A Unified Generative Retriever for Knowledge-Intensive Language Tasks via Prompt Learning

Jiangui Chen
Ruqing Zhang[*]
CAS Key Lab of Network Data Science and Technology, ICT, CAS
University of Chinese Academy of Sciences
Beijing, China
{chenjiangui18z,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

Maarten de Rijke
University of Amsterdam
Amsterdam, The Netherlands
m.derijke@uva.nl

Yiqun Liu
Department of Computer 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 (KILTs) benefit from retrieving high-quality relevant contexts from large external knowledge corpora. Learning task-specific retrievers that return relevant contexts at an appropriate level of semantic granularity, such as a document retriever, passage retriever, sentence retriever, and entity retriever, may help to achieve better performance on the end-to-end task. But a task-specific retriever usually has poor generalization ability to new domains and tasks, and it may be costly to deploy a variety of specialised retrievers in practice.

We propose a *unified generative retriever* (UGR) that combines task-specific effectiveness with robust performance over different retrieval tasks in KILTs. To achieve this goal, we make two major contributions: (i) To unify different retrieval tasks into a single generative form, we introduce an n-gram-based identifier for relevant contexts at different levels of granularity in KILTs. And (ii) to address different retrieval tasks with a single model, we employ a prompt learning strategy and investigate three methods to design prompt tokens for each task. In this way, the proposed UGR model can not only share common knowledge across tasks for better generalization, but also perform different retrieval tasks effectively by distinguishing task-specific characteristics.

We train UGR on a heterogeneous set of retrieval corpora with well-designed prompts in a supervised and multi-task fashion. Experimental results on the KILT benchmark demonstrate the effectiveness of UGR on in-domain datasets, out-of-domain datasets, and unseen tasks.[1]

[*]Research conducted when the author was at the University of Amsterdam.
[†]Jiafeng Guo is the corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGIR '23, July 23–27, 2023, Taipei, Taiwan
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9408-6/23/07…$15.00
https://doi.org/10.1145/3539618.3591631

CCS CONCEPTS

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

KEYWORDS

Knowledge-intensive language tasks, Generative retrieval, Unified retriever

ACM Reference Format:
Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Yiqun Liu, Yixing Fan, and Xueqi Cheng. 2023. A Unified Generative Retriever for Knowledge-Intensive Language Tasks via Prompt Learning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23), July 23–27, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3539618.3591631

1 INTRODUCTION

Knowledge-intensive language tasks (KILTs), such as fact checking [41] and slot filling [27], have gained much attention in recent years. They require that knowledge from large external corpora is surfaced [34]. Practical solutions to these tasks usually combine a search system with a machine reader [34]. Given an input query, the search component retrieves a limited subset of relevant contexts from a given corpus. Then, the reader component examines the retrieved information to perform the end task. High-quality relevant contexts retrieved by the search component are the foundation to support end tasks. Importantly, the granularity of relevant contexts required by KILTs varies from task to task, e.g., documents,

[1]The code can be found at https://github.com/ict-bigdatalab/UGR.

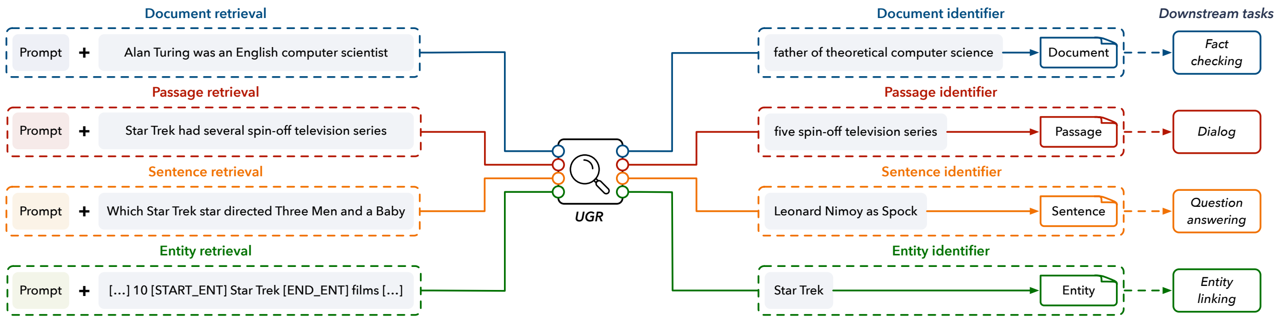

Document retrieval
Prompt + Alan Turing was an English computer scientist
Passage retrieval
Prompt + Star Trek had several spin-off television series
Sentence retrieval
Prompt + Which Star Trek star directed Three Men and a Baby
Entity retrieval
Prompt + [...] 10 [START_ENT] Star Trek [END_ENT] films [...]
UGR
Document identifier
father of theoretical computer science
Document
Passage identifier
five spin-off television series
Passage
Sentence identifier
Leonard Nimoy as Spock
Sentence
Entity identifier
Star Trek
Entity
Downstream tasks
Fact checking
Dialog
Question answering
Entity linking

Figure 1: Overview of the unified generative retriever (UGR), a Seq2Seq model that consumes queries and produces identifiers of relevant contexts. We design n-gram-based identifiers for relevant contexts at different granularities to unify different retrieval tasks. We employ a prompt learning strategy and design prompt tokens for each task to capture task specifications.

passages, sentences, and entities. For instance, for fact checking one needs to retrieve a set of documents [34] or passages [41] to verify the truthfulness of a claim. Open-domain question answering requires that we find a set of candidate sentences containing correct answer spans from the corpus [11, 22, 26, 46]. And for entity linking one needs to search the target entity from a knowledge source based on the given mention and its context [16, 19].

Retrieval tasks. Retrieval tasks that support KILTs can be grouped into four types: document retrieval, passage retrieval, sentence retrieval, and entity retrieval. Therefore, how to effectively perform different retrieval tasks at different levels of granularity becomes a critical challenge for KILT practitioners. Without loss of generality, previous efforts on search for KILTs come in two kinds: (i) The first one simply employs a single document retriever to support the downstream readers for all the KILTs, ignoring the different levels of granularity of relevant contexts each task actually needs [8, 32]. This method can share useful features or knowledge by training a universal document retriever across different KILTs. However, the returned documents are often too coarse-grained to support many KILTs [34]. (ii) The second one focuses on learning specific retrievers for different retrieval tasks to support KILTs. This includes document retrievers [29, 34], passage retrievers [2, 23], sentence retrievers [22, 26, 46], and entity retrievers [14, 44]. Although task specification contributes to improved retrieval performance, such retrievers often have poor generalization abilities to out-of-domain data and unseen tasks [32, 34]. A possible reason is that a task-specific retriever may overemphasize the specific knowledge and data bias in each task. Furthermore, deploying multiple task-specific retrievers over the same corpus, i.e., Wikipedia in KILTs, may be prohibitive in terms of memory or computational costs. *Can we develop a search system that has the merits of both types of approach while avoiding their disadvantages?*

A unified generative retriever. Building on the vision expressed in [33], we propose to train a single generative model to perform a variety of information retrieval (IR) tasks, a model that directly generates the identifiers of a set of relevant contexts given an input query. Such a generative retrieval model for KILTs can not only share common knowledge across different tasks, like the universal document retriever, but it also returns relevant contexts at an appropriate level of granularity for different tasks, like the task-specific retrievers. There have so far been some studies in this direction, but they only applied the idea of designing a generative IR system for a specific retrieval task [see, e.g., 2, 5, 8, 40].

