for information retrieval: A survey. *CoRR*, abs/2308.07107.

Honglei Zhuang, Zhen Qin, Kai Hui, Junru Wu, Le Yan, Xuanhui Wang, and Michael Bendersky. 2024. Beyond yes and no: Improving zero-shot LLM rankers via scoring fine-grained relevance labels. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Short Papers, NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 358–370. Association for Computational Linguistics.

Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. 2023a. Rankt5: Fine-tuning T5 for text ranking with ranking losses. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*, pages 2308–2313.

Shengyao Zhuang, Bing Liu, Bevan Koopman, and Guido Zuccon. 2023b. Open-source large language models are strong zero-shot query likelihood models for document ranking. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 8807–8817. Association for Computational Linguistics.

Model	Training Data	DL19	DL20
MVP	RankDistiLLM	**74.3**	**69.2**
ListT5	RankDistiLLM	72.5	68.5

Table 5: nDCG@10 results comparing MVP and a ListT5 variant trained on RankDistiLLM data, using tournament sort

A Implementation Details

A.1 Passage Length Configuration

During Inference, We follow the passage length configuration from ListT5 (Yoon et al., 2024), where the maximum passage length for each dataset is selected from 256, 512, 1024 based on the average number of tokens in the query-passage pair. For the *signal* dataset, however, we use a smaller maximum length of 128, considering its short input length. We found that this reduced setting did not negatively impact performance. The final maximum input lengths used for each dataset are summarized as follows:

‘dl19’: 256, ‘dl20’: 256, ‘trec-covid’: 512, ‘nf-corpus’: 512, ‘signal’: 128, ‘news’: 1024, ‘robust04’: 1024, ‘scifact’: 512, ‘touche’ : 1024, ‘dbpedia-entity’ : 256

B Additional Experiments

B.1 Comparison with Generation-Based Re-ranking

To further validate our approach, we trained the ListT5 framework on our dataset. Following prior work (Yoon et al., 2024), the model was configured to take 5 passages as input and generate the top 2 passages. Results are shown in Table 5.

Despite being trained on the same dataset, our Anchor based relevance estimation with multi-view representation and reranking approach consistently outperformed the generation-based model. We attribute this performance gap to two main factors: (1) generation-based models are trained with language modeling objectives, which are not inherently aligned with ranking tasks, and (2) our method evaluates relevance from multiple perspectives and aggregates the results, enabling more accurate and robust ranking estimation.

B.2 Designs for View Tokens

To analyze the impact of view token design on reranking performance, we compare three alternative configurations: (1) First 4 Tokens: Following the

	EXTRA_ID	FIRST 4	Numeric	Alphabetic
DL19	**74.3**	74.2	73.4	73.2
DL20	**69.2**	68.7	67.7	68.1
COVID	**80.2**	78.4	78.5	78.8
NFCorpus	**36.0**	35.7	35.5	35.3
Signal	**33.0**	32.2	32.0	33.0
NEWS	49.1	**49.4**	48.7	49.0
Robust04	**55.1**	54.2	54.1	54.9
SciFact	**75.0**	74.6	73.5	73.8
Touche	39.1	39.7	**40.4**	**40.4**
DBPedia	43.8	**44.3**	43.9	43.8
BEIR Avg.	**51.4**	51.0	50.8	51.1

Table 6: nDCG@10 results for different view token designs.

	5	10	20	100
DL19	**74.3**	73.7	74.0	68.1
DL20	**69.2**	68.0	67.0	62.4
COVID	**80.2**	80.1	80.1	76.3
NFCorpus	**36.0**	35.9	35.3	34.3
Signal	**32.7**	32.1	31.3	32.0
NEWS	**49.1**	48.6	47.2	46.0
Robust04	55.1	**55.4**	53.8	52.5
SciFact	**75.0**	74.4	74.1	69.3
Touche	**39.1**	39.0	36.9	35.2
DBPedia	43.8	**44.0**	43.4	40.7
BEIR Avg.	**51.4**	51.2	50.3	48.3

Table 7: nDCG@10 performance with varying candidate sampling sizes during training.

FiD-Light (Hofstätter et al., 2023) approach, the first four tokens in the input prompt are reused without introducing dedicated special tokens; (2) Numeric Tokens: View tokens are replaced with number-based tokens (1, 2, 3, 4); (3) Alphabetic Tokens: Character-based tokens (A, B, C, D) are used as view tokens.

Table 6 demonstrate that the <extra_id> tokens defined in the T5 tokenizer, as adopted by the proposed MVP framework, are the most effective. This result suggest that:

(1) Learnable token embeddings specifically trained to encode query-passage relevance are more effective than simply reusing initial prompt tokens;

(2) Moreover, numeric and alphabetic tokens may already carry semantic meaning from pretraining, leading to potential conflicts with their intended function as compression tokens, ultimately resulting in degraded performance.

