CorpusBrain: Pre-train a Generative Retrieval Model for Knowledge-Intensive Language Tasks

Jiangui Chen
CAS Key Lab of Network Data Science and Technology, ICT, CAS
University of Chinese Academy of Sciences
Beijing, China
chenjiangui18z@ict.ac.cn

Ruqing Zhang
CAS Key Lab of Network Data Science and Technology, ICT, CAS
University of Chinese Academy of Sciences
Beijing, China
zhangruqing@ict.ac.cn

Jiafeng Guo*
CAS Key Lab of Network Data Science and Technology, ICT, CAS
University of Chinese Academy of Sciences
Beijing, China
guojiafeng@ict.ac.cn

Yiqun Liu
Dept. CS&T, Beijing National Research Center for Information Science and Technology, Tsinghua University
Beijing, China
yiqunliu@tsinghua.edu.cn

Yixing Fan
CAS Key Lab of Network Data Science and Technology, ICT, CAS
University of Chinese Academy of Sciences
Beijing, China
fanyixing@ict.ac.cn

Xueqi Cheng
CAS Key Lab of Network Data Science and Technology, ICT, CAS
University of Chinese Academy of Sciences
Beijing, China
cxq@ict.ac.cn

ABSTRACT

Knowledge-intensive language tasks (KILT) usually require a large body of information to provide correct answers. A popular paradigm to solve this problem is to combine a search system with a machine reader, where the former retrieves supporting evidences and the latter examines them to produce answers. Recently, the reader component has witnessed significant advances with the help of large-scale pre-trained generative models. Meanwhile most existing solutions in the search component rely on the traditional "index-retrieve-then-rank" pipeline, which suffers from large memory footprint and difficulty in end-to-end optimization. Inspired by recent efforts in constructing model-based IR models, we propose to replace the traditional multi-step search pipeline with a novel single-step generative model, which can dramatically simplify the search process and be optimized in an end-to-end manner. We show that a strong generative retrieval model can be learned with a set of adequately designed pre-training tasks, and be adopted to improve a variety of downstream KILT tasks with further fine-tuning. We name the pre-trained generative retrieval model as *CorpusBrain* as all information about the corpus is encoded in its parameters without the need of constructing additional index. Empirical results show that CorpusBrain can significantly outperform strong baselines for the retrieval task on the KILT benchmark and establish new state-of-the-art downstream performances. We also show that CorpusBrain works well under zero- and low-resource settings.

*Jiafeng Guo is the corresponding author.

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557271

CCS CONCEPTS

• **Information systems** → **Retrieval models and ranking**.

KEYWORDS

Model-based IR; Pre-training; Generative Retrieval

ACM Reference Format:
Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Yiqun Liu, Yixing Fan, and Xueqi Cheng. 2022. CorpusBrain: Pre-train a Generative Retrieval Model for Knowledge-Intensive Language Tasks. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3511808.3557271

1 INTRODUCTION

Knowledge-intensive language tasks (KILT) such as fact checking [45] and open-domain question answering [48], have received much attention in recent years. Compared with traditional information processing tasks, they are usually more complex and require to surface knowledge from a large body of information. A popular paradigm to approach these tasks is to combine a search system with a machine reader [38]. The former retrieves a limited subset of supporting evidences from large corpora, and the latter then examines the retrieved information to produce final answers. Recently, the reader component has witnessed significant advances with the help of the large-scale pre-trained generative models (e.g., BART [28] and T5 [28]), and such models have become the de-facto implementation of the reader. However, the search component has yet to benefit from these tremendous advances in generative models.

Most existing solutions in the search component follow the paradigm of "index-retrieve-then-rank" [7, 13, 21, 49, 51] which includes three sequential steps: (1) building an index for each document in the corpus; (2) retrieving an initial set of candidate documents for a query; and (3) determining the relevance degree of each candidate. Despite its wide usage, this paradigm has clear limitations. At the training stage, heterogeneous ranking components are usually

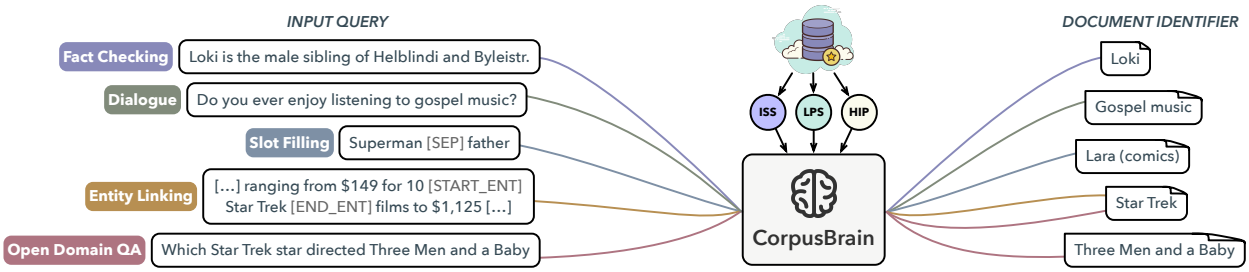

Figure 1: Overview of how the CorpusBrain can be adopted to solve a variety of downstream KILT tasks. At the pre-training stage, we design three pre-training tasks (i.e., ISS, LPS and HIP) to encode the knowledge about the corpus, as shown in Figure 2. If downstream supervised data is available, CorpusBrain can be further fine-tuned to improve the retrieval performance. Given a query, the retrieval task is cast as a Seq2Seq learning problem, to generate the identifiers of relevant documents.

difficult to be optimized in an end-to-end way towards the global objective. At the inference stage, a large document index is needed to search over the corpus, leading to significant memory consumption and computational overhead. Besides, errors would accumulate and propagate among the sequential components.

Recently, Metzler et al. [37] envisioned a fundamentally different paradigm called model-based IR, to replace the long-standing "index-retrieve-then-rank" paradigm. Specifically, with model-based IR, the indexing, retrieval, and ranking components of traditional IR systems are collapsed into a single consolidated model. This enables us to mitigate the aforementioned technical issues. Firstly, the knowledge of all documents in the corpus is encoded into the model parameters, which can be optimized directly in an end-to-end manner. Secondly, the memory and computational cost is greatly reduced because the document index is eliminated. Documents are returned using model parameters only, which can dramatically simplify the heavy search process. Inspired by this blueprint, researchers [6, 9, 44] have proposed to generate identifier strings for documents via generative language models. A typical way is to directly fine-tune the off-the-shelf pre-trained generative models (e.g., BART [28]) for specific KILT task, e.g., entity linking [9] and fact checking [6]. Unfortunately, these methods depend on large-scale application-specific supervision, which are often not available for many real-world applications.

Therefore, in this paper, we propose to pre-train a novel single-step generative model, called *CorpusBrain*, to encode all information of the corpus within its parameters in a general way. In this work, we assume that the pre-training data is defined as positive pairs of query and document identifier. To resemble the relevance relationship between query and document in downstream KILT tasks, the pre-training task should be designed to meet the following requirements: (R1) It should capture different granularities of semantics between the query and document. For example, the input queries of different KILT tasks range from sentence-level [12, 20, 24, 45, 48] to paragraph-level [10, 17, 18]. (R2) It should be flexible to predict dynamic numbers of relevant documents for different queries. For example, the question answering usually requires multiple documents to support the answer, while the slot filling task [11, 27] usually needs one. (R3) It should capture inter-document semantic relation since multiple documents may share similar characteristics with respect to a query.