In this work, we put the idea of a generative IR model into practice for KILTs. We introduce the *unified generative retriever* (UGR), a single retriever that can perform robustly across a variety of retrieval tasks in KILTs (see Figure 1). We need to solve two major challenges when building such a generative model: (i) How to unify different retrieval tasks that return relevant contexts at different levels of granularity, into a single generative form? And (ii) how to properly learn different retrieval tasks with a single model so that it can capture task specifications while obtaining the shared knowledge?

To solve the first challenge, we propose *n-gram-based identifiers* to unify different retrieval tasks that target different relevant contexts, i.e., documents, passages, sentences, and entities. The idea is to leverage the important n-grams in a context as its possible identifiers without the need for appropriate metadata and human annotation. We first concatenate the query and its relevant context together, and use BERT [24] to encode the concatenated text. Then, we directly sample important n-grams from an n-gram distribution as the identifier by summing up and re-normalizing the vanilla `[CLS]`-token attention weights over distinct n-grams. At inference time, we exploit an FM-Index [12] with constrained beam search [8] to ensure that each generated n-gram occurs at least once in the corpus. Moreover, we introduce an interactive scoring function on multiple generated n-grams to break ties among candidate contexts whose highest scoring n-gram is the same.

To solve the second challenge, we focus on training the retriever in a supervised and massively multi-task fashion. Specifically, motivated by prompt learning [30], we propose to plug a task-specific instruction, i.e., the prompt, into the query as the model input. Concretely, we carefully investigate three methods to design a prompt token for each retrieval task to keep specialized knowledge and alleviate the blurring-out problem [37]. Then, we train our model via a standard Seq2Seq objective [38], i.e., maximizing the likelihood of the output identifier with teacher forcing, over a training mixture consisting of the four retrieval tasks specified in well-designed prompts. This way, on the one hand, multi-task supervision steers our model to explore the common knowledge for better generalization instead of overemphasizing task-specific knowledge. On the other hand, the task-specific prompts help our model to perform a specific retrieval task.

Empirical findings. We conduct an empirical study on the comprehensive KILT benchmark [34] with eleven datasets that require retrieval at four levels of granularity. Our experimental results show that compared with prevailing baselines, UGR yields better retrieval performance on in-domain datasets, out-of-domain datasets and unseen tasks. In addition, the contexts retrieved by UGR contribute to new state-of-the-art downstream results on multiple datasets when paired with existing reader models.

2 RELATED WORK

Knowledge-intensive language tasks. Knowledge-intensive language tasks (KILTs) require access to extensive world knowledge due to their nature. For instance, a dialogue system needs to find the proper answer from a knowledge source for a given context [9]. Existing methods for KILT typically contain a search component and a reader component [4, 22, 26, 41, 46]. The search component retrieves relevant contexts from large knowledge sources, and the reader component produces final results by capturing the relationship between the input and the retrieved information. Various KILT datasets have been proposed, with different formats and assumptions [9, 26, 27, 41]. The knowledge sources they depend on range from different versions of Wikipedia to entirely different corpora. To enable comparisons on end-to-end tasks, a comprehensive benchmark, *knowledge-intensive language task* (KILT) [34], has been proposed. It formulates several KILTs in a common format and grounds them in the same snapshot of Wikipedia, and spans five KILT tasks: fact checking, open domain question answering, slot filling, entity linking, and dialogue.

Information retrieval for KILT. Retrieving relevant contexts from a large corpus is a crucial step for KILTs. Existing work can be grouped into two lines. First, some prior work proposes to search a set of relevant documents with respect to the given query [4, 29, 34, 36]. While such methods benefit from sharing knowledge among multiple document retrieval tasks, they ignore the granularity of relevant contexts each KILT needs. Such methods are ill-suited for many KILTs, as the tasks require more fine-grained contexts to produce their final answers. Another line of research develops task-specific models to retrieve relevant contexts at different levels of granularity [1, 23, 32, 35, 44]. However, task-specific training likely hurts the generalization ability of the model to different tasks.

Motivated by the success of unifying NLP tasks in a single generative model (e.g., T0 [37], T5 [35] and FLAN [43]), Metzler et al. [33] envision a generative approach to IR that encodes all information in a corpus within the model parameters. The proposal is to learn a Seq2Seq model to map a query to a list of relevant contexts, typically represented by short strings called identifiers. Generating short identifiers rather than original long contexts, promises greater ease of optimization and memorization for generative models [21], and is relatively easy to constrain beam search decoding. Compared with modularized pipelines [14, 23, 29, 32, 36], a generative formulation has several advantages: (i) For effectiveness, a single generative model can encode the global information in a corpus and be easily optimized towards the global objective, while the pipeline framework models each document independently and has difficulty in end-to-end optimization. (ii) For efficiency, the storage footprint and computational cost are greatly reduced by overcoming the embedding space bottleneck and large document index that comes with a pipeline framework.

There have been some initial explorations to operationalize a generative vision of IR [2, 5, 6, 8, 40, 42, 47]. For example, Zhou et al. [47] assign each document an arbitrary unique identifier and train relations between query and identifiers. Some researchers [5, 8] identify Wikipedia pages by their titles and induce structure in the search space, which can be easier to memorize than unstructured identifiers. Nonetheless, existing generative IR models have all been proposed for a specific retrieval task, e.g., document [40], passage [2], and entity retrieval [8]. Besides, the identifiers are designed individually, making it difficult to adapt the model. In contrast, in this paper, we develop a unified generative retriever for KILTs.

Prompt learning. Recently, we have witnessed the bloom of pre-trained language models (PLMs) in many NLP tasks [7, 32, 39]. PLMs are usually pre-trained with general language modeling tasks and then fine-tuned with different objectives on downstream tasks. To alleviate the discrepancy between pre-training and fine-tuning, prompt learning has been proposed [30]. Prompt learning reformulates the fine-tuning data into a format identical to pre-training to leverage the implicit knowledge stored in PLMs. Recently, the development of text-to-text PLMs has shown that prompt learning could achieve notable results [13, 35, 37, 43]. For example, Wei et al. [43] fine-tune language models on a collection of datasets described via natural language instruction templates, and substantially improved the generalization ability of the model. In this work, we formulate the four retrieval tasks in the form of prompt learning, and design specific prompts for each task.

