future internet

MDPI

Article

A Hybrid Text Generation-Based Query Expansion Method for Open-Domain Question Answering

Wenhao Zhu [1,†], Xiaoyu Zhang [1,†], Qiuhong Zhai [1] and Chenyun Liu [2,*]

[1] School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China; whzhu@shu.edu.cn (W.Z.); xiaoyu121@shu.edu.cn (X.Z.); qiuhongzhai@shu.edu.cn (Q.Z.)
[2] Shanghai Municipal Big Data Center, Shanghai 200433, China
* Correspondence: liuchenyun189@163.com
† These authors contributed equally to this work.

Abstract: In the two-stage open-domain question answering (OpenQA) systems, the retriever identifies a subset of relevant passages, which the reader then uses to extract or generate answers. However, the performance of OpenQA systems is often hindered by issues such as short and semantically ambiguous queries, making it challenging for the retriever to find relevant passages quickly. This paper introduces Hybrid Text Generation-Based Query Expansion (HTGQE), an effective method to improve retrieval efficiency. HTGQE combines large language models with Pseudo-Relevance Feedback techniques to enhance the input for generative models, improving text generation speed and quality. Building on this foundation, HTGQE employs multiple query expansion generators, each trained to provide query expansion contexts from distinct perspectives. This enables the retriever to explore relevant passages from various angles for complementary retrieval results. As a result, under an extractive and generative QA setup, HTGQE achieves promising results on both Natural Questions (NQ) and TriviaQA (Trivia) datasets for passage retrieval and reading tasks.

Keywords: query expansion; Pseudo-Relevance Feedback; text generation

Citation: Zhu, W.; Zhang, X.; Zhai, Q.; Liu, C. A Hybrid Text Generation-Based Query Expansion Method for Open-Domain Question Answering . *Future Internet* **2023**, *15*, 180. https://doi.org/10.3390/fi15050180

Academic Editors: Wei Emma Zhang, Chenliang Li and Michael Sheng

Received: 27 April 2023
Revised: 8 May 2023
Accepted: 9 May 2023
Published: 12 May 2023

1. Introduction

Open-Domain Question Answering (OpenQA) aims to answer questions without pre-specified domains, including end-to-end question-answering systems and pipeline-based systems. The former regards the OpenQA system as a whole for joint training, while the latter decomposes the system into multiple components for separate training. Currently, pipeline-based OpenQA systems are more common, such as those developed after 2017, which generally follow the two-stage Retriever-Reader (R2) architecture proposed by Chen et al. [1]. This architecture first retrieves a small subset of passages based on the query, and then the reader extracts or generates answers from the retrieved passages. Therefore, when the retriever retrieves more relevant passages, the reader is more likely to extract the answer, and the performance of the OpenQA system can be improved. However, queries are often short, incompletely expressed semantically, and full of ambiguities. So the model cannot fully understand the query's intent, thus retrieving irrelevant passages, and readers extract or generate incorrect answers.

To improve retrieval efficiency, we use the query expansion technique. Query expansion is the process of reformulating a given query to improve retrieval performance in information retrieval operations, especially in the context of query comprehension, which expands the initial query to match additional relevant passages. Traditional query expansion methods include (1) linguistics-based, (2) corpus-based, (3) search log-based, and (4) network-based approaches. These methods all involve selecting expansion terms that are semantically similar to the original terms from existing knowledge resources or large corpora. Another important method is Pseudo-Relevance Feedback (PRF), which assumes that top-ranked passages at the retrieval stage contain relevant signals and leverages these

signals to modify initial queries, reducing the impact of lexical mismatches between queries and passages and improving search efficiency. BERT [2], T5 [3], and other transformer-based deep language models combined with PRF techniques have significantly improved performance in different search tasks [4,5] compared with traditional PRF methods such as Rocchio, association model, RM3 and KL expansion models.

The two keys of query expansion are the types of expansion terms and the selection methods. (1) The expansion terms extracted by the above methods are more inclined to be semantically close to the items in the original query. However, choosing only semantically related words as expansion terms is insufficient for retrieving the most relevant passages. Utilizing diverse contextual information can help queries retrieve relevant passages from different angles for complementarity. (2) Regarding the method for selecting expansion terms, traditional statistical methods and modern neural networks can be used to select terms from relevant passages. Recently, generative models have played a significant role in OpenQA, performing exceptionally as readers and query expansion generators [6,7]. However, due to the closed-book nature of these models, the generation process can be challenging. Providing some reference materials for the generative models and transforming them into "open-book" generation can speed up the generation process and improve the quality of the generated results [8,9].

In this paper, we propose HTGQE (https://github.com/XY2323819551/TGQE_code, accessed on 27 April 2023), a Hybrid Text Generation-based Query Expansion method to improve retrieval efficiency. We improve the quality of query expansion terms in two ways: the type of expansion generated and the way it is generated. We utilize the generation model as query expansion generator, combining PRF techniques in information retrieval with text generation technologies. By providing "reference materials" for the generation model during text generation, compared to closed-book text generation, HTGQE reduces the difficulty of text generation and accelerates the model's convergence speed and improves the quality of the generated text. Furthermore, diverse query expansion contexts are highly beneficial for query expansion. HTGQE follows Mao et al. [10] and generates three types of expansion contexts for the query: the answer to the question, the sentence containing the answer, and the passage's title where the query is located. In the following, we denote these three contexts as $ctx(A)$, $ctx(S)$, and $ctx(T)$, respectively. Finally, the retrieval results are combined to form new retrieval results. Comprehensive experiments on the NQ and Trivia demonstrate that HTGQE performs well in passage retrieval and reading tasks.

HTGQE increases the lexical overlap between the expanded query and candidate passages, significantly improving sparse retrieval performance. Furthermore, we combine the enhanced sparse retrieval results with dense retrieval results, achieving better retrieval accuracy and question-answering system performance due to their complementary nature. Besides, HTGQE is a plug-and-play method that can be applied to any existing R2 system architecture without altering the structure of the retriever and reader. This query expansion method, independent of the specific retriever and reader structures, means that the benefits of query expansion can be obtained without fine-tuning or training the retriever and reader.