	View 1	View 2	View 3	View 4	MAX	Mean
DL19	72.8	71.5	72.7	73.5	73.5	**74.3**
DL20	66.3	65.9	68.0	68.3	68.4	**69.2**
COVID	78.9	78.8	79.9	80.0	80.1	**80.2**
NFCorpus	35.7	32.4	31.4	**36.2**	33.1	36.0
Signal	30.6	**33.1**	32.6	32.7	33.4	32.7
NEWS	47.5	44.7	**49.9**	48.0	46.9	49.1
Robust04	52.2	51.4	54.4	**55.3**	53.4	55.1
SciFACT	74.1	58.5	57.0	74.7	73.4	**75.0**
Touche	34.2	37.2	38.2	38.6	37.9	**39.1**
DBPedia	43.1	42.0	41.9	43.6	43.7	**43.8**
BEIR Avg.	49.5	47.2	48.2	51.1	50.2	**51.4**

Table 8: nDCG@10 comparison of view-wise score aggregation methods, including individual views, *MAX*, and *Mean*. The *Mean* strategy corresponds to the default aggregation method used in our proposed framework MVP.

	Relevance Vectors	Anchor Vectors
MVP	0.4910 (0.0229)	-0.0025 (0.0010)
w/o Orthogonal	0.8815 (0.0232)	0.9800 (0.0062)

Table 9: Mean (standard deviation) of pairwise cosine similarities. Similarities are calculated respectively among relevance vectors and anchor vectors.

B.3 Effect of Sampling Size

We further analyze the impact of the number of candidate passages used during training on model performance. The setting with 100 candidates follows the original configuration of Rank-DistilLM, while the settings with 10 and 20 candidates involve randomly sampling 10 or 20 passages per training instance, respectively. The results are summarized in Table 7.

Interestingly, we observe a performance degradation as the number of candidate passages increases. We attribute this to the fact that, when the number of candidates is large, the target distribution becomes overly flattened, thereby diminishing the distinction between highly relevant and irrelevant passages.

C Additional Analysis of MVP

C.1 Impact of Orthogonal Regularization

To verify whether orthogonality promotes separation across views, we analyze the pairwise cosine similarities within anchor vectors and relevance vectors on the DL20 dataset, which contains 54 queries, each associated with 100 candidate passages. We compare the results between MVP and

Model	COVID	NFCorpus	Signal	News	Robust04	SciFact	Touche	DBPedia	Avg.
MVP	**80.2**	**36.0**	**32.7**	**49.1**	55.1	**75.0**	**39.1**	43.8	**51.4**
w/o $\mathcal{L}_{\text{Orthogonal}}$	79.1	35.6	31.4	48.1	54.6	74.3	38.4	43.9	50.7
w/o Multi-view Encoding	78.8	35.8	32.6	48.7	**55.2**	73.2	**39.1**	**44.2**	50.9

Table 10: Full BEIR results for the ablation study on training strategies.

its variant without orthogonality regularization. For anchor vectors, we compute the average pairwise similarities among the 4 anchors for each query[3] and report the average across 54 queries. For relevance vectors, we also compute the average pairwise similarities among the four vectors produced for each query-passage pair, and report the average over all 5,400 pairs.

The results are presented in Table 9. As shown, removing the orthogonality constraint leads to a substantial increase in similarity among both anchor and relevance vectors. This indicates that the relevance vectors capture highly similar signals, and the anchor vectors assess relevance using overlapping criteria. Consequently, this reduces view diversity and leads to performance degradation.

C.2 Effectiveness of View Aggregation

We conducted an additional analysis to verify whether the proposed model effectively aggregates information from each view. Table 8 presents the results of this analysis, where each column represents a different aggregation strategy. Specifically, columns labeled *View 1*, ..., *View 3* show performance when reranking is performed using scores from each individual view alone, while the column labeled *Max* indicates performance obtained by selecting the highest relevance score among all views as the final relevance score. Lastly, the column labeled *Mean* corresponds to our proposed MVP approach, where the final relevance score is calculated by averaging scores across all views.

Experimental results demonstrate that, the MVP approach of averaging scores across views consistently outperforms in most scenarios. In contrast, the *MAX* strategy results in decreased performance, which can be attributed to the inconsistency introduced by selecting the final score from different views. Since each view captures distinct relevance perspectives, relying on a single highest score may lead to instability and undermine the overall ranking consistency.

C.3 Full Reranking Results from Ablation Studies

Following the analysis in Section 4.5.1, Table 10 reports the full reranking results from the ablation experiments.

[3] four vectors generate six unique pairs.