3 OUR APPROACH

In this section, we introduce the *unified generative retriever* (UGR) that performs robustly across a variety of retrieval tasks for KILTs.

3.1 Retrieval task description

In this work, we make use of the KILT benchmark [34], where all tasks require a retriever to fetch relevant contexts from a knowledge source, i.e., the same snapshot of Wikipedia, to support the final downstream task. The retrieval tasks in KILT can be categorized into four classes according to the level of granularity of relevant contexts: document retrieval, passage retrieval, sentence retrieval, and entity retrieval. To put the idea of generative IR into practice for KILTs, we formulate the four retrieval tasks as a unified Seq2Seq problem, i.e., directly generating identifiers of relevant contexts with respect to the given query.

Formally, let $C = \{C_1, C_2, \dots\}$ denote a knowledge corpus used for a retrieval task, $C_i = \{c_1, c_2, \dots, c_{|C|}\}$ denotes a textual context, which could be a document, a passage, a sentence, or an entity, and $R_i = \{r_1, r_2, \dots, r_{|R|}\}$ denotes the identifier of context C_i. Given a query $Q = \{q_1, q_2, \dots, q_{|Q|}\}$ with $|Q|$ tokens, different retrieval tasks can be uniformly formulated as a Seq2Seq problem,

$$r_k = Model(Q, r_1, r_2, \dots, r_{k-1}; \theta), \tag{1}$$

where *Model* denotes the generative retriever accomplished by decoding relevant context identifiers given an input query and θ denotes the model parameters.

3.2 Overview of the approach

Based on the above task formulation, we develop a novel *unified generative retriever* (UGR) to serve a variety of retrieval tasks in KILTs. To implement such a unified approach, we need to address two major challenges: (i) how to unify different retrieval tasks into a single generative form (Section 3.3), and (ii) how to properly specialize for different retrieval tasks when using a single model (Section 3.4). In what follows, we will introduce the two parts in detail, as well as the training and inference process (Section 3.5). The overall architecture of UGR is illustrated in Figure 1.

3.3 N-gram-based identifiers

In this section we propose how to represent relevant contexts generated in different retrieval tasks, i.e., documents, passages, sentences, and entities, in a unified way. A good identifier should have the following properties: (i) It should capture the semantic information of its associated context and equip the identifier space with semantic structure. (ii) It should be cost-efficient to be created, ideally without the need for additional human supervision or the availability of special metadata. And (iii) it should be as unique as possible to distinguish different contexts.

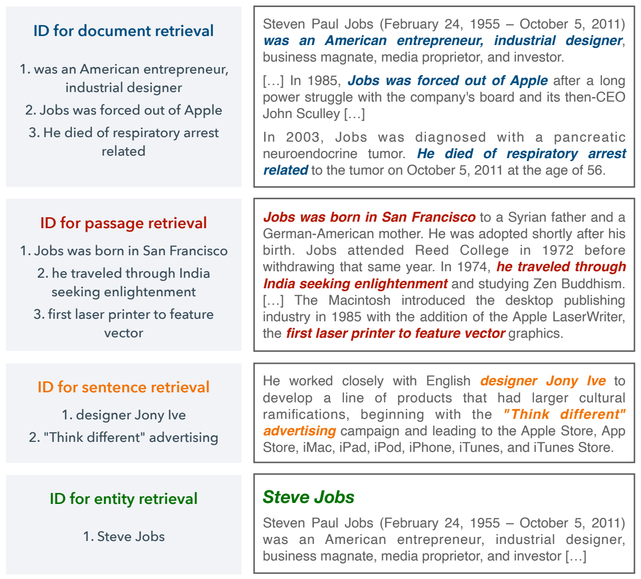
To satisfy the above requirements, we present n-gram-based identifiers to represent different contexts in a fully unsupervised way. The key idea is to use the important n-grams occurring in a context as its identifiers without the need for any structure in the search space. An example of unified n-gram-based identifiers is shown in Figure 2. During the training phase, the importance of each n-gram is estimated based on BERT's [CLS]-token attention [24], which includes three main steps: (i) n-gram importance, (ii) n-gram distribution, and (iii) important n-gram sampling.

N-gram importance. We first concatenate the query Q (note that a query is given in the training phase) and its relevant context C with special delimiter tokens as a single input sequence, i.e., [CLS]+Q+[SEP]+C+[SEP], and feed it into the original BERT model to get a d-dimensional hidden vector of [CLS], denoted as $\mathbf{h}_{\texttt{[CLS]}}$. Then, we obtain the attention weight a_i^h of the i-th token (i.e., 1-gram) c_i in context C from the h-th attention head for [CLS] in BERT's final layer, i.e.,

$$a_i^h = \text{softmax}\left(\frac{W_{query}^h \mathbf{h}_{\texttt{[CLS]}} \cdot W_{key}^h \mathbf{h}_i}{\sqrt{d/H}}\right), \tag{2}$$

where H is the number of self-attention heads and $W_{query}^h \in \mathbb{R}^{d/h \times d}$, $W_{key}^h \in \mathbb{R}^{d/h \times d}$ are the learned matrices; $\mathbf{h}_i$ denotes a d-dimensional hidden vector of the i-th token c_i. The final token importance a_i is computed by averaging the [CLS]-token attention weights across H attention heads, i.e., $a_i = \frac{1}{H}\sum_{h=1}^{H} a_i^h$. Then, the importance for the n-gram M_j spanning from c_j to c_{j+n-1} of C is denoted as $w_j = \frac{1}{n}\sum_{i=j}^{j+n-1} a_i$, where n is the length of M_j.

N-gram distribution. Generally, an n-gram may appear multiple times within a context. To mitigate this issue, we first add up the n-gram importance of the same n-gram M_j over different positions in C, i.e., $\hat{\pi}_{M_j} = \sum_{M_i = M_j} w_i, M_i \in C$. Then, inspired by the term saturation function in BM25 [36], we compute the distinct n-gram importance score as, $\pi_{M_j} = \frac{\hat{\pi}_{M_j}}{\rho + \hat{\pi}_{M_j}}$, where ρ is a hyperparameter controlling the shape of the saturation curve. Finally, the n-gram distribution in context C is obtained by normalizing the distinct n-gram importance of all the n-grams in C, i.e.,



ID for document retrieval
1. was an American entrepreneur, industrial designer
2. Jobs was forced out of Apple
3. He died of respiratory arrest related

Steven Paul Jobs (February 24, 1955 – October 5, 2011) ***was an American entrepreneur, industrial designer***, business magnate, media proprietor, and investor.
[…] In 1985, ***Jobs was forced out of Apple*** after a long power struggle with the company's board and its then-CEO John Sculley […]
In 2003, Jobs was diagnosed with a pancreatic neuroendocrine tumor. ***He died of respiratory arrest related*** to the tumor on October 5, 2011 at the age of 56.

