# GENRA: Enhancing Zero-shot Retrieval with Rank Aggregation

Georgios Katsimpras and Georgios Paliouras
NCSR Demokritos, Athens, Greece
{gkatsibras,paliourg}@iit.demokritos.gr

***Abstract*—Large Language Models (LLMs) have been shown to effectively perform zero-shot document retrieval, a process that typically consists of two steps: i) retrieving relevant documents, and ii) re-ranking them based on their relevance to the query. This paper presents GENRA, a new approach to zero-shot document retrieval that incorporates rank aggregation to improve retrieval effectiveness. Given a query, GENRA first utilizes LLMs to generate informative passages that capture the query's intent. These passages are then employed to guide the retrieval process, selecting similar documents from the corpus. Next, we use LLMs again for a second refinement step. This step can be configured for either direct relevance assessment of each retrieved document or for re-ranking the retrieved documents. Ultimately, both approaches ensure that only the most relevant documents are kept. Upon this filtered set of documents, we perform multi-document retrieval, generating individual rankings for each document. As a final step, GENRA leverages rank aggregation, combining the individual rankings to produce a single refined ranking. Extensive experiments on benchmark datasets demonstrate that GENRA improves existing approaches, highlighting the effectiveness of the proposed methodology in zero-shot retrieval.**

***Index Terms*—Information Retrieval, Zero-shot Retrieval, Large Language Models, Rank Aggregation.**

## I. Introduction

Recent studies in zero-shot retrieval have demonstrated remarkable advancements, significantly improving the effectiveness of retrievers with the use of encoders like BERT [10] and Contriever [17]. With the emergence of Large Language Models (LLMs) [6], [41], [46], research focused on how to leverage LLMs for information retrieval tasks, such as zero-shot retrieval. Early zero-shot ranking with LLMs relied on methods that score each query-document pair and select the top-scoring pairs [26]. Researchers have attempted to boost these methods by enriching contextual information to help LLMs understand the relationships between queries and documents. This often involves using LLMs to generate additional queries, passages, or other relevant content [32], [24]. These enhancements have significantly improved retrieval performance, especially for unseen (zero-shot) queries.

In a typical retrieval setting, as shown in Figure 1, queries and documents are embedded in a shared representation space to enable efficient search. The success of the entire approach depends strongly on the quality of the results of the retrieval step. However, LLMs can generate potentially non-factual or nonsensical content (e.g. "hallucinations"), and their performance is susceptible to factors like prompt order and input length, which can hurt the performance of the retriever [51].

To address this problem, some studies [26], [45] propose employing LLMs as relevance assessors, providing individual relevance judgments for each query-document pair. These approaches aim to enhance trustworthiness by leveraging the LLM's strengths in understanding nuances and identifying potentially irrelevant content. Additionally, recent work [43], [37] suggests incorporating re-ranking models into the retrieval process. Such models process a ranked list of documents and directly produce a reordered ranking.

However, most existing methods focus solely on retrieval without a separate relevance assessment step, which could be beneficial. To address this gap, our approach utilizes rank aggregation techniques to combine individual rankings generated by separate retrieval and relevance assessment subprocesses. This allows our method to combine the strengths of the two stages, leading to a more refined and accurate final ranking of documents.

While combining multiple rankings (rank aggregation) has proven highly effective in various domains, like bioinformatics [47] and recommendation systems [4], its use with LLMs in zero-shot retrieval has not been explored thus far.

Our method (Figure 1), named GENRA, first utilizes the LLM to generate informative passages that capture the query's intent. These passages serve as query variants, guiding the search for similar documents. Next, we leverage the LLM's capabilities to further refine the initial retrieval. This can be achieved through either direct relevance assessment (generating 'yes' or 'no' judgments) or by employing a re-ranking model to optimize the document order and select the top-ranked ones. This step acts as a verification filter, ensuring the candidate documents can address the given query. Using each verified document as a query, we retrieve new documents from the corpus, generating document-specific rankings that capture diverse facets of the query. By combining these individual rankings through a rank aggregation method, we mitigate potential biases inherent in any single ranking and achieve a more accurate final ranking.

Thus, the main contributions of the paper are the following:

We propose a new pipeline for zero-shot retrieval, which is based on the synergy between LLMs and rank aggregation. We confirm through experimentation on several benchmark datasets the effectiveness of the proposed approach.

GENRA can be combined with different LLMs and different rank aggregation methodologies.

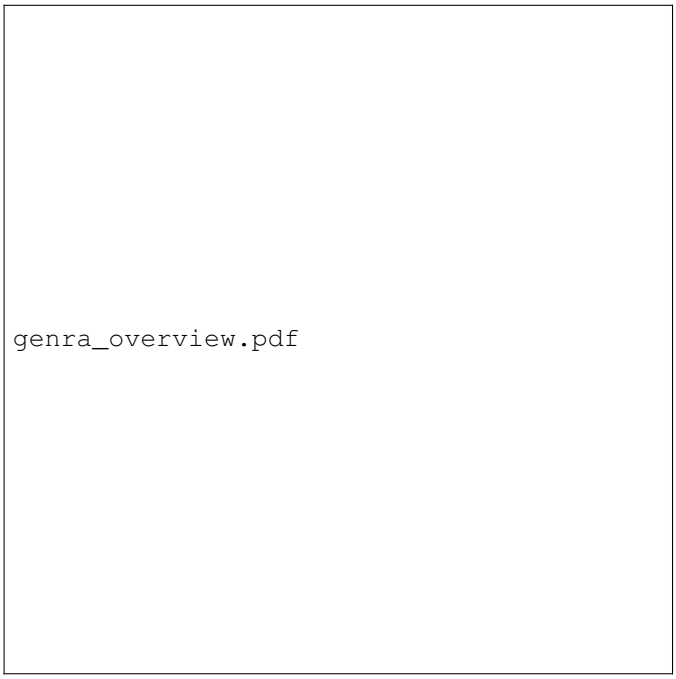


**Fig. 1:** GENRA retrieval can capture both relevance and validity of the documents.

## II. Related Work

Zero-shot retrieval has experienced great advancements in recent years, largely driven by the development and adoption of pre-trained models [20], [7], [42]. Researchers have explored a diverse range of approaches, including contrastive pre-training [14], [17], contextualized models [21], and hybrid settings [15], [29].

With the emergence of LLMs, research focused on the capabilities of generative models to improve the query representation, through query generation and rewriting [13], [18], [50], or context generation [32]. In the same line of work [5] and [30], LLMs are used to create synthetic queries for documents. These artificial query-document pairs serve as training data for retrievers or re-rankers, aiming to enhance retrieval effectiveness. Similarly, HyDE [16] employs LLMs to enrich queries by generating hypothetical documents.

Beyond retrieval, LLMs have also been employed for relevance assessment, helping to filter out irrelevant or off-topic documents generated at the retrieval stage. The goal of LLM assessors is to provide a relevance label to each query-document pair. Such methods [26] and [40] have been used to refine the retrieved document sets and enhance relevance. Other recent work [25], [52] proposes the use of more fine-grained relevance judgements instead of binary filters. Moreover [11] suggest to incorporate these fine-grained relevance judgements into the LLM prompting process.

Taking fine-grained relevance judgements one step further, re-ranking models [43], [31] have been shown to achieve improved retrieval performance with the use of popular generative models like chatGPT and GPT-4 [1]. In the same line of work [37] utilize passage relevance labels, obtained either from human judgments or through a GPT-based teacher model. However, the computational cost associated with these models can be significant, requiring substantial resources for both training and inference. Furthermore, relying on black-box models poses significant challenges for academic researchers, including substantial cost barriers and restricted access due to their proprietary nature. Previous studies have also explored methods for aggregating query or document representations to improve performance in zero-shot document retrieval [35], [23]. However, the question of how to effectively aggregate per-document rankings has received limited attention. While various rank aggregation techniques exist [4], their potential in the context of zero-shot retrieval has not been explored.

Our study bridges this gap, by incorporating different rank aggregation strategies within the GENRA pipeline. Drawing inspiration from and building upon previous work, GENRA introduces an effective approach to zero-shot document retrieval, and demonstrates the potential of incorporating rank aggregation techniques for improved retrieval results.

## III. Preliminaries

Given a query $q$ and a set of documents $D = \{d_1, d_2, \ldots, d_n\}$ the goal of retrieval is to rank $D$ in descending order of relevance to query $q$ . Sparse retrieval methods, like BM25 [39], rely on keyword-based similarity to estimate relevance. The similarity metric, denoted by $s_{q,d}$ , is typically based on term frequencies and document lengths, ignoring semantic relationships between terms. On the other hand, dense retrieval leverages embedding similarity to assess the relevance between a query $q$ and document $d$ . Using an encoder $\epsilon$ , queries and documents are converted into vectors, $v_q$ and $v_d$ , respectively. The inner product of these vectors serves as the similarity metric $s_{q,d} = \langle \epsilon(q), \epsilon(d) \rangle$ . Upon inferring the similarity scores for each document, we can readily construct a ranked document list $\sigma_q^n = \{\{d_i\}_{i=1}^n; s_{q,d_1} > s_{q,d_2} > \ldots > s_{q,d_n}\}$ . In zero-shot retrieval, the whole process is performed without the aid of labeled training data, posing a significant challenge.

## IV. Methodology

In Figure 2, we present the proposed GENRA approach, which consists of three main steps. The initial step aims to retrieve potentially relevant documents from a large collection. The retriever at this stage is assisted by a LLM that generates passages, based on the query. In the second step, the relevance of each retrieved document is further assessed and only highly-relevant documents are kept. This is achieved either by asking a LLM to filter-out irrelevant documents or by employing a pre-trained model to re-rank the documents and keep the top most-relevant ones. Once the relevant documents are selected, similar documents are retrieved, generating a ranking per document. In the third and last step of GENRA, a rank aggregation method combines the individual rankings into a single more accurate ranking. Each of the three steps is detailed in the following subsections. Notably, the method


**Fig. 2:** The key steps of GENRA: In step (a), LLMs generate informative passages that capture the intent of the query. These passages are then used to guide the retrieval process, selecting relevant documents from a large collection. In step (b), LLMs assess the relevance of each retrieved document, keeping only the $m$ most relevant results. Finally, step (c) employs a rank aggregation technique combining the individual rankings into a single one.

relies solely on LLM inference, without the requirement for dedicated training.

### A. Passage Generation

Our method draws inspiration from related work [16], [32] that demonstrates the significant benefits of enriching query and document vector representations with generated contexts. Taking a similar approach, we seek to expand the query beyond its original text by incorporating complementary information.

The proposed method, GENRA, achieves this by instructing a LLM to generate a set of informative passages $P = \{p_1, p_2, \ldots, p_n\}$ that capture the context and intent behind the query as $P = LLM(instruction, q)$ . Our prompt template for generating the passages is depicted in Figure 2a.

Subsequently, we encode these generated passages using a pre-trained encoder $\epsilon$ to obtain a dense vector representation for the query as $v_{\bar{q}} = \frac{1}{n}\sum_{i=1}^{n} \epsilon(p_i)$ , similar to Gao et al. (2022). This vector, encompassing the aggregated knowledge of the generated passages, serves as the query representation for the initial retrieval processes. With this enhanced query representation, we retrieve the $k$ most relevant documents $D_k = \{d_1, d_2, \ldots, d_k\}$ , with $\sigma_q^k = \{\{d_i\}_{i=1}^k; s_{q,d_1} > s_{q,d_2} > \ldots > s_{q,d_k}\}$ , ensuring the most promising candidates are prioritized for further analysis. As an alternative to the query encoder, we use BM25 on the concatenated passages generated by the LLM.

### B. Relevance Assessment

Recent studies [26], [40] have highlighted the potential benefits of leveraging LLM insights for enhancing the relevance of retrieved documents. In line with these findings, we incorporate a relevance assessment step. This step enables users to select between LLM-based relevance judgments or a pre-trained re-ranking model.

*1) LLM-based filtering:* Given the ranking $\sigma_q^k$ , we select a subset of documents, while maintaining their relative order. Our key objective is to ensure that documents containing the correct answer are included and prioritized within this filtered set. To achieve this we instruct a LLM to compute a relevance judgement (yes or no) for each document. In other words, given the query $q$ , we examine whether each $d \in D_k$ can support answering $q$ with $\rho_{q,d} = LLM(instruction, q, d) = (yes, no)$ . Note that the LLM's relevance judgments are based on the original query $q$ , to ensure direct alignment with the query's intent. While it is common to instruct LLMs to provide relevance assessments for multiple documents simultaneously, recent research by Liu et al. (2023) and Wang et al. (2023) revealed that LLMs tend to lose focus when processing lengthy contexts and that the order of prompts can significantly impact their responses. Motivated by these insights, we opt to process each document independently, in order to produce more accurate judgements. Additionally, relevance judgements are generated sequentially and in a zero-shot fashion without any fine-tuning. The instruction is simply concatenated with the document. Eventually, the documents judged to be relevant make it to the next stage. If the number of these documents exceeds a user-defined threshold $m$ , the top-$m$ are kept.

*2) Re-ranking:* As an alternative to the LLM-based relevance judgments, GENRA employs a pretrained re-ranking model for refining the initial retrieval results. Given the ranking $\sigma_q^k$ , a new ranking $\tilde{\sigma}_q^k$ is generated by the RankVicuna method [37]. From the re-ranked list of the documents, the top-$m$ ranked ones proceed to the next stage.

### C. Rank Aggregation

With the help of passage generation and relevance assessment, a refined document set $D_m \subset D_k$ is generated, comprising highly relevant real documents. For each of these documents, our method produces a separate set of relevant documents from the collection (ranking) and all the rankings are aggregated into a single high-quality one.

In this way, we aim to improve the diversity of the rankings and reduce the impact of documents incorrectly placed at high rank positions by an individual ranker [3]. We have tested several aggregation methods, including Outrank [12] and Dibra [2], and we found that a simple linear approach [38], which aggregates multiple rankings by summing the individual normalized scores of each item across all rankings, performed best.

An overview of GENRA is also given in Algorithm 1 (Appendix A.1). It is worth noting that, the zero-shot nature of each individual step enables our pipeline to operate across

diverse document collections, without the need for dataset-specific models or tuning.

## V. Results and Analysis

### A. Setup

In line with previous studies, we evaluated our method on the TREC 2019 and 2020 Deep Learning Tracks (DL19 and DL20) [8], [9], and five datasets from the BEIR benchmark (Covid, News, NFCorpus, Signal, and Touche) [44]. We directly assess our method's performance on the respective test sets. Following established practice, we report MAP, nDCG@10, Recall@10, and Recall@100 metrics for DL19 and DL20, and nDCG@10 for the BEIR datasets.

In our experiments, we utilize the pre-built indexes of the aforementioned datasets, extracted from the Pyserini toolkit [27]. For initial retrieval in step (a), we experimented with BM25 and Contriever, and chose the retrieval size to be $k = 100$ . Regarding the choice of LLMs, we opted for open-source ones that are publicly available through Huggingface [49]. Specifically, we conducted experiments using Solar [22] and Mistral [19]. We set the maximum number of new tokens for each generated passage to be 512.

Given our focus on zero-shot retrieval, our primary baselines consist of retrieval methods that do not require labeled data. Specifically, we use BM25 and Contriever as zero-shot lexicon-based and dense retrieval baselines, respectively. To strengthen our evaluation, we also include HyDE [16], a state-of-the-art approach in LLM-based retrieval, and RankVicuna [37], a state-of-the-art re-ranking model. For these models, we used the default settings suggested by their authors.