In light of the above requirements, we carefully devise three pre-training tasks to capture the query-document relevance in different views, namely Inner Sentence Selection (ISS), Lead Paragraph Selection (LPS), and Hyperlink Identifier Prediction (HIP). The key idea of ISS and LPS is to sample sentences or paragraphs from documents and adopt them as pseudo queries, while that of HIP is to utilize hyperlinks and anchor texts to approximate the relationship between two documents. Based on the three proposed tasks, we can build large-scale pseudo pairs of query and document identifiers without additional human supervision. Then, initialized with BART [28], we pre-train our model via a standard seq2seq objective, i.e., maximizing the likelihood of the output sequence with teacher forcing [43]. As shown in Figure 1, once such a strong generative retrieval model is learned, we expect it can be seamlessly adapted to a wide variety of downstream KILT tasks without the need of additional index.

We pre-train CorpusBrain on English Wikipedia, which contains tens of millions of well-formed Wiki articles. We fine-tune CorpusBrain on the comprehensive KILT benchmark [38] which consists of eleven datasets spanning five distinct KILT tasks. Empirical experimental results demonstrated that CorpusBrain can achieve significant improvements over strong baseline solutions for the retrieval task and push forward the SOTA performance on a number of downstream tasks. We simulate both zero- or low-resource settings and show that CorpusBrain works well even when they were fine-tuned with very little supervision.

2 RELATED WORK

In this section, we briefly review three lines of the related works, including traditional pipeline IR framework, model-based IR approaches and knowledge-intensive language tasks.

2.1 Traditional Pipeline IR Framework

Most existing IR methods follow a common thee-step pipeline framework, i.e., "index-retrieve-then-rank". For the indexing and retrieval stage, existing approaches can be divided into two categories [15] from the view of representation type and index mode, including sparse retrieval and dense retrieval models. For sparse retrieval, they generally build the inverted index based on the corpus, which encode term-based features like term frequencies and term position. The typical methods in this category are TF-IDF and BM25 [40]. Besides, several works [13, 51] employed word embedding to enhance the semantic matching. With the development of pre-training techniques, researchers explore to utilize pre-trained models to estimate term weights. For example, DeepCT [7] used

BERT [22] to obtain term weights for the inverted index. For dense retrieval, they usually project documents into dense representations to build index and turn to approximate nearest neighbor search algorithms for fast retrieval. Karpukhin et al. [21] demonstrated dense retrieval models could outperform BM25 by utilizing in-batch negatives. Recently, various fine-tuning techniques [19, 23, 32, 47, 49, 49] are explored to enhance dense retrieval.

For the ranking stage, many different ranking models have been proposed, including vector space models [42], probabilistic models [41], learning to rank models [3, 30, 31] and neural ranking models [8, 16]. Recently, researcher have shown that developed pre-training objectives tailed for IR could further enhance the performance on downstream ranking tasks [4, 25, 33–35]. For example, Lee et al. [25] introduced Inverse Cloze Task (ICT) for passage retrieval by randomly sampling a sentence from passage as pseudo query and taking the rest sentences as the document. Ma et al. [33] presented Representative Words Prediction (ROP) task for pre-training, which sampled representation words from the document according to a unigram document language model. Ma et al. [35] proposed HARP by leveraging the large-scale hyperlinks and anchor texts to pre-train the language model for ad-hoc retrieval. Despite their success, however, such pipeline framework has has drawbacks on the end-to-end optimization and memory resources.

2.2 Model-based IR Approaches

To replace the long-lived "index-retrieve-then-rank" paradigm, the model-based IR paradigm is proposed to collapse the indexing, retrieval, and ranking components of traditional IR systems into a single consolidate model [37]. There have been some preliminary explorations in model-based IR [2, 6, 9, 44, 52] over the past year. For example, De Cao et al. [9] presented to generate entity names in an autoregressive fashion for entity linking task. Chen et al. [6] proposed GERE for fact checking task, which generates the document titles as well as evidence sentence identifiers. Tay et al. [44] explored different ways to obtain document identifiers, including unstructured atomic identifiers and semantically structured identifiers. However, these approaches are generally designed for specific task, resulting low flexibility for other tasks. In addition, a large number of annotated data is needed to learn a well behaved model. In this work, we propose to pre-train a general-purpose generative model which can serve a wide range of downstream KILT tasks.

2.3 Knowledge-Intensive Language Tasks

Knowledge-intensive language tasks (KILT) require access to large and external knowledge sources. For example, fact checking requires to find trustworthy evidences to determine the veracity of a claim [45]. Open-domain question answering needs to reason over a knowledge source to produce the correct answer for a question [12, 20, 24, 48]. Practical solutions to these tasks usually apply a two-step pipeline framework [5, 45]. First, an efficient retrieval component is employed to retrieve relevant information from a large knowledge source. Then, dedicated downstream models are adopted to produce the final results by capturing the relationship between the input and the context of the retrieved information.

Up to now, numerous datasets for KILT tasks [10, 24, 45, 48] have been proposed to facilitate the research. Generally, these datasets have different formats, and are processed or evaluated with different assumptions. Besides, their knowledge sources vary from different versions of Wikipedia to entirely different corpora. To facilitate the task-to-task comparisons, a benchmark named KILT [38] was introduced, which consists of eleven datasets spanning five distinct tasks (i.e., fact checking, open domain question answering, slot filling, entity linking and dialogue). Crucially, all tasks in KILT are formulated into a common interface and grounded in the same snapshot of Wikipedia.

3 OUR APPROACH

In this section, we present the novel pre-trained generative retrieval model CorpusBrain in detail. We first introduce our motivation on the design of our method. We then describe the model architecture as well as the pre-training tasks.

3.1 Motivation

KILT tasks are usually approached by combining a search component with a reader component [38]. Recently, thanks to the popularity of pre-trained models and their strong natural language understanding capabilities, the large-scale pre-trained generative models have been the de-facto implementation of the reader component. Unfortunately, these tremendous advances in pre-trained generative models has yet to bring similar transformational changes in how the search component is approached. Most existing solutions in the search component follow the traditional "index-retrieve-then-rank" paradigm. To fully parameterize the traditional pipeline framework, Metzler et al. [37] outlined a high-level vision of the next generation of IR systems, called model-based IR, to replace the long-lived pipeline framework with a single consolidated model. In this way, we can not only optimize the entire model directly in an end-to-end way, but also consume largely reduced memory resources and computation cost.

Motivated by this blueprint, there have been some preliminary works [6, 9, 44, 52] dedicated to mapping a query to a document identifier for the search component. Typically, these solutions directly fine-tune the off-the-shelf pre-trained generative models (e.g., BART) on specific downstream KILT tasks. However, they are designed to suit the need of specific tasks. It may not be practical to deploy separate specialised models for each application due to considerable memory resources or computation overhead. Besides, a large amount of application-specific annotated data is required to learn the model, which is unsuitable for low data regimes, e.g., zero and few-shot settings.