ID for passage retrieval
1. Jobs was born in San Francisco
2. he traveled through India seeking enlightenment
3. first laser printer to feature vector

Jobs was born in San Francisco to a Syrian father and a German-American mother. He was adopted shortly after his birth. Jobs attended Reed College in 1972 before withdrawing that same year. In 1974, ***he traveled through India seeking enlightenment*** and studying Zen Buddhism. […] The Macintosh introduced the desktop publishing industry in 1985 with the addition of the Apple LaserWriter, the ***first laser printer to feature vector*** graphics.

ID for sentence retrieval
1. designer Jony Ive
2. "Think different" advertising

He worked closely with English ***designer Jony Ive*** to develop a line of products that had larger cultural ramifications, beginning with the ***"Think different" advertising*** campaign and leading to the Apple Store, App Store, iMac, iPad, iPod, iPhone, iTunes, and iTunes Store.

ID for entity retrieval
1. Steve Jobs

Steve Jobs
Steven Paul Jobs (February 24, 1955 – October 5, 2011) was an American entrepreneur, industrial designer, business magnate, media proprietor, and investor […]

Figure 2: Examples of n-gram-based identifiers for different contexts in different retrieval tasks. We sample the important n-grams occurring in a context according to the importance of each n-gram as its identifiers.

$$p(M_j|C) = \frac{\exp(\pi_{M_j})}{\sum_{M_j \in C}\exp(\pi_{M_j})}. \tag{3}$$

Important n-gram sampling. Given a context C, we sample v important n-grams as its identifiers according to $p(M_j|C)$. According to the experiments in Section 5.3, only 0.04% of the documents collide over the same n-grams in the training phase with 10 10-grams on the Wikipedia English corpus. Assuming their essential topics are almost the same, it is reasonable if (very) few documents share the same identifiers. In this work, we ignore the negligible identifier repetition problem at the training phase following [2]. As to the inference phase, recall that the KILT benchmark sets the number of most ground-truth relevant contexts as 1; we solve the repetition problem for the inference phase in Section 3.5.2.

3.4 Prompt engineering

Different retrieval tasks may compete with one another and thus "blur out" features learned by individual tasks [17]. To mitigate this issue, we plug a task-specific prompt into the query as the model input. Such a prompt could give better descriptions of each task and stimulate the model's capacity to perform a specific task. Concretely, we design three types of prompts, i.e., discrete, continuous and hybrid prompts, to encode task-specific knowledge:

- **Discrete prompts**: Inspired by recent successes in applying discrete prompts in many natural language processing tasks [35, 37, 43], we manually create cloze templates based on the characteristics of each retrieval task. In discrete prompts, prompt tokens are entirely composed of natural language without additional training. The discrete prompts designed for four different retrieval tasks are shown in Table 1.

Table 1: Discrete prompts for four different retrieval tasks.

Retrieval task	Discrete prompt
Document retrieval	Find the relevant document:
Passage retrieval	Find the relevant passage:
Sentence retrieval	Find the relevant sentence:
Entity retrieval	Find the relevant entity:

- **Continuous prompts**: To reduce human efforts for manually annotating templates so as to obtain discrete prompt, we propose to automatically learn prompts in continuous space [31]. In continuous prompts, a prompt is a trainable dense vector instead of natural language text instructions used in discrete prompts. We utilize a bidirectional long-short term memory network (LSTM) [18] as the prompt encoder to learn the prompt embedding [31].
- **Hybrid prompts**: In practice, directly learning a good continuous prompt that can effectively describe the retrieval task, is not easy since there is no prior information about the retrieval task other than training data. Therefore, we propose a hybrid prompt [15] to combine the discrete and continuous prompts. Specifically, we add anchor texts, e.g., "document," in prompts instead of using a prompt encoder to generate a purely continuous vector. This enforces the model to squeeze the essential information from the input for different retrieval tasks.

After prompt engineering, the descriptions of each retrieval task are mapped to specific well-designed prompts. Then, we guide the generative model to learn the common and shared knowledge on a mixture of different retrieval tasks phrased as prompts. Besides, the task-specific prompts stimulate the model capacity in distinguish different retrieval tasks and achieving good generalization.

3.5 Training and inference

We introduce a multi-task training strategy that augments the model's ability with different corpora and the inference process to efficiently find matching contexts that contain the generated n-grams.

3.5.1 Multi-task training. In the multi-task training process, each training sample is represented as $u_i^t = (S_i^t, Q_i^t, R_i^t)$, where t is the retrieval task, i.e., document retrieval, passage retrieval, sentence retrieval, or entity retrieval. S_i^t is the task-specific prompt for the i-th input query of task t, Q_i^t is the i-th input query of task t, and R_i^t is the identifier of the i-th relevant context of task t. For each context, its identifier R_i^t can be defined by v important n-grams.

The model that we propose, UGR, is built on a transformer-based Seq2Seq model, BART [28]. We train UGR with a standard Seq2Seq objective, i.e., maximizing the likelihood of the output identifier strings with teacher forcing, and the parameters of the model are optimized in an end-to-end manner by the cross entropy loss, i.e.,

$$\mathcal{L} = \arg\max_{\theta} \sum_{t} \sum_{i} \sum_{k} \log p(r_k^t \mid r_{<k}^t, S_i^t, Q_i^t; \theta), \tag{4}$$

where θ denotes the model parameters.

3.5.2 Inference process. For all four retrieval tasks, we construct an FM-index [12], which provides information on all the contexts in the given corpus, i.e., Wikipedia. This forces each generated string to be a valid identifier, i.e., n-grams occurring in all the contexts.

Specifically, an FM-index is an index combining the Burrows-Wheeler Transform (BWT) [3] with a few small auxiliary data structures. The core of an FM-index consists of **F** and **L** from BWT, where **F** is an array of runs and **L** is the string's BWT. Because the relative rank of **F** and **L** stays the same, we employ an FM-index to identify the list of possible token successors with constrained beam search [8]: (i) Given the starting token, we first use **F** to find the contiguous range of rows corresponding to the token. (ii) Then, we switch to **L** to examine the same range of rows to obtain the list of the next valid tokens. And (iii) the valid tokens in **F** are selected based on the same ranks in **L**. By iteratively repeating the above procedure, we can find a valid n-gram with arbitrary size.

A shortcoming with n-gram-based identifiers is that different contexts may contain the same important n-grams in document retrieval, passage retrieval and sentence retrieval. Besides, considering all generated n-grams can better capture the information within a context. Entity retrieval does not have this problem, since we use the unique document titles as identifiers of relevant entities [8].