Contributions: (1) HTGQE integrates large language models and PRF techniques, providing "reference materials" for the generation model, making the generation process faster and the generated content of higher quality. (2) HTGQE utilizes the query's answer $ctx(A)$, title $ctx(T)$, and relevant sentences $ctx(S)$ as diversified contexts, expanding the initial query from different perspectives and complementing the retrieval results. (3) Compared to the closed-book text generation method GAR, HTGQE achieves improved EM scores on both NQ and Trivia regardless of the configuration of extractive or generative readers.

The Materials and Methods section introduced the framework of the HTGQE method, the datasets, evaluation metrics, related model structures, and hyperparameter settings for the experiments. The Experimental Results and Analysis section designed detailed experiments to validate HTGQE, including passage retrieval experiments, passage reading experiments, discussions on mixed strategies, investigations on the optimal number of

expansion terms, and case analysis. The Discussion section discussed some limitations of the study. Finally, the Conclusions section concludes the paper.

2. Materials and Methods

2.1. HTGQE Framework and Task Definition

The framework of HTGQE is illustrated in Figure 1. In the retrieval stage, given a corpus $C = \{D_1, D_2, ...D_m\}$, the task of each query q is to return a list $R = \{p_1, p_2, ..., p_k\}$ from C containing k ($k << C$) most relevant passages to the query, i.e., the passages most likely to contain the correct answer for the current query. Then, top-k passages (we set k = 3 in the paper) were selected from the initial retrieval results R as reference materials and concatenated them with the initial query q as $[CLS]q[SEP]p_1...p_k[SEP]$, which is input into the expansion term generation model G(including G_A, G_S, G_T); Each query expansion generator generates the corresponding query expansion terms, namely G_A, G_S and G_T generates $ctx(A)$, $ctx(S)$ and $ctx(T)$, respectively; the three expansion terms, together with the initial query, form new queries q_A, q_S, and q_T to obtain their respective retrieval results R_A, R_S, and R_T; Combine the retrieval results to obtain the final search result $R\prime$. Finally, the obtained retrieval results are sent to the reader to verify the performance of HTGQE on the OpenQA system. To further validate the effectiveness of HTGQE, we equip it with extractive and generative readers for end-to-end OpenQA evaluation. Notably, HTGQE first uses an initial query from a specific dataset to retrieve relevant passages as reference material for the generator. Next, three types of query expansion generators are trained on the specific dataset, and the results of these generators are used to expand the initial query. HTGQE only utilizes the retrieval results of the retriever and has nothing to do with the structure of the retriever or reader. Using a different structure for the retriever or reader does not affect the HTGQE process.

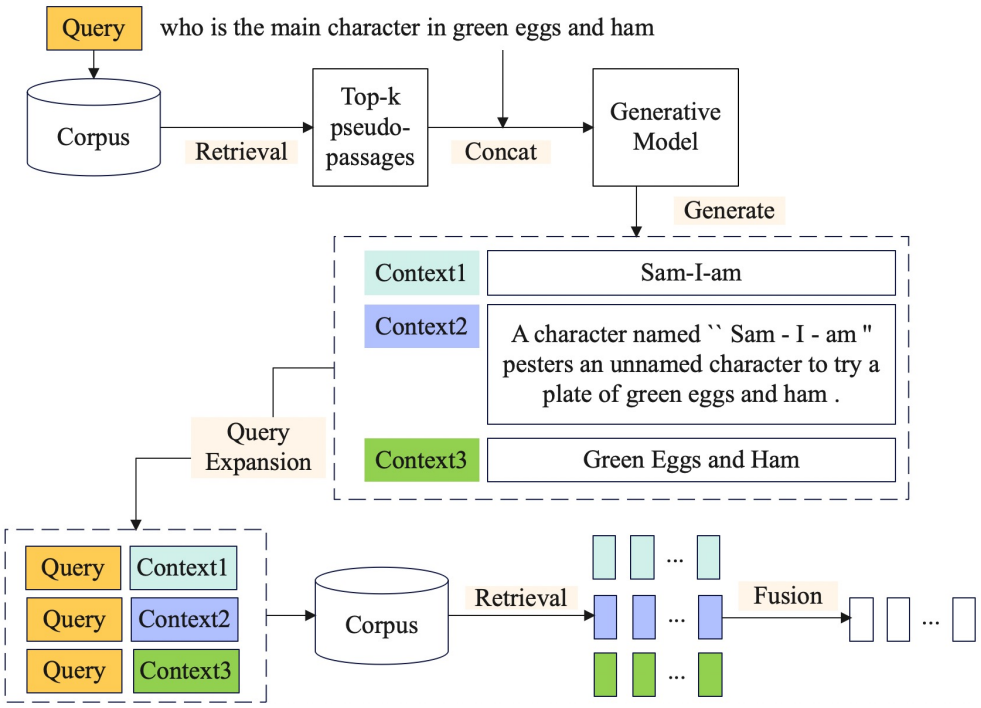


Figure 1. The framework of HTGQE.

Unlike GAR [10], which regards query as input. We combine PRF techniques from information retrieval when generating various contexts, transforming the closed-book generation model into an open-book generation model. Specifically, GAR only treats queries as input, and HTGQE takes the first three pseudo-relevant passages as input to the generated model. This not only reduces the difficulty of text generation but also

significantly improves the quality of the generated text. Below, we introduce the meanings of the three contexts:

$ctx(A)$: Using the answer as query expansion term is valuable for retrieving passages containing the target. This is advantageous because even if the generated answers are only partially accurate or incorrect, the retriever can still retrieve the generated answers if they are related to (e.g., appear with) the passage containing the correct answer.

$ctx(S)$: Sentence containing the default target. Similar to employing the answer as the generation objective, such sentences can enhance the retrieval of relevant passages even if they lack the answer, as their semantics closely align with the question/answer context.

$ctx(T)$:Title of the target passage. Titles of relevant passages frequently represent notable entities and occasionally provide answers to queries.

Although some generated query contexts may include unrealistic or non-factual contexts due to hallucinations in text generation and introduce noise during retrieval, they prove more advantageous than detrimental. Our experiments demonstrate that HTGQE significantly enhances retrieval and question-answering performance compared to BM25. Moreover, generating three different (complementary) query contexts and fusing their retrieval results further mitigated the impact of hallucinated content.

2.2. Datasets

The dataset mainly consists of two parts, one is the passage candidate pool, where the retriever retrieves relevant passages; the other is which benchmark datasets to use for experiments.