Therefore, in this work, we propose to pre-train a general-purpose generative retrieval model, named *CorpusBrain*, by encoding all information about the corpus in the parameters. Namely, we target a strong generative retrieval model learned with several pre-training tasks, that can be used for a diverse range of KILT tasks without the need of additional index. Specifically, we carefully design three self-supervised pre-training objectives to satisfy the following three properties: (R1) The semantic granularities between the query and document vary greatly in different tasks; (R2) Various tasks usually require different numbers of retrieved supporting documents; (R3) Different documents could share similar characteristics with the queries. Then, we pre-train an encoder-decoder architecture to generate document identifiers via a standard seq2seq objective.

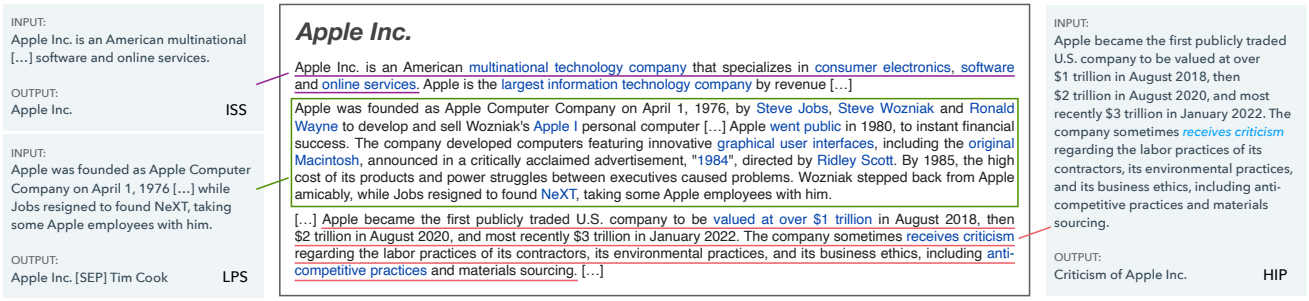

Figure 2: An illustrative example of the three pre-training tasks where each query is highlighted in different colors. Given a document p_i, the input of the ISS is a randomly sampled sentence from p_i, while the input of the LPS is a paragraph from the first few paragraphs in p_i. The output of both the ISS and LPS is the document identifier of p_i and destination pages linked by o anchor texts sampled from a_i. For the HIP, the input is the contextual information of an anchor a_i^k while the output is the document identifier of the destination page.

3.2 Model Architecture

To tackle the generative retrieval problem, we leverage a Transformer-based encoder-decoder architecture, which contains the following two dependent components: (1) Query Encoder, a bidirectional encoder to achieve the query representation; (2) Identifier Decoder, a sequential generation process to yield document identifiers.

3.2.1 Query Encoder. The query encoder is to map the input query $q = \{w_1, w_2, \ldots, w_{|q|}\}$ into a compact vector that can capture its essential topics. Specifically, the encoder represents the query q as a series of hidden vectors, i.e.,

$$H_q = \text{Encoder}(w_1, w_2, \ldots, w_{|q|}),$$

where H_q denotes the query representation.

3.2.2 Identifier Decoder. The decoder is to generate a sequence of document identifiers of the relevant documents for each query. Note the document identifier can be defined in different ways, such as the unique title and url of the document. Specifically, the probability of generating the n-th token $w_{m,n}$ in the m-th document identifier t_m is defined as,

$$p(w_{m,n}|w_{\leq m,<n}, q) = \text{Decoder}(w_{\leq m,<n}, H_q).$$

At the inference time, we apply the constrained beam search strategy [9] to limit each generated identifier to be in the valid pre-defined candidate set, i.e., the identifiers of all the documents in the given corpus. Concretely, we define our constrain in terms of a prefix tree where nodes are annotated with tokens from the predefined candidate set. For each node in the tree, its children indicate all the valid continued token list from the prefix defined traversing the tree from the root to it.

3.3 Pre-training Tasks

Formally, suppose $C = \{p_0, p_1, \ldots\}$ denotes a large-scale corpus where p_i denotes an individual document. Each document $p_i = \{t_i, s_i, a_i\}$ consists of a unique document identifier t_i, a sequence of sentences s_i, and an anchor set a_i. Specifically, $s_i = \{s_i^0, s_i^1, \ldots, s_i^n\}$ contains n sentences where s_i^j denotes the j-th sentence in p_i. $a_i = \{a_i^0, a_i^1, \ldots, a_i^m\}$ contains m anchors where a_i^k is the k-th anchor in p_i. $t_{a_i} = \{t_{a_i}^0, t_{a_i}^1, \ldots, t_{a_i}^m\}$ are the document identifiers of destination pages linked by each anchor text in a_i, where $t_{a_i}^l$ is linked by a_i^l.

What pre-training tasks are useful for improving Transformer-based encoder-decoder models is a crucial step in large-scale generative retrieval. It is generally hypothesized that using a pre-training task that more closely resembles the downstream task contributes to better fine-tuning performance [50]. To satisfy three requirements discussed above, we first assume that the pre-training data is defined as positive pairs of query and document identifier. Specifically, the document identifier can be defined in different ways, such as the unique title or url of the document. Then, as shown in Figure 2, we carefully design three pre-training tasks, i.e., Inner Sentence Selection (ISS), Lead Paragraph Selection (LPS), and Hyperlink Identifier Prediction (HIP). These three tasks target to learn the query-document relevance in different views, to generate pseudo pairs of query and document identifier to simulate the downstream retrieval task. The training data for these tasks can be freely obtained from the given corpus without an additional manual labeling process. In the following, we will present the proposed three pre-training tasks in detail.

3.3.1 Inner Sentence Selection (ISS). Given a document p_i, to satisfy the requirement (R1) where the query for many KILT tasks (e.g., fact checking and question answering) is in the sentence level, the ISS treats the inner sentence s_i^j randomly drawn from p_i as the pseudo query q. This task contributes to capturing the semantic context of a sentence. To satisfy the requirement (R2), the ISS treats the p_i and destination pages linked by $o(o < m)$ anchor texts $\{a_i^{s_1}, a_i^{s_2}, \ldots, a_i^{s_o}\}$ randomly sampled from a_i as the relevant documents. ISS uses the concatenated document identifiers as the generation target $[t_i, t_{a_i}^{s_1}, t_{a_i}^{s_2}, \ldots, t_{a_i}^{s_o}]$. In this way, it is feasible to achieve dynamic predictions of relevant documents for different downstream tasks.

3.3.2 Lead Paragraph Selection (LPS). Different from ISS, LPS samples a paragraph from the document p_i and adopts it as the pseudo query q. With such pre-training task, we can fit the requirement (R1) where the query of entity linking, dialog and other KILT tasks is in the paragraph level. Generally, the top paragraphs can be a surrogate summary of better quality than the remains. Inspired by this, we regard the leading l paragraphs as the pseudo queries. To satisfy the requirement (R2), the output $[t_i, t_{a_i}^{s_1}, t_{a_i}^{s_2}, \ldots, t_{a_i}^{s_o}]$ of the LPS is consistent with that of the ISS.

Table 1: Retrieval datasets statistics of the KILT benchmark. #P denotes the average number of relevant Wikipedia pages for each query in the train, dev and test set. '-' denotes that the task does not provide a ground-truth document in the training set.

Dataset	Task	#P	Train Size	Dev Size	Test Size
FEV [45]	Fact Checking	1.13	104,966	10,444	10,100
AY2 [18]	Entity linking	1	18,395	4,784	4,463
WnWi [17]	Entity Linking	1	-	3,396	3,376
WnCw [17]	Entity Linking	1	-	5,599	5,543
T-REx [11]	Slot Filling	1.26	2,284,168	5,000	5,000
zsRE [27]	Slot Filling	1	147,909	3,724	4,966
NQ [24]	Open Domain QA	1.57	87,372	2,837	1,444
HoPo [48]	Open Domain QA	2	88,869	5,600	5,569
TQA [20]	Open Domain QA	1.68	61,844	5,359	6,586
ELI5 [12]	Open Domain QA	1.18	-	1,507	600
WoW [10]	Dialogue	1	63,734	3,054	2,944

3.3.3 Hyperlink Identifier Prediction (HIP). Based on the classical anchor intuition [35], the hyperlink information could indicate the inter-document semantic relation to some extent. To satisfy the requirement (R3), we first view the anchor text as a pseudo query. Unfortunately, the anchor texts are usually too short to carry enough semantics. Therefore, we resort to the anchor context, i.e., the surrounding contextual information in the anchor's corresponding sentence. Specifically, given a page p_i, we randomly select an anchor a_i^k from the anchor set a_i and locate the sentence s_q containing a_i^k. Based on s_q, we get the anchor context by looking at its previous and successive sentence, i.e., $[s_{q-1}, s_q, s_{q+1}]$ as the pseudo query q. Then, the generation target is the document identifier $t_{a_i}^k$ of the destination page linked by the anchor a_i^k.

3.3.4 Learning Objective. Based on the three pre-training tasks, we can build a large number of pseudo pairs of query and document identifiers. All the tasks are formulated by a standard seq2seq objective, i.e., cross entropy loss, for the pre-training,

$$\mathcal{L} = \arg\max_{\theta} \sum_{\mathcal{D}} \sum_{m} \sum_{n} \log p(w_{m,n} | w_{\leq m, <n}, q; \theta),$$

where θ denotes the model parameters and $\mathcal{D}$ denotes the pre-training dataset. All parameters are optimized by the loss $\mathcal{L}$, and the whole model is trained in an end-to-end fashion.

4 EXPERIMENTAL SETTINGS

In this section, we introduce our experimental settings.

4.1 Datasets

We first introduce the large text corpora for pre-training and eleven downstream KILT datasets.

4.1.1 Pre-training Corpus. We use the English Wikipedia as the pre-training corpus, since (1) Wikipedia is publicly available and easy to collect; and (2) A large collection of documents could well support our pre-training method. English Wikipedia contains tens of millions of documents which has been widely used in many pre-training methods. Following [9], we use 2019/08/01 Wikipedia dump pre-processed by Petroni et al. [38].

4.1.2 Downstream Tasks. To verify the effectiveness of our method, we conduct experiments on the KILT benchmark [38] with eleven datasets spanning five distinct knowledge-intensive language tasks. In this work, we consider the retrieval task on the KILT benchmark, in which the model should provide a set of Wikipedia pages as evidences for final prediction with respect to the input query. Table 1 shows the overall statistics of the retrieval tasks in KILT.

4.2 Baselines

We adopt two types of baseline methods for comparison, including traditional IR models and model-based IR models.

4.2.1 Traditional IR Models. We take several representative models that are widely used for KILT tasks as the baselines, including the sparse retrieval and dense retrieval methods.

- **BM25** [40] is a highly effective retrieval model that represents the classical probabilistic retrieval model.
- **TF-IDF** [5] is a traditional sparse vector space retrieval model that combines bigram hashing and TF-IDF matching to return relevant documents.
- **DPR** [21] is a BERT-based dual-encoder model trained with in-batch negatives and a few hard negatives selected with BM25.
- **DPR+BERT** [38] combines a BERT-base classifier with passages returned from DPR where the query and retrieved passages are the input.
- **DPR+BART** [38] incorporates an explicit retrieval step in addition to the generative pre-training together with DPR and BART.
- **RAG** [29] combines pre-trained parametric and non-parametric memory for generation.
- **MT-DPR** [36] jointly trains a DPR model on an extensive selection of retrieval tasks.
- **BLINK+flair** [38] combines BLINK [46] and flair [1] retrieval solution that ranks pages according to entities in the input.

4.2.2 Generative Retrieval Models. Furthermore, we consider several advanced generative retrieval baselines.

- **BART** [28] is a denoising autoencoder built with a Seq2Seq model that is applicable to sequence generation tasks. Following [6, 9], we extract the query-title pairs from each downstream task and directly fine-tune the BART for generative retrieval. Specifically, we denote BART fine-tuned under three settings, i.e., zero-shot, specific-task fine-tuning and multi-task fine-tuning, as BART_{zs}, BART_{ft} and BART_{mt}, respectively. Note we report the official performance of BART_{ft} on KILT test data.
- **T5** [39] is a pre-trained encoder-decoder model on a multi-task mixture of unsupervised and supervised tasks. We report the official performance of fine-tuned T5 (T5_{ft}) on test and dev data.
- **SEAL** [2] combines an autoregressive language model with a compressed full-text substring index. SEAL fine-tunes BART via multi-task training on all the datasets in the KILT benchmark.
- **GENRE** [9] retrieves entities by generating their unique names. GENRE takes advantage of fine-tuning BART via multi-task training on the supervised BLINK data (i.e., 9M unique triples document-mention-entity from Wikipedia) and all the data in the KILT benchmark.

4.3 Evaluation Metrics

To measure the retrieval performance on the downstream KILT dataset, we use R-precision (%) as the evaluation metric, which is suggested in the official instructions and widely used in previous

works on KILT [2, 9, 29, 36, 38]. R-precision is calculated as $\frac{r}{R}$, where R is the number of Wikipedia pages inside each provenance set and r is the number of relevant pages among the top-R retrieved pages. Besides, we also report the average performance of all the eleven datasets. For the zero-shot and full fine-tuning setting, we report the performance results on both the test and dev sets. For the zero- and low-resource settings, we report the performance results on the dev sets since the KILT leaderboard[1] limits the frequency of the submission for test performance.

4.4 Implementation Details

In this section, we describe the implementation details of CorpusBrain[2], including model architecture, pre-training process and fine-tuning process.

4.4.1 Model Architecture. We use the Transformer-based encoder-decoder architecture similar to BART_{large} version, where the number of Transformer layers is 12, the hidden size is 1024, the feed-forward layer size is 4096 and the number of self-attention heads is 16, for both the encoder and decoder. The total parameters is 406M. For a fair comparison, we leverage the same architecture in the experiments for our CorpusBrain and the baseline BART. Besides, we use sequence modeling toolkit fairseq[3] for the implementation of CorpusBrain.

4.4.2 Pre-training Process. In this work, we leverage the Wikipedia article title as the document identifier and leave other identifiers (e.g., url and hashcode) in the future work. Given an article, for the ISS task, we first select the lead 2 sentences, and then randomly sample up to 10 from the rest sentences set as the pseudo queries (i.e., 12 sentences in total). For the LPS task, we select the top 3 paragraphs (i.e., l=3) as the pseudo queries. For the output of ISS and RPS, we use the current article title joint with titles of destination Wikipedia pages linked by o anchors. Specifically, o is in [0, 1, 2, 3, 4] with the probability of [70%, 20%, 5%, 3%, 2%], respectively. For the HIP task, we randomly select 3 anchors from the anchor set, and then denote the anchor context as input. The output is the title of the destination Wikipedia page. In total, for each article, we construct 18 (i.e., 12+3+3) pseudo pairs.

Considering the large cost of training from scratch, we initialize the parameters of the encoder-decoder architecture from the official BART's checkpoint. We use a learning rate of $3e^{-5}$ and Adam optimizer with the warmup technique, where the learning rate increases over the first 10% of batches, and then decays linearly to zero. The label smoothing is 0.1, the weight decay is 0.01, and the gradient norm clipping is 0.1. We train in batches of 8192 tokens on two NVIDIA Tesla V100 32GB GPUs.

4.4.3 Fine-tuning Process. For the retrieval tasks in the downstream KILT benchmark, we first process the original data into a Seq2Seq pair format. Specifically, the original input (e.g., claim, text chunk and question) remains unchanged. For the output, we use the `[SEP]` token to concatenate the titles of several ground-truth relevant pages. Then, we fine-tune CorpusBrain with the processed data, with the learning rate as $3e^{-5}$. For each task, We train in batches of 4096 tokens on one NVIDIA Tesla V100 32GB GPUs. At the inference time, we use constrained beam search with 10 beams, and maximum decoding steps of 15. For the entity linking sub-task, we restrict the input sequence to be at most 384 tokens cutting the left, right, or both parts of the context around a mention. We normalize the log-probabilities by sequence length.

In this work, we fine-tune CorpusBrain via three different strategies: (1) We fine-tune CorpusBrain on specific downstream KILT task, denoted as CorpusBrain_{ft}. (2) We fine-tune CorpusBrain via multi-task training on all KILT retrieval data following [2, 9, 36], denoted as CorpusBrain_{mt}. Note that not all dataset available in KILT have a training set as shown in Table 1. To address all tasks, multi-task training has been a common approach for the KILT benchmark. (3) We follow the fine-tuning setting used in GENRE [9] to train CorpusBrain on BLINK [46] and all KILT data simultaneously, denoted as $\text{CorpusBrain}_{mt+BLINK}$.

5 EXPERIMENTAL RESULTS

Our experiments mainly target the following research questions:

- **RQ1:** How does CorpusBrain perform compared with strong retrieval baselines across both the unsupervised and supervised evaluations?
- **RQ2:** What is the performance difference of CorpusBrain under specific-task fine-tuning and multi-task fine-tuning?
- **RQ3:** How do the three pre-training tasks of CorpusBrain affect the retrieval performance?
- **RQ4:** How does CorpusBrain perform under the low-resource setting?
- **RQ5:** How does CorpusBrain perform compared with traditional baselines in terms of memory footprint and inference time?
- **RQ6:** Can we better understand how different models perform via some case studies?

5.1 Baseline Comparison

To answer **RQ1**, we compare CorpusBrain with various strong baselines on the KILT benchmark. Table 2 and Table 3 show the R-precision performance on the test and dev set, respectively. We can observe that: (1) Among the traditional IR models, the TF-IDF model performs pretty well on the test set and even outperforms the strong dense retrieval method DPR with enough supervised data. (2) The generative retrieval models can outperform traditional IR models significantly across all the datasets, indicating the effectiveness of integrating all the components in traditional pipelines into a single unified model. (3) Our CorpusBrain_{zs} under zero-shot setting already achieves comparable results to most traditional baselines via full fine-tuning. For example, the R-precision of CorpusBrain_{zs} and DPR is 94.84% and 28.9%, respectively, on the zsRE dataset.

When we look at the generative retrieval baselines, we find that: (1) GENRE performs the best among these baselines in terms of most datasets, which benefits from multi-task training on supervised BLINK and entire KILT dataset at the same time using 128 GPUs. (2) CorpusBrain_{ft} only fine-tuned on specific tasks can obtain comparable results over GENRE, which could better serve the practical use of search systems. $\text{CorpusBrain}_{mt+BLINK}$ outperforms GENRE on all the dev sets significantly (p-value < 0.05) and 10 out of 11 test sets. These results suggest that the pre-training tasks do help

[1] https://eval.ai/challenge/689/leaderboard
[2] The code can be found at https://github.com/ict-bigdatalab/CorpusBrain
[3] https://github.com/pytorch/fairseq

Table 2: R-precision (%) for the page-level retrieval task on the KILT test data. Bold and underline indicates the best and second model respectively. The results are reported on the KILT leaderboard.

Model Type	Model	FC FEV	Entity Linking AY2	WnWi	WnCw	Slot Filling T-REx	zsRE	Open Domain QA NQ	HoPo	TQA	ELI5	Dial. WoW	Avg.
						Zero-shot							
Unspervised IR Models	TF-IDF [38]	50.9	3.7	0.24	2.1	44.7	60.8	28.1	34.1	**46.4**	**13.7**	**49.0**	30.3
	CorpusBrain$_{zs}$	**70.38**	**5.47**	**0.56**	**7.78**	**69.48**	**94.84**	**28.25**	**44.84**	42.76	12.17	29.64	**36.92**
						Full fine-tuning							
Traditional IR Models	DPR [38]	55.3	1.8	0.3	0.5	13.3	28.9	54.3	25	44.5	10.7	25.5	23.5
	DPR + BERT [38]	72.9	-	-	-	-	40.1	60.7	25	43.4	-	-	-
	DPR + BART [38]	55.3	75.5	45.2	46.9	13.3	28.9	54.3	25.0	44.4	10.7	25.4	38.6
	RAG [38]	61.9	72.6	48.1	47.6	28.7	53.7	59.5	30.6	48.7	11.0	57.8	47.3
	BLINK + flair [38]	63.7	81.5	80.2	68.8	59.6	78.8	24.5	46.1	65.6	9.3	38.2	56.0
	MT-DPR [36]	74.5	26.5	4.9	1.9	69.5	80.9	59.4	42.9	61.5	15.5	41.1	43.5
Model-based IR Models	T5$_{ft}$ [38]	-	74.0	47.1	49.3	-	-	-	-	-	-	-	-
	BART$_{ft}$ [38]	-	77.6	45.9	49.2	-	-	-	-	-	-	-	-
	SEAL [2]	81.4	-	-	-	62.1	91.6	**63.2**	**58.8**	68.4	-	57.5	-
	GENRE [9]	83.64	89.85	87.44	**71.22**	79.42	95.81	60.25	51.27	69.16	15.83	62.88	69.71
Our Approach	CorpusBrain$_{mt+BLINK}$	**84.07**	**89.98**	**88.12**	70.58	**79.98**	**98.27**	60.32	51.80	**70.19**	**17.50**	**64.79**	**70.51**

Table 3: R-precision (%) for the page-level retrieval task on the KILT dev data. Bold and underline indicates the best and second model respectively.

Model Type	Model	FC FEV	Entity Linking AY2	WnWi	WnCw	Slot Filling T-REx	zsRE	Open Domain QA NQ	HoPo	TQA	ELI5	Dial. WoW	Avg.
						Zero-shot							
Unsupervised IR Models	BM25 [36]	50.13	3.47	-	-	58.6	66.43	25.83	43.95	29.44	-	**27.5**	-
	BART$_{zs}$	10.2	0.04	0.03	0.09	6.82	6.98	0.14	1.18	1.10	0.20	1.38	2.56
	CorpusBrain$_{zs}$	**71.69**	**5.43**	**0.53**	**6.05**	**69.44**	**90.95**	**27.88**	**45.07**	**43.40**	**9.56**	26.23	**36.02**
						Full fine-tuning							
Traditional IR Models	DPR + BART [38]	55.46	-	44.96	45.70	13.62	45.60	54.25	24.62	45.36	10.32	-	-
	RAG [38]	63.50	77.40	49.00	46.70	29.26	65.36	60.31	30.76	49.26	10.39	46.66	48.05
	MT-DPR [36]	74.72	83.78	-	-	69.18	77.23	61.51	44.21	61.95	-	39.70	-
Model-based IR Models	T5$_{ft}$ [38]	-	86.62	47.35	46.58	-	-	-	-	-	-	-	-
	BART$_{ft}$	80.03	87.98	-	-	74.46	93.91	50.96	39.21	66.13	-	50.75	-
	BART$_{mt}$	81.92	89.17	67.58	62.33	75.18	91.08	58.62	48.69	67.64	12.08	50.98	64.12
	GENRE [9]	84.68	92.75	87.69	70.57	79.68	94.84	64.26	51.82	71.11	13.47	56.32	69.74
Our Approach	CorpusBrain$_{ft}$	81.77	90.36	-	-	76.90	**98.49**	57.67	50.62	69.25	-	53.60	-
	CorpusBrain$_{mt}$	82.06	90.84	72.26	66.23	77.62	98.26	59.10	50.07	68.78	12.88	53.75	66.53
	CorpusBrain$_{mt+BLINK}$	**85.03**	**92.86**	**88.64**	**71.35**	**80.22**	**98.49**	**64.61**	**52.23**	**71.71**	**14.33**	**59.72**	**70.84**

obtain a better understanding of the corpus for document retrieval. (3) CorpusBrain$_{mt+BLINK}$ is able to achieve the best performance among all the baselines for all 11 dev sets, while for 8 of 11 test sets and the second place for other 3 test sets. Compared with baselines which are applicable to all 11 tasks, CorpusBrain$_{mt+BLINK}$ performs the best on both the dev and test set.

Finally, the observations from the KILT leaderboard[4] with 11 KILT tasks are as follows: (1) As a universal retrieval model, our CorpusBrain$_{mt+BLINK}$ wins the 1st place for 3 tasks (AY2, WnWi and WoW), the 2nd place for 3 tasks (WnCw, zsRE and ELI5), and the 3rd place for 4 tasks (FEV, T-REx, HoPo and TQA). These results indicate the good generalization ability of our method, which is able to memorize the knowledge about the corpus. (2) For the leaderboard top methods, generally speaking, they are specially optimized for each task, or incorporate extra information to enhance the retrieval performance. For example, Re2G leverages the information in the reader component to train a separate model for each task, while the top-1 model TABi [26] for T-REx leverages the knowledge graph, which is quite effective for the entity-related task. We refer readers to the leaderboard for the performance of these methods. (3) CorpusBrain under-performs others especially for QA datasets (e.g., NQ, HoPo, and TQA). The reason might be that the pseudo queries in our pre-training tasks are mainly declarative sentences, which are quite different from the questions in QA. For ELI5, since there is no training data, all leaderboard methods perform poorly but our CorpusBrain still achieves the 2nd. We will investigate to further enhance the generalization of our method.

[4]We report the KILT leaderboard results on August 16, 2022.

5.2 Impact of Fine-tuning Strategies

To answer **RQ2**, we further analyze the effect of different fine-tuning strategies, i.e., specific-task fine-tuning and multi-task fine-tuning, on the KILT dev set. We conduct a comparison of the retrieval performance between CorpusBrain and BART.

As seen in Table 3, we can see that: (1) CorpusBrain under multi-task fine-tuning achieves better results than under specific-task

Table 4: The zero-shot R-precision (%) of CorpusBrain with different pre-training tasks. Best results are marked bold.

Pre-training task	FC	Entity Linking			Slot Filling		Open Domain QA				Dial.	
	FEV	AY2	WnWi	WnCW	T-REx	zsRE	NQ	HoPo	TQA	ELI5	WoW	Avg.
ISS+LPS+HIP	**71.69**	5.43	0.53	6.05	**69.44**	**90.95**	**27.88**	**45.07**	**43.4**	**9.56**	**26.23**	**36.02**
ISS	64.89	4.91	0.65	5.70	58.8	79.27	22.91	44.09	42.10	7.70	27.14	32.56
LPS	61.5	4.39	0.47	5.88	53.7	69.44	23.48	43.12	35.45	7.10	25.61	30.01
HIP	58.19	**5.77**	**0.85**	**6.14**	51.1	68.34	21.75	42.21	32.79	7.10	18.30	28.41

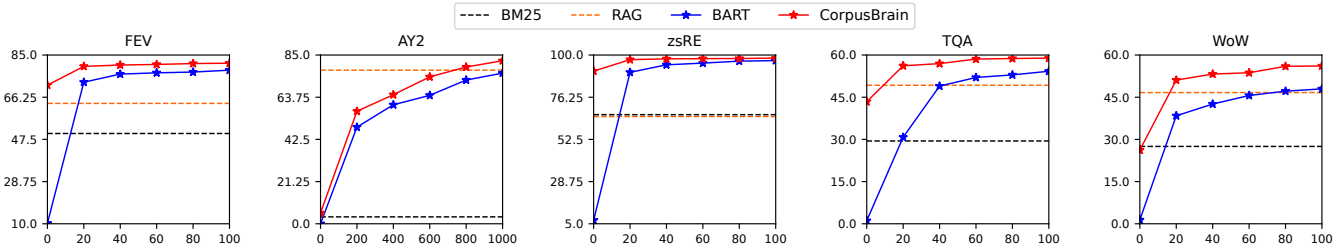

Figure 3: Fine-tuning with limited supervised data. The red and blue solid lines denote CorpusBrain and BART, respectively. The orange and gray dashed lines are RAG fine-tuned using the full supervised data and unsupervised BM25, respectively. We report the R-precision (%) results on the dev set.

fine-tuning. This reason may be that all tasks share a common objective to generate document identifiers and boost each other in the training process. (2) Under both the specific-task and multi-task fine-tuning, CorpusBrain outperforms BART over all the KILT tasks. This indicates that the designed three objective in CorpusBrain which resembles the relevance relationship between the query and document identifier could contribute to the retrieval task. (3) CorpusBrain converges faster than BART at both fine-tuning strategies, indicating that with the encoded corpus information, CorpusBrain could easily adapt to the downstream KILT tasks via the same learning objective as in the pre-training stage.

5.3 Impact of Pre-training Tasks

To answer **RQ3**, we conduct a thorough ablation study on different pre-training tasks in CorpusBrain. Specifically, we pre-train the encoder-decoder model with ISS, LPS and HTP, respectively and evaluate the retrieval performance under the zero-shot setting on the dev set. For fair comparison, we set the number of pseudo (query, document identifiers) pairs extracted from each Wikipedia article for each pre-training task as 18.

Table 4 shows the individual performance of three pre-training tasks. We can see that: (1) In general, ISS has the best performance, followed by LPS, and then HIP. One possible reason is that the input of 7 downstream KILT tasks is sentence-level, the (query, document identifier) pairs defined by the sampled sentences are suitable for these generative retrieval task. (2) HIP outperforms ISS and LPS on the entity linking tasks, i.e., AY2, WnWi and WnCw. This is because that the HTP task can capture the inter-page relation, which is similar to the entity linking formulation. Note CorpusBrain without fine-tuning performs poorly on entity linking under all the pre-training tasks. The reason is that we do not add any tokens (e.g., `[START_ENT]` and `[END_ENT]`) to denote the mention of entities at pre-training stage and thus CorpusBrain easily generates the same titles for different entites. (3) With all the pre-training tasks, CorpusBrain achieves the best performance. This results demonstrate the effectiveness of all the three pre-training tasks and the importance to measure different granularities of semantics between the query and document.

5.4 Zero- and Low-Resource Settings

In real-world practice, it is often time-consuming and difficult to collect a large number of relevance labels to train or fine-tune a retrieval model for various KILT tasks. To answer **RQ4**, we simulate the low-resource retrieval setting for five datasets, i.e., FEV, AY2, zsRE, TQA and WoW, spanning five varied classes of KILT tasks. Following [14], we randomly select different numbers of instances from the original training set for fine-tuning CorpusBrain. Specifically, we randomly pick 20, 40, 60, 80 and 100 instances for the FEV, zsRE, TQA and WoW dataset, and pick 200, 400, 600, 800 and 1000 instances for AY2. We fine-tune CorpusBrain and BART on each downstream dataset, with batches of 4096 tokens, learning rate as 3e-5, and pick the last checkpoint to evaluate the performance on the original dev set.

As shown in Figure 3, we can observe that: (1) CorpusBrain outperforms BART on all the five datasets by fine-tuning on the same limited supervised data, demonstrating that CorpusBrain is able to encode the relevance information about a given corpus through the pre-training. Furthermore, CorpusBrain achieves much better zero-shot performance than BART. (2) For the five datasets, CorpusBrain fine-tuned on limited supervised data can achieve competitive results with RAG fine-tuned on the full supervised datasets. For example, CorpusBrain fine-tuned with only 20 examples has outperformed RAG on TQA and WoW datasets. The results demonstrate that by fine-tuning with small numbers of supervised pairs, CorpusBrain is able to adapt to the target task quickly. (3) Under the zero resource setting, for example, CorpusBrain can outperform RAG significantly for the FEV (71.69% vs. 63.50%) and zsRS dataset (90.95% vs. 65.36%). (4) CorpusBrain also beats previous state-of-the-art baseline, i.e., GENRE, with limited fine-tuning examples. For the zsRE dataset, with just 100 examples, CorpusBrain could be fine-tuned to retrieval documents at comparable quality (i.e., R-precision = 98.28%) to GENRE (i.e., R-precision = 94.84%). It is worth noting that GENRE was trained on the full supervised 11

Table 5: An example from the ELI5 dev set. The query is "Are there any actual laws against false advertisement?", and the titles of ground-truth relevant articles are "False advertising" and "Media regulation". We show the top-5 beams returned by GENRE, BART_{mt} and $\text{CorpusBrain}_{mt+BLINK}$. Correct results are marked bold.

Beam	GENRE	BART_{mt}	CorpusBrain
1	False advertising	False advertising [SEP] U.S. Patent No. 1	**False advertising [SEP] Media regulation**
2	United States trademark law	Federal Trade Commission	False advertising
3	Anti-discrimination law	Misrepresentation [SEP] Ad serving	False advertising [SEP] Misrepresentation
4	Advertising to children	Misleading or deceptive conduct	False advertising [SEP] Legal liability
5	Anti-terrorism legislation	Federal Trade Commission	Advertising

Table 6: Comparisons on the memory footprint, the number of model parameters and inference time.

Model	Memory	Parameter	Time
DPR	70.9GB	220M	14.01ms
RAG	40.4GB	626M	9.86ms
CorpusBrain	**2.1GB**	**406M**	**5.32ms**

datasets in the KILT benchmark plus 9M BLINK data. These results further validate that the pre-training stage encodes all the information about the corpus into the model parameters and CorpusBrain does work like an expert with a knowledgeable brain.

5.5 Memory and Inference Efficiency

To answer **RQ5**, we compare CorpusBrain with two representative traditional IR models, i.e., DPR and RAG, in terms of the memory footprint and inference time. We compare the memory footprint (disk space) required by different models. Here, we evaluate the end-to-end inference time of the retrieval phase in the fact checking task (i.e., FEVER dataset).

As shown in Table 6, we can see that: (1) CorpusBrain requires the least memory footprint and model parameter regardless of the size of the corpus. For example, CorpusBrain occupied 34 times less memory than DPR and 20 times less memory than RAG. It is because that CorpusBrain uses its model parameters to store all information of the corpus while traditional IR methods store dense representations for the whole corpus increased along with the increase of corpus. (2) CorpusBrain has a significant reduction of the inference time of document retrieval. As we can see, dense retrieval models like DPR need to compute relevance over all the document dense representations, while the inference process of CorpusBrain is relatively quite simple, where the inference time is directly proportional to the beam size with a limited overhead by constrained decoding. These results show that CorpusBrain can be well deployed in resource-limited platforms due to the memory-efficiency and time-efficiency.

5.6 Case Study

To answer **RQ6**, we conduct some case studies to better understand how different models perform. We take one query from the ELI5 dev set (open-domain QA task) as an example, and show the generated Wikipedia article titles from our $\text{CorpusBrain}_{mt+BLINK}$ model as well as that from the strong baselines GENRE and BART_{mt}.

As shown in Table 5, we have the following observations: (1) GENRE only predicts one title at each beam, but can obtain multiple titles via a post-processing way, i.e., selecting a fixed number of top-ranked beams. In this case, although GENRE could successfully predict one relevant document "False advertising", it fails to predict another relevant document in other beams. The reason may be that using beams to return multiple documents can not well model the dependency information between documents. (2) BART_{mt} has the ability to model the document dependency for generating multiple titles. Nonetheless, without adequate pre-training tasks used for encoding the knowledge about the corpus, BART_{mt} may not be able to make totally correct ground-truth documents. (3) $\text{CorpusBrain}_{mt+BLINK}$ can generate right document titles. Note in the other beams, $\text{CorpusBrain}_{mt+BLINK}$ can still generate the right title (i.e., "false advertising") or potential right titles (e.g., "legal liability"). As we look at the "legal liability" article, we found that it is actually a very proper supporting article. These results again demonstrate the effectiveness of the proposed pre-training tasks.

6 CONCLUSION

In this paper, we have proposed *CorpusBrain*, a novel pre-trained generative retrieval model to encode all information about the corpus into its parameters. To train such a strong generative model, we delicately devised a set of pre-training tasks to emphasize different aspects of semantics between queries and documents. The key idea is to sample a context from one document as a pseudo query and generate the document identifiers of source or destination documents based on hyperlinks. CorpusBrain just needs to pre-train one model and could be then adapted to improve a diversity of downstream KILT tasks without the need of constructing additional index. Through experiments on the KILT benchmark in terms of the retrieval task, CorpusBrain achieved significant improvements over strong baseline approaches. We also showed that CorpusBrain can achieve strong performance under both the zero- and low-resource settings.

In future work, we would like to explore other document identifiers, e.g., page url and HashCode, and investigate new ways to further enhance the pre-training tailored for generative retrieval. Besides, it is worthwhile to go beyond the search component in the KILT tasks. We would try to test the ability of CorpusBrain over other types of downstream IR tasks, such as ad-hoc retrieval, passage retrieval in QA or response retrieval in dialog systems. Furthermore, it is valuable to design an end-to-end KILT system in a fully generative way.