Therefore, inspired by [2], given a query in the test data, we first obtain all the candidate contexts that contain the n-grams generated by beam search and then introduce an interactive scoring function to rank the candidate contexts, which combines the contribution of several different n-grams contained in the same context.

Formally, let $F(M, C)$ denote the frequency of the n-gram M in the corpus C used for the retrieval task. The unconditional n-gram probabilities can be computed as $p(M) = \frac{F(M,C)}{\sum_{M \in C} |M|}$. Then, given an input query Q, we obtain the weight of M by

$$w(M, Q) = \max\left(0, \log \frac{p(M \mid S, Q)(1 - p(M))}{p(M)(1 - p(M \mid S, Q))}\right), \tag{5}$$

where $p(M \mid S, Q)$ is the probability of the generative model decoding M conditioned on the query Q and its prompt S. Given multiple generated n-gram identifiers R, we can obtain its corresponding contexts and the score of each context C for Q as:

$$W(C, Q) = \sum_{R \in K^C} w(R, Q)^{\alpha} \cdot \text{cover}(R, K), \tag{6}$$

where α is a hyperparameter, K is the set of all generated n-grams for Q and K^C is the subset of n-grams in K that appear in C. cover(R, K) is defined as, $\text{cover}(R, K) = 1 - \beta + \beta \cdot \frac{|\text{set}(R) \setminus V(K)|}{|\text{set}(R)|}$, where β is a hyperparameter, set(R) is the set of tokens in R, and $V(K)$ is the union of all tokens in K with top-g highest scores. In this work, we select the context C with the highest score $W(C, Q)$ as the relevant context for the test query Q.

4 EXPERIMENTAL SETUP

Next, we introduce our experimental settings, including datasets, baseline methods, evaluation metrics, and implementation details.

4.1 Datasets

We conduct a series of experiments on the KILT benchmark [34]. Detailed statistics of the benchmark datasets are shown in Table 2. A listing of retrieval datasets grouped by retrieval task is provided in Table 3, including document retrieval (*DR*), passage retrieval (*PR*), sentence retrieval (*SR*), and entity retrieval (*ER*). For each retrieval task, we construct the datasets that are in the training mixture (i.e., in-domain datasets) and are not seen during training (i.e., out-of-domain datasets), respectively.

Table 2: Statistics of datasets in the KILT benchmark. '–' denotes that the task does not provide ground-truth documents in the training set.

Label	Dataset	Train size	Dev size	Test size
FEV	FEVER [41]	104,966	10,444	10,100
AY2	AIDA CoNLL-YAGO [19]	18,395	4,784	4,463
WnWi	WNED-WIKI [16]	–	3,396	3,376
WnCw	WNED-CWEB [16]	–	5,599	5,543
T-REx	T-REx [10]	2,284,168	5,000	5,000
zsRE	Zero Shot RE [27]	147,909	3,724	4,966
NQ	Natural Questions [26]	87,372	2,837	1,444
HoPo	HotpotQA [46]	88,869	5,600	5,569
TQA	TriviaQA [22]	61,844	5,359	6,586
ELI5	ELI5 [11]	–	1,507	600
WoW	Wizard of Wikipedia [9]	63,734	3,054	2,944

4.2 Baselines

To verify the effectiveness of UGR, we first implement variants of the model: (i) **BART^{sp}** is a basic BART$_{large}$ model that uses all datasets in each retrieval task listed in Table 2 for training. It takes the query as input and the n-gram-based identifiers as output. (ii) **BART$^{sp}_{hp}$** extends BART^{sp} by adding a hybrid prompt to the query as the model input. (iii) **BART^{mt}** removes the prompt learning strategy used in UGR, which can be regarded as an adaption of multi-task learning to generative retrieval.

We adopt several baseline methods for comparison: (i) **BM25** [36] is a classical probabilistic retrieval model. (ii) **GENRE** [8] performs retrieval by generating the document titles for *DR*. (iii) **SEAL** [2] generates passage identifiers to retrieve relevant passages for *PR*. (iv) **MT-DPR** [32] jointly trains a DPR model [23] on an extensively selected retrieval datasets for *SR*. (v) **BLINK** [34] combines BLINK [44] and the flair [1] retrieval solution that ranks pages according to entities in the input for *ER*. We take the best results of these models for the corresponding retrieval task from the original papers.

4.3 Evaluation metrics

Following previous work [2, 8, 29, 32, 34] and the official KILT instructions,[2] we use R-precision (%) as the evaluation metric for all four retrieval tasks. R-precision is calculated as $\frac{r}{R}$, where R is the number of contexts inside each provenance set and r is the number of relevant contexts among the top-R retrieved contexts. For downstream evaluation, we adopt specific metrics for different downstream tasks. Specifically, as suggested in the KILT resource paper [34], we use Accuracy (*ACC*) for FEV, AY2, WnWi, WnCw, T-REx and zsRE; Exact Match (*EM*) for NQ, TQA and HoPo; *ROUGE-L* for ELI5; and *F1* for WoW. Following previous work [2, 32, 44], we report all performance results on the dev sets since the KILT leaderboard limits the frequency of the submission for test performance.

4.4 Implementation details

In this section, we describe implementation details of UGR.

- **Model architecture:** UGR is based on the transformer-based encoder-decoder architecture, where the hidden size is 1024, the feed-forward layer size is 4096, the number of transformer layers is 12, and the number of self-attention heads is 16, for both the encoder and decoder. The total number of parameters is 406M. We implement UGR in PyTorch based on the fairseq library.[3]
- **Identifier construction:** During the training phase, we use the original BERT$_{base}$ [24] to encode the concatenated text. The length n of n-grams used is 10 and the number of n-grams v is 10 for *DR* and *PR*, while n is 10 and v is 5 for *SR*. The value of ρ in the saturation function is 0.01. For *ER*, since entity names are unique and their length is short, we directly use the entity name as the identifier, i.e., n is the length of an entity name and v is 1.
- **Prompt engineering:** For discrete prompts, we directly add the natural language prompts listed in Table 1 to the input query in specific tasks as the model input. For continuous and hybrid prompts, the length of prompt tokens is set to 6 and the hidden size of the LSTM is set to 1024. The anchor texts in the hybrid prompt are "document", "passage", "sentence", and "entity" for *DR*, *PR*, *SR* and *ER*, respectively.
- **Training hyperparameters:** We initialize the parameters of the encoder-decoder architecture from the official checkpoint of BART$_{large}$ [28]. We use a learning rate of $3e^{-5}$ and the Adam optimizer [25] 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 four NVIDIA Tesla A100 40GB GPUs. Following [31], we first only train the prompt encoder, and then train the generative model while fixing the prompt encoder. We refer to our UGR model with discrete prompt, continuous prompt and hybrid prompt as **UGR$_{dp}$**, **UGR$_{cp}$** and **UGR$_{hp}$**, respectively.
- **Inference hyperparameters:** We use the C++ implementation of an FM-index in sdsl-lite.[4] We build an FM-index on the Wikipedia English corpus, which is the knowledge source for the four retrieval tasks. At inference time, we adopt constrained beam search to decode the identifier with 10 timesteps and 15 beams. The value of α is set to 2.0, β is set to 0.8, and g is set to 5.