(1) Passage Candidate Pool: This study adopts the same version of Wikipedia used in other OpenQA research to construct a candidate pool [1,11], specifically using the English Wikipedia from 20 December 2018, as the source passages for answering questions. By applying the preprocessing code released in DrQA [1], the data is cleaned and each article is divided into multiple unrelated 100-word text blocks, which serve as basic search units. In the end, 21,015,324 passages are obtained. Pyserini [12] has pre-built sparse and dense indexed for these passages, and achieved offline encoding for these passages. The offline indices are used directly in this study for inference, significantly improving efficiency.

(2) Benchmark Datasets: We conduct experiments on the open-domain versions of two widely used QA benchmarks: Natural Questions (NQ) [13] and TriviaQA (Trivia) [14], corresponding to DPR [11]. The questions in NQ were mined from real Google search queries and the answers were spans in Wikipedia articles identified by annotators. Trivia contains a set of trivia questions with answers that were originally scraped from the Web. Each question in both datasets corresponds to a list of answers, and as long as the reader's top-ranked predicted answer appears in the answer list, the current question is considered to be answered correctly. The final statistical information of the datasets used in the experiments is shown in Table 1. *ori* represents the amount of data in each part of the original dataset. We adjusted the dataset according to GAR [10], removing entries that did not qualify, such as queries without corresponding context or title. Finally, we obtain $ctx(A)$, $ctx(s)$ and $ctx(T)$, which are used to train three types of query expansion generators.

Table 1. Statistics of the number of data sets after processing.

	Dataset	Train	Val	Test
NQ	*ori*	79,168	8757	3610
	$ctx(A)$	79,168	8757	3610
	$ctx(S)$	79,168	8757	3610
	$ctx(T)$	67,969	7514	3610
Trivia	*ori*	78,785	8837	11,313
	$ctx(A)$	78,785	8837	11,313
	$ctx(S)$	65,375	7388	11,313
	$ctx(T)$	69,595	7848	11,313

2.3. Evaluation Metrics

Following previous research [1,6,10,11], this study uses Top-k Retrieval Accuracy to evaluate the performance of the retriever and EM (Exact Match) metric to measure the performance of the reader. Top-k Retrieval Accuracy refers to the proportion of questions in which at least one answer span is contained within the top-k retrieved passages. EM is the proportion of predicted answer spans that are exactly the same as one of the ground truth answers after string normalization.

2.4. Model Structure and Initialization

Dual-Tower Retriever. This study uses Pyserini [12] as the retriever. Pyserini [12] is an information retrieval toolkit that supports sparse retrieval by integrating the Anserini toolkit and dense retrieval by integrating Facebook's Faiss Library. Pyserini [12] also supports hybrid retrieval methods by combining dense retrieval results and sparse retrieval results with a fusion factor α, which is set to 1.0 and 0.95 for the NQ and Trivia in the experiments, respectively.

$$h\_score = Sim(q, D_j) + \alpha \cdot BM25(q, D_j) \tag{1}$$

$BM25(q, D_j)$ represents the relevance score between the query and passage in the sparse retrieval; $Sim(q, D_j)$ represents the relevance score in the dense retrieval; *h_score* represents the relevance score for hybrid retrieval.

Extractive Reader. The settings for the extractive reader GAR [10] are in full compliance with the design and implementation of PyGaggle [15], which integrates passage-level voting and evidence fusion techniques. The retrieved passage list and its individual passage relevance scores are denoted as $D = [d_1, d_2, ...d_i, ...d_k]$. The top n text spans in passage d_i are denoted as $S_i = [s_1, s_2, ...s_j..., s_n]$. Text spans are ranked by calculating the relevance scores $\vec{S_i}$. Finally, the prediction scores for the same text span from different retrieved passages are aggregated using normalized answer span scoring techniques. PyGaggle [15] uses evidence fusion techniques, fusing $\vec{D}$ and $\vec{R}$ through the formula $\beta \cdot \vec{D} + \gamma \cdot \vec{R_i}$, where $\vec{R}$ represents the retrieval scores assigned to each passage by the retriever. Depending on the chosen retrieval method, $\vec{R}$ can be sparse retrieval scores or hybrid retrieval scores. The final combined score replaces the original $\vec{D}$ as the relevance score for all contexts in the answer span scoring step. For example, the calculation of the answer span by Mao et al. [10] using evidence fusion techniques is as follows:

$$p(S_i[j]) = softmax(\beta \cdot \vec{D} + \gamma \cdot \vec{R_i})[i] \times softmax(\vec{S_i})[j] \tag{2}$$

We called this reader GAR*.

Generative Reader. In this paper, we choose the model published by Izacard [6] as the generative reader. This generative model for OpenQA is based on a sequence-to-sequence network model, pre-trained on unsupervised data such as the T5 [3] model. Each retrieved passage and its title are connected to the question as a single data sample for processing, independent of other passages; the decoder applies attention mechanisms on

the distributed representation of all data (output by the encoder) to obtain the answer. The encoder processes passages independently, which can be scaled to large datasets as it computes self-attention mechanisms for only one data sample at a time. However, jointly processing passages in the decoder can better aggregate evidence from multiple passages.

Query Expansion Generator. In this paper, we utilize BART-large [16] as baseline models to obtain G_A, G_S and G_T, which generate three types of query expansion contexts, respectively. The top three pseudo-relevant passages from the retrieval results are used as "reference materials" for the generative model, denoted as p_1, p_2, and p_3. These passages are concatenated with the initial query q using [SEP], resulting in an input format of $[CLS]q[SEP]p_1[SEP]p_2[SEP]p_3[SEP]$ for the generative model.

Model Initialization. The retriever uses a dual-tower retrieval model and has been validated on both extractive and generative readers. We use the models released by the Pyserini (https://github.com/castorini/pyserini/blob/master/docs/experiments-dpr.md), accessed on 2 March 2022 and PyGaggle (https://github.com/castorini/pygaggle/blob/master/docs/experiments-dpr-reader.md) toolkits, accessed on 31 January 2021 as the two main modules in the R2 system. For the NQ dataset, we use the RetrieverNQ and ReaderNQ-Single models; for the Trivia dataset, we use the RetrieverMulti and ReaderTQA-Multi models (https://github.com/facebookresearch/DPR, accessed on 11 February 2023). (2) Generative Reader: For the generative reader, we use the models released by Izacard [6] (https://github.com/facebookresearch/FiD, accessed on 1 February 2023), using the ReaderNQ-T5 base version model on the NQ dataset and the ReaderTQA-T5 base version model on the Trivia dataset.

2.5. Hyperparameter Settings

(1) Evidence Mixing Parameters β and γ. Following the GAR* implementation in the PyGaggle library [15], for sparse retrieval, we set the parameters β (i.e., the relevance score obtained from the reader) and γ (i.e., the retrieval score obtained from the retriever) to 0.46 and 0.308 on the NQ dataset, and 0.78 and 0.093 on the Trivia dataset, respectively. For mixed retrieval results, we set β and γ to 0.32 and 0.1952 on the NQ dataset, and 0.76 and 0.152 on the Trivia dataset, respectively.

(2) In the paper, BART-large is employed as the query expansion generator, with the generative model training conducted on two Tesla V100 GPUs. For both the NQ and Trivia, the learning rate can be uniformly set at 3×10^{-5}; however, when training the G_T, it is advisable to increase the learning rate, such as by adjusting it to 1×10^{-4}. While training the G_A, G_S and G_T, the batch sizes are set at 64, 16, and 8, respectively; for the Trivia dataset, they are 16, 8 and 8. Furthermore, due to the incorporation of PRF, the maximum input length is set to 768 tokens, with the maximum generation lengths being 16 ($ctx(A)$), 64 ($ctx(S)$), and 32 ($ctx(T)$) tokens, respectively. However, if the experimental conditions are changed, the above parameters also need to be adjusted appropriately.

(3) When we select top-k pseudo-relevant passages as references for generative models, we set k = 3 in our experiments. If k is too large, the input for the query expansion generation model will become too long, affecting the training and inference speed of the generation model. If k is too small, the reference material provided to the generation model will be insufficient, making it difficult to produce correct answers. Of course, other values of k are also worth trying.

3. Experimental Results and Analysis

This section primarily focuses on conducting comprehensive experiments to validate the proposed HTGQE. Section 3.1 deals with the passage retrieval task based on the HTGQE approach; Section 3.2, building upon the retrieval results of Section 3.1, investigates the advantages of the HTGQE in OpenQA; Section 3.3 thoroughly explores and discusses various fusion strategies for different retrieval results; Section 3.4 examines the impact of the number of generated expansion terms on retrieval accuracy; finally, to better understand

the process and rationale of the HTGQE, Section 3.5 presents and analyzes several case studies.

3.1. Passage Retrieval with HTGQE

This section focuses on highlighting the effectiveness of the HTGQE. In addition to the BM25 and the BM25+RM3 [10], other comparisons are carried out between GAR [10] and HTGQE. As shown in Table 2, we examine the Top-k retrieval accuracy of both sparse and hybrid retrieval methods. In sparse retrieval, The result of GAR is a combination of GAR($ctx(A)$), GAR($ctx(S)$), and GAR($ctx(T)$); the specific fusion method can be found in Section 3.3. HTGQE follows the same principle.

Table 2. Top-k retrieval accuracy of various on NQ and Trivia.

	Dataset	**NQ**					**Trivia**				
	Methods	**Top-5**	**Top-20**	**Top-100**	**Top-500**	**Top-1000**	**Top-5**	**Top-20**	**Top-100**	**Top-500**	**Top-1000**
Sparse	BM25(ours)	43.8	62.9	78.3	85.6	88.0	66.3	76.4	83.2	87.3	8.5
	BM25+RM3 [10]	44.6	64.2	79.6	86.8	88.9	67.0	77.1	83.8	87.7	88.9
	GAR ($ctx(A)$)	53.2	67.1	79.4	86.7	89.2	66.5	76.0	83.7	88.0	89.2
	HTGQE ($ctx(A)$)	64.2	73.7	82.7	87.8	89.5	74.2	**79.8**	**85.0**	88.5	89.5
	GAR ($ctx(S)$)	52.7	66.6	79.5	86.5	89.0	62.3	72.6	81.0	86.7	88.0
	HTGQE ($ctx(S)$)	64.4	73.2	82.2	88.1	89.6	72.6	78.0	83.2	87.1	88.4
	GAR ($ctx(T)$)	50.8	67.2	79.8	87.0	88.8	65.1	75.3	82.6	87.0	88.3
	HTGQE ($ctx(T)$)	55.7	69.8	81.0	87.4	89.2	70.2	77.8	83.8	87.7	88.8
	GAR(ours)	58.5	72.6	83.8	89.2	90.9	68.7	77.7	84.2	88.2	89.3
	HTGQE	**67.3**	**76.2**	**84.5**	**89.8**	**91.3**	**74.2**	79.6	84.9	**88.6**	**89.6**
Hybrid	BM25+DPR [12]	60.3	76.1	86.0	90.1	91.7	72.1	80.9	86.3	89.0	89.8
	GAR+DPR(ours)	66.2	79.1	87.3	91.3	92.4	73.2	81.5	86.6	89.2	90.0
	HTGQE+DPR	**69.7**	**80.6**	**87.4**	**91.5**	**92.5**	**75.2**	**81.8**	**86.7**	**89.5**	**90.2**

For sparse retrieval, GAR and HTGQE significantly outperform the initial BM25 and BM25+RM3, demonstrating the effectiveness of query expansion based on generative models. Under various conditions, HTGQE's Top-k retrieval accuracy surpasses that of GAR, indicating that with proper "reference materials," the accuracy of open-domain generative models can be significantly improved. HTGQE uses the top three Pseudo-Relevance passages of the initial query retrieval results as these reference materials. In the NQ/Trivia dataset, GAR's Top-5 and Top-20 accuracy are 14.7%/2.4% and 9.7%/1.3% higher than BM25, respectively, while HTGQE's Top-5 and Top-20 retrieval accuracy are 8.8%/5.5% and 3.6%/1.9% higher than GAR, respectively. Furthermore, regardless of how many passages are retrieved, HTGQE consistently outperforms GAR, and the gap between GAR and HTGQE narrows as the value of k increases. In contrast, the classic QE method RM3 slightly improves over regular BM25 but does not perform comparably to HTGQE or GAR. According to Table 2, the effectiveness of different query contexts varies, with $ctx(A)$ and $ctx(S)$ being more effective than $ctx(T)$. However, the hybrid results are higher than any individual retrieval performance, suggesting that merging each query context will yield better retrieval results. The complementary nature of different generative-enhanced queries is also mentioned in Section 3.4.

For hybrid retrieval, as seen in Table 2, the retrieval accuracy of HTGQE+DPR is higher than that of GAR+DPR and BM25+DPR. On NQ, HTGQE+DPR's Top-5/Top-20 accuracy are 9.4% 4.5% and 3.5%/1.5% higher than BM25+DPR and GAR+DPR, respectively. On Trivia, HTGQE+DPR's Top-5/Top-20 accuracy are 3.1%/2.0% and 0.9%/0.3% higher than BM25+DPR and GAR+DPR, respectively. The specific fusion method is described in Section 3.3.

3.2. Passage Reading with HTGQE

To evaluate whether the HTGQE approach can offer OpenQA improvements, we conducted experiments comparing the open-book HTGQE and closed-book GAR models using the retrieval results from Section 3.1. The outstanding performance of GAR is already

known in previous study [10], this section aims to verify that the HTGQE is more effective than GAR. It is important to note that both the extractive and generative readers in this paper have not undergone training, so they will not achieve the results reported in the GAR [10].

As shown in Table 3, the top 50 candidate passages from the retrieval results are selected as inputs for the reader. HTGQE outperforms GAR in both extractive and generative settings. For the extractive reader, the former surpasses the latter by 4.57% and 6.35% on the NQ and Trivia, respectively. In the generative reader setting, HTGQE exceeds GAR by 14.34% and 17.09%, respectively. This further demonstrates that text generation incorporating PRF techniques is more effective because it provides useful reference materials during the text generation process. Moreover, the different contextual environments also contribute significantly to the experimental results.

Furthermore, while HTGQE focuses on improving sparse retrieval, the new sparse retrieval results bring improvements in EM scores that outperform many other neural network methods, even if they take 50, 100, or even more passages as the input of the readers. However, since we directly use the existing reader checkpoint and do not train the reader on the new retrieval results, the reader has yet to learn the knowledge in the new retrieval results. So there is still a gap between our method and SOTA methods. If you retrain the reader, expect to see a significant performance improvement.

Table 3. OpenQA system performance comparison between HTGQE and other methods.

Methods		**NQ**	**Trivia**
Extractive	Hard EM [17]	28.1	50.9
	Path Retriever [18]	32.6	-
	ORQA [19]	33.3	45.0
	Graph Retriever [20]	34.5	56.0
	REALM [21]	40.4	-
	DPR [11]	41.5	57.9
2-4 **Generative**	GPT3 [22]	29.9	-
	T5 [3]	36.6	60.5
	SpanSeqGen [23]	42.2	-
	RAG [4]	44.5	56.1
	FiD [6]	**51.4**	**67.6**
2-4 **Sparse**	BM25 (E,ours)	27.2	35.4
	GAR (E)	30.4	44.4
	HTGQE (E)	35.0	50.7
	BM25 (G,ours)	28.8	47.1
	GAR (G)	24.0	39.7
	HTGQE (G)	**38.3**	**56.8**

3.3. Fusion Strategies for Retrieval Results

In this section, we explore how to fuse the three types of retrieval results better to achieve improved retrieval accuracy. We have experimented with four fusion methods: $ctx(AS)$, $ctx(AT)$, $ctx(ST)$ and $ctx(AST)$ (for example, $ctx(AS)$ refers to the combination of results retrieved using $ctx(A)$ and $ctx(S)$ as query expansion terms). According to Table 4, the overall performance is better when fusing the results of all three types, with $ctx(AS)$ being the second best. This shows that the retrieval results obtained using $ctx(A)$, $ctx(S)$, and $ctx(T)$ as query expansion terms are complementary.

Furthermore, this paper employs a uniform fusion approach, in which an equal number of candidate passages are extracted from each retrieval result to form a new candidate passage set. For instance, in the $ctx(AS)$ method, for each query, we extract the top 500 candidate passages from the retrieval results corresponding to the query with $ctx(A)$, and the top 500 candidate passages from the retrieval results corresponding to the query with $ctx(S)$. These two sets are then intermixed to form the final retrieval results. This fusion approach is relatively simple, but other fusion methods, such as rank reciprocal, can also be considered.

Table 4. Discussion on hybrid strategies.

	NQ						Trivia					
	Top-1	**Top-5**	**Top-20**	**Top-100**	**Top-500**	**Top-1000**	**Top-1**	**Top-5**	**Top-20**	**Top-100**	**Top-500**	**Top-1000**
$ctx(AS)$	**53.4**	67.0	75.3	84.2	89.5	90.8	**65.3**	**74.3**	**79.8**	**84.9**	**88.6**	89.5
$ctx(AT)$	34.3	64.2	75.1	83.7	89.2	90.2	65.1	74.0	79.6	84.8	88.5	89.5
$ctx(ST)$	34.3	65.5	75.9	**84.5**	89.6	91.1	**65.3**	73.4	79.2	84.5	88.2	89.2
$ctx(AST)$	**53.4**	**67.3**	**76.2**	**84.5**	**89.8**	**91.3**	**65.3**	74.2	79.6	**84.9**	**88.6**	**89.6**

3.4. Exploring the Optimal Number of Expansion Terms

In this paper, we use three query expansion generators to generate various contextual information for initial query. However, there is yet to be a definitive answer regarding the optimal number of expansion terms for each context. The previous experiments have all assumed a default number of expansion terms to be 1. This section aims to explore the optimal number of expansion terms for various contexts. We conducted experiments with the number of expansion terms set to 1, 2, 3, 4, and 5, respectively.

On NQ, as shown in Figure 2, we investigated the impact of different numbers of expansion terms on the Top-k retrieval accuracy. The four graphs in Figure 2 show a fairly consistent trend: for $ctx(A)$, the performance is better when the number of expansion terms is 2 (orange line); for $ctx(S)$, the performance is relatively better when the number of expansion terms is 1 (light blue line); for $ctx(T)$, the performance is better when the number of expansion terms is 1 or 5(light blue line or blue line). Lastly, we combined the three individual retrieval results corresponding to the number of expansion terms to obtain a mixed effect. Overall, the optimal result can be achieved when the number of expansion terms is 5.

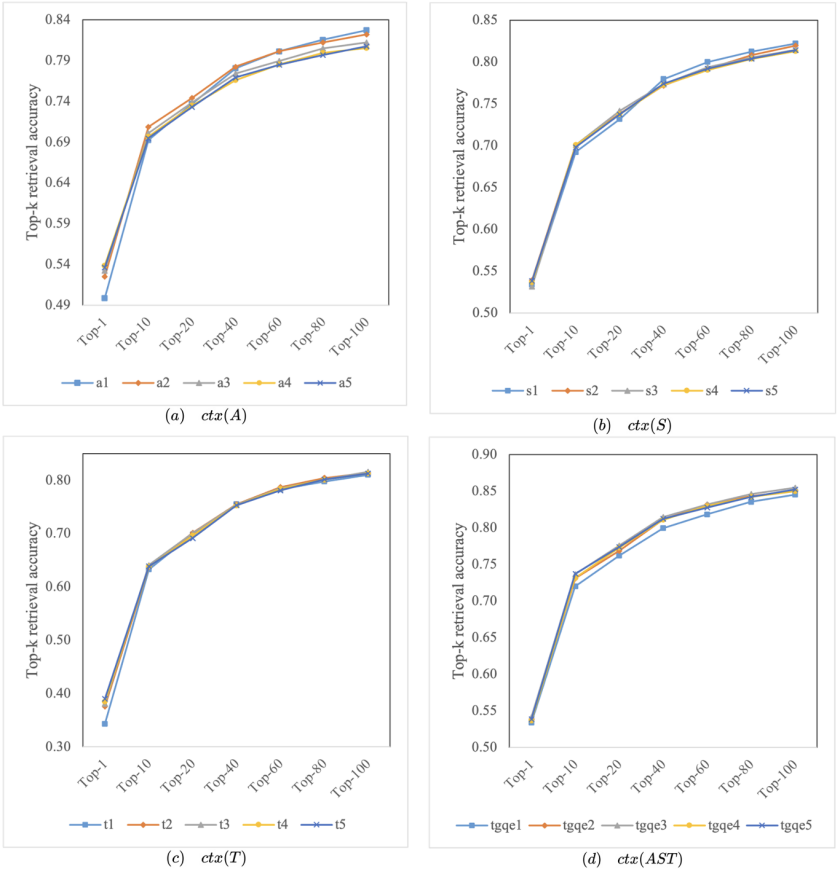
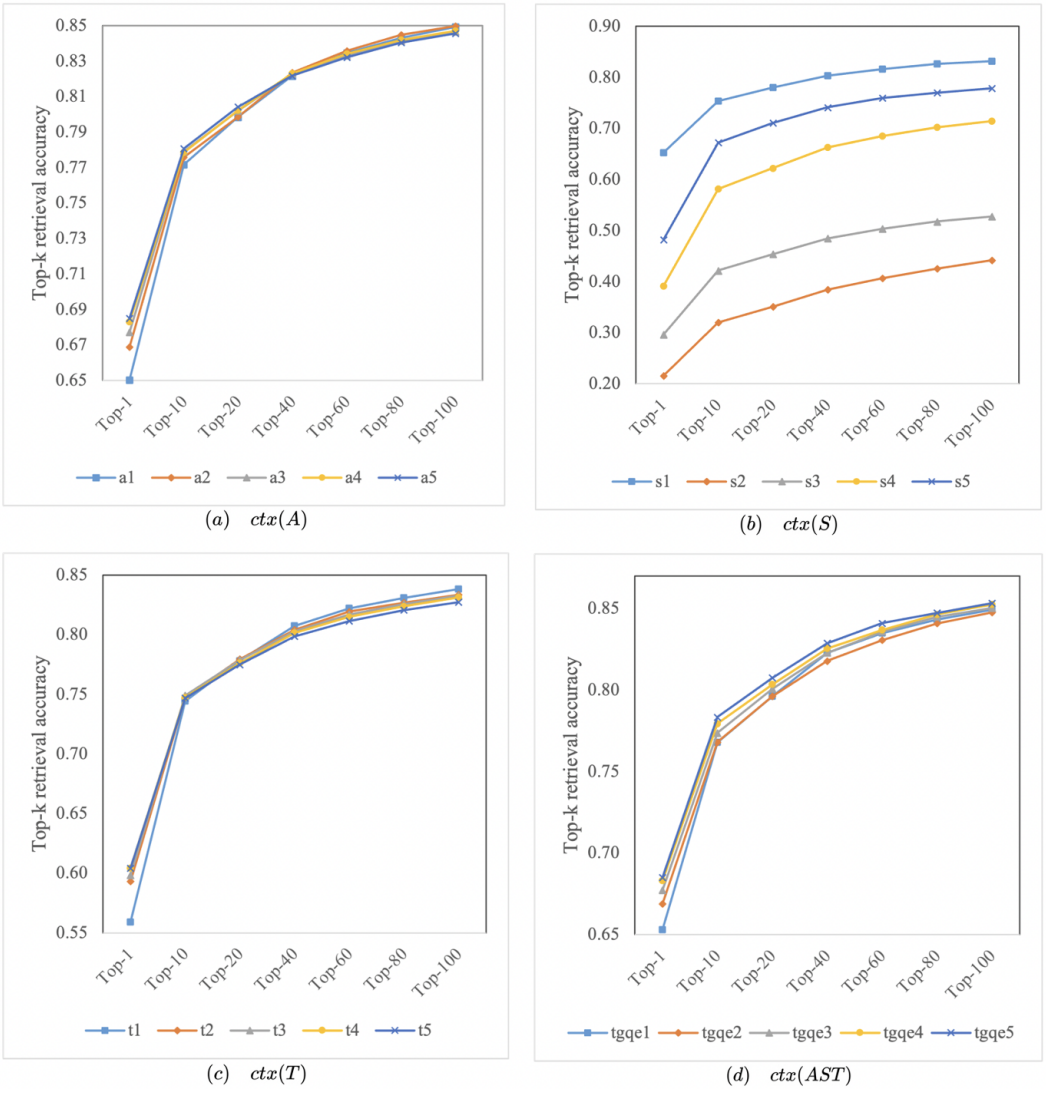


Figure 2. Analysis of the number of expansion terms on NQ.

As shown in Figure 3, we observed a fairly consistent trend on Trivia: for $ctx(A)$, the performance is relatively good when the number of expansion terms is 2 or 5 (represented by the orange and blue line, respectively); for $ctx(S)$, the performance is comparatively better when the number of expansion terms is 1 (indicated by the light blue line); for $ctx(T)$, the optimal number of expansion terms is 1 or 5. Lastly, fusing the three individual retrieval results according to the corresponding expansion term quantities achieves the overall best performance when the number of expansion terms is 5. Although the influence of the expansion terms quantity on retrieval accuracy in $ctx(S)$ query expansion is significant, this gap is greatly alleviated after combining the three contexts, further highlighting their complementarity.



(*a*) $ctx(A)$ (*b*) $ctx(S)$

(*c*) $ctx(T)$ (*d*) $ctx(AST)$

Figure 3. Analysis of the number of expansion terms on Trivia.

In summary, the best results can be achieved when the number of expansion terms is set to 5. However, the optimal performance is similar when the number of expansion terms is 1, 2, 3, or 4. Additionally, too many expansion terms can lead to an excessively long initial query, which may reduce the retrieval speed. Therefore, whether to choose a quantity of 1 or 5 for expansion terms depends on the specific application requirements. If the goal is to achieve higher accuracy, it is advisable to set the number of expansion terms to 5. On the other hand, if the focus is inference speed, opting for a single expansion term is a better choice.

3.5. Case Study

To facilitate a better understanding of the HTGQE process and its underlying ideas for the readers, we present three examples in Table 5. Correct answers are highlighted in green, incorrect ones in red, and the correct reference answers are enclosed in blue brackets. For each query, we concatenate it with the top three pseudo-relevant passages from the initial retrieval results and input them into different query expansion generators. The $ctx(A)$, $ctx(S)$ and $ctx(T)$ are three query expansion contexts. Finally, we use the three generated contexts as expansion terms for the initial query and conduct retrieval, fusing all retrieval results. In the first example, the target appears in $ctx(A)$, but not in $ctx(S)$ and $ctx(T)$. In the second example, the target does not appear in $ctx(A)$; fortunately, $ctx(S)$ and $ctx(T)$ both provide the target. In addition, the final examples, only $ctx(S)$ provides the correct target. While not every context provides the correct target, three contexts increase the probability of finding the correct target, all of which are relevant to the query. This demonstrates that different query contexts complement each other, reducing noise during query context generation and improving the final retrieval accuracy.

Table 5. Examples of generated query contexts. Ground-truth references are shown in the {braces}. Incorrect predictions are shown in red. Correct predictions are shown in green.

case1	**query:** who started the trojan war in the iliad? $ctx(A)$**: Paris** {[Paris of Troy]} $ctx(S)$: the Trojan War was started by the **Achaeans (Greeks)**. {In Greek mythology , the Trojan War was waged against the city of Troy by the Achaeans (Greeks) after Paris of Troy took Helen from her husband Menelaus.} $ctx(T)$**: Trojan War** {Helen of Troy (film)}
case2	**query:** what is the european recovery program of 1947 better known as? $ctx(A)$**: European Recovery Plan** {[The Marshall Plan]} $ctx(S)$: **The Marshall Plan, officially the European Recovery Program.** (born November 26, 1962) is an American country music singer. {The Marshall Plan was an American initiative to aid Western Europe, in which the United States gave over $13 billion in economic support to help rebuild Western European economies after the end of World War II.} $ctx(T)$**: as Marshall Plan** [SEP] **Committee for the Marshall Plan** [SEP] I Do (Reba McEntire album) {Committee of European Economic Co-operation [SEP] Federal Ministry of Matters of the Marshall Plan.}
case3	**query:** who is the original singer of and i am telling you? $ctx(A)$**: Amber Riley** {[Jennifer Holliday]} $ctx(S)$**: Jennifer Holliday** for its re-release in 1982 for which it became a number - one R&B hit for Holliday. {Jennifer Holliday originated the role on Broadway in 1981 and won a Tony Award for her performance as well as the Grammy for Best R&B Performance.} $ctx(T)$**: And I Am Telling You I'm Not Going** {I Am Changing [SEP] Jennifer Hudson [SEP] Jennifer Holliday [SEP] Dreamgirls}

4. Discussion

HTGQE can also be improved in the following aspects: (1) Relevant reference materials. HTGQE combines the PRF techniques in information retrieval and the text generation task to provide reference materials, transforming the closed-book task into an open-book task, making the generation of targets easier and more accurate. In addition to utilizing the content in the retrieval results, the reference materials can also use any useful knowledge, such as knowledge base content related to the query terms, web knowledge, chart content, etc., as long as it provides the correct guidance for the text generation task. (2) Various contextual information. Regarding the generation targets, this paper mainly generates three kinds of query expansion contexts such as GAR [10]: the answer corresponding to the query, the sentence containing the answer, and the title of the passage containing the query.

Of course, many other expansion contexts can improve the retrieval performance, such as generating the background knowledge of the queries or directly generating multiple context information related to the query, ranking these contexts, and selecting the most relevant ones as query expansion terms. This issue is worth further exploration. (3) Fusion methods. Additionally, the fusion methods of individual retrieval results still require in-depth investigation. The fusion approach used in this paper is rather simplistic and crude; in the future, we will attempt various fusion techniques further to enhance the retrieval stage and OpenQA system performance. (4) Generative models. Furthermore, with the increasing popularity of the ChatGPT model, the potential of generative models has been demonstrated. Therefore, it is worth exploring more query expansion methods based on generative models in the future.

5. Conclusions

In this work, we propose HTGQE, which improves retrieval efficiency and effectiveness by combining generation models and PRF techniques to generate various query expansion contexts. The pseudo-relevant passages served as reference material for the generation model, which increases the model's input while reducing the pressure on generative models and improving the quality of generated terms. Various query expansion contexts complement each other, reducing noise during query context generation and improving the final retrieval accuracy. Notably, compared to the closed-book text generation method GAR, when configured with extractive readers, HTGQE achieves EM score improvements of 4.57% and 6.35% on NQ and Trivia, respectively. When equipped with generative readers, HTGQE achieves EM score improvements of 14.34% and 17.09% on NQ and Trivia datasets, respectively. In addition, HTGQE is a plug-and-play approach that can be applied to existing R2 systems under specific datasets, independent of the specific structure of the retriever and reader models, which allows easy and quick integration into existing R2 systems. However, our method still has its limitations. Adding expansion terms to the initial query can improve the retrieval efficiency of the QA system compared to the standard R2 system, but also slow down the inference speed within an acceptable range.

Author Contributions: Conceptualization, X.Z. In addition, W.Z.; methodology, X.Z. In addition, W.Z.; software, X.Z. In addition, W.Z.; validation, Q.Z. In addition, C.L.; formal analysis, X.Z. In addition, W.Z.; investigation, Q.Z.; resources, C.L.; data curation, X.Z. In addition, Q.Z.; writing—original draft preparation, X.Z., W.Z. In addition, Q.Z.; writing—review and editing, C.L.; visualization, X.Z.; supervision, C.L.; project administration, C.L.; funding acquisition, W.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by National Natural Science Foundation of China (No. 61572434) and Shanghai Science and Technology Committee (No. 19DZ2204800).

Data Availability Statement: The data used in the experiment are all presented in the Github Repository: https://github.com/XY2323819551/TGQE_code. All the data is public, accessed on 27 April 2023.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

OpenQA	Open-Domain Question Answering
PRF	Pseudo-Relevance Feedback
HTGQE	Hybrid Text Generation-based Query Expansion
QE	Query Expansion
NQ	Natural Questions
Trivia	TriviaQA

References

1. Chen, D.; Fisch, A.; Weston, J.; Bordes A. Reading wikipedia to answer open-domain questions. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, BC, Canada, 30 July–4 August 2017; pp. 1870–1879.
2. Devlin, J.; Chang, M.; Lee, K.; Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, 2–7 June 2019; pp. 4171–4186.
3. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* **2020**, *21*, 5485–5551.
4. Lewis, P.S.H.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.T.; Rocktäschel, T.; et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Proceedings of the Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, Virtual, 6–12 December 2020.
5. Yu, H.; Dai, Z.; Callan, J. PGT: Pseudo Relevance Feedback Using a Graph-Based Transformer. In *Advances in Information Retrieval, Proceedings of the 43rd European Conference on IR Research, ECIR 2021, Virtual Event, 28 March–1 April 2021*; Springer: Berlin/Heidelberg, Germany, 2021; Volume 12657, pp. 440–447.
6. Izacard, G.; Grave, E. 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, Online, 19–23 April 2021; pp. 874–880.
7. Liu, J.; Liu, A.; Lu, X.; Welleck, S.; West, P.; Bras, R.L.; Choi, Y.; Hajishirzi, H. Generated knowledge prompting for commonsense reasoning. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, ACL 2022, Dublin, Ireland, 22–27 May 2022; pp. 3154–3169.
8. Yu, W.; Zhu, C.; Li, Z.; Hu, Z.; Wang, Q.; Ji, H.; Jiang, M. A survey of knowledge-enhanced text generation. *ACM Comput. Surv.* **2022**, *54*, 1–38. [CrossRef]
9. Li, H.; Su, Y.; Cai, D.; Wang, Y.; Liu, L. A Survey on Retrieval-Augmented Text Generation. *arXiv* **2022**, arXiv:2202.01110.
10. Mao, Y.; He, P.; Liu, X.; Shen, Y.; Gao, J.; Han, J.; Chen, W. Generation-augmented retrieval for open-domain question answering. In the Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, Virtual Event, 1–6 August 2021; pp. 4089–4100.
11. Karpukhin, V.; Oguz, B.; Min, S.; Lewis, P.; Wu, L.; Edunov, S.; Chen, D.; Yih, W.T. Dense passage retrieval for open-domain question answering. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, 16–20 November 2020; pp. 6769–6781.
12. Lin, J.; Ma, X.; Lin, S.C.; Yang, J.H.; Pradeep, R.; Nogueira, R. 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, Virtual Event, 11–15 July 2021; pp. 2356–2362.
13. Seo, M.J.; Lee, J.; Kwiatkowski, T.; Parikh, A.; Farhadi A.; Hajishirzi H. Real-time open-domain question answering with dense-sparse phrase index. In Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, 28 July–2 August 2019; pp. 4430–4441.
14. Joshi, M.; Choi, E.; Weld, D.S.; Zettlemoyer, L. Triviaqa: A large scaled istantly supervised challenge dataset for reading comprehension. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, BC, Canada, 30 July–4 August 2017; pp. 1601–1611.
15. Nogueira, R.; Jiang, Z.; Lin, J. Document ranking with a pretrained sequence-to-sequence model. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16–20 November 2020; pp. 708–718.
16. Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; Zettlemoyer, L. 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, ACL 2020, Online, 5–10 July 202; pp. 7871–7880.
17. Min, S.; Chen, D.; Hajishirzi, H.; Zettlemoyer, L. A discrete hard EM approach for weakly supervised question answering. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, 3–7 November 2019; pp. 2851–2864.
18. Asai, A.; Hashimoto, K.; Hajishirzi, H.; Socher, R.; Xiong, C. Learning to retrieve reasoning paths over wikipedia graph for question answering. In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020.
19. Lee, K.; Chang, M.; Toutanova, K. Latent retrieval for weakly supervised open domain question answering. In Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, 28 July–2 August 2019; pp. 6086–6096.
20. Min, S.; Chen, D.; Zettlemoyer, L.; Hajishirzi, H. Knowledge guided text retrieval and reading for open domain question answering. *arXiv* **2019**, arXiv:1911.03868.
21. Guu, K.; Lee, K.; Tung, Z.; Pasupat, P., Chang, M.W. REALM: Retrieval-Augmented Language Model Pre. *Training* **2020**. Available online: https://www.semanticscholar.org/paper/REALM%3A-Retrieval-Augmented-Language-Model-Guu-Lee/832fff14d2ed50eb7969c4c4b976c35776548f56 (accessed on accessed on 1 May 2023).

22. Brown, T.B.; Mann, B.; Ryder, N. Language models are few-shot learners. In Proceedings of the Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, Virtual, 6–12 December 2020.
23. Min, S.; Michael, J.; Hajishirzi, H.; Zettlemoyer, L. Ambigqa: Answering ambiguous open-domain questions. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, 16–20 November 2020; pp. 5783–5797.