Generative Relevance Feedback using Large Language Models

Student Name: Mohammad Tejabwala
Enrollment ID: **202001406**
B. Tech. Project (BTP) Report
BTP Mode: ***On Campus* - in a Group***
Dhirubhai Ambani Institute of ICT (DA-IICT)
Gandhinagar, India
202001406[at]daiict.ac.in

Mentor's Name: Prof. Gaurav Kumar Singh
Dhirubhai Ambani Institute of ICT (DA-IICT)
Near Indroda Circle
Gandhinagar 382007, India
gauravkumar_singh[at]daiict[dot]ac[dot]in

***Abstract*—Integrating Large Language Models (LLMs) like GPT-4 and Gemini into search engines is a groundbreaking leap in information retrieval. This paper introduces a simple yet effective query expansion approach utilizing the Large Language Models. Pseudo-relevance feedback (PRF) is used by current query expansion models to increase first-pass retrieval effectiveness; however, this approach is ineffective when the initial results are irrelevant. Furthermore, our results indicate that our studied generative models can outperform various statistical query expansion approaches by improving the Recall@1000 by 13.49%, MAP by 34.02%, and NDCG@10 by 23.59%.**

I. Introduction

Of late, there has been a lot of hype about using Large language models (LLMs) to improve various tasks and one such task is the integration of LLMs with Search Engines to improve user experience. In this paper, we focus on improving the information retrieval task by forming a new query with the feedback from LLMs. We have used GPT-3.5, GPT-4, BARD and Gemini models for this purpose with their default parameters. The feedback generated by the LLM is varied by providing various subtasks to obtain a wide range of distinct results. Subsequently, we form the new query by concatenating the feedback with the original query [12]. Since the original query is typically much shorter than the generated feedback, to balance the relative weights of the query and the feedback, we boost the query term weights by repeating the query n times before concatenating [17]. Following that, we find average values of n for each evaluating parameter and perform pseudo-relevance feedback using the Bo1 query expansion model [7] for the values of n with the highest average for each evaluation parameter. Finally, we select the subtasks with the highest values in each evaluation parameter in the previous step and evaluate them by generating different combinations and merging them into a single subtask.

II. Related work

When a query does not fully capture the user's information demand, vocabulary mismatch is a significant problem in information retrieval. Lexical-based sparse retrieval, like BM25, and embedding-based dense retrieval are the two main paradigms for information retrieval [13], [18]. While dense retrievers tend to outperform in scenarios with ample labeled data, it has been noted that BM25 maintains a competitive advantage, particularly in datasets that lie outside its domain. [10], [15].

Early research on query expansion concentrated on utilizing either lexical knowledge bases [2], [16] or Pseudo-relevance feedback (PRF) [8], [14], [19] typically through the training or fine-tuning of a model. The first round of retrieval's top-ranked documents are presumed relevant for a particular user query by the pseudo-relevance feedback (PRF). The expansion terms are then removed from the collection of papers with pseudo-relevance using various term weighting techniques and selection criteria. The benefit of PRF is that it greatly reduces the gap between query and document vocabularies since the extracted terms are automatically obtained based on the context analysis of local document collections.

Reformulations based on the usage of dictionaries and thesauri [6] or the examination of comparable queries in query logs [4], [9] were two other methods of query expansion. For example, Zukerman et al. employed an iterative approach to discover comparable terms to a query based on WordNet [20], and Jones et al. [9] substituted phrases within queries with those in similar queries. Furthermore, the use of word embeddings for query expansion has also been investigated [5], [11]. Nevertheless, in order to improve the query representation, these query expansion models rely on data that is taken from the documents that are retrieved or from other sources (such as WordNet or click data). On the other hand, our method makes use of a text generation model in conjunction with pseudo-relevance feedback to generate refined textual inquiries.

LLMs have been used in recent work by Mackie et al. to propose generative-relevance feedback (GRF) that generates text independently of first-pass retrieval [12]. In particular, they leverage an LLM to produce various forms of text depending on the original query. Then, they feed these termed "generated documents" into term-based expansion models. They test several generated content formats (facts, news stories, chain-

*Group Members's Name: Aditi Das, Enrollment ID: 202001259, Email: 202001259 [at] daiict.ac.in, Institute: DA-IICT

of-thought reasoning, etc.) and discover that integrating all the generations yields the greatest results. Building on this work, we execute combinations instead of combining all the subtasks and expand GRF with PRF.

Arora et al. used the retrieval model first to fetch potentially relevant documents, which then become the input to the large language model to produce answers for the query directly [1]. Thus, they use few-shot learning by passing the relevant documents as a context to the large language model and then generating the answer for the query. This is known as the retrieval-augmented generation (RAG) paradigm.

III. Subtasks

The 12 query-specific generation subtasks are:

- Entities
- CoT-Entities
- Keywords
- CoT-Keywords
- Facts
- Summary
- Document
- Essay
- News
- Alternate Queries
- Expanded Queries
- Web page content

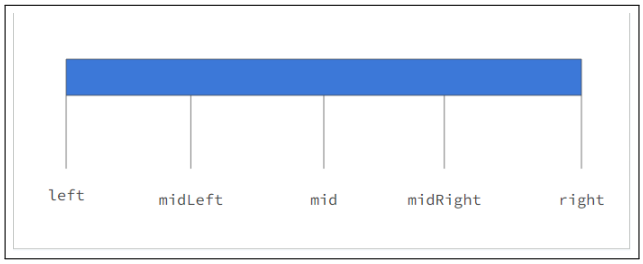
The first 9 subtasks are referred from the work of Mackie et al. [12]. The prompt for the Large Langauge model is generated as follows: Generate subtask for each of the queries. For the CoT subtasks, generate chain-of-thought (CoT) reasoning to explain "why" a list of subtasks are relevant is added to the prompt.

IV. Method

A. Tuning the parameters for the BM25 model

The bag-of-words retrieval function BM25 ranks a collection of documents according to the query phrases that exist in each one, regardless of how close together they occur. There are two free parameters, k_1 and b, that are used in the calculation of the relevance score for the document. In most cases, we tune the parameters by simulating all the values in a particular range by using methods such as GridSearch belonging to Pyterrier. However, to reduce the computation time, we applied binary search to tune the parameters.

For each of the evaluation parameters, We first set the k_1 parameter to its default value and perform a binary search in the range from 0 to 1 to tune the value of the parameter b. The binary search is applied as follows:



Initially, the $left$ value is set to 0, and the $right$ value is set to 1. Then, we calculate the mid by $(left + right)/2$. We calculate the $midLeft$ using $(left + mid)/2$ and $midRight$ using $(mid + right)/2$. We then calculate the value of the selected evaluation parameter by passing the value of b to both $midLeft$ and $midRight$. We then move toward having a higher value in the evaluation parameter. If the evaluation parameter has a higher value using $midLeft$, then $right$ is set to $midLeft$. Otherwise, we set $left$ to $midRight$. We continue this until the absolute difference between $left$ and $right$ is greater than the selected threshold. After obtaining the value of b, we then repeat the same method for k_1 for the range of 0 to 2. This method might not give us a global maximum, but we can obtain a local maximum in this way.

If we use the traditional method, then we would need to perform 210 simulations for each of the subtasks (Considering the 0.1 step in both ranges). With the introduction of the Binary search method, the simulations reduce from 210 to only 3.

Further, as the query length varies in the different subtasks, we tuned the parameters for three different subtask categories, namely keywords, entities (Short-length queries), CoT Entities, CoT keywords, facts, summary, alternate queries, expanded queries (medium-length queries), an essay, news, document, and web page content (long-length queries).

B. First retrieval

We boost the query word weights by repeating the query n times before concatenating with the generated feedback F_{GRF}, as the query q is usually considerably shorter than the generated feedback. This allows us to balance the relative weights of the query and the generated feedback.

$$q' = \text{concat}(\{q\} \times n, F_{\text{GRF}})$$

We pass the newly generated query q' to the BM25 to retrieve the first set of documents and obtain the value for each evaluation parameter.

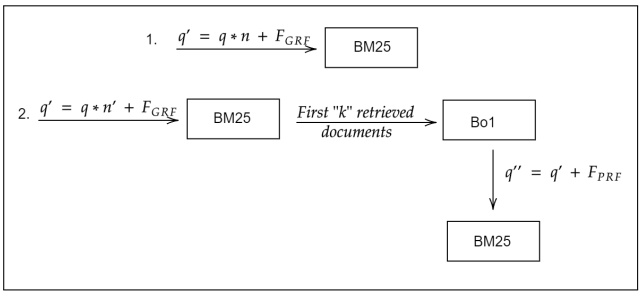
C. Second retrieval

After getting the results for the first retrieval, we calculate the average values of each evaluation parameter of all the subtasks for each turn n. From the calculated average values, we select the values of n with the highest average value in each evaluation parameter and term it n'. By filtering the values of n in this way, we avoid irrelevant computation in this step and decrease the computation time needed for the simulation.

Now, we again pass q' for the selected values of n to the pipeline containing BM25 + Bo1 + BM25. Bo1 Divergence from Randomness query expansion model rewrites the query based on the occurrences of terms in the feedback documents provided for each query. After passing through the Bo1 model, the new query contains additional terms F_{PRF} added by the PRF Bo1 model. We use the Bo1 model with its default parameters set by Pyterrier.

$$q'' = \text{concat}(\{q\} \times n', F_{\text{GRF}}, F_{\text{PRF}})$$

We then obtain the new values of evaluation parameters for each of the subtasks.



D. Combinations

After obtaining the results of each subtask for the selected value n' from the previous pipeline, we select the task having the overall best Recall@1000, best MAP, best ndcg@10 score, and an overall best sum of the values of the evaluation parameters. We then perform $\binom{4}{2} + \binom{4}{3} + \binom{4}{4} = 11$ combinations using these four distinct selected tasks and pass them through the same pipeline consisting of BM25 + Bo1 + BM25 and the values of n' used in the previous step.

E. Combining all subtasks

Having obtained the results from generating combinations using the most optimum subtasks; we wanted to check if the evaluation metrics generated in this manner would out perform the metrics generated by combining all the subtasks for each LLM.

V. Dataset

Studying ad hoc ranking in a massive data regime is the aim of the Deep Learning Track, a new track for TREC 2019. It's the first track with sizable training sets labeled by humans, introducing two sets for each task, each with reusable test sets and a strict blind assessment process akin to TREC. A reusable test set of 43 queries is developed for the 3.2 million document corpus with 367 thousand training queries used in the document retrieval assignment [3].

VI. Evaluation Parameters

Following standard practice in the TREC Deep Learning track [3], we measure the effectiveness of the reformulated queries through their ranking performance in terms of Mean Average Precision (MAP), Recall, and normalized discounted cumulative gain calculated to rank depth 10.

VII. Results

A. Without GRF

TABLE I
Performance Metrics for $n = 2$

Task	Recall@1000	MAP	NDCG@10
BM25 (default parameters)	0.706	0.335	0.539
BM25 (using GridSearch)	0.703	0.335	0.531
BM25 (using our parameters)	0.713	0.344	0.561

B. BARD

Step 1: Using GRF:

TABLE II
Performance Metrics for Entities

Recall@1000	MAP	NDCG@10	n
0.778	0.395	0.539	1
0.778	0.397	0.539	2
0.766	0.393	0.545	3
0.752	0.385	0.545	4
0.741	0.375	0.542	5

TABLE III
Performance Metrics For Chain of Thought Entities

Recall@1000	MAP	NDCG@10	n
0.783	0.438	0.653	1
0.786	0.443	0.652	2
0.787	0.442	0.654	3
0.786	0.438	0.65	4
0.783	0.433	0.642	5

TABLE IV
Performance Metrics For Keywords

Recall@1000	MAP	NDCG@10	n
0.769	0.371	0.549	1
0.768	0.387	0.561	2
0.766	0.389	0.566	3
0.758	0.388	0.561	4
0.75	0.385	0.56	5

TABLE V
Performance Metrics For Chain Of Thought Keywords

Recall@1000	MAP	NDCG@10	n
0.764	0.386	0.578	1
0.768	0.39	0.585	2
0.77	0.395	0.587	3
0.769	0.397	0.588	4
0.769	0.398	0.586	5

TABLE VI
Performance Metrics For Facts

Recall@1000	MAP	NDCG@10	n
0.802	0.429	0.601	1
0.803	0.432	0.603	2
0.8	0.429	0.61	3
0.797	0.425	0.608	4
0.79	0.418	0.6	5

TABLE VII
PERFORMANCE METRICS FOR SUMMARY

Recall@1000	MAP	NDCG@10	n
0.79	0.397	0.579	1
0.785	0.395	0.589	2
0.778	0.386	0.58	3
0.768	0.376	0.577	4
0.751	0.365	0.565	5

TABLE VIII
PERFORMANCE METRICS FOR DOCUMENT

Recall@1000	MAP	NDCG@10	n
0.816	0.455	0.609	1
0.812	0.456	0.609	2
0.803	0.45	0.601	3
0.793	0.44	0.589	4
0.787	0.43	0.584	5

TABLE IX
PERFORMANCE METRICS FOR ESSAY

Recall@1000	MAP	NDCG@10	n
0.786	0.423	0.595	1
0.791	0.428	0.594	2
0.792	0.426	0.599	3
0.793	0.427	0.604	4
0.793	0.424	0.606	5

TABLE X
PERFORMANCE METRICS FOR NEWS

Recall@1000	MAP	NDCG@10	n
0.799	0.438	0.592	1
0.796	0.44	0.59	2
0.793	0.434	0.582	3
0.788	0.425	0.577	4
0.782	0.416	0.578	5

TABLE XI
PERFORMANCE METRICS FOR ALTERNATE SENTENCES

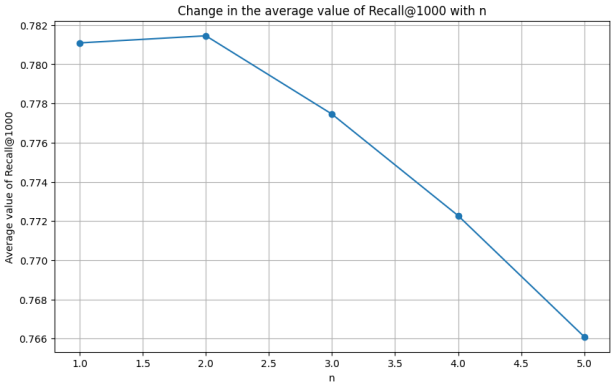
Recall@1000	MAP	NDCG@10	n
0.755	0.397	0.578	1
0.757	0.394	0.574	2
0.746	0.386	0.572	3
0.743	0.379	0.567	4
0.737	0.371	0.568	5

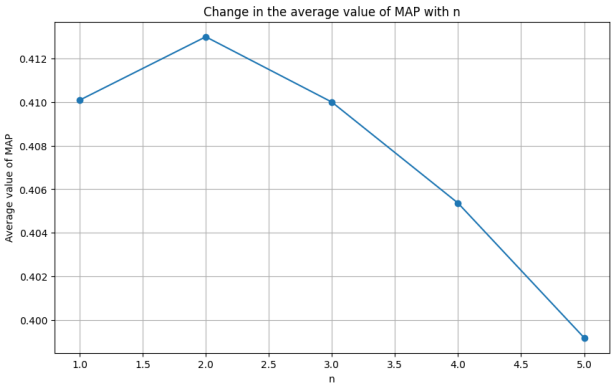
TABLE XII
PERFORMANCE METRICS FOR EXPANDED QUERIES

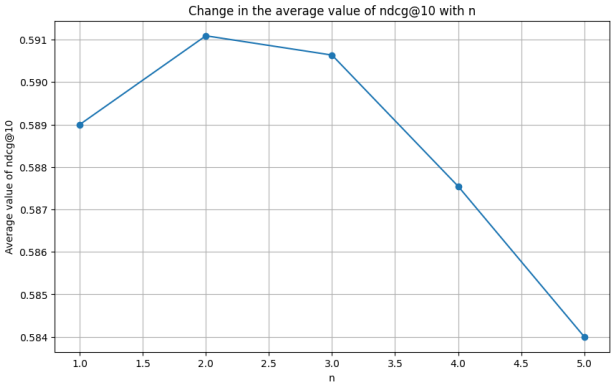
Recall@1000	MAP	NDCG@10	n
0.742	0.381	0.583	1
0.731	0.375	0.578	2
0.725	0.365	0.574	3
0.714	0.353	0.564	4
0.699	0.343	0.548	5

TABLE XIII
PERFORMANCE METRICS FOR WEB PAGE

Recall@1000	MAP	NDCG@10	n
0.749	0.38	0.568	1
0.756	0.387	0.582	2
0.754	0.394	0.587	3
0.75	0.392	0.59	4
0.745	0.389	0.591	5







Step 2 : Using GRF + PRF :

TABLE XIV
PERFORMANCE METRICS FOR $n = 2$

Subtask	Recall@1000	MAP	NDCG@10
entities	0.800	0.417	0.546
entities_cot	0.806	0.467	0.651
keywords	0.794	0.414	0.574
keywords_cot	0.785	0.413	0.579
facts	0.822	0.460	0.609
summary	0.820	0.443	0.613
document	0.823	0.474	0.612
essay	0.820	0.457	0.616
news	0.813	0.465	0.590
alt_sentences	0.776	0.412	0.582
expanded_queries	0.765	0.398	0.597
web_page	0.797	0.432	0.595

Step 3 : GRP + PRF + Combining subtasks

TABLE XV
PERFORMANCE METRICS FOR $n = 2$

Subtasks	Recall@1000	MAP	NDCG@10
document, entities_cot	0.817	0.483	0.644
document, essay	0.835	0.487	0.645
document, facts	0.831	0.483	0.632
entities_cot, essay	0.818	0.482	0.644
entities_cot, facts	0.816	0.484	0.656
essay, facts	0.825	0.481	0.636
document, entities_cot, essay	0.823	0.491	0.655
document, entities_cot, facts	0.818	0.487	0.639
document, essay, facts	0.837	0.493	0.638
entities_cot, essay, facts	0.819	0.488	0.659
document, entities_cot, essay, facts	0.825	0.495	0.659

Step 4 : Combining all subtasks:

TABLE XVI
PERFORMANCE METRICS FOR BARD

Recall@1000	MAP	NDCG@10	n
0.83	0.483	0.645	2

C. GPT

Step 1 : Using GRF:

TABLE XVII
PERFORMANCE METRICS FOR ENTITIES

Recall@1000	MAP	NDCG@10	n
0.796	0.409	0.562	1
0.787	0.41	0.572	2
0.779	0.402	0.572	3
0.767	0.388	0.562	4
0.75	0.374	0.547	5

TABLE XVIII
PERFORMANCE METRICS FOR CHAIN OF THOUGHT ENTITIES

Recall@1000	MAP	NDCG@10	n
0.787	0.371	0.515	1
0.787	0.391	0.544	2
0.783	0.394	0.549	3
0.778	0.391	0.545	4
0.77	0.387	0.541	5

TABLE XIX
PERFORMANCE METRICS FOR KEYWORDS

Recall@1000	MAP	NDCG@10	n
0.771	0.368	0.544	1
0.766	0.37	0.548	2
0.749	0.362	0.542	3
0.736	0.35	0.536	4
0.723	0.338	0.528	5

TABLE XX
PERFORMANCE METRICS FOR CHAIN OF THOUGHT KEYWORDS

Recall@1000	MAP	NDCG@10	n
0.775	0.385	0.545	1
0.785	0.396	0.552	2
0.782	0.396	0.55	3
0.78	0.393	0.546	4
0.773	0.389	0.543	5

TABLE XXI
PERFORMANCE METRICS FOR FACTS

Recall@1000	MAP	NDCG@10	n
0.792	0.388	0.56	1
0.798	0.403	0.571	2
0.802	0.403	0.571	3
0.798	0.4	0.571	4
0.785	0.394	0.567	5

TABLE XXII
PERFORMANCE METRICS FOR SUMMARY

Recall@1000	MAP	NDCG@10	n
0.781	0.39	0.576	1
0.782	0.39	0.572	2
0.773	0.386	0.573	3
0.763	0.378	0.565	4
0.757	0.37	0.56	5

TABLE XXIII
PERFORMANCE METRICS FOR DOCUMENT

Recall@1000	MAP	NDCG@10	n
0.809	0.431	0.578	1
0.812	0.443	0.593	2
0.811	0.442	0.601	3
0.808	0.437	0.61	4
0.803	0.428	0.606	5

TABLE XXIV
PERFORMANCE METRICS FOR ESSAY

Recall@1000	MAP	NDCG@10	n
0.778	0.395	0.566	1
0.782	0.404	0.576	2
0.784	0.406	0.58	3
0.78	0.402	0.574	4
0.77	0.398	0.574	5

TABLE XXV
PERFORMANCE METRICS FOR NEWS

Recall@1000	MAP	NDCG@10	n
0.781	0.393	0.596	1
0.784	0.396	0.595	2
0.777	0.39	0.597	3
0.767	0.387	0.59	4
0.761	0.381	0.587	5

TABLE XXVI
PERFORMANCE METRICS FOR ALTERNATE SENTENCES

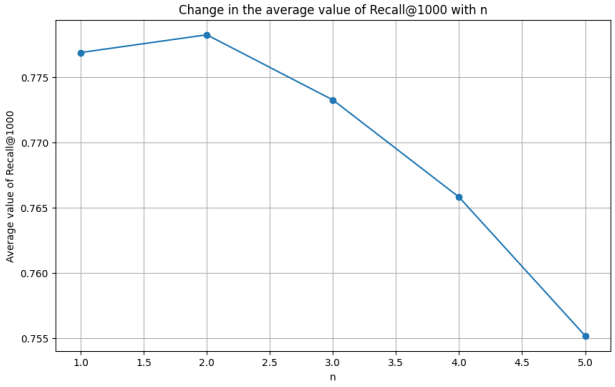
Recall@1000	MAP	NDCG@10
0.769	0.38	0.593
0.766	0.383	0.595
0.764	0.384	0.596
0.752	0.379	0.593
0.746	0.373	0.587

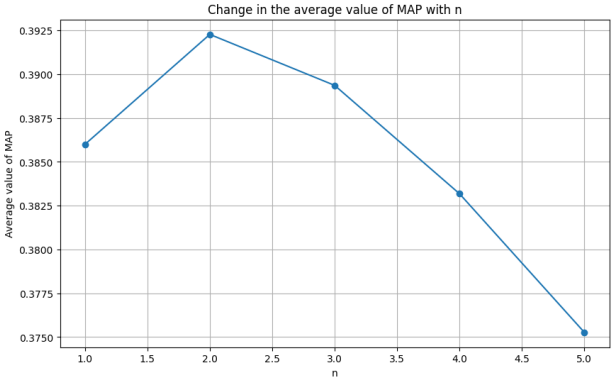
TABLE XXVII
PERFORMANCE METRICS FOR EXPANDED QUERIES

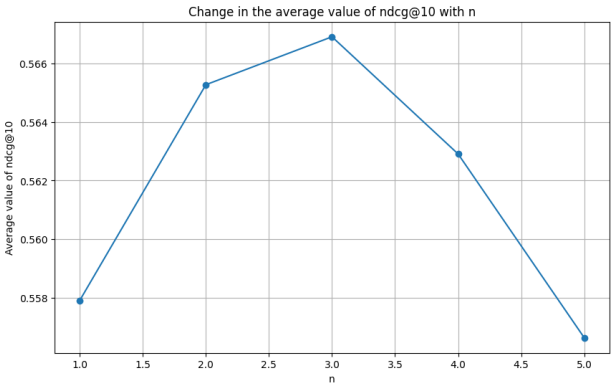
Recall@1000	MAP	NDCG@10	n
0.71	0.349	0.555	1
0.712	0.343	0.548	2
0.703	0.334	0.547	3
0.694	0.323	0.539	4
0.671	0.309	0.522	5

TABLE XXVIII
PERFORMANCE METRICS FOR WEB PAGE

Recall@1000	MAP	NDCG@10	n
0.744	0.355	0.53	1
0.742	0.357	0.54	2
0.74	0.357	0.538	3
0.732	0.355	0.541	4
0.72	0.351	0.542	5







Step 2 : Using GRF + PRF :

TABLE XXIX
PERFORMANCE METRICS FOR $n = 2$

Subtask	Recall@1000	MAP	NDCG@10
entities	0.81	0.426	0.55
entities_cot	0.814	0.425	0.545
keywords	0.804	0.408	0.551
keywords_cot	0.811	0.42	0.543
facts	0.821	0.432	0.565
summary	0.812	0.43	0.582
document	0.829	0.462	0.59
essay	0.809	0.432	0.583
news	0.814	0.445	0.612
alt_sentences	0.802	0.424	0.605
expanded_queries	0.77	0.394	0.568
web_page	0.789	0.41	0.578

TABLE XXX
PERFORMANCE METRICS FOR $n = 3$

Subtask	Recall@1000	MAP	NDCG@10
entities	0.812	0.418	0.547
entities_cot	0.814	0.431	0.551
keywords	0.797	0.402	0.549
keywords_cot	0.812	0.424	0.548
facts	0.822	0.438	0.573
summary	0.812	0.429	0.578
document	0.831	0.47	0.603
essay	0.81	0.438	0.588
news	0.813	0.44	0.608
alt_sentences	0.8	0.419	0.6
expanded_queries	0.764	0.385	0.552
web_page	0.78	0.401	0.564

Step 3 : GRP + PRF + Combining subtasks

TABLE XXXI
PERFORMANCE METRICS FOR $n = 2$

Subtasks	Recall@1000	MAP	NDCG@10
document, news	0.831	0.47	0.624
document, facts	0.837	0.467	0.6
document, alt_sentences	0.841	0.478	0.635
news, facts	0.829	0.451	0.589
news, alt_sentences	0.825	0.454	0.622
facts, alt_sentences	0.834	0.466	0.614
document, news, facts	0.834	0.464	0.607
document, news, alt_sentences	0.837	0.471	0.629
document, facts, alt_sentences	0.841	0.474	0.614
news, facts, alt_sentences	0.836	0.459	0.609
document, news, facts, alt_sentences	0.832	0.463	0.616

TABLE XXXII
PERFORMANCE METRICS FOR $n = 3$

Subtasks	Recall@1000	MAP	NDCG@10
document, news	0.83	0.468	0.62
document, facts	0.836	0.473	0.605
document, alt_sentences	0.837	0.476	0.631
news, facts	0.827	0.452	0.602
news, alt_sentences	0.823	0.453	0.626
facts, alt_sentences	0.835	0.466	0.612
document, news, facts	0.835	0.464	0.603
document, news, alt_sentences	0.835	0.47	0.634
document, facts, alt_sentences	0.84	0.476	0.616
news, facts, alt_sentences	0.833	0.46	0.61
document, news, facts, alt_sentences	0.831	0.463	0.612

Step 4 : Combining all subtasks:

TABLE XXXIII
PERFORMANCE METRICS FOR GPT

Recall@1000	MAP	NDCG@10	n
0.833	0.459	0.595	2

TABLE XXXIV
PERFORMANCE METRICS FOR GPT

Recall@1000	MAP	NDCG@10	n
0.83	0.462	0.601	3

D. GPT-4

Step 1 : Using GRF:

TABLE XXXV
PERFORMANCE METRICS FOR ENTITIES

Recall@1000	MAP	NDCG@10	n
0.794	0.416	0.584	1
0.792	0.415	0.585	2
0.784	0.406	0.583	3
0.766	0.392	0.584	4
0.748	0.378	0.568	5

TABLE XXXVI
PERFORMANCE METRICS FOR CHAIN OF THOUGHT ENTITIES

Recall@1000	MAP	NDCG@10	n
0.76	0.4	0.585	1
0.771	0.404	0.585	2
0.77	0.399	0.583	3
0.764	0.394	0.579	4
0.757	0.385	0.581	5

TABLE XXXVII
PERFORMANCE METRICS FOR KEYWORDS

Recall@1000	MAP	NDCG@10	n
0.79	0.406	0.588	1
0.789	0.409	0.595	2
0.781	0.401	0.599	3
0.771	0.388	0.589	4
0.748	0.373	0.578	5

TABLE XXXVIII
PERFORMANCE METRICS FOR CHAIN OF THOUGHT KEYWORDS

Recall@1000	MAP	NDCG@10	n
0.772	0.385	0.591	1
0.774	0.393	0.595	2
0.773	0.389	0.589	3
0.767	0.383	0.582	4
0.761	0.377	0.572	5

TABLE XXXIX
PERFORMANCE METRICS FOR FACTS

Recall@1000	MAP	NDCG@10	n
0.794	0.428	0.636	1
0.795	0.421	0.625	2
0.78	0.409	0.615	3
0.772	0.396	0.607	4
0.75	0.381	0.598	5

TABLE XL
PERFORMANCE METRICS FOR SUMMARY

Recall@1000	MAP	NDCG@10	n
0.79	0.425	0.613	1
0.791	0.422	0.614	2
0.788	0.413	0.609	3
0.775	0.398	0.597	4
0.766	0.385	0.592	5

TABLE XLI
PERFORMANCE METRICS FOR DOCUMENT

Recall@1000	MAP	NDCG@10	n
0.819	0.442	0.638	1
0.818	0.45	0.642	2
0.812	0.448	0.642	3
0.807	0.444	0.643	4
0.805	0.439	0.64	5

TABLE XLII
PERFORMANCE METRICS FOR ESSAY

Recall@1000	MAP	NDCG@10	n
0.816	0.439	0.618	1
0.818	0.451	0.619	2
0.819	0.452	0.623	3
0.817	0.452	0.618	4
0.816	0.449	0.61	5

TABLE XLIII
PERFORMANCE METRICS FOR NEWS

Recall@1000	MAP	NDCG@10	n
0.807	0.438	0.636	1
0.81	0.444	0.634	2
0.804	0.441	0.626	3
0.803	0.433	0.626	4
0.794	0.422	0.623	5

TABLE XLIV
PERFORMANCE METRICS FOR ALTERNATE SENTENCES

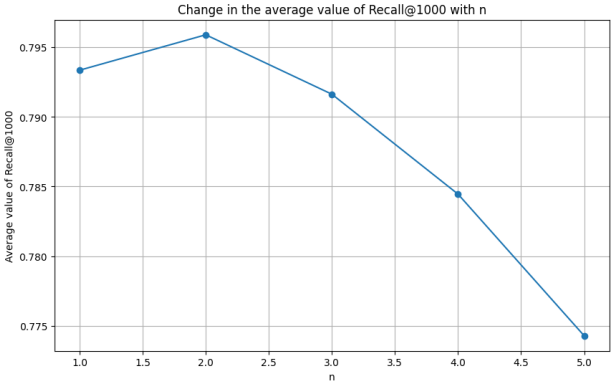
Recall@1000	MAP	NDCG@10	n
0.737	0.37	0.563	1
0.73	0.367	0.567	2
0.73	0.362	0.569	3
0.727	0.359	0.566	4
0.726	0.353	0.559	5

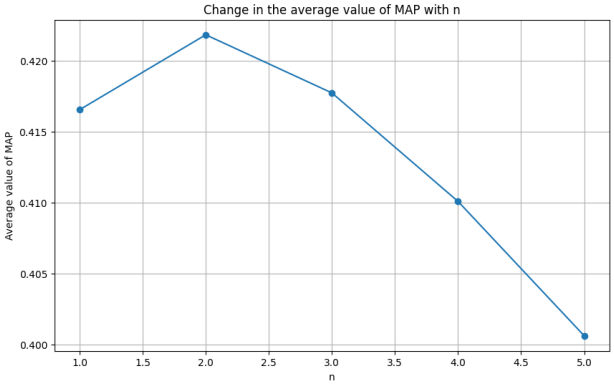
TABLE XLV
PERFORMANCE METRICS FOR EXPANDED QUERIES

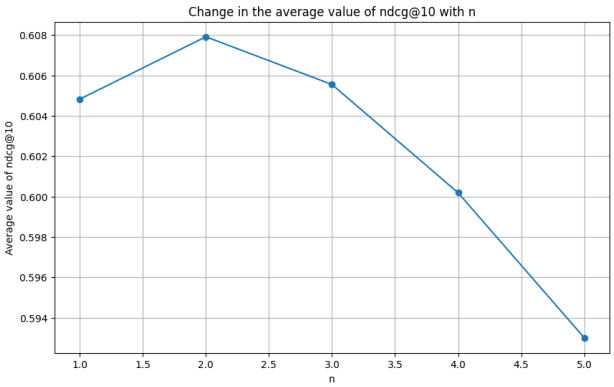
Recall@1000	MAP	NDCG@10	n
0.784	0.383	0.557	1
0.79	0.398	0.577	2
0.788	0.398	0.572	3
0.779	0.391	0.563	4
0.766	0.38	0.551	5

TABLE XLVI
PERFORMANCE METRICS FOR WEB PAGE

Recall@1000	MAP	NDCG@10	n
0.783	0.397	0.61	1
0.786	0.403	0.61	2
0.782	0.399	0.606	3
0.773	0.392	0.601	4
0.761	0.385	0.597	5







Step 2 : Using GRF + PRF :

TABLE XLVII
PERFORMANCE METRICS FOR $n = 2$

Subtask	Recall@1000	MAP	NDCG@10
entities	0.810	0.434	0.590
entities_cot	0.803	0.447	0.597
keywords	0.828	0.442	0.604
keywords_cot	0.810	0.432	0.594
facts	0.822	0.457	0.640
summary	0.824	0.457	0.623
document	0.842	0.472	0.636
essay	0.843	0.469	0.630
news	0.835	0.473	0.638
alt_sentences	0.762	0.4	0.592
expanded_queries	0.826	0.443	0.598
web_page	0.808	0.437	0.615

Step 3 : GRP + PRF + Combining subtasks

TABLE XLVIII
PERFORMANCE METRICS AT $n = 2$

Subtasks	Recall@1000	MAP	NDCG@10
document, news	0.845	0.492	0.654
document, essay	0.847	0.491	0.642
document, summary	0.848	0.485	0.64
news, essay	0.842	0.481	0.631
news, summary	0.84	0.484	0.638
essay, summary	0.847	0.479	0.632
document, news, essay	0.843	0.486	0.644
document, news, summary	0.844	0.49	0.644
document, essay, summary	0.847	0.489	0.649
news, essay, summary	0.839	0.481	0.647
document, news, essay, summary	0.844	0.486	0.647

Step 4 : Combining all subtasks:

TABLE XLIX
PERFORMANCE METRICS FOR GPT-4

Recall@1000	MAP	NDCG@10	n
0.835	0.473	0.638	2

E. Gemini

Step 1 : Using GRF:

TABLE L
PERFORMANCE METRICS FOR ENTITIES

Recall@1000	MAP	NDCG@10	n
0.809	0.43	0.59	1
0.803	0.427	0.587	2
0.788	0.416	0.589	3
0.774	0.403	0.582	4
0.761	0.386	0.567	5

TABLE LI
PERFORMANCE METRICS FOR CHAIN OF THOUGHT ENTITIES

Recall@1000	MAP	NDCG@10	n
0.782	0.407	0.563	1
0.784	0.409	0.574	2
0.78	0.404	0.577	3
0.776	0.398	0.574	4
0.768	0.392	0.567	5

TABLE LII
PERFORMANCE METRICS FOR KEYWORDS

Recall@1000	MAP	NDCG@10	n
0.806	0.407	0.554	1
0.802	0.411	0.578	2
0.786	0.405	0.572	3
0.77	0.393	0.565	4
0.755	0.38	0.559	5

TABLE LIII
PERFORMANCE METRICS FOR CHAIN OF THOUGHT KEYWORDS

Recall@1000	MAP	NDCG@10	n
0.808	0.405	0.572	1
0.808	0.41	0.58	2
0.802	0.41	0.589	3
0.793	0.402	0.583	4
0.783	0.394	0.57	5

TABLE LIV
PERFORMANCE METRICS FOR FACTS

Recall@1000	MAP	NDCG@10	n
0.804	0.426	0.639	1
0.808	0.433	0.64	2
0.808	0.435	0.643	3
0.803	0.434	0.648	4
0.799	0.429	0.64	5

TABLE LV
PERFORMANCE METRICS FOR SUMMARY

Recall@1000	MAP	NDCG@10	n
0.787	0.407	0.61	1
0.778	0.4	0.616	2
0.761	0.39	0.612	3
0.747	0.379	0.602	4
0.737	0.367	0.586	5

TABLE LVI
PERFORMANCE METRICS FOR DOCUMENTS

Recall@1000	MAP	NDCG@10	n
0.772	0.429	0.639	1
0.778	0.435	0.642	2
0.783	0.43	0.633	3
0.781	0.422	0.624	4
0.775	0.41	0.611	5

TABLE LVII
PERFORMANCE METRICS FOR ESSAYS

Recall@1000	MAP	NDCG@10	n
0.787	0.405	0.61	1
0.792	0.42	0.626	2
0.787	0.42	0.625	3
0.783	0.409	0.616	4
0.776	0.398	0.611	5

TABLE LVIII
PERFORMANCE METRICS FOR NEWS

Recall@1000	MAP	NDCG@10	n
0.781	0.397	0.602	1
0.782	0.394	0.602	2
0.776	0.389	0.602	3
0.768	0.382	0.6	4
0.756	0.371	0.591	5

TABLE LIX
PERFORMANCE METRICS FOR ALTERNATE SENTENCES

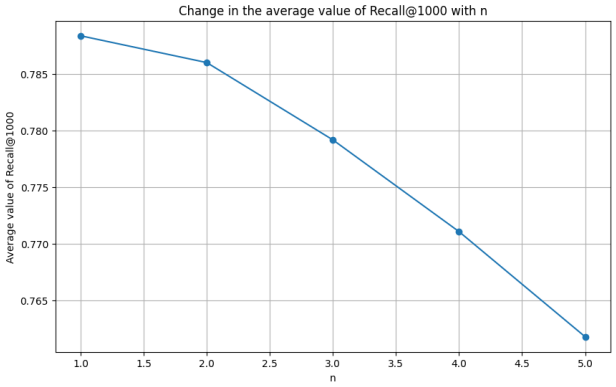
Recall@1000	MAP	NDCG@10	n
0.776	0.406	0.647	1
0.763	0.402	0.643	2
0.762	0.396	0.644	3
0.761	0.39	0.636	4
0.757	0.383	0.628	5

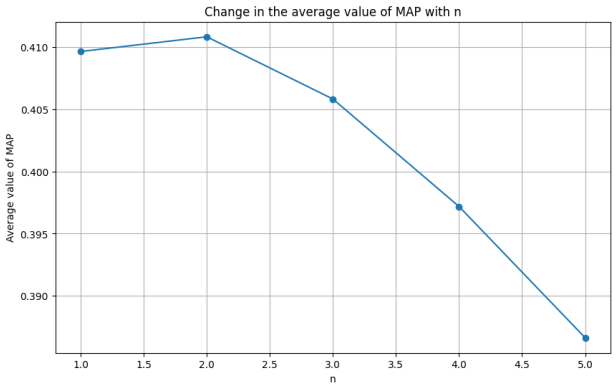
TABLE LX
PERFORMANCE METRICS FOR EXPANDED QUERIES

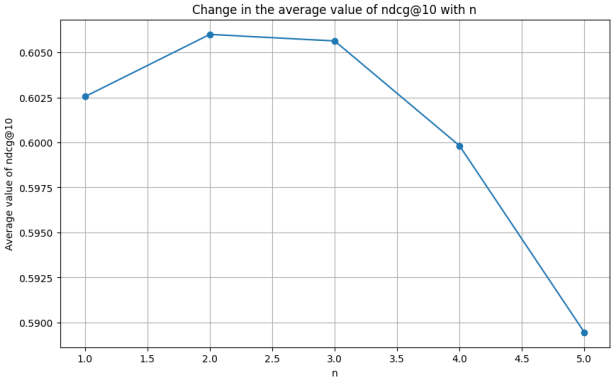
Recall@1000	MAP	NDCG@10	n
0.76	0.387	0.602	1
0.748	0.378	0.578	2
0.738	0.369	0.576	3
0.726	0.357	0.568	4
0.713	0.343	0.554	5

TABLE LXI
PERFORMANCE METRICS FOR WEB PAGES

Recall@1000	MAP	NDCG@10	n
0.794	0.406	0.612	1
0.789	0.402	0.612	2
0.777	0.398	0.611	3
0.768	0.388	0.606	4
0.762	0.378	0.598	5







Step 2 : Using GRF + PRF :

TABLE LXII
PERFORMANCE METRICS FOR $n = 1$

Subtask	Recall@1000	MAP	NDCG@10
entities	0.823	0.448	0.591
entities_cot	0.8	0.427	0.581
keywords	0.825	0.423	0.553
keywords_cot	0.828	0.435	0.583
facts	0.832	0.458	0.648
summary	0.811	0.435	0.611
document	0.814	0.47	0.636
essay	0.807	0.448	0.614
news	0.822	0.436	0.609
alt_sentences	0.817	0.446	0.655
expanded_queries	0.785	0.417	0.612
web_page	0.833	0.454	0.622

TABLE LXIII
PERFORMANCE METRICS FOR $n = 2$

Subtask	Recall@1000	MAP	NDCG@10
entities	0.825	0.45	0.587
entities_cot	0.798	0.425	0.577
keywords	0.829	0.429	0.564
keywords_cot	0.828	0.44	0.595
facts	0.829	0.458	0.662
summary	0.815	0.436	0.621
document	0.818	0.475	0.639
essay	0.823	0.467	0.624
news	0.819	0.431	0.612
alt_sentences	0.812	0.442	0.657
expanded_queries	0.791	0.413	0.602
web_page	0.835	0.45	0.62

Step 3 : GRP + PRF + Combining subtasks

TABLE LXIV
PERFORMANCE METRICS FOR $n = 1$

Subtasks	Recall@1000	MAP	NDCG@10
facts, document	0.834	0.484	0.66
facts, alt_sentences	0.845	0.491	0.702
facts, web_page	0.849	0.483	0.663
document, alt_sentences	0.842	0.495	0.691
document, web_page	0.846	0.497	0.677
alt_sentences, web_page	0.841	0.463	0.674
facts, document, alt_sentences	0.841	0.499	0.671
facts, document, web_page	0.846	0.49	0.674
facts, alt_sentences, web_page	0.851	0.494	0.692
document, alt_sentences, web_page	0.847	0.505	0.699
facts, document, alt_sentences, web_page	0.855	0.506	0.698

TABLE LXV
PERFORMANCE METRICS FOR $n = 2$

Subtasks	Recall@1000	MAP	NDCG@10
facts, document	0.834	0.484	0.658
facts, alt_sentences	0.842	0.488	0.7
facts, web_page	0.849	0.488	0.667
document, alt_sentences	0.835	0.491	0.682
document, web_page	0.843	0.493	0.669
alt_sentences, web_page	0.835	0.458	0.666
facts, document, alt_sentences	0.839	0.499	0.675
facts, document, web_page	0.845	0.494	0.671
facts, alt_sentences, web_page	0.851	0.494	0.695
document, alt_sentences, web_page	0.847	0.502	0.692
facts, document, alt_sentences, web_page	0.855	0.505	0.697

Step 4 : Combining all subtasks:

TABLE LXVI
PERFORMANCE METRICS FOR GEMINI

Recall@1000	MAP	NDCG@10	n
0.857	0.504	0.675	1

TABLE LXVII
PERFORMANCE METRICS FOR GEMINI

Recall@1000	MAP	NDCG@10	n
0.856	0.504	0.677	2

VIII. RAG: RETRIEVAL-AUGMENTED GENERATION

RAG paradigm fetches (using a retrieval model) relevant documents from the corpus as context for the language model and then generates an answer for the input query directly using the language model. As our dataset contains about 3.2 million documents, storing the whole document content in the index would be quite difficult, so we stored only the document titles for each of the queries after the first retrieval. Then, we prompted the Large Language models with the queries, giving the document titles as a context for generating an answer for each of the queries.

TABLE LXVIII
PERFORMANCE METRICS FOR RAG BARD

Recall@1000	MAP	NDCG@10	n
0.49	0.265	0.351	1
0.502	0.263	0.348	2
0.528	0.268	0.366	3
0.572	0.281	0.398	4
0.583	0.277	0.397	5

TABLE LXIX
PERFORMANCE METRICS FOR RAG GPT

Recall@1000	MAP	NDCG@10	n
0.799	0.436	0.572	1
0.801	0.442	0.571	2
0.803	0.442	0.57	3
0.801	0.439	0.569	4
0.797	0.431	0.571	5

TABLE LXX
PERFORMANCE METRICS FOR RAG GPT4

Recall@1000	MAP	NDCG@10	n
0.812	0.46	0.648	1
0.814	0.461	0.648	2
0.814	0.458	0.647	3
0.809	0.455	0.64	4
0.808	0.453	0.637	5

TABLE LXXI
PERFORMANCE METRICS FOR RAG GEMINI

Recall@1000	MAP	NDCG@10	n
0.811	0.436	0.617	1
0.809	0.444	0.619	2
0.81	0.444	0.609	3
0.809	0.443	0.608	4
0.809	0.439	0.601	5

IX. COMBINING LLMS

Here, we have tried combining different combination LLM outputs to see if it has an overall improvement in the recall@1000, map and ndcg@10 values.

A. Combining LLMs subtasks - 1

Here, we combined the subtasks having the best recall@1000 values generated from the combinations generated using each LLM. (See tables XV, XXXI, XLVIII, LXV for the same). The following subtasks per LLM were used:

LLM Name	Subtask Name
Bard	document, essay
GPT	document, alt_sentences
GPT-4	document, summary
Gemini	facts, document, alt_sentences, web_page

TABLE LXXII
PERFORMANCE METRICS FOR $(n = 2)$

Recall@1000	MAP	NDCG@10
0.856	0.508	0.676

B. Combining LLMs subtasks - 2

Here, we combined the subtasks having the best MAP values generated from the combinations generated using each LLM. (See tables XV,XXXI, XLVIII, LXV for the same). The following subtasks per LLM were used:

LLM Name	Task Name
Bard	facts, document, entities_cot, essay
GPT	document, alt_sentences
GPT-4	document, news
Gemini	facts, document, alt_sentences, web_page

TABLE LXXIII
PERFORMANCE METRICS FOR $n = 2$

Recall@1000	MAP	NDCG@10
0.857	0.516	0.695

C. Combining LLMs subtasks - 3

Here, we combined the subtasks having the best NDCG@10 values generated from the combinations generated using each LLM. (See tables XV,XXXI, XLVIII, LXV for the same). The following subtasks per LLM were used:

LLM Name	Task Name
Bard	Facts, Document, Entities CoT, Essay
GPT	Document, Alt Sentences
GPT-4	Document, News
Gemini	Facts, Alt Sentences

TABLE LXXIV
PERFORMANCE METRICS FOR ($n = 2$)

Recall	MAP	NDCG
0.858	0.515	0.691

X. COMPARISON WITH OTHER METHODS AND PREVIOUS GRF RESULTS

TABLE LXXV
PERFORMANCE METRICS FOR DL-19

Model	Recall@1000	MAP	NDCG@10
BM25+Bo1	0.756	0.385	0.568
CEQE-MaxPool	0.746	0.378	0.518
SPLADE+RM3	0.651	0.328	0.566
TCT+PRF	0.684	0.378	0.670
ColBERT-PRF	0.625	0.385	0.668
GRF ([12])	0.797	0.441	0.620
Combining LLMs subtasks - 1	**0.858**	0.515	0.691
Combining LLMs subtasks - 2	0.857	**0.516**	0.695
Combination(facts, alt_sentences) - Gemini	0.845	0.491	**0.702**

XI. LEARNING OUTCOMES

This section will contain a list (strictly) of skills, both technical and non-technical, you acquired while you worked on the project.

A. Technical Skills:

1) **Information Retrieval Techniques:** The project involves the use of retrieval models like BM25 for ranking documents based on query relevance.
2) **Query Expansion:** Techniques such as Bo1 Divergence from Randomness query expansion model are employed to enhance the original query by incorporating terms from feedback documents.
3) **Parameter Tuning:** The process of tuning parameters, such as k_1 and b in the BM25 model, using techniques like binary search to optimize retrieval performance.
4) **Algorithm Optimization:** Optimization techniques are employed to reduce computation time, as evidenced by the reduction in simulations required from 210 to 3 using binary search.
5) **Data Analysis:** Evaluation parameters like MAP, Recall, and NDCG are used to measure the effectiveness of different approaches, indicating the need for data analysis skills.
6) **Experiment Design and Evaluation:** Designing experiments, selecting appropriate evaluation metrics, and interpreting results are crucial aspects of the project.
7) **Pipeline Development:** Developing complex pipelines involving multiple components like BM25, Bo1 model, and large language models.
8) **Python Programming:** Given that Python is commonly used in NLP and ML projects, proficiency in Python programming is essential for implementing algorithms and building the project pipeline.
9) **Documentation:** Documenting the project's methodologies, findings, and code is essential for reproducibility and knowledge sharing within the team or community.

,

B. Non-Technical Skills:

1) **Problem-solving**: Developing solutions to challenges encountered during the project.
2) **Time management**: Prioritizing tasks, setting deadlines, and managing time effectively to ensure timely completion of project milestones.
3) **Adaptability**: Flexibility in adapting to changes in project requirements, scope, or constraints.
4) **Research**: Conducting research to explore new methodologies, algorithms, or approaches to address project goals and challenges.

XII. CONCLUSION

1) **Binary Search optimization**: By tuning the BM25 parameters using the Binary Search method compared to the traditional Grid Search method, we gained an increase of 1.42% in Recall@1000, 2.69% in MAP and 5.65% in ndcg@10. Also, we decreased the computation time by 70, as earlier in the Grid Search method, it took 210 simulations per subtask, while our Binary Search tuning method reduces it to 3 simulations per subtask. (Each simulation contains specific parameters for appropriate query lengths).
2) **The weighted term** n: By our calculations, it shows that $n = 2$ is generally good enough to obtain the results, but it varies according to the subtasks and the Large Language model.
3) **Performing combinations of the best subtasks rather than combining them all**: Mackie et al. [12] obtained the best results using GRF by combining all the subtasks, but we prove in this paper that instead of combining all, performing combination on some of the best subtasks can significantly improve the results. For GPT-3.5, Recall@1000 increased by 0.96%, MAP by 3.46%, and NDCG@10 by 5.66%. For GPT-4, Recall@1000 increased by 1.56%, MAP by 4.02%, and NDCG@10 by 2.51%. For BARD, Recall@1000 increased by 0.84%, MAP by 2.48%, and NDCG@10 by 2.17%. And for Gemini, Recall@1000 decreased by 0.23%, MAP increased by 0.40%, and NDCG@10 increased by 3.10%.

So, we observed an increase in all the cases except the Recall@1000 in Gemini.

4) **Comparison with the previous methods** We obtained the best Recall@1000, MAP and NDCG@10 compared with the overall results in [12]. Our Recall@1000 is better by 7.65%, MAP by 17%, and NDCG@10 by 4.77%. We also observe that these results are obtained either by combining some of the best subtasks within the same LLM or within the different LLMs.

XIII. Contributions

- Analyzing the research papers in the relevant domain.
- Implementing the binary search method to reduce the computation time.
- Adding the pseudo-relevant feedback after retrieving the results from the queries generated by the Generative Relevance Feedback.
- Adding the term weights to retain the original query,
- Filtering the value of n from the first step to reduce the computation in the second step.
- Generating the different subtasks using GPT-3.5, GPT-4, BARD, and Gemini.
- Performing combinations on the best subtasks to improve the results.
- Performing Retrieval Augmented Generation.

A. My Individual Contribution

- Analyzing the research papers in the relevant domain.
- Adding the pseudo-relevant feedback after retrieving the results from the queries generated by the Generative Relevance Feedback.
- Performing combinations on the best subtasks to improve the results.
- Generating the different subtasks using GPT-3.5 and GPT-4.
- Performing Retrieval Augmented Generation.

B. Other Members' Contributions

- Analyzing the research papers in the relevant domain.
- Implementing the binary search method to reduce the computation time.
- Adding the term weights to retain the original query.
- Filtering the value of n from the first step to reduce the computation in the second step.
- Generating the different subtasks using BARD and Gemini.

References

[1] Arora, D., Kini, A., Chowdhury, S. R., Natarajan, N., Sinha, G., and Sharma, A. Gar-meets-rag paradigm for zero-shot information retrieval, 2023.

[2] Bhogal, J., Macfarlane, A., and Smith, P. A review of ontology based query expansion. *Inf. Process. Manage. 43*, 4 (jul 2007), 866–886.

[3] Craswell, N., Mitra, B., Yilmaz, E., Campos, D., and Voorhees, E. M. Overview of the trec 2019 deep learning track, 2020.

[4] Craswell, N., and Szummer, M. Random walks on the click graph. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2007), SIGIR '07, Association for Computing Machinery, p. 239–246.

[5] Diaz, F., Mitra, B., and Craswell, N. Query expansion with locally-trained word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (Berlin, Germany, Aug. 2016), K. Erk and N. A. Smith, Eds., Association for Computational Linguistics, pp. 367–377.

[6] Feng, D. D. F., Siu, W.-C., and Zhang, H.-J. *Multimedia Information Retrieval and Management: Technological Fundamentals and Applications*. Springer, 01 2003.

[7] Giambattista, A. Probability models for information retrieval based on divergence from randomness, 2003.

[8] Imani, A., Vakili, A., Montazer, A., and Shakery, A. Deep neural networks for query expansion using word embeddings. In *Advances in Information Retrieval: 41st European Conference on IR Research, ECIR 2019, Cologne, Germany, April 14–18, 2019, Proceedings, Part II* (Berlin, Heidelberg, 2019), Springer-Verlag, p. 203–210.

[9] Jones, R., Rey, B., Madani, O., and Greiner, W. Generating query substitutions. In *Proceedings of the 15th International Conference on World Wide Web* (New York, NY, USA, 2006), WWW '06, Association for Computing Machinery, p. 387–396.

[10] Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and tau Yih, W. Dense passage retrieval for open-domain question answering, 2020.

[11] Kuzi, S., Shtok, A., and Kurland, O. Query expansion using word embeddings. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management* (New York, NY, USA, 2016), CIKM '16, Association for Computing Machinery, p. 1929–1932.

[12] Mackie, I., Chatterjee, S., and Dalton, J. Generative relevance feedback with large language models. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (July 2023), SIGIR '23, ACM.

[13] Qu, Y., Ding, Y., Liu, J., Liu, K., Ren, R., Zhao, W. X., Dong, D., Wu, H., and Wang, H. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering, 2021.

[14] Roy, D., Paul, D., Mitra, M., and Garain, U. Using word embeddings for automatic query expansion, 2016.

[15] Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., and Gurevych, I. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models, 2021.

[16] Voorhees, E. M. Query expansion using lexical-semantic relations. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Berlin, Heidelberg, 1994), SIGIR '94, Springer-Verlag, p. 61–69.

[17] Wang, L., Yang, N., and Wei, F. Query2doc: Query expansion with large language models, 2023.

[18] Xiong, L., Xiong, C., Li, Y., Tang, K.-F., Liu, J., Bennett, P., Ahmed, J., and Overwijk, A. Approximate nearest neighbor negative contrastive learning for dense text retrieval, 2020.

[19] Zheng, Z., Hui, K., He, B., Han, X., Sun, L., and Yates, A. Contextualized query expansion via unsupervised chunk selection for text retrieval. *Inf. Process. Manage. 58*, 5 (sep 2021).

[20] Zukerman, I., and Raskutti, B. Lexical query paraphrasing for document retrieval. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1* (USA, 2002), COLING '02, Association for Computational Linguistics, p. 1–7.

APPENDIX A
BINARY SEARCH METHOD FOR FINDING THE PARAMETERS

```
def binary_search_for_parameters (topics, idx = 0):
  left, right = 0, 1
  precision = 1e-2

  while abs(right - left) > precision:
    mid = (left + right) / 2
    midLeft = (left + mid) / 2
    midRight = (mid + right) / 2

    resLeft = get_evaluation(pt.BatchRetrieve(index,
     wmodel="BM25", controls={"bm25.b" : midLeft}),
    topics, midLeft, None)
    resRight = get_evaluation(pt.BatchRetrieve(index
    , wmodel="BM25", controls={"bm25.b" : midRight})
    , topics, midRight, None)

    if resLeft[idx] >= resRight[idx]:
      right = mid
    else:
      left = mid

  return (left + right) / 2
```