[2]https://eval.ai/challenge/689/leaderboard

Table 3: Datasets for different retrieval tasks in KILT. For each retrieval task, we include some datasets in the training mixture, while some are reserved as held-out datasets.

Task	Training-mixture datasets	Held-out datasets
DR	FEV, T-REx, NQ, HoPo, TQA, WoW	zsRE, ELI5
PR	FEV, T-REx, WoW	zsRE
SR	NQ, HoPo, TQA	ELI5
ER	AY2	WnWi, WnCw

5 EXPERIMENTAL RESULTS

Our experiments are organized around five research questions: **(RQ1)** How does UGR perform compared to strong retrieval baselines on both in-domain and out-of-domain datasets? **(RQ2)** How is the adaptability of UGR to unseen tasks? **(RQ3)** How does the n-gram-based identifier affect retrieval performance? **(RQ4)** Can relevant contexts retrieved by UGR improve the performance of downstream tasks in KILT? **(RQ5)** How does UGR perform compared to traditional retrieval methods and generative methods in

[3]https://github.com/pytorch/fairseq
[4]https://github.com/simongog/sdsl-lite

Table 4: R-precision (%) for four retrieval tasks on in-domain datasets. Best results are marked in boldface. $*$ indicates statistically significant improvements over all baselines (p-value < 0.05).

	DR						PR			SR			ER
Model	**FEV**	**T-REx**	**NQ**	**HoPo**	**TQA**	**WoW**	**FEV**	**T-REx**	**WoW**	**NQ**	**HoPo**	**TQA**	**AY2**
BM25	50.13	58.60	25.83	43.95	29.44	27.50	40.10	51.60	18.40	14.20	38.40	16.20	3.47
Previous SOTA	84.68	79.68	60.25	51.82	71.11	56.32	67.59	58.24	35.82	42.66	50.70	39.98	89.39
Task-specific retriever for each DR, PR, SR and ER task													
BART^{sp}	82.48	72.23	63.17	55.91	69.78	55.48	67.64	58.49	36.08	42.17	52.65	39.40	90.56
BART$^{sp}_{hp}$	83.41	73.76	64.09	54.23	70.83	56.27	67.82	58.36	36.34	43.36	52.51	40.55	90.83
Multi-task retriever for all DR, PR, SR and ER tasks													
BART^{mt}	82.13	71.52	62.25	52.88	66.45	54.55	66.50	56.76	34.93	40.52	53.38	41.19	89.67
UGR$_{dp}$	85.41*	80.75*	64.89*	57.73*	72.36*	59.91*	68.34	59.25*	36.60	44.49*	54.87*	42.11*	91.94*
UGR$_{cp}$	85.88*	80.93*	65.21*	57.24*	72.65*	60.46*	69.72*	59.81*	37.18*	44.75*	55.31*	42.26*	93.01*
UGR$_{hp}$	**86.29***	**81.21***	**65.47***	**58.74***	**73.04***	**61.22***	**69.91***	**60.17***	**37.74***	**45.29***	**55.86***	**42.44***	**93.13***

Table 5: R-precision (%) for four retrieval tasks on out-of-domain datasets. Best results are marked in boldface. $*$ indicates statistically significant improvements over all baselines (p-value < 0.05).

	DR		PR	SR	ER	
Model	**zsRE**	**ELI5**	**zsRE**	**ELI5**	**WnWi**	**WnCw**
BM25	66.43	8.23	52.98	0.28	0.35	1.74
Previous SOTA	90.46	11.26	74.50	1.46	85.26	68.57
Task-specific retriever for each DR, PR, SR and ER task						
BART^{sp}	94.58	12.49	78.27	2.11	86.69	69.57
BART$^{sp}_{hp}$	95.62	12.83	78.85	3.03	86.94	70.10
Multi-task retriever for all DR, PR, SR and ER tasks						
BART^{mt}	90.15	10.47	75.82	1.59	84.23	67.75
UGR$_{dp}$	97.52*	13.54*	78.26	3.15	87.81*	70.72*
UGR$_{cp}$	98.08*	13.81*	78.89	3.77*	88.49*	70.96*
UGR$_{hp}$	**98.66***	**14.60***	**79.25***	**4.97***	**88.83***	**71.40***

terms of computational cost? In the following subsections we answer our research questions.

5.1 Evaluation on in-domain and out-of-domain datasets

To answer **RQ1**, we compare UGR with baselines on both in-domain datasets and out-of-domain datasets.

5.1.1 In-domain performance. We train our model over the mixture of training datasets listed in Table 2 and evaluate the performance on the dev datasets of each specific task. Table 4 shows the results. We observe that: (i) The traditional retrieval model BM25 is a strong baseline that performs well on most retrieval tasks. (ii) For previous SOTA models, the document-focused retriever GENRE achieves promising results on *DR* by sharing useful features across different datasets. However, it is difficult to adapt it to other retrieval tasks due to its designed mechanism (e.g., requiring unique titles for contexts in GENRE). Existing task-specific retrievers (i.e., SEAL, MT-DPR and BLINK) obtain good performance by effectively learning the task-specific characteristics. Nonetheless, these methods have poor generalization abilities, as shown in Table 5, since they are trained for a single specific task.

When we look at variants of UGR, we find that: (i) BART^{sp} achieves better results than BART^{mt}. This indicates that fine-tuning PLM on the simply mixed datasets, is not effective as it ignores the task-specific characteristics. (ii) BART$^{sp}_{hp}$ outperforms BART^{sp}, showing that prompt learning utilizes the signals shared by each task and distinguishes different tasks, thereby improving the performance on each retrieval task.

Finally, we observe that UGR$_{hp}$ significantly outperforms all baseline methods. Specifically, (i) The improved results of UGR compared to BART^{mt} demonstrate the effectiveness of the prompt learning strategy. That is, by introducing task-specific prompts, UGR effectively learns general knowledge across tasks, while handling them based on the characteristics of different tasks. (ii) Among the three variants of UGR, UGR with hybrid prompts outperforms UGR with discrete and continuous prompts, showing that it is effective to use natural language to control the learning of continuous prompt tokens to describe the retrieval task being addressed. Overall, the improvements of UGR over previous SOTA on in-domain performance suggests that generative methods for IR deserve further exploration.

5.1.2 Out-of-domain performance. We also evaluate the generalization ability of UGR to out-of-domain datasets. Specifically, we train UGR on the mixture datasets and test it on the held-out datasets listed in Table 2. Table 5 lists the results. We find that: (i) BART^{sp} and BART$^{sp}_{hp}$ perform better than BART^{mt}, but worse than UGR. This shows that UGR is able to capture knowledge of each task in multi-task learning. (ii) UGR$_{hp}$ outperforms the baselines on all out-of-domain datasets. This result demonstrates the generalization ability of UGR on new domains compared to existing methods. Each specific task is described by the corresponding prompt tokens, which facilitates knowledge transfer among different datasets under the same task. Moreover, by jointly training on multiple tasks, UGR is able to improve the generalization robustness.

5.2 Adaptability to unseen tasks

To answer **RQ2**, we explore the zero-shot and few-shot learning capability of UGR on unseen tasks. Concretely, for the four retrieval tasks, we select three of them for mixed training and evaluate UGR on the dev set of the remaining one. Following [45], we average the prompt tokens of three training tasks as that of the unseen task

Table 6: R-precision (%) on unseen tasks under zero-shot setting. Each task in the first line is an unseen task and we train UGR on the other three tasks. We also evaluate the performance under few-shot setting by providing little data in the four tasks. Best results are marked in boldface.

	DR						PR			SR			ER
Model	**FEV**	**T-REx**	**NQ**	**HoPo**	**TQA**	**WoW**	**FEV**	**T-REx**	**WoW**	**NQ**	**HoPo**	**TQA**	**AY2**
BM25	50.13	58.60	25.83	43.95	29.44	27.50	40.10	51.60	18.40	14.20	38.40	16.20	3.47
Multi-task retriever for the other three tasks under a zero-shot setting													
BART^{mt}	66.38	64.14	32.61	40.21	32.78	40.84	43.52	30.19	15.66	18.99	30.18	28.44	5.67
UGR$_{dp}$	69.81	68.84	36.41	44.83	42.05	45.93	49.19	34.07	18.07	21.55	36.65	31.20	13.45
UGR$_{cp}$	70.54	68.41	36.66	**46.21**	42.57	45.95	50.45	34.62	18.61	21.79	38.41	31.82	15.27
UGR$_{hp}$	**72.25**	**70.34**	**38.48**	46.02	**44.43**	**46.28**	**51.24**	**36.21**	**20.37**	**22.36**	**39.06**	**32.51**	**18.23**
Multi-task retriever for the other three tasks under a few-shot setting													
BART^{mt}	79.92	73.45	57.47	48.50	55.94	51.27	55.17	45.44	25.00	35.46	41.73	35.64	78.89
UGR$_{dp}$	81.49	74.01	57.81	49.01	58.97	52.81	58.33	49.81	27.93	37.41	45.01	38.72	83.61
UGR$_{cp}$	**82.60**	73.94	58.29	49.96	59.41	54.07	61.08	48.36	28.45	38.05	46.33	39.55	83.93
UGR$_{hp}$	82.46	**75.10**	**61.37**	**50.48**	**61.10**	**55.60**	**62.91**	**51.28**	**29.08**	**38.26**	**46.81**	**40.14**	**85.58**

for UGR$_{cp}$ and UGR$_{hp}$. And for UGR$_{dp}$, the prompt tokens of the unseen task are shown in Table 1. Under the few-shot setting, we randomly pick 1,000 instances from each task, and fine-tune UGR on each unseen task. We pick the last checkpoint to evaluate the performance on the original dev set.

The results are displayed in Table 6. We find that: (i) Under the zero-shot setting, UGR achieves competitive results to the strong baseline BM25. This indicates that the information of other tasks can improve the adaptability of the model to unseen tasks via multi-task prompt learning. (ii) Under the few-shot setting, UGR$_{hp}$ adapts well to unseen tasks to achieve a strong performance, showing that it has learned common retrieval knowledge. And (iii) with limited fine-tuning examples, UGR$_{hp}$ outperforms previous SOTA baselines on some datasets. For the NQ dataset on *DR*, UGR achieves comparable quality to previous SOTA (i.e., 61.37% vs 60.25%) with full supervised learning. This demonstrates that UGR is able to utilize the limited information from unseen tasks to facilitate the generalization ability to unseen retrieval tasks.

5.3 Analysis of n-gram-based identifiers

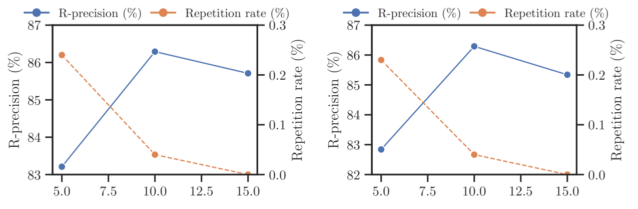
To answer **RQ3**, in this section, we conduct analyses on the n-gram-based identifier.

Impact of important n-gram sampling strategy. To assess whether the proposed n-gram sampling is effective for *DR*, *PR*, and *SR* in the training phrase, we compare it with a random sampling strategy, which randomly samples multiple n-grams from the context. Recall that we do not need to sample n-grams for *ER*. We sample ten 10-grams for *DR* and *PR*, five 10-grams for *SR*, and train the UGR$_{hp}$ under the same setting described in Section 4.4. We write UGR$_{important}$ and UGR$_{random}$ for the models that use important n-gram sampling and random sampling, respectively. Due to space limitations, we only show the results on selected datasets for each task, i.e., *DR* (FEV, TQA), *PR* (T-REx, WoW), and *SR* (NQ, HoPo). See Table 7.

We observe that UGR$_{random}$ performs worse than UGR$_{important}$ by a large margin on most retrieval tasks. The n-grams sampled in a random way are not able to effectively represent the essential semantics of contexts, making it difficult for the model to learn the mapping relationship between query and relevant contexts.

Table 7: Comparison of UGR$_{hp}$ with the proposed important n-gram sampling strategy and random sampling strategy. Best R-precision (%) results are marked in boldface.

	DR		PR		SR	
Model	**FEV**	**TQA**	**T-REx**	**WoW**	**NQ**	**HoPo**
UGR$_{random}$	80.41	65.67	54.38	20.42	33.14	48.95
UGR$_{important}$	**86.29**	**73.04**	**60.17**	**37.74**	**45.29**	**55.86**



(a) Impact of the length of n-grams n (b) Impact of the number of n-grams v

Figure 3: R-precision (%) performance and repetition rate (%) of the UGR$_{hp}$ method with different lengths n and different numbers of n-grams v.

This result further validates the effectiveness of our strategy on sampling important n-grams from the relevant contexts.

Impact of the length and number of n-grams. Next, we analyze the impact of the length n and the number of n-grams v for UGR$_{hp}$. Due to space limitations, we only show the performance on the FEV dataset for *DR*; qualitatively similar observations can been made for *PR* and *SR*. We report the performance in terms of R-precision as well as the repetition rate of n-grams, which denotes the percentage of the number of contexts with repeated n-grams among the total number of contexts. We first fix v to ten, and test the performance of UGR$_{hp}$ over different lengths of n-grams, varying n in $\{5, 10, 15\}$. Then we fix n to ten, and test the effect of different values of v, i.e., $\{5, 10, 15\}$. See Figure 3.

We observe that by setting the length of the n-grams to 5 or 15, UGR$_{hp}$ seems to either represent insufficient semantic information or to represent noisy information that may hurt the identifier generation. For short n-grams, UGR$_{hp}$ has a high repetition rate with

Table 8: Downstream results for four retrieval tasks on the KILT dev sets. The metrics used are accuracy (%) for fact checking, slot filling, and entity linking (EL); exact match (%) for QA; and F1 score (%) for dialogue. Best results are marked in boldface.

Model	Fact checking		Slot filling		Dialogue		Open domain QA						EL
	FEV		T-REx		WoW		NQ		HoPo		TQA		AY2
	DR	PR	DR	PR	DR	PR	DR	SR	DR	SR	DR	SR	ER
Previous SOTA+FiD	86.74	88.29	76.53	82.41	15.33	17.54	48.68	51.84	36.92	40.06	70.08	71.65	89.39
UGR$_{hp}$+FiD	**87.36**	**89.83**	**79.40**	**83.91**	**15.52**	**18.47**	**51.83**	**54.05**	**38.37**	**41.68**	**71.54**	**72.32**	**93.13**

other identifiers (e.g., 4.36% for 10 5-grams), which may hurt the distinctiveness among documents. Longer n-grams may more likely lead to error accumulation of generation. UGR$_{hp}$ with v set to 15 performs worse than $v = 10$, which shows that too many identifiers may introduce noisy information that may hurt retrieval performance. Therefore, it is important to achieve a trade-off between the length and number of n-grams and the performance.

5.4 Downstream performance

To answer **RQ4**, we pair UGR$_{hp}$ with a representative and widely-used downstream reader model for KILT, i.e., Fusion-in-Decoder (FiD) [20], following the setup in [2]. For comparison, we pair previous SOTA retrieval models for each KILT task with FiD. Specifically, the reader component takes a query and the first five relevant contexts retrieved by the UGR$_{hp}$ model and the SOTA retrieval models respectively as input, to generate the final answer. The results are displayed in Table 8.

We observe that: (i) Coarse-grained contexts do not support downstream tasks that need fine-grained contexts in the reader component. For example, in the slot filling task, the drop rate of UGR$_{hp}$+FiD with retrieved documents as input of the reader compared to retrieved passages is about 4.51%. (ii) Compared with previous SOTA, UGR$_{hp}$+FiD achieves significantly better performance on all the datasets. This result further demonstrates that by introducing n-gram-based identifiers and prompt learning, UGR can effectively make use of shared information to retrieve more precise contexts for specific downstream tasks.

5.5 Memory and inference efficiency

Finally, to answer **RQ5**, we compare UGR with traditional retrieval models (MT-DPR and BLINK) and advanced generative retrieval models (GENRE and SEAL), in terms of memory and inference time. The memory footprint is the disk space required by each model. For the inference time, we compute the average time UGR takes to run four retrieval tasks, and directly use the time each baseline takes to run the specific retrieval task. Table 9 lists the results.

We find that: (i) Compared with traditional retrieval models, generative retrieval models have more model parameters, but a smaller memory footprint and faster inference speed. The reason is that traditional retrieval models need to store the dense representations for the whole corpus besides the model parameters, while the parameters of generative retrieval models scale linearly with the vocabulary size, not document count. Moreover, the inference time of generative retrieval models is directly proportional to the beam size with a limited overhead by constrained decoding. (ii) Compared with GENRE, UGR has a larger memory footprint and higher inference time, since UGR constructs the FM-index to constrain the generation process, which is larger than the prefix tree adopted in GENRE. Besides, the document identifier designed in GENRE is unique, saving time in the de-duplication process. (iii) Compared with SEAL, UGR achieves comparable inference time. The reason may be that UGR is an all-around model for a diversity of retrieval tasks, while SEAL can only be used for one specific retrieval task.

Table 9: Comparisons on memory footprint, number of model parameters, and inference time.

Model	Task	Memory	Parameters	Time
MT-DPR	*SR*	70.9 GB	220M	15.34 ms
BLINK	*ER*	24.2 GB	220M	12.71 ms
GENRE	*DR*	**2.1** GB	**406M**	**5.81** ms
SEAL	*PR*	8.8 GB	406M	10.16 ms
UGR	*ALL*	8.8 GB	406M	10.58 ms

6 CONCLUSION

We have proposed UGR, a novel Unified Generative Retriever, which can robustly serve different retrieval tasks for knowledge-intensive language tasks. To unify retrieval tasks, we formulated the retrieval problem as a conditional generation problem and introduced an n-gram-based identifier for relevant contexts at different levels of granularity. To learn different retrieval tasks with a single model, we mapped the descriptions of tasks to a few prompt tokens for keeping task specifications. Empirical results on the KILT benchmark demonstrated the superiority of the proposed method.

Efficiently integrating knowledge from different retrieval tasks in UGR has the potential to save significant time and computational resources in both academic and industrial environments. However, UGR needs a complex scoring function to solve the identifier repetition problem; we encourage future work that explores other effective and efficient semantic identifiers for generative retrieval. Beyond KILT, training a more general unified generative retrieval model to serve different retrieval applications under multiple corpora and modalities seems a promising future direction.

ACKNOWLEDGMENTS

This work was funded by the National Natural Science Foundation of China (NSFC) under Grants No. 62006218 and 61902381, the China Scholarship Council under Grants No. 202104910234, 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. This work was also (partially) funded by the Hybrid Intelligence Center, a 10-year program funded by the Dutch Ministry of Education, Culture and Science through the Netherlands Organization for Scientific Research, https://hybrid-intelligence-centre.nl. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

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 *NAACL 2019*. 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*.

[3] Michael Burrows and David Wheeler. 1994. A Block-sorting Lossless Data Compression Algorithm. In *Digital SRC Research Report*. Citeseer.

[4] 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.

[5] Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Yixing Fan, and Xueqi Cheng. 2022. GERE: Generative evidence retrieval for fact verification. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2184–2189.

[6] 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 *CIKM*. 191–200.

[7] Ronan Collobert and Jason Weston. 2008. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *Proceedings of the 25th international conference on Machine learning*. 160–167.

[8] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2020. Autoregressive Entity Retrieval. In *International Conference on Learning Representations*.

[9] 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*.

[10] 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 2018*.

[11] Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. ELI5: Long Form Question Answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 3558–3567.

[12] Paolo Ferