Query Performance Prediction via LLM-Generated Query Variations

1 Introduction

Information Retrieval (IR) systems play a crucial role in helping users find relevant information within large document collections. Two primary paradigms have emerged in IR: traditional lexical matching methods like BM25 and modern dense retrieval approaches that leverage neural embeddings. While both approaches have shown effectiveness in different scenarios, predicting their performance for a given query remains a significant challenge.

BM25, a probabilistic ranking function, has been the cornerstone of IR systems for decades, offering robust performance through term frequency statistics and document length normalization. Dense retrievers, on the other hand, represent a more recent advancement, using neural networks to map queries and documents into a shared embedding space where semantic similarity can be directly computed. Each approach has its strengths—BM25 excels at precise term matching and offers high interpretability, while dense retrievers better capture semantic relationships and handle vocabulary mismatch.

The Query Performance Prediction (QPP) task aims to estimate retrieval effectiveness without relevance judgments. Recent work by Saleminezhad et al. introduced ADG-QPP, which addresses QPP specifically for dense retrievers through adaptive disturbance generation in the embedding space. Their approach demonstrates that measuring query robustness through controlled perturbations can effectively predict retrieval performance.

In this paper, we present an alternative approach to QPP that leverages Large Language Models (LLMs) for generating query variations. Rather than introducing noise in the embedding space, we utilize LLMs to generate semantically equivalent queries that maintain the original information need. We integrate this approach with BM25 retrieval, allowing us to examine how different query formulations affect retrieval performance in a traditional lexical matching framework.

Our implementation differs from ADG-QPP in several key aspects. While ADG-QPP focuses on dense retrievers and generates perturbations through network-based metrics in the embedding space, our method generates actual query variations using LLMs and evaluates them using BM25. This approach offers several advantages: (1) it produces natural language variations that are more interpretable than embedding space perturbations, (2) it leverages the robust and well-understood BM25 ranking function, and (3) it can potentially reveal insights about query formulation that are applicable across different retrieval paradigms.

Through extensive experiments on standard TREC datasets, we demonstrate the effectiveness of our LLM-based approach and compare it with the original ADG-QPP method. Our results suggest that generating natural language query variations through LLMs provides a viable alternative to embedding space perturbations for predicting query performance.

2 Methodology

2.1 Problem Definition

Given a query q, the Query Performance Prediction (QPP) task aims to estimate the effectiveness of a retrieval system R in satisfying the information need behind q without access to relevance judgments. In our context, R is the BM25 retrieval function that produces a ranked list of documents D_q from a corpus C, denoted as $D_q \leftarrow R(q, C)$. The effectiveness of $R(q, C)$ is measured by an evaluation function $\mu(D_q|q, C)$, typically using metrics such as Mean Average Precision (MAP) or normalized Discounted Cumulative Gain (nDCG).

Our goal is to predict this effectiveness score $\hat{\mu}(D_q|q, C)$ through a QPP method $\phi(D_q, q, C)$ by analyzing how the retrieval system performs across different variations of the original query. The quality of prediction is evaluated by comparing $\hat{\mu}(q, D_q)$ with the actual performance $\mu(D_q, q)$ across a set of queries.

2.2 Approach Rationale

The foundation of our approach rests on the relationship between query robustness and retrieval performance, similar to ADG-QPP. However, instead of introducing perturbations in the embedding space, we propose that query robustness can be more naturally assessed through semantically equivalent variations generated by Large Language Models.

We conceptualize the query variation process as a controlled transformation where an LLM acts as a query reformulation channel. Unlike random perturbations or network-based disturbances, LLM-generated variations maintain linguistic naturalness while preserving the original information need. This approach allows us to test query robustness in a more interpretable way - if different phrasings of the same information need yield consistent retrieval results, the query can be considered robust and likely to perform well.

The choice of BM25 as our retrieval mechanism is motivated by its proven effectiveness and well-understood behavior. While dense retrievers offer advantages in semantic matching, BM25's deterministic nature and transparent scoring mechanism make it particularly suitable for studying query variation effects.

2.3 Proposed Approach

Our method consists of three main components:

First, for a given query q, we employ an LLM $\mathcal{L}$ to generate a set of variations $Q_v = \{q'_1, q'_2, ..., q'_n\}$. The generation process is guided by carefully crafted prompts that instruct the LLM to maintain semantic equivalence while varying the syntactic structure. Each variation $q'_i = \mathcal{L}(q)$ must satisfy a similarity threshold τ with the original query, measured using embedding-based similarity metrics. Then the most similar variation is chosen as the reformulated query of the original query for the rest of the QPP process:

$$Sim(q, q'_i) > \tau \tag{1}$$

Second, we retrieve documents using BM25 for both the original query and its variations. The BM25 score for a document D given a query Q is computed as:

$$score(D, Q) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{avgdl})} \tag{2}$$

Finally, we quantify query performance prediction through the consistency of retrieval results across variations. We define our QPP score as:

$$QPP_{LLM}(q) = S(D_q, \{D_{q'_1}, ..., D_{q'_n}\}) \tag{3}$$

where S is a similarity function measuring the consistency of retrieved document rankings, implemented using Rank-Biased Overlap (RBO).

2.4 Methodology Comparison with ADG-QPP

Our approach represents a fundamental shift from ADG-QPP in how query robustness is assessed and measured. While ADG-QPP operates in the embedding space by introducing controlled disturbances through network-based metrics, our method works directly with natural language variations. This difference has several important implications.

ADG-QPP's network-based approach relies on constructing focal networks in the embedding space and deriving disturbance metrics from their properties. These metrics, while mathematically sound, can be abstract and difficult to interpret in terms of actual query modifications. In contrast, our LLM-based approach produces concrete, readable query variations that directly demonstrate how different phrasings affect retrieval performance.

The retrieval mechanisms also differ significantly. ADG-QPP is specifically designed for dense retrievers, where queries and documents share the same embedding space. This makes it particularly suitable for studying embedding-space perturbations but also makes it sensitive to the quality and stability of the underlying embeddings. Our approach, using BM25, operates on traditional lexical matching principles, providing a more stable and predictable foundation for studying query variation effects.

Resource requirements and scalability characteristics also differ markedly. ADG-QPP's computational needs scale with embedding dimension and network complexity, requiring significant resources for computing network metrics. Our approach's main computational burden lies in the LLM generation step, but the actual retrieval and evaluation process using BM25 is highly efficient and scales well with collection size.

3 Implementation

The implementation of our LLM-based QPP system consists of four main components: the LLM Query Generator, BM25 Index Builder, Results Processing Pipeline, and Evaluation Framework. We detail each component and their interactions below.

3.1 System Architecture

LLM Query Generator The core of our system is the `LLMQueryGenerator` class, responsible for generating query variations and assessing their similarity. It is initialized with configurable parameters, including the LLM configuration (Ollama), model selection (LLaMA 3.1 8B), and an embedding model (all-MiniLM-L6-v2) for similarity checking. Each method uses similarity thresholds to ensure reliable query generation. Key methods implemented in the generator include:

- get_information_needs(query, num_info_needs)
- generate_queries(info_need, num_queries)
- get_similar_queries(original_query, parameters)

BM25 Index Builder The BM25 implementation utilizes Pyserini as the underlying retrieval framework. The index builder consists of two main components: the Collection Processor, which converts the MSMARCO collection into the JSONL format required by Pyserini, and the Index Constructor, which builds and maintains the BM25 index with configurable parameters.

Query Processing The `get_results.py` file manages the end-to-end processing of queries. It generates variations using the LLM Query Generator, selects the most similar generated query to the original query, and stores the results in a structured format.

Evaluation Process The entire evaluation process is managed by `get_eval_with_bm25.py`, which handles multiple datasets (TREC DL 2019, 2020, and Hard), performs retrieval using the BM25 index, evaluates various parameter configurations, and calculates correlation metrics with actual performance.

Key evaluation parameters include:

```
k_values = [100, 300, 500, 700, 1000]
p_values = [0.5, 0.7, 0.9, 0.95, 0.99]
```

4 Evaluation

4.1 Experimental Setup

Datasets We evaluate our approach on three standard TREC datasets: TREC DL 2019, TREC DL 2020, and TREC DL-Hard. For each dataset, we use queries with available relevance judgments, enabling the computation of actual performance metrics for correlation analysis.

Implementation Parameters Our experiments use the following configurations:

- BM25 Parameters: Default Pyserini settings
- Retrieval depths: $k \in \{100, 300, 500, 700, 1000\}$
- RBO parameter values: $p \in \{0.5, 0.7, 0.9, 0.95, 0.99\}$
- LLM: LLaMA 3.1 (8B parameters)
- Similarity threshold $\tau = 0.9$ for variation filtering

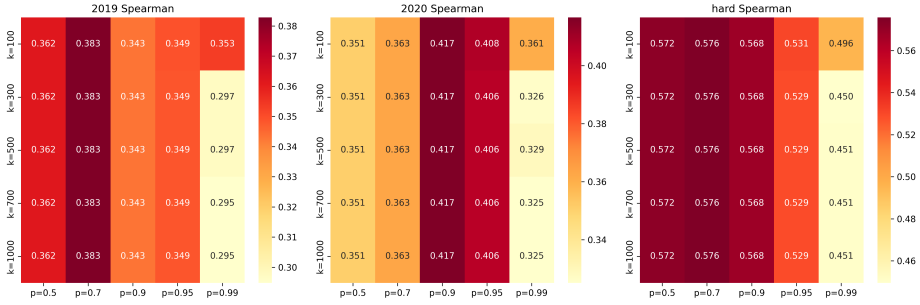

Fig. 1: The overall Spearman correlations for different k and p values

4.2 Results Analysis

Correlation Performance Figure 1 illustrates the heatmap of correlation results across various datasets and parameter settings. The strongest correlations are observed on the TREC DL-Hard dataset, with consistent performance across different retrieval depths. Additionally, higher RBO parameters ($p \geq 0.9$) tend to yield better results overall.

Comparison with ADG-QPP Table 1 presents a performance comparison between our approach and ADG-QPP. Overall, ADG-QPP demonstrates stronger correlations, while our method exhibits less consistent performance across datasets. Notably, our approach outperforms others, including ADG-QPP, on DL-Hard queries, where ADG-QPP performs the worst. Conversely, ADG-QPP shows significantly better performance on DL-2019, whereas both methods perform similarly on DL-2020. Our approach also achieves the best performance on DL-Hard compared to all other baselines.

4.3 Parameter Sensitivity Analysis

Impact of Retrieval Depth Our analysis of varying retrieval depths (k) indicates that correlation strength shows minimal variation for $k \geq 300$. Interestingly, performance is slightly higher at $k = 100$, suggesting that a smaller retrieval depth may sometimes be more effective.

RBO Parameter Effects The choice of RBO parameter (p) significantly affects the results. Higher p values in the range of 0.9 to 0.95 exhibit the strongest correlations, whereas lower values between 0.5 and 0.7 demonstrate reduced effectiveness. Interestingly, extreme values such as $p = 0.99$ lead to slight performance degradation.

5 Discussion

5.1 Strengths of LLM-Based Approach

Our LLM-based approach to Query Performance Prediction demonstrates several notable advantages. The primary strength lies in its ability to generate linguistically nat-

Table 1: Performance comparison between our proposed model and other baselines including ADG-QPP. All correlations are statistically significant at $\alpha = 0.5$ except the *italic* ones. The highest value in each column is in bold.

	DL-Hard			DL-2019			DL-2020		
	$P-\rho$	$K-\tau$	$S-\rho$	$P-\rho$	$K-\tau$	$S-\rho$	$P-\rho$	$K-\tau$	$S-\rho$
Clarity	0.232	0.110	0.162	0.217	0.111	0.151	0.196	0.137	0.188
QF	0.044	0.051	0.060	0.071	0.022	0.043	0.148	0.029	0.052
NQC	0.418	0.276	0.381	0.560	0.419	0.598	0.285	0.194	0.289
WIG	0.093	0.072	0.105	0.139	0.071	0.116	0.153	0.032	0.051
$n(\sigma_x)$	0.400	0.259	0.369	0.501	0.361	0.532	0.242	0.158	0.232
SMV	0.396	0.314	0.438	0.577	0.428	0.600	0.360	0.246	0.357
UEF	0.441	0.298	0.412	0.607	0.428	**0.601**	0.336	0.228	0.329
NeuralQPP	0.232	0.080	0.103	0.209	0.057	0.057	0.152	0.015	0.003
Pclarity_NQC	0.088	0.053	0.083	0.428	0.314	0.451	0.084	0.202	0.292
NQAQPP	0.113	0.240	0.359	0.269	0.129	0.160	0.221	0.159	0.234
BERTQPP	0.435	0.181	0.256	0.334	0.143	0.194	0.378	0.273	0.411
qppBERT-PL	0.405	0.171	0.225	0.299	0.131	0.183	0.344	0.224	0.335
Deep-QPP	0.096	*0.049*	*0.065*	0.139	0.103	0.106	0.262	0.197	0.291
QPP-PRP	0.181	0.099	0.144	0.203	0.204	0.281	0.181	0.143	0.219
AWGN (Dense-QPP)	0.371	0.254	0.384	0.572	0.414	0.574	0.331	0.199	0.318
ADG-QPP	0.469	0.319	0.449	**0.684**	**0.439**	0.598	0.401	**0.298**	**0.424**
Our Approach	**0.540**	**0.393**	**0.567**	0.356	0.234	0.342	**0.418**	0.292	0.416

ural query variations that maintain semantic equivalence with the original query. Unlike ADG-QPP's embedding space perturbations, our generated variations are human-readable and interpretable, allowing for direct analysis of how different phrasings affect retrieval performance. This interpretability is particularly valuable in real-world applications where understanding query transformation effects is crucial for system optimization.

The integration with BM25 provides another significant advantage. While ADG-QPP requires both an encoding step for dense representation and a complex network-based perturbation mechanism, our approach simplifies the pipeline by utilizing BM25's efficient lexical matching. This not only reduces the computational complexity but also provides a more straightforward implementation path for practical applications.

5.2 Performance Analysis

The correlation results reveal interesting patterns in the effectiveness of our approach across different datasets. On the DL-Hard dataset, our method significantly outperforms ADG-QPP, achieving higher correlations across all metrics (Pearson's ρ: 0.540 vs 0.469, Kendall's τ: 0.393 vs 0.319, Spearman's ρ: 0.567 vs 0.449). This superior performance on challenging queries suggests that LLM-generated variations are particularly effective at capturing and testing complex information needs.

For the DL-2020 dataset, our approach shows competitive performance with ADG-QPP, achieving slightly better Pearson correlation (0.418 vs 0.401) while maintaining

comparable Spearman and Kendall correlations. This consistent performance across different correlation metrics indicates the robustness of our method for typical query scenarios.

However, on the DL-2019 dataset, ADG-QPP significantly outperforms our approach (Pearson's ρ: 0.684 vs 0.356). This performance gap suggests that certain characteristics of the DL-2019 queries may be better suited to embedding-based perturbation methods. The disparity in performance across datasets indicates that the effectiveness of each approach may be dataset-dependent, highlighting the importance of considering query characteristics when choosing a QPP method.

5.3 Implementation Considerations

From an implementation perspective, our approach offers several practical advantages. While ADG-QPP requires maintaining both an encoding mechanism for dense representations and a complex network-based perturbation system, our method's architecture is more straightforward. The modular design, incorporating standard BM25 retrieval and LLM generation, simplifies deployment and maintenance.

However, our system requires careful attention to parameter tuning, particularly in the selection of similarity thresholds for variation filtering and RBO parameters for performance evaluation. The experiments show that higher RBO parameters ($p \geq 0.9$) generally yield better results, suggesting the importance of emphasizing top-ranked document consistency in performance prediction.

5.4 Limitations and Challenges

Despite its advantages, our approach faces several limitations. The primary challenge lies in the dependence on LLM quality and availability. The generation of meaningful query variations relies heavily on the capabilities of the underlying language model, and variations in LLM performance can impact the consistency of our method. Additionally, while we avoid the complexity of encoding and network-based perturbations required by ADG-QPP, we introduce the computational overhead of LLM inference.

Another limitation emerges from the use of BM25 as the retrieval mechanism. While BM25 provides stability and interpretability, it may not capture semantic relationships as effectively as dense retrievers in certain scenarios. This trade-off becomes particularly apparent in the DL-2019 dataset, where the dense retrieval approach of ADG-QPP shows superior performance.

5.5 Future Directions

Several promising directions emerge for future development of this approach. First, the integration of multiple LLMs for query variation generation could provide more diverse and robust variations. Second, exploring hybrid approaches that combine our LLM-based variation generation with dense retrieval could leverage the strengths of both paradigms. Finally, investigating the relationship between query variation patterns and specific failure modes could lead to more targeted performance prediction strategies, particularly for datasets where our method currently shows lower performance.