We conducted our experiments using two Nvidia RTX A6000-48GB GPUs on an AMD Ryzen Threadripper PRO 3955WX CPU. We used PyTorch [36], RankLLM and PyFlag to implement GENRA and relevant baselines. Our code is available at https://github.com/nneinn/genra.

### B. Ablation Study

In order to assess the importance of different features of the proposed approach, we ran a set of experiments on the DL-19 and DL-20 datasets.

*1) Number of Passages Generated:* First, we conducted experiments varying the number of passages generated per query (1, 2, 5, 10, 20). Each passage set underwent the same encoding and rank aggregation process within GENRA. We then evaluated the retrieved documents on the test data using nDCG@10.

Our results in Figure 3, reveal an initial performance boost as more passages are used, capturing more diverse information. However, this improvement plateaued beyond 10 passages for DL-19 and 5 passages for DL-20. This result aligns with findings from previous studies, which suggest that information diversity in query representations can enhance retrieval performance, but excessive redundancy can ultimately hinder it [33]. Determining the optimal number of passages depends on the specific information retrieval context and the desired balance between accuracy and efficiency. In the rest of the experiments we generate 10 passages per query.

num_passages.pdf

**Fig. 3:** Impact of the number of passages.

*2) Number of Relevant Documents:* Next, we assessed how the number $m$ of documents selected at the relevance assessment step, influences the overall retrieval effectiveness. We conducted experiments varying the number of relevant documents using 1, 5, 10 and 20 per query. Each configuration undergoes the complete GENRA pipeline, including passage retrieval, relevance assessment, and rank aggregation. We evaluated the final ranking using nDCG@10.

Based on the results presented in Figure 4, we observe a performance improvement as the number of relevant documents increases. However, this benefit diminishes beyond 5 documents. While verified passages can enhance trust and potentially improve relevance, incorporating too many can expose the method to possible misjudgements made by the LLM, leading to a decline in performance.

*3) Different Aggregations:* In this section we investigate the impact of different rank aggregation methods on GENRA's retrieval performance. We utilize various approaches available in PyFlag, including Linear, Borda, Outrank and DIBRA. For comparison, we also include models that don't use any rank aggregation, named w/oRA. These models follow the initial two stages of GENRA, but employ simple retrieval in the last step, using the aggregated document representations $(D_m)$ as a query. The rankings produced by each method were evaluated using the nDCG@10 and MAP metrics.

As shown in Table I, the Linear method consistently outperformed the other methods, achieving the highest NDCG and MAP scores in most cases. Interestingly, the other rank aggregation methods did not improve the model without rank aggregation. The Linear method's effectiveness might be

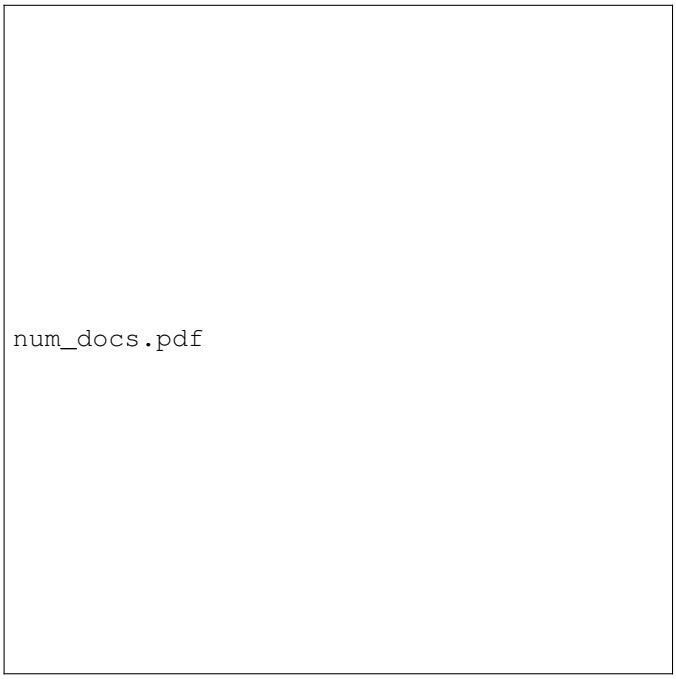


**Fig. 4:** Impact of the number of relevant documents


**Fig. 5:** Average time (seconds) required for processing a query.

**TABLE I:** Evaluation of different rank aggregation methods within the GENRA pipeline. The scores represent the results of each model using BM25 for retrieval (scores in parentheses correspond to the results using Contriever). The best result is indicated in bold.

| GENRA | DL19 | | DL20 | |
|---|---|---|---|---|
| | MAP | nDCG | MAP | nDCG |
| +Judgements w/oRA | 35.2 (42.0) | 56.1 (62.2) | 33.6 (33.2) | 50.5 (52.9) |
| +Borda | 32.9 (36.5) | 49.5 (52.8) | 30.3 (25.7) | 44.4 (39.7) |
| +DIBRA | 32.3 (37.1) | 49.7 (53.1) | 26.5 (24.8) | 39.9 (36.3) |
| +Outrank | 33.8 (37.6) | 52.2 (53.9) | 30.9 (27.3) | 45.6 (42.2) |
| +Linear | **35.5 (42.3)** | **57.2 (62.8)** | **34.2 (34.7)** | **52.0 (53.3)** |
| +RankVicuna w/oRA | 32.9 (32.5) | 67.1 (67.2) | 38.0 (35.3) | 65.5 (63.2) |
| +Borda | 32.2 (33.7) | 55.4 (61.1) | 31.2 (34.1) | 50.6 (55.1) |
| +DIBRA | 28.3 (30.7) | 52.9 (58.5) | 26.4 (29.3) | 46.4 (51.3) |
| +Outrank | 33.9 (34.1) | 56.8 (61.4) | 32.9 (34.8) | 53.4 (57.1) |
| +Linear | **35.7 (36.4)** | **63.4 (68.4)** | **38.0 (39.6)** | **64.8 (65.5)** |

attributed to the fact that it uses the actual ranking similarity scores, in contrast to positional methods, such as Borda and Outrank. Furthermore, the more sophisticated DIBRA method might require more careful tuning for each dataset, in order to achieve optimal results. Exhaustive parameter tuning of each rank aggregation method could have produced different results, but it would hurt the generality of the GENRA method. Overall, the effect of the linear rank aggregation seems positive, improving the results obtained without it.

*4) Model Efficiency:* Beyond effectiveness, it is crucial to consider the computational cost associated with the use of multiple retrieval steps. Each step requires resources and can impose additional time constraints on the retrieval process. This effect is highlighted in Figure 5, which presents the average processing time (in seconds) required for different models to process a single query on datasets DL19 and DL20. As expected, HyDE has the lowest processing time across both datasets, making it suitable for scenarios where efficiency is important. On the other hand, GENRA combined with relevance judgements is slower than HyDE but faster than RankVicuna. This suggests that GENRA+judgements might be a reasonable compromise between efficiency and potential effectiveness for some retrieval tasks.

### C. Passage Ranking

Having examined the role of each different component within GENRA, we then assessed its performance in a number of different retrieval tasks.

*1) TREC datasets:* As shown in Table II, GENRA consistently outperforms the baseline methods across both TREC datasets, DL19 and DL20. Additionally, GENRA achieves a significant improvement of 0.5 to 7.4 percentage points in MAP and nDCG@10, over its LLM-based competitor, HyDE. Additionally, the combination of GENRA with RankVicuna brings considerable improvements of 1.6 to 9.9 percentage points in MAP and $\mathbb{R}$@100 over the vanilla RankVicuna model, while maintaining or improving the scores of nDCG@10. Overall, GENRA consistently improves baseline systems leading to a boost in retrieval effectiveness across various metrics. These results highlight the effectiveness of combining relevance assessment and rank aggregation.

*2) BEIR datasets:* The situation is similar in the BEIR datasets (Table III). GENRA consistently outperforms the other baselines across all datasets, HyDE approaches the performance of GENRA in some cases (notably in the NFCorpus). Also, RankVicuna performs well on the Covid dataset, but GENRA achieves a higher performance on average ($+2$ percentage points). Comparing the results of different

**TABLE II:** Results on DL19 and DL20. The best result is shown in bold. The second-best is underlined for comparison.

| | DL19 | | | | DL20 | | | |
|---|---|---|---|---|---|---|---|---|
| | MAP | nDCG@10 | R@10 | R@100 | MAP | nDCG@10 | R@10 | R@100 |
| BM25 | 30.1 | 50.6 | 17.8 | 45.2 | 28.6 | 48.0 | 16.9 | 55.8 |
| Contriever | 24.0 | 44.5 | 14.1 | 48.9 | 24.0 | 42.1 | 23.1 | 51.4 |
| HyDE+mistral | 38.2 | 54.8 | 22.5 | 57.4 | 35.0 | 49.5 | 26.8 | 59.6 |
| GENRA+mistral+BM25 | 38.4 | 60.4 | 23.5 | 55.4 | 32.4 | 52.1 | 26.5 | 60.9 |
| GENRA+mistral+Contriever | 39.1 | 56.6 | 22.7 | 57.5 | 33.8 | 51.5 | 26.7 | 60.4 |
| HyDE+solar | 37.4 | 55.4 | 22.3 | 56.9 | 32.7 | 52.8 | 30.2 | 62.3 |
| GENRA+solar+BM25 | 35.5 | 57.2 | 20.3 | 55.9 | 34.2 | 52.0 | 26.9 | 62.3 |
| GENRA+solar+Contriever | 42.3 | 62.8 | 25.4 | 58.5 | 34.7 | 53.3 | 28.5 | 61.4 |
| RankVicuna | 32.9 | 67.1 | 24.9 | 49.1 | 38.0 | 65.5 | 34.4 | 55.9 |
| GENRA+RankVicuna+BM25 | 35.7 | 63.4 | 25.5 | 57.9 | 38.0 | 64.8 | 32.6 | 63.8 |
| GENRA+RankVicuna+Contriever | **36.4** | **68.4** | **25.5** | **54.3** | **39.6** | **65.5** | **35.4** | **63.5** |

**TABLE III:** Results on BEIR. Best performing are marked bold.

| Avg | nDCG@10 | Covid | News | NFCorpus | Signal | Touch |
|---|---|---|---|---|---|---|
| BM25 | 59.4 | 39.5 | 30.7 | 33.0 | 44.2 | 35.2 |
| Contriever | 27.3 | 34.8 | 31.7 | 23.3 | 16.0 | 26.7 |
| HyDE+mistral | 55.9 | 34.3 | 30.8 | 21.6 | 31.1 | 31.5 |
| GENRA+mistral+BM25 | 61.2 | 40.9 | 31.5 | 33.4 | 44.7 | 42.3 |
| GENRA+mistral+Contriever | 58.4 | 40.7 | 26.7 | 17.9 | 18.3 | 32.4 |
| HyDE+solar | 56.7 | 35.1 | 31.3 | 21.5 | 15.0 | 31.9 |
| GENRA+solar+BM25 | 61.5 | 41.2 | 32.3 | 33.6 | 45.4 | 42.8 |
| GENRA+solar+Contriever | **63.4** | **41.4** | **31.9** | **18.1** | **18.9** | **34.7** |
| RankVicuna | 81.1 | 46.9 | 33.0 | 33.2 | 29.6 | 44.7 |
| GENRA+RankVicuna+BM25 | 78.0 | 47.1 | 33.2 | 33.4 | 41.7 | 46.7 |
| GENRA+RankVicuna+Contriever | 70.2 | 43.3 | 30.4 | 25.6 | 24.5 | 38.8 |

retrievers within GENRA, it seems that BM25 leads more consistently to good results. GENRA with Contriever performs very well in some datasets, but not so well in others.

Among the tested LLMs, Solar achieved marginally better overall performance compared to Mistral. This could potentially be attributed to its larger size, consisting of 10 billion parameters compared to Mistral's 7 billion.

### *D. Summarizing Crisis Events*

Moving beyond the standard benchmarks, we evaluated our method on the CrisisFACTS 2022 task [34]. This task focuses on summarizing multiple streams of social media and news data related to a specific short-term crisis event, aiming to include factual information relevant to pre-defined queries. Given a set of queries $Q$ and documents $D$ , the goal is to return a list of $k$ most-relevant text snippets (namely "facts") along with their importance scores, forming a daily summary.

The CrisisFACTS dataset offers multi-stream data, tagged with ground truth summaries sourced from ICS-2009, Wikipedia, and NIST annotations. Following [34], we used Rouge-2 F1-Score and BERT-Score metrics to evaluate the performance of GENRA on the task. For comparison, we selected the top-performing methods from the CrisisFACTS 2022 challenge, namely unicamp and ohmkiz. In our method, we utilized Contriever for retrieval and Solar as the LLM, generating 10 candidate passages for each query. We set the LLM relevant documents to $m = 5$ and employed the linear rank aggregation method.

**TABLE IV:** Results on the CrisisFACTS 2022 dataset. The best performing result is indicated in bold. The second-best is underlined for comparison.

| | WIKI | | NIST | | ICS | |
|---|---|---|---|---|---|---|
| | BERT | Rouge | BERT | Rouge | BERT | Rouge |
| unicamp | 53.2 | 0.2 | 55.7 | 13.3 | 45.9 | 0.5 |
| ohmkiz | 56.4 | 0.3 | 56.4 | 14.7 | 45.0 | 0.1 |
| HyDE | 52.2 | 0.3 | 53.4 | 11.1 | 44.4 | 0.0 |
| GENRA+judgments | 53.0 | 0.3 | 56.1 | 12.6 | 46.1 | 0.5 |
| GENRA+RankVicuna | **54.2** | **0.3** | **56.0** | **12.3** | **46.5** | **0.6** |

As illustrated in Table IV, our approach produces summaries with fluency and factual accuracy comparable to the best methods, as measured by BERTScore and ROUGE-F1 scores. This level of performance is maintained across all ground truth summaries, even outperforming all other methods in some cases. Furthermore, in contrast to ohmkiz, which requires fine-tuning on question-document pairs, and unicamp, which utilizes proprietary OpenAI API calls, GENRA operates entirely unsupervised, leveraging readily available, open-source LLMs. This distinction eliminates the need for additional training data, and promotes usability.

## VI. Conclusion

In this work, we have introduced GENRA, a new approach to information retrieval that leverages LLMs and rank aggregation to enhance the effectiveness of zero-shot document retrieval. Our method involves three main steps. First, GENRA utilizes LLMs to create informative passages that serve as refined query representations. Then, the retrieved documents undergo a LLM-based relevance assessment, keeping only highly relevant ones. Finally, multi-document retrieval generates individual rankings for each verified document, which are further refined through rank aggregation, leading to the final ranking.

Extensive experiments on benchmark datasets demonstrated that GENRA consistently outperforms existing zero-shot approaches, in some cases achieving considerable improvements. Furthermore, the modular nature of GENRA facilitates further experimentation with different, possibly better, LLMs and rank aggregation methods.

In addition, we will be investigating alternative ways to incorporate relevance judgments into the passage generation instructions. Finally, we are interested in ways to improve the computational efficiency of GENRA, particularly for large-scale retrieval.


## Acknowledgements

This work was partially supported by the EU project CREXDATA (Critical Action Planning over Extreme-Scale Data), grant agreement ID: 101092749.


## Limitations

Our study focus on zero-shot retrieval with open-source LLMs, and shows that the combination of relevance assessment and rank aggregation can improve the quality of

the retrieval. However, the computational cost of GENRA is relatively high due to the need for multiple iterations of LLM-based inferences and document retrieval. This could compromise its usability in very large-scale scenarios, or when using systems with restricted computational resources.

While incorporating relevance judgments demonstrably enhances retrieval performance, our methodology utilizes only binary assessments. Recent research suggests that finer-grained relevance levels could yield further improvements. Therefore, while our approach prioritizes simplicity, it may sacrifice some potential performance gains compared to approaches using more granular relevance scales.

Our work prioritizes open-source LLMs to foster open and reproducible research within the academic community. This approach contrasts closed-source commercial APIs, like ChatGPT, which may achieve higher performance. Therefore, we appreciate the value of a broader benchmarking of GENRA's performance across various LLM models.

## REFERENCES


[1] Josh Achiam, Steven Adler, Sandhini Agarwal, et al. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

[2] Leonidas Akritidis, Athanasios Fevgas, Panayiotis Bozamis, and Yannis Manolopoulos. 2022. An unsupervised distance-based model for weighted rank aggregation with list pruning. *Expert Systems with Applications*, 202:117435.

[3] Javier Alcaraz, Mercedes Landete, and Juan F Monge. 2022. Rank aggregation: Models and algorithms. In *The Palgrave Handbook of Operations Research*, pages 153-178. Springer.

[4] Michal Balchanowski and Urszula Boryczka. 2023. A comparative study of rank aggregation methods in recommendation systems. *Entropy*, 25(1):132.

[5] Luiz Bonifacio, Hugo Abonizio, Marzieh Fadaee, and Rodrigo Nogueira. 2022. Inpars: Data augmentation for information retrieval using large language models. *arXiv preprint arXiv:2202.05144*.

[6] Tom Brown, Benjamin Mann, Nick Ryder, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877-1901.

[7] Wei-Cheng Chang, Felix X Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. 2020. Pre-training tasks for embedding-based large-scale retrieval. *arXiv preprint arXiv:2002.03932*.

[8] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. Overview of the TREC 2019 deep learning track. *ArXiv:2003.07820 [cs]*.

[9] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the TREC 2020 deep learning track. *ArXiv:2102.07662 [cs]*.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[11] Guglielmo Faggioli, Laura Dietz, Charles LA Clarke, et al. 2023. Perspectives on large language models for relevance judgment. In *Proceedings of the 2023 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 39-50.

[12] Mohamed Farah and Daniel Vanderpooten. 2007. An outranking approach for rank aggregation in information retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '07*, page 591-598, New York, NY, USA. Association for Computing Machinery.

[13] Jiazhan Feng, Chongyang Tao, Xiubo Geng, et al. 2023. Knowledge refinement via interaction between search engines and large language models. *arXiv preprint arXiv:2305.07402*.

[14] Luyu Gao and Jamie Callan. 2021. Unsupervised corpus aware language model pre-training for dense passage retrieval. *arXiv preprint arXiv:2108.05540*.

[15] Luyu Gao, Zhuyun Dai, Tongfei Chen, Zhen Fan, Benjamin Van Durme, and Jamie Callan. 2021. Complement lexical retrieval model with semantic residual embeddings. In *Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28- April 1, 2021, Proceedings, Part I 43*, pages 146-160. Springer.

[16] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2022. Precise Zero-Shot Dense Retrieval without Relevance Labels. *ArXiv:2212.10496 [cs]*.

[17] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised Dense Information Retrieval with Contrastive Learning. *ArXiv:2112.09118 [cs]*.

[18] Rolf Jagerman, Honglei Zhuang, Zhen Qin, Xuanhui Wang, and Michael Bendersky. 2023. Query expansion by prompting large language models. *arXiv preprint arXiv:2305.03653*.

[19] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

[20] Vladimir Karpukhin, Barlas Oguz, Sewon Min, et al. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

[21] Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39-48.

[22] Dahyun Kim, Chanjun Park, Sanghoon Kim, et al. 2023. Solar 10.7b: Scaling large language models with simple yet effective depth upscaling.

[23] Canjia Li, Andrew Yates, Sean MacAvaney, Ben He, and Yingfei Sun. 2020. Parade: Passage representation aggregation for document reranking. *arXiv preprint arXiv:2008.09093*.

[24] Minghan Li, Honglei Zhuang, Kai Hui, et al. 2023. Generate, Filter, and Fuse: Query Expansion via Multi-Step Keyword Generation for Zero-Shot Neural Rankers. *arXiv:2311.09175 [cs]*.

[25] Xiaonan Li, Changtai Zhu, Linyang Li, Zhangyue Yin, Tianxiang Sun, and Xipeng Qiu. 2023. LLatrieval: LLM-Verified Retrieval for Verifiable Generation. *arXiv:2311.07838 [cs]*.

[26] Percy Liang, Rishi Bommasani, Tony Lee, et al. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.

[27] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2356-2362.

[28] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*.

[29] Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics*, 9:329-345.

[30] Guangyuan Ma, Xing Wu, Peng Wang, Zijia Lin, and Songlin Hu. 2023. Pre-training with large language model-based document expansion for dense passage retrieval. *arXiv preprint arXiv:2308.08285*.

[31] Xueguang Ma, Xinyu Zhang, Ronak Pradeep, and Jimmy Lin. 2023. Zero-shot listwise document reranking with a large language model. *arXiv preprint arXiv:2305.02156*.

[32] Iain Mackie, Shubham Chatterjee, and Jeffrey Dalton. 2023. Generative Relevance Feedback with Large Language Models. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2026-2031. *arXiv:2304.13157 [cs]*.

[33] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802-9822.

[34] Richard McCreadie and Cody Buntain. 2023. Crisis-FACTS: Building and Evaluating Crisis Timelines.

[35] Shahrazd Naseri, Jeffrey Dalton, Andrew Yates, and James Allan. 2021. Ceqe: Contextualized embeddings for query expansion. In *Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28- April 1, 2021, Proceedings, Part I 43*, pages 467-482. Springer.

[36] Adam Paszke, Sam Gross, Francisco Massa, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

[37] Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. Rankvicuna: Zero-shot listwise document reranking with open-source large language models. *arXiv preprint arXiv:2309.15088*.

[38] M Elena Renda and Umberto Straccia. 2003. Web metasearch: rank vs. score based rank aggregation methods. In *Proceedings of the 2003 ACM symposium on Applied computing*, pages 841-846.

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

[40] Devendra Singh Sachan, Mike Lewis, Dani Yogatama, Luke Zettlemoyer, Joelle Pineau, and Manzil Zaheer. 2023. Questions are all you need to train a dense passage retriever. *Transactions of the Association for Computational Linguistics*, 11:600-616.

[41] Teven Le Scao, Angela Fan, Christopher Akiki, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

[42] Devendra Singh, Siva Reddy, Will Hamilton, Chris Dyer, and Dani Yogatama. 2021. End-to-end training of multi-document reader and retriever for open-domain question answering. *Advances in Neural Information Processing Systems*, 34:25968-25981.

[43] 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. *ArXiv:2304.09542 [cs]*.

[44] Nandan Thakur, Nils Reimers, Andreas Ruckle, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models. *ArXiv:2104.08663 [cs]*.

[45] Paul Thomas, Seth Spielman, Nick Craswell, and Bhaskar Mitra. 2023. Large language models can accurately predict searcher preferences. *arXiv preprint arXiv:2309.10621*.

[46] Hugo Touvron, Louis Martin, Kevin Stone, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

[47] Bo Wang, Andy Law, Tim Regan, et al. 2022. Systematic comparison of ranking aggregation methods for gene lists in experimental results. *Bioinformatics*, 38(21):4927-4933.

[48] Peiyi Wang, Lei Li, Liang Chen, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926*.

[49] Thomas Wolf, Lysandre Debut, Victor Sanh, et al. 2023. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

[50] Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2022. Generate rather than retrieve: Large language models are strong context generators. *arXiv preprint arXiv:2209.10063*.

[51] Wenhao Yu, Chenguang Zhu, Zaitang Li, Zhiting Hu, Qingyun Wang, Heng Ji, and Meng Jiang. 2022. A survey of knowledge-enhanced text generation. *ACM Computing Surveys*, 54(11s):1-38.

[52] Honglei Zhuang, Zhen Qin, Kai Hui, Junru Wu, Le Yan, Xuanhui Wang, and Michael Bendersky. 2023. Beyond Yes and No: Improving Zero-Shot LLM Rankers via Scoring Fine-Grained Relevance Labels. *ArXiv:2310.14122 [cs]*.

**TABLE V:** Statistics of datasets.

| Dataset | Domain | #Query | #Documents |
|---|---|---|---|
| TREC DL19 | Web | 43 | 8,841,823 |
| TREC DL20 | Web | 200 | 8,841,823 |
| BEIR Covid | Bio-Medical | 50 | 171,332 |
| BEIR NFCorpus | Bio-Medical | 323 | 3,633 |
| BEIR News | News | 57 | 594,977 |
| BEIR Signal | Twitter | 97 | 2,866,316 |
| BEIR Touche | Misc. | 49 | 382,545 |
| CrisisFACTS | Social Media | 52 | 468,788 |

# APPENDIX

## A. GENRA Algorithm

**Algorithm 1** GENRA

**Require:** query $q$ documents $D$ $\epsilon$ generation instruction $I_g$ , judgement instruction $I_j$ , number of retrieved documents $k$ , number of verified documents $m$ , number of generated passages $n$

**Ensure:** ranking $\sigma_q^k$

**procedure** PASSAGE GENERATION
  Generate passages $\mathbf{P} = LLM(I_g, q)$
  Enrich query $v_{\bar{q}} = \frac{1}{n} \sum \epsilon(P)$
  Retrieve top documents $\sigma_q^k$
**end procedure**
**procedure** RELEVANCE ASSESSMENT
  **procedure** RE-RANKING
    Re-order documents $\hat{\sigma}_q^k = LLM(\sigma_q^k)$
    Add top-$m$ documents in $D_m$
  **end procedure**
  **procedure** LLM-BASED JUDGEMENTS
    **for** $d = 1, 2, \ldots, k$ **do**
      Obtain $\rho_{q,d} = LLM(I_j, d)$
      **if** $\rho_{q,d} =$ yes **then**
        Add $d$ in $D_m$
      **end if**
    **end for**
  **end procedure**
**end procedure**
**procedure** RANKING AGGREGATION
  $S = []$
  **for** $d = 1, 2, \ldots, m$ **do**
    Retrieve top documents $\sigma_d^k$
    Add $\sigma_d^k$ in $S$
  **end for**
**end procedure**
Obtain final ranking $\sigma_q^k = agg(S)$

## B. Datasets Statistics

Table V presents the statistics of the TREC, BEIR and CrisisFACTS datasets.