ACKNOWLEDGMENTS

This work was funded by the National Natural Science Foundation of China (NSFC) under Grants No. 62006218 and 61902381, the Youth Innovation Promotion Association CAS under Grants No. 20144310, and 2021100, the Young Elite Scientist Sponsorship Program by CAST under Grants No. YESS20200121, and the Lenovo-CAS Joint Lab Youth Scientist Project.

REFERENCES

[1] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. 54–59.

[2] Michele Bevilacqua, Giuseppe Ottaviano, Patrick Lewis, Wen tau Yih, Sebastian Riedel, and Fabio Petroni. 2022. Autoregressive Search Engines: Generating Substrings as Document Identifiers. In *arXiv pre-print 2204.10628*. https://arxiv.org/abs/2204.10628

[3] Christopher Burges, Robert Ragno, and Quoc Le. 2006. Learning to rank with nonsmooth cost functions. *NIPS* 19 (2006).

[4] Wei-Cheng Chang, X Yu Felix, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. 2019. Pre-training Tasks for Embedding-based Large-scale Retrieval. In *ICLR*.

[5] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *55th Annual Meeting of the Association for Computational Linguistics, ACL 2017*. 1870–1879.

[6] Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Yixing Fan, and Xueqi Cheng. 2022. GERE: Generative Evidence Retrieval for Fact Verification. *arXiv preprint arXiv:2204.05511* (2022).

[7] Zhuyun Dai and Jamie Callan. 2020. Context-aware term weighting for first stage passage retrieval. In *SIGIR*. 1533–1536.

[8] Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *WSDM*. 126–134.

[9] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2020. Autoregressive Entity Retrieval. In *ICLR*.

[10] Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2018. Wizard of Wikipedia: Knowledge-Powered Conversational Agents. In *International Conference on Learning Representations*.

[11] Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. 2018. T-rex: A large scale alignment of natural language with knowledge base triples. In *LREC*.

[12] Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. ELI5: Long Form Question Answering. In *ACL*. Association for Computational Linguistics, Florence, Italy, 3558–3567. https://doi.org/10.18653/v1/P19-1346

[13] Jibril Frej, Philippe Mulhem, Didier Schwab, and Jean-Pierre Chevallet. 2020. Learning term discrimination. In *SIGIR*. 1993–1996.

[14] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *EMNLP*. 6894–6910.

[15] Jiafeng Guo, Yinqiong Cai, Yixing Fan, Fei Sun, Ruqing Zhang, and Xueqi Cheng. 2022. Semantic models for the first-stage retrieval: A comprehensive review. *TOIS* 40, 4 (2022), 1–42.

[16] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *CIKM*. 55–64.

[17] Zhaochen Guo and Denilson Barbosa. 2018. Robust named entity disambiguation with random walks. *Semantic Web* 9, 4 (2018), 459–479.

[18] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *EMNLP*. 782–792.

[19] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently teaching an effective dense retriever with balanced topic aware sampling. In *SIGIR*. 113–122.

[20] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *ACL*. Association for Computational Linguistics, Vancouver, Canada, 1601–1611.

[21] 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 *EMNLP*. 6769–6781.

[22] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*. 4171–4186.

[23] Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *SIGIR*. 39–48.

[24] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics* 7 (2019), 453–466.

[25] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent Retrieval for Weakly Supervised Open Domain Question Answering. In *ACL*. 6086–6096.

[26] Megan Leszczynski, Daniel Y Fu, Mayee F Chen, and Christopher Ré. 2022. TABi: Type-Aware Bi-Encoders for Open-Domain Entity Retrieval. *arXiv preprint arXiv:2204.08173* (2022).

[27] Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-Shot Relation Extraction via Reading Comprehension. In *CoNLL*. 333–342.

[28] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *ACL*. 7871–7880.

[29] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *NIPS* 33 (2020), 9459–9474.

[30] Hang Li. 2014. Learning to rank for information retrieval and natural language processing. *Synthesis lectures on human language technologies* 7, 3 (2014), 1–121.

[31] Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval* 3, 3 (2009), 225–331.

[32] Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, and Xueqi Cheng. 2022. Pre-train a Discriminative Text Encoder for Dense Retrieval via Contrastive Span Prediction. *arXiv preprint arXiv:2204.10641* (2022).

[33] Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, Xiang Ji, and Xueqi Cheng. 2021. Prop: Pre-training with representative words prediction for ad-hoc retrieval. In *WSDM*. 283–291.

[34] Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, Yingyan Li, and Xueqi Cheng. 2021. B-PROP: bootstrapped pre-training with representative words prediction for ad-hoc retrieval. In *SIGIR*. 1513–1522.

[35] Zhengyi Ma, Zhicheng Dou, Wei Xu, Xinyu Zhang, Hao Jiang, Zhao Cao, and Ji-Rong Wen. 2021. Pre-training for Ad-hoc Retrieval: Hyperlink is Also You Need. In *CIKM*. 1212–1221.

[36] Jean Maillard, Vladimir Karpukhin, Fabio Petroni, Wen-tau Yih, Barlas Oguz, Veselin Stoyanov, and Gargi Ghosh. 2021. Multi-Task Retrieval for Knowledge-Intensive Tasks. In *ACL*. 1098–1111.

[37] Donald Metzler, Yi Tay, Dara Bahri, and Marc Najork. 2021. Rethinking search: making domain experts out of dilettantes. In *ACM SIGIR Forum*, Vol. 55. ACM New York, NY, USA, 1–27.

[38] Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a Benchmark for Knowledge Intensive Language Tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Online, 2523–2544. https://doi.org/10.18653/v1/2021.naacl-main.200

[39] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21 (2020), 1–67.

[40] Stephen Robertson and Hugo Zaragoza. 2009. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc.

[41] Stephen E Robertson and K Sparck Jones. 1976. Relevance weighting of search terms. *Journal of the American Society for Information science* 27, 3 (1976), 129–146.

[42] Gerard Salton, Anita Wong, and Chung-Shu Yang. 1975. A vector space model for automatic indexing. *Commun. ACM* 18, 11 (1975), 613–620.

[43] Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *ICML*.

[44] Yi Tay, Vinh Q Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, et al. 2022. Transformer memory as a differentiable search index. *arXiv preprint arXiv:2202.06991* (2022).

[45] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a Large-scale Dataset for Fact Extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. 809–819.

[46] Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable Zero-shot Entity Linking with Dense Entity Retrieval. In *EMNLP*. 6397–6407.

[47] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In *ICLR*.

[48] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *EMNLP*. Association for Computational Linguistics, Brussels, Belgium, 2369–2380. https://doi.org/10.18653/v1/D18-1259

[49] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Optimizing dense retrieval model training with hard negatives. In *SIGIR*. 1503–1512.

[50] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*. PMLR, 11328–11339.

[51] Guoqing Zheng and Jamie Callan. 2015. Learning to reweight terms with distributed representations. In *SIGIR*. 575–584.

[52] Yujia Zhou, Jing Yao, Zhicheng Dou, Ledell Wu, and Ji-Rong Wen. 2022. DynamicRetriever: A Pre-training Model-based IR System with Neither Sparse nor Dense Index. *arXiv preprint arXiv:2203.00537* (2022).