RBR: RAG-Based Open-Domain Question Answering using a Ranking Approach to Document Retrieval

Priyatam Naravajhula, Vincent Ng
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75080-0688, USA
priyatam.naravajhula@utdallas.edu, vince@hlt.utdallas.edu

Abstract

Retrieval-Augmented Generation (RAG) has emerged as a promising approach to ODQA. A RAG-based ODQA system is typically composed of two components: a *retriever* that retrieves the passages that are most relevant to a given query, and a *generator* that generates the answer to the query by combining the information from the retrieved passages. Existing retrievers typically identify the most relevant passages by computing the similarity between the query and each passage in a given collection. In other words, they do *not* compare which of two passages is more relevant to the given query. We hypothesize, however, that we can improve RAG-based ODQA systems by modeling the relationship among the passages to be retrieved, specifically by learning which passages are more relevant than the others to the given query. To do so, we propose a novel ranking-based approach to passage retrieval, where we first rank the candidate passages w.r.t. the query and subsequently refine the score associated with each of these passages using a Graph Attention Network. We evaluate our approach to ODQA, RBR (Ranking-Based Retrieval), on two commonly-used ODQA datasets, Natural Questions and TriviaQA. Experimental results show that RBR slightly outperforms PA-RAG, a state-of-the-art ODQA system, by 0.45 points and 1.01 points in Exact Match score on Natural Questions and TriviaQA, respectively.

Keywords: Question answering, Retrieval augmented generation

1. Introduction

Open-Domain Question Answering (ODQA) represents a sophisticated task within natural language processing (NLP) that involves answering arbitrary questions expressed in natural language by leveraging extensive collections of unstructured or semi-structured text documents. Unlike specialized QA systems constrained to domains or reliant on carefully structured knowledge bases, ODQA systems must possess the versatility to retrieve, process, and synthesize pertinent information from massive, heterogeneous text corpora spanning diverse topics and domains. These corpora typically include encyclopedic resources like Wikipedia, journalistic content, scientific literature, and social media discourse. The fundamental challenge lies in creating systems capable of understanding questions in their full semantic complexity, efficiently identifying relevant information across vast repositories, and generating accurate, contextually appropriate responses that satisfy the user's informational needs

While ODQA has been tackled since the late 1960s (Simmons, 1969), it was not until the turn of the 21st century that the task started to gain significant attention in the NLP community when ARDA (Advanced Research and Development Activity) funded research projects through the ACQUAINT program (Saint-Dizier, 2013). ACQUAINT sought to create advanced QA tools to help analysts quickly sort through massive amounts of information to find facts relevant to their intelligence work.

Early QA systems are typically composed of two components: (1) a *retriever* that returns the documents/passages from a text collection that are most relevant to the given query, where relevance is measured using cosine similarity and term importance is computed using TF-IDF (Sparck Jones, 1972) or BM25 (Robertson et al., 2009), and (2) an *extractor* that employs either a heuristic- or learning-based method to extract a text span from one of the retrieved documents/passages and return it as an answer to the query.

The advent of the neural NLP era has revolutionized the way ODQA systems are developed in at least two ways. First, given that pre-trained models or LLMs, regardless of whether they are fine-tuned on QA collections, contain a vast amount of knowledge, researchers abandoned the retriever in the aforementioned traditional QA systems, relying solely on the knowledge embedded within the weights of the neural models (a.k.a. *parametric* knowledge) to answer questions. Second, given the generative capability of neural models, answers can now be *generated* rather than *extracted* from the given knowledge. A key drawback of this approach is that parametric knowledge is *static* (i.e., it does not change over time): as new facts appear over time, these neural models may not be able to answer questions that rely on them unless they are being re-trained on these new facts. Though conceptually feasible, retraining large neural models is

Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026), pages 805–817
11–16 May 2026.

computationally very expensive and is therefore not considered a practical solution to this drawback.

To address this limitation, researchers introduced the concept of Retrieval Augmented Generation (RAG): rather than retrain a neural model on up-to-date knowledge, complement the parametric knowledge possessed by a neural model with the up-to-date knowledge in external knowledge bases (a.k.a. *non-parametric* knowledge). Hence, applying RAG to ODQA entails re-introducing the retriever commonly-used in traditional QA systems, where the retriever first retrieves the passages that are most relevant to the given query, and then the generator generates the answer by synthesizing information from the retrieved passages.

Our goal in this paper is to improve RAG approaches to ODQA. As noted above, a RAG-based ODQA approach is typically composed of two components, a passage retriever and an answer generator. Our *contribution* lies in improving RAG-based ODQA by improving the retriever. Specifically, a retriever typically retrieves the k passages most relevant to a given query by measuring the similarity between a passage and a query. In other words, a retriever does *not* compare which of two passages is more relevant to the given query. We hypothesize, however, that we can improve RAG-based ODQA by modeling the relationship among the passages to be retrieved, specifically by learning which passages are more relevant than the others to the given query. To do so, we propose a novel ranking-based approach to passage retrieval, where we first rank the candidate passages and subsequently refine the score associated with each of these passages using a Graph Attention Network (GAT). We evaluate our approach to ODQA, RBR (ranking-based retrieval), on two commonly-used ODQA datasets, Natural Questions and TriviaQA. Experimental results show that RBR performs comparably with the state of the art.

The rest of the paper is organized as follows. Section 2 presents an overview of related work on RAG-based approaches to ODQA. In Section 3, we describe two baseline systems, both of which RBR is built on. Section 4 describes the details of our ranking approach. Finally, we present evaluation results in Section 5 presents evaluation results and conclusions in Section 6.

2. Related Work

In this section, we review related work on RAG approaches to ODQA.

2.1. Naive RAG Architecture

Naive RAG represents the foundational implementation of this approach. It is a two-component architecture combining dense retrieval with answer generation, as described below.

The first component, the Dense Passage Retriever (DPR) (Karpukhin et al., 2020), forms the backbone of the retrieval mechanism in Naive RAG. DPR employs a dual-encoder architecture with separate BERT-based networks for encoding queries and documents. The similarity between a query and document is computed as the dot product of their respective embeddings. Specifically, the probability of retrieving a document z given a query x is modeled as proportional to the exponential of the inner product between document and query embeddings: $p_\eta(z|x) \propto e^{d(z)\cdot q(x)}$. This formulation enables efficient identification of the most relevant documents, which are then provided to the generation component. DPR is trained using a combination of in-batch negatives and hard negatives, maximizing the log-likelihood of retrieving relevant passages. This training approach allows the retriever to identify semantically related documents rather than relying solely on lexical overlap, addressing limitations of previous TF-IDF-based methods.

For answer generation, Naive RAG employs a sequence-to-sequence transformer model, specifically BART-Large (Lewis et al., 2020a), This model contains approximately 400M parameters and represents the parametric memory component of the system. which conditions on both the input query and retrieved documents to generate responses, effectively combining the broad language understanding capabilities encoded in its parameters with the specific factual information retrieved from external sources. Lewis et al. (2020b) introduced two variants of Naive RAG that differ in how they marginalize over retrieved documents: RAG-Sequence uses a single document for generating the entire output sequence, marginalizing document probabilities at the sequence level, whereas RAG-Token allows different documents to influence each token in the generation process, with document marginalization occurring at the token level.

2.2. Improvements to Naive RAG

Various attempts have been made to improve Naive RAG for ODQA.

2.2.1. Improving the Inputs

Some researchers attempt to change neither the retriever nor the generator. Rather, they focus on changing the *input* to one of these components.

Rausdashl (2023) changed the input to the retriever. Specifically, they introduced RAG-Fusion, which leverages multiple query variations to enhance retrieval robustness. Instead of relying on a single query formulation, RAG-Fusion generates

multiple semantically equivalent queries from the original user query. For instance, given a query "What are the challenges faced by people with ADHD?", the system might generate variations like "What are the common social difficulties experienced by individuals with ADHD?" and "What are the challenges in accessing healthcare and treatment for ADHD?". These queries are processed independently through the retrieval pipeline, and the results are aggregated using Reciprocal Rank Fusion (RRF). This technique assigns scores to documents based on their reciprocal ranks across all queries, prioritizing documents that appear consistently across multiple query formulations. Rackauckas (2024) demonstrated that RAG-Fusion produced more accurate and comprehensive answers than traditional RAG approaches when applied to private datasets.

Other researchers change the input to the generator. Shi et al. (2024) developed REPLUG, which processes each document-query pair separately through the LLM before ensemble combination and after retrieving relevant documents using a dense retriever. The authors demonstrated that even without fine-tuning the language model, substantial performance gains could be achieved through optimized retrieval and thoughtful integration strategies. Izacard and Grave (2021) developed the Fusion-in-Decoder (FiD) approach, which builds upon Naive RAG by modifying how retrieved documents are processed. Specifically, FiD encodes each retrieved passage independently along with the query, then concatenates these representations for the decoder to process jointly. This design allows FiD to scale efficiently to a larger number of passages (up to 100) while still enabling cross-passage reasoning in the generator. FiD demonstrated superior performance compared to Naive RAG on several open-domain QA benchmarks, highlighting the importance of effective passage integration. Wang et al. (2024) introduced REAR, which addresses LLMs' inability to accurately assess retrieved document relevance by incorporating an explicit assessment module that performs fine-grained relevance evaluation using continuous scores rather than binary judgments. The system uses the resulting relevance scores to guide generation, helping the LLM adaptively choose between internal parametric knowledge (for low-relevance documents) and external evidence (for high-relevance documents).

2.2.2. Improving the LLM Generator

Some researchers focus on improving the generator. For instance, Guu et al. (2020) presented REALM (Retrieval-Augmented Language Model Pre-Training), which shares conceptual similarities with RAG but differs in its training approach. Specifically, REALM integrates a neural retriever into the LLM when pre-training it, treating retrieval as a latent variable during masked language modeling. This approach allows the model to learn what information is useful to retrieve during pre-training. While REALM forms an important precursor to RAG, its indexing process requires periodic refreshing during training, introducing additional computational complexity.

2.2.3. Improving the Training Process

End-to-end training is a crucial aspect of Naive RAG, which is trained to minimize the negative log-likelihood of generating correct answers given the query, with gradients backpropagated through both the generator and query encoder. However, for computational efficiency, the document encoder typically remains fixed during training to avoid the need for frequent index updates. Siriwardhana et al. (2021) questioned this assumption and explored training the entire retriever, including the document encoder. Despite implementation challenges involving re-encoding and re-indexing, their approach demonstrated approximately 12% performance improvement over the original RAG on the SQuAD dataset. Building on this work, Siriwardhana et al. (2023) introduced a reconstruction signal for pre-training RAG models. This auxiliary training objective requires the model to reconstruct input statements using only the retrieved documents, ensuring a stronger alignment between the retrieval and generation components. The authors reported significant improvements in exact match scores and retrieval accuracy, particularly in domain-specific applications such as COVID-19 question answering.

Wu et al. (2024) also aimed to improve the training procedure. Specifically, they proposed PA-RAG, a multi-stage preference optimization framework addressing informativeness, robustness, and citation quality. This method employs a two-phase approach: first conducting instruction fine-tuning with high-quality data constructed using ChatGPT and a citation rewrite mechanism, then applying Direct Preference Optimization (DPO) sequentially for informativeness, robustness, and citation quality. Extensive experiments across multiple question-answering datasets and LLM architectures demonstrated substantial improvements in answer correctness (13.97%), citation recall (49.77%), and citation precision (39.58%) compared to baseline approaches.

2.2.4. Improving the Retriever

Most closely related to our approach is work that attempts to improve the retriever. Sawarkar et al. (2024) presented the Blended RAG approach, which focuses on improving RAG accuracy by en-

hancing the retriever component rather than the language model. It integrates three distinct search strategies: keyword-based similarity search (BM25 index), dense vector-based search (KNN index), and semantic-based sparse encoders, combined with multiple hybrid query formulations including match queries, multi-match queries (cross fields, most fields, best fields, and phrase prefix). The methodology systematically evaluates these combinations across different datasets to identify optimal retriever configurations. Unlike our approach, however, Blended RAG does not model the relationship among the candidate passages.

2.2.5. Incorporating a Reranker

While a RAG-based ODQA system is typically composed of a pipeline of two components, a retriever and a generator, Ke et al. (2024) introduced a third component into the pipeline, the bridge model, which is situated between the retriever and generator in order to address what they termed the "preference gap" between the retrievers and the LLM generator. The bridge model is a sequence-to-sequence model that functions as an intermediary that adapts the retrieved information to better match the LLM's preferences. Ke et al. observed that while retrievers might rank documents based on lexical or semantic similarity, these rankings do not necessarily align with the documents' utility for the LLM's generation process. The bridge model takes the query and a list of initially retrieved passages as input, with each passage prefixed by a unique sentinel token. It then generates a sequence of passage IDs representing a reordered and possibly filtered set of passages better suited to the LLM's needs. This model is trained through a combination of supervised learning on synthetically generated optimal passage sequences and reinforcement learning, using downstream task performance as the reward signal. Experiments showed substantial improvements over traditional ranking methods, particularly for complex queries requiring integration of information from multiple sources.

3. Baseline Systems

We employ two Baseline systems, both of which our ranking-based retrieval (RBR) approach is built on.

3.1. Baseline 1

Baseline 1 is modeled after Naive RAG in that it has two components, the retriever and the generator.

Retriever. We implement the retriever using Dense Passage Retrieval (DPR), which employs a dual-encoder architecture with separate networks for encoding queries and documents. In our experiments, we use BERT-base-uncased as the encoder. During training, for each query, we randomly sample N positive (i.e., relevant) passages and $N(M-1)$ negative (i.e., irrelevant) passages as training instances, where N and M are tunable hyperparameters in our experiments. The training objective is the standard negative log-likelihood with contrastive loss, which maximizes the similarity between the query and positive passage while minimizing similarity with the 15 negative passages. The loss computes the log probability of the N positive passages relative to all $16N$ passages. In our experiments, we send the top 10 passages retrieved to the generator.

Generator. We use Llama 3.1 8B (Grattafiori et al., 2024) as the LLM generator. This decision was based on the fact that our computational resources do not allow us to run larger models. Nevertheless, Llama3.1 8B demonstrates performance comparable to significantly larger models from previous generations. As part of the RAG pipeline, we *prompt* the LLM with the query and the retrieved, re-ranked passages as context for answer generation.

3.2. Baseline 2

As mentioned in Section 2, Ke et al. (2024) observed that in the standard retriever-generator pipeline, retrievers typically rank documents based on lexical or semantic similarity, but these rankings do not necessarily align with what is most useful for the LLM's generation process. Hence, they proposed the incorporation of a reranker into the pipeline in order to bridge the "preference gap" between these components.

Baseline 2 simply augments Baseline 1 with a reranker, but is otherwise identical to Baseline 1. Following Ke et al. (2024), the reranker is implemented as a sequence-to-sequence model that takes the query and initially retrieved passages as input and outputs a reordered and possibly filtered set of passages better suited to the LLM's needs. The reranker undergoes the three-phase training process proposed by Ke et al. In Step 1 (Silver Sequence Generation), we create optimal passage sequences through a greedy search procedure driven by performance. In Step 2 (Supervised Learning), we use a T5 encoder-decoder architecture optimized with negative log-likelihood on the silver sequences. In Step 3 (Reinforcement Learning), we fine-tune the model using policy gradient methods with rewards derived from answer quality. The reward function incorporates multiple components including overlap with silver sequences, order preservation using Kendall's Tau correlation, and positioning high-relevance passages earlier.

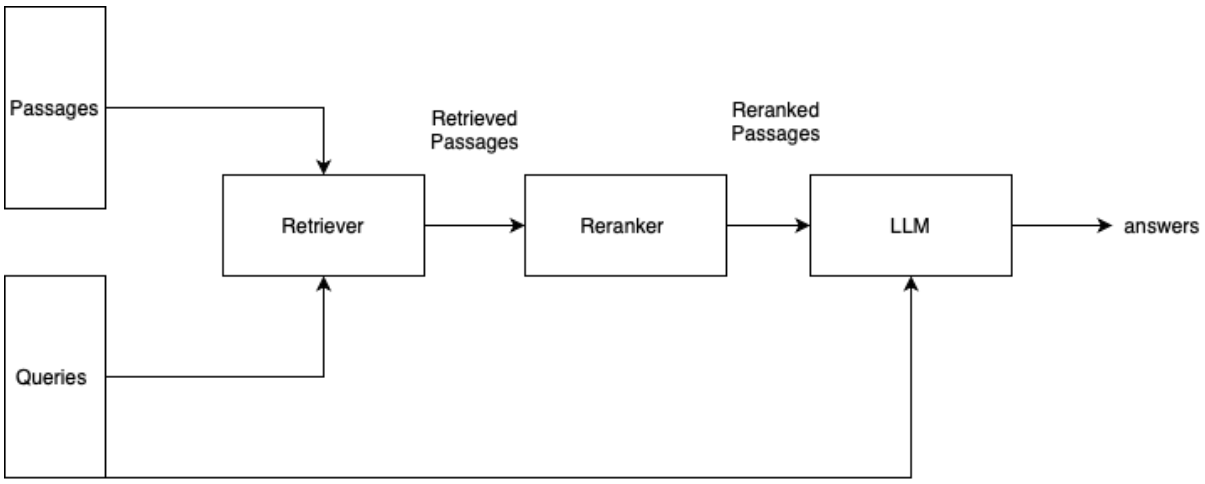

Figure 1: Flow chart of RAG Architecture with a Reranker (Baseline 2).

In our experiments, we send the top 20 passages retrieved by the retriever to the reranker, which then sends the top 10 reranked passages to the LLM.

Figure 1 shows the architecture for Baseline 2, which consists of three components, the retriever, the reranker and the LLM generator.

4. RBR: Ranking-Based Retrieval

In this section, we describe RBR, our ranking-based approach to retrieving relevant passages. Given a *test* query, RBR performs *intra-set* ranking (Section 4.1) followed by *inter-set* ranking (Section 4.2).

4.1. Intra-Set Ranking: Training

Recall that existing document retrievers compute the similarity between a passage and a given query. In other words, each query-passage pair is considered independently of other query-passage pairs. In particular, existing retrievers do *not* model the relationship among the candidate passages, meaning that they fail to answer the question of which of two candidate passages are *more* relevant to the query. To address this weakness, we propose to model retrieval as a *ranking* task.

As mentioned above, our approach, RBR, is composed of two steps, intra-set ranking and inter-set ranking, both of which involve training a ranker. To understand the difference between intra- and inter-set ranking, we need to better understand our evaluation datasets (see Section 5 for details). In these datasets, each query has 80–100 positive (i.e., relevant) passages and 200–250 negative (i.e., irrelevant) passages; moreover, each positive passage is associated with a relevance score: the higher the score is, the more relevant the positive passage is to the query. Intra-Set ranking aims to rank a positive instance above the negative instances, whereas inter-set ranking aims to rank the positive instances of a query. In this subsection, we focus on intra-set ranking, describing how we train an intra-set ranker.

Given the large number of positive and negative passages associated with a query, it will be computationally very expensive to train a ranker on all of them. As a result, for each query in the training set, we create N sets (where N is hyperparameter to be tuned on development data), each of which is composed of one positive passage and $M-1$ negative passages (where M is another hyperparameter to be tuned on development data). Both the positive passage and the negative passages used to create a set is randomly sampled from the available positive and negative passages *without* replacement.

Given the sets created from the training queries, we can train a ranker to rank the positive passage above the negative passages within each set. We use as the ranking loss a margin-based contrastive objective (Zhang et al., 2022). For each query representation $q \in \mathbb{R}^d$ and its corresponding passage representations $p_0, p_1, ..., p_n \in \mathbb{R}^d$ (without loss of generality, we assume that p_0 is the positive passage and others are negatives), the similarity is computed using cosine similarity with temperature scaling:

$$s(q,p) = \frac{\cos(q,p)}{\tau}$$

where τ is a learnable temperature parameter subjected to softplus activation (Zheng et al., 2015) to ensure positivity:

$$\tau_{\text{effective}} = \text{softplus}(\tau_{\text{raw}})$$

The margin-based contrastive loss, $\mathcal{L}_{\text{intra}}$, is then calculated as:

$$\frac{1}{B \cdot N} \sum_{i=1}^{B} \sum_{j=1}^{N} \frac{1}{M-1} \sum_{k=1}^{M-1} \max(0, \alpha - s(q_{ij}, p_{ij0}) + s(q_{ij}, p_{ijk}))$$

where B is the batch size, N is the number of sets per batch, M is the number of samples per set, p_{ij0} represents the positive passage sample, p_{ijk} represents the negative passage samples, and α is the margin hyperparameter.

The margin-based pairwise loss enforces a specific ranking preference: the similarity between a query and its positive passage must exceed the similarity with any negative passage by at least a margin α. This margin serves as a buffer zone that makes the model more robust during inference, ensuring a clear separation between relevant and non-relevant documents in the embedding space. By pushing positive pairs closer and negative pairs farther apart, the model develops a more discriminative understanding of document relevance.

4.2. Inter-Set Ranking: Training

So far we have trained a ranker to rank the positive instance above its negative counterparts *within* each set. Since we have N sets per query, we will end up with N sets of ranked results for each query. However, given a query during *testing*, the retriever will only output one set of ranked results (i.e., the top k passages), not N sets. The purpose of *inter-set* ranking is to rank the N positive instances from the N sets.

In this subsection, we describe how we train a ranker to perform inter-set ranking of the positive instances. Specifically, we leverage a Graph Attention Network (GAT) (Veličković et al., 2018) to model the interactions among the positive passages associated with a query.

Step 1: Graph Formation

GATs are typically employed for downstream tasks with graph-structured inputs. The design of an appropriate graph structure becomes crucial for the successful application of GAT to ODQA. Hence, we first describe how we form the graph given a query.

Node construction. Given a query with N sets, we create a graph with $2N$ nodes. N of them correspond to the N positive instances taken from the N sets, while the remaining N nodes all correspond to the query. As part of the node construction process, we need to associate each node with an *initial* node representation, as described below.

Let $q \in \mathbb{R}^D$ be the query embedding and $p_i \in \mathbb{R}^D$ be the embedding for positive passage i ($1 \leq i \leq N$), where D is the embedding dimension. We first project these embeddings into a lower-dimensional space:

$$q_c = \text{GELU}(\text{LN}(W_q q)) \in \mathbb{R}^{D/2}$$
$$p_c i = \text{GELU}(\text{LN}(W_p p_i)) \in \mathbb{R}^{D/2}$$

where $W_q, W_p \in \mathbb{R}^{D \times D/2}$ are learnable projection matrices, LN denotes layer normalization, and GELU is the activation function.

The reduction in dimensionality from D to $D/2$ serves multiple purposes: it reduces computational complexity for subsequent attention operations, forces the model to distill the most relevant semantic information, and helps prevent overfitting by reducing parameter count. Layer normalization stabilizes training by standardizing the activations, which is important when working with variable-length documents. The GELU activation function GELU introduces non-linearity while maintaining smooth gradients, which improves optimization dynamics.

These projected embeddings then undergo a node transformation:

$$q_n = \text{ReLU}(\text{LN}(W_n q_c)) \in \mathbb{R}^{D/2}$$
$$p_{n,i} = \text{ReLU}(\text{LN}(W_n p_{c,i})) \in \mathbb{R}^{D/2}$$

This additional transformation serves to (1) enforce sparsity through the ReLU activation, helping focus attention on the most salient features, (2) further normalize embeddings to improve attention mechanism stability, and (3) create a common representation space where query and passage features can effectively interact.

Recall that each of the $2N$ nodes in the graph has an initial node representation. Each of the N nodes that correspond to the query will have q_n as its initial node representation. Each of the N passage nodes will have a $p_{n,i}$ as its initial node representation.

One may wonder why we need N query nodes that all have the same initial representation. This is so for two reasons. The first is structural symmetry: since we have N passage nodes, for symmetry we have N query nodes, though this is not a requirement for GAT to wor. Second, despite the fact that they have the same initial representation, their representations will change as the graph passes through the GAT layers (see below), and having multiple query nodes offers the flexibility of different query nodes to specialize through attention.

Edge construction. After node construction, we form a fully connected graph by connecting every node to every other node with a directed edge. This will allow information to flow between all node pairs during the forward pass, as described below.

The resulting graph structure enables (1) direct attention from query to passages and vice versa; (2) attention between different passages, allowing

for comparative assessment; and (3) self-attention within queries and within passages to strengthen internal representations.

Step 2: Iterative Refinement

Let $H^{(0)}$ be the graph having the initial node representations as described in Step 1. In this step, we perform a forward pass on $H^{(0)}$. In other words, we apply multiple sequential GAT layers to $H^{(0)}$ in an iterative refinement process to refine the node representations. Specifically:

$$H^{(l+1)} = \text{GAT}_l(H^{(l)})$$

where $0 \leq l \leq L$, and $H^{(l)}$ denotes the graph with the node representations obtained after the l-th GAT layer, starting with l=0. Each GAT_l layer performs the multi-head attention operations.

This iterative structure enables (1) first-order relationships (direct connections) to be modeled in the initial layer; (2) second-order relationships (connections between connections) to emerge in subsequent layers; and (3) progressive contextualization of node representations with increasing layer depth.

Step 3: Node Separation and Normalization

Now that we have refined the node representations, we can separate each (contextualized) node representation into a query component and a passage component, followed by L2 normalization:

$$q_{\text{enhanced}} = H^{(L)}_{1:N}, \quad p_{\text{enhanced}} = H^{(L)}_{N+1:2N}$$
$$q_{\text{normalized}} = \frac{q_{\text{enhanced}}}{||q_{\text{enhanced}}||_2}, \quad p_{\text{normalized}} = \frac{p_{\text{enhanced}}}{||p_{\text{enhanced}}||_2}$$

Step 4: Similarity Score Computation

Finally, we compute the similarity score between the query and a passage by taking the temperature-scaled dot product of their contextualized representations:

$$S = \frac{q_{\text{normalized}} \cdot p^T_{\text{normalized}}}{\tau} \in \mathbb{R}^{N \times N}$$

where τ denotes the temperature parameter that controls the sharpness of the resulting distribution.

Training objective. The training objective employs a pairwise margin loss function operating on the contextualized similarity scores:

$$\mathcal{L}_{\text{margin}} = \frac{1}{|\mathcal{P}|} \sum_{(i,j)\in\mathcal{P}} \max(0, \gamma - (S_{i,i} - S_{j,j}))$$

where $\mathcal{P}$ encompasses all valid pairs (i, j) for which the ground truth relevance of document i exceeds that of document j by a predetermined threshold, and γ represents the margin hyperparameter.

4.3. Joint Training

While the above discussion seems to suggest that the intra-set ranker and the inter-set ranker are trained separately using their own training objectives, in reality we perform *joint* training of the two rankers, where the training loss is the unweighted sum of the losses employed by the two rankers.

4.4. Applying the Two Rankers

Now that we have described how we jointly train the intra- and inter-set rankers, in this subsection we describe how to apply them.

Applying the intra-set ranker. Recall that when training the intra-set ranker, we created the N sets by randomly sampling positive and negative passages given a query. During testing, however, we do not know whether a passage is positive or negative. To address this problem, we first predict whether a passage is positive or negative and use the resulting predicted labels to create the N sets for intra-set ranking.

Specifically, we first use the retriever employed by the two Baselines (see Section 3) to compute a similarity between each query and each of its passages (using DPR). We then create the N sets as follows. The N top-ranked will each be used as the (predicted) positive instance for each of the N sets. Since each set has M passages, we will use the next $N(M-1)$ passages in the ranked list of passages returned by the Baseline retriever as the negative passages, distributing them evenly over the N sets. Finally, we use the resulting N sets for intra- and inter-set ranking.

Applying the inter-set ranker. Recall that when training the inter-set ranker, we used the *gold* positive instances to construct the graph. During testing, however, gold positive instances are not available. Hence, we use instead the *predicted* positive instances, where we assume that the highest-ranked instance in each of the N sets for the given query q is assumed to be its positive instance and will be used for graph formation.

4.5. Returning the Ranked List

As mentioned at the beginning of Section 4.2, the retriever needs to return *one* set of ranked results. Hence, given a test query, we need to merge the results of intra-set ranking and inter-set ranking into one set of ranked results.

We employ a simple merging procedure. Each of the N highest-ranked passages according to the

Metric	NQ	TriviaQA
Training set size	79,168 docs	78,785 docs
Development set size	8,757 docs	8,837 docs
Test set size	3,610 docs	11,313 docs
Avg. question length	9.2 tokens	14 tokens
Avg. answer length	3.5 tokens	2.0 tokens

Table 1: Statistics on the two datasets.

intra-set ranker underwent inter-set ranking and now has a GAT/semantic-refined score. The remaining passages retain their original intra-set similarity scores. We sort the passages from all N sets by these scores to produce a complete ranking. Following the two Baseline systems (see Section 3), if a reranker is not used (as in Baseline 1), we return the top 10 passages to the generator. However, if a reranker is used (as in Baseline 2), we feed the top 20 passages to the reranker.

5. Evaluation

5.1. Experimental Setup

In this subsection, we discuss the datasets, the implementation details, and the metric we use in our evaluation.

Datasets. We employ two commonly-used evaluation datasets. The **Natural Questions (NQ)** dataset (Kwiatkowski et al., 2019) was introduced by Google in 2019 as a benchmark for open-domain question answering research. It consists of real questions that users asked Google Search, paired with answers found in Wikipedia articles. These questions represent genuine information needs expressed by real users, making it more challenging and realistic than previous QA datasets. The questions are collected from Google search queries, and the answers are annotated in Wikipedia passages by human annotators. **TriviaQA** (Joshi et al., 2017) was released in 2017 and was one of the first large-scale reading comprehension datasets to feature complex trivia-style questions. The questions come from trivia websites and quiz-league websites, paired with evidence documents from Wikipedia and web searches. Unlike NQ, TriviaQA questions were specifically created to be trivia challenges, making them more complex and often requiring multi-hop reasoning. TriviaQA is notable for having multiple answer aliases for each question (e.g., "JFK" and "John F. Kennedy"), which allows for more flexible evaluation. Statistics on these two datasets are shown in Table 1.

Implementation details and hyperparameter settings used to implement our retriever. We set the dimensionality of the Transformer representations to 768, the batch size to 4, the gradient accumulation (the number of steps before parameter update) to 4, the learning rate to $1 \times 10^{-}5$, the weight decay to 0.01, the maximum graident norm to 1, the warmup ratio to 0.1, the maximum token length of each query to 128, the maximum token length of each passage to 128, the number of epochs to 175, the temperature used for intra-set ranking to 0.1, the temperature used for inter-set ranking to 0.3, the margin for the pairwise loss to 0.2, and the number of epochs without improvement before stopping (early stopping) to 5. These values were chosen using the development set. Note, however, that given the large number of hyperparameters, it is not computationally feasible to do a grid search over all of them. What we did instead was to perform iterative hyperparameter tuning, where in each iteration, we optimize exactly one hyperparameters while keeping the rest fixed.

After determining these hyperparameter values, we use them to perform a grid search on the development set over the remaining two hyperparameters, which are core to our approach, namely the number of candidates per set, which we search out of {8, 16, 32}, and the number of sets, which we search out of {6, 8, 10, 12}.[1]

Evaluation metric. To evaluate RAG-based ODQA systems, the Exact Match (EM) score (Lewis et al., 2020b) serves as the primary evaluation metric, calculated as a normalized binary assessment across all test queries. The formula for EM score is:

$$\text{EM} = \frac{1}{N} \sum_{i=1}^{N} \delta(\text{pred}_i, \text{ref}_i) \quad (1)$$

Where N represents the total number of queries evaluated, pred_i is the normalized predicted answer for query i, ref_i is the set of normalized reference answers for query i, and $\delta(\text{pred}_i, \text{ref}_i)$ equals 1 if the prediction matches any reference answer, and 0 otherwise. Following previous work, we employ a lenient matching approach where a match is recognized when there is substantial term overlap between the candidate answer and references, determined through multiple criteria: exact match after normalization, containment analysis, fuzzy matching with similarity thresholds, or significant token overlap ratio. This balanced methodology maintains objectivity while avoiding the excessive strictness of character-by-character comparison, providing a practical measure of answer correctness that remains reproducible and interpretable.

5.2. Results and Discussion

In this subsection, we gauge the performance of RBR by comparing it with our two baselines as well

[1]According to the dev set, the best number of candidates per set for both datasets is 16. As for the number of sets, the optimal number is 8 for NQ and 10 for TriviQA.

	System	NQ	TriviaQA
1	DPR only (Baseline 1)	40.50	42.90
2	DPR+Reranker (Baseline 2)	42.00	45.46
3	DPR+Intra+Inter+Reranker	**62.74**	**74.50**
4	DPR+Intra+Inter	53.56	66.03
5	DPR+Intra+Reranker	49.97	62.14
6	DPR+Intra only	45.65	59.00

Table 2: EM results of different variants of RBR.

RAG type	NQ	TriviaQA
Naïve RAG (Lewis et al., 2020b)	44.50	55.80
REPLUG (Shi et al., 2024)	45.50	72.39
Blended RAG (Sawarkar et al., 2024)	42.63	-
Bridging RAG (Ke et al., 2024)	45.37	-
REAR (Wang et al., 2024)	51.41	66.26
PA-RAG (Wu et al., 2024)	62.29	73.49
RBR (Our approach)	**62.74**	**74.50**

Table 3: Comparison with state-of-the-art models. All results are expressed in terms of EM.

as the state-of-the-art RAG-based ODQA systems.

5.2.1. Comparison with the Baselines

EM results of Baselines 1 and 2 are shown in rows 1 and 2 of Table 2 respectively. Recall that the two Baselines differ only w.r.t. whether a reranker is used. As can be see, incorporating a reranker yields an improvement of 1.5–2.56 EM points on the two datasets. These results demonstrate the usefulness of a reranker for RAG-based ODQA.

Results of RBR are shown in row 3 of Table 2. Comparing the results in row 3 with those in the first two rows, we see that RBR considerably improves both Baselines by 20.74–29.04 EM points on both datasets. These results demonstrate the effectiveness of RBR for ODQA.

5.2.2. Ablation Results

To gain insights into RBR, we conduct three ablation experiments.

First, to understand the contribution of the reranker in our approach, we remove the reranker from our approach, feeding the top-10 passages ranked by the retriever directly to the LLM generator. Results of this experiment are shown in row 4 of Table 2. As can be seen, without the reranker, the EM scores of our approach drop abruptly by 8.47–9.18 points on the two datasets. These results suggest that the reranker plays a crucial role in our approach.

Second, to understand the contribution of the GAT-based inter-set ranking in our approach, we omit inter-set ranking by re-training our retriever using intra-set ranking only. Results of this ablated system are shown in row 5 of Table 2. As can be seen, without inter-set ranking, the EM scores of our approach drop abruptly by 12.36–12.77 points on the two datasets. These results suggest that inter-set ranking plays a crucial role in our approach. Moreover, comparing rows 4 and 5, we see that inter-set ranking contributed more to the overall performance of our approach than the reranker.

Finally, we conduct an ablation experiment in which we remove both the reranker and inter-set ranking from our approach. Results of this ablated system are shown in row 6 of Table 2. As can be seen, without the reranker and inter-set ranking, the EM scores of our approach drop further in comparison to those in rows 4 and 5, where exactly one of them is ablated. Specifically, when both are ablated from our approach, the EM scores drop considerably by 15.5–17.09 points on the two datasets. Comparing the results in row 1 and row 6, we see that the addition of intra-set ranking to Baseline 1 improves the EM scores by 5.15–16.1 points on the two sets, suggesting the usefulness of intra-set ranking in the absence of the reranker.

5.2.3. Comparison with the State of the Art

Despite RBR's considerable improvements over the two Baselines, a natural question is: how does RBR perform relative to the state-of-the-art RAG-based ODQA systems? Results are shown in Table 3. Specifically, the first six rows of the table show the results of six state-of-the-art RAG-based ODQA systems on the two datasets taken directly from the respective papers[2], and the last row of the table shows the results of RBR. As can be see, RBR slightly outperforms PA-RAG, the best state-of-the-art system, on both datasets. These results provide suggestive evidence that RBR is an effective approach to RAG-based ODQA that performs comparably to the state of the art.

6. Conclusion

We presented RBR, a novel approach that improves RAG-based ODQA systems by improving the retriever. Unlike existing retrievers, which models each query-passage pair independently of other pairs, RBR first models the relationship among the passages by ranking them w.r.t. the given query, then refines the relevance scores using a GAT. Empirical results on two commonly-used evaluation datasets, Natural Questions and TriviaQA, showed that RBR slightly outperforms PA-RAG, a state-of-the-art ODQA system, by 0.45 points and 1.01 points in EM score on Natural Questions and TriviaQA, respectively.

[2]Dashes are used in the table when an approach does not report results on a dataset.

Bibliographical References

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet,

Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The Llama 3 Herd of Models. ArXiv:2407.21783 [cs.AI].

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: Retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

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

Zixuan Ke, Weize Kong, Cheng Li, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. 2024. Bridging the preference gap between retrievers and LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10438–10451, Bangkok, Thailand. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020b. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Zackary Rackauckas. 2024. RAG-Fusion: a new take on retrieval-augmented generation. ArXiv:2402.03367 [cs.IR].

Adrian H Rausdashl. 2023. Forget RAG, the Future is RAG-Fusion. *Towards Data Sciecne*.

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

Patrick Saint-Dizier. 2013. Advanced question-answering and discourse semantics. In *Emerging Applications of Natural Language Processing: Concepts and New Research*. IGI Global.

Kunal Sawarkar, Abhilasha Mangal, and Shivam Raj Solanki. 2024. Blended RAG: Improving RAG (Retriever-Augmented Generation) accuracy with semantic search and hybrid query-based retrievers. In *2024 IEEE 7th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, volume 24, page 155–161. IEEE.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. REPLUG: Retrieval-augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8371–8384, Mexico City, Mexico. Association for Computational Linguistics.

Robert F. Simmons. 1969. Natural language question-answering systems. *Communications of the ACM*, 13(1):15–30.

Shamane Siriwardhana, Rivindu Weerasekera, Elliott Wen, Tharindu Kaluarachchi, Rajib Rana, and Suranga Nanayakkara. 2023. Improving the domain adaptation of retrieval augmented generation (RAG) models for open domain question answering. *Transactions of the Association for Computational Linguistics*, 11:1–17.

Shamane Siriwardhana, Rivindu Weerasekera, Elliott Wen, and Suranga Nanayakkara. 2021. Fine-tune the entire RAG architecture (including DPR retriever) for question-answering. ArXiv:2106.11517 [cs.IR].

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of the 6th International Conference on Learning Representations*, Vancouver, BC, Canada.

Yuhao Wang, Ruiyang Ren, Junyi Li, Xin Zhao, Jing Liu, and Ji-Rong Wen. 2024. REAR: A relevance-aware retrieval-augmented framework for open-domain question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 5613–5626, Miami, Florida, USA. Association for Computational Linguistics.

Jiayi Wu, Hengyi Cai, Lingyong Yan, Hao Sun, Xiang Li, Shuaiqiang Wang, Dawei Yin, and Ming Gao. 2024. PA-RAG: RAG alignment via multi-perspective preference optimization. ArXiv:2412.14510 [cs.CL].

Yuhao Zhang, Hongji Zhu, Yongliang Wang, Nan Xu, Xiaobo Li, and Binqiang Zhao. 2022. A contrastive framework for learning sentence representations from pairwise and triple-wise perspective in angular space. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4892–4903, Dublin, Ireland. Association for Computational Linguistics.

Hao Zheng, Zhanlei Yang, Wenju Liu, Jizhong Liang, and Yanpeng Li. 2015. Improving deep neural networks using softplus units. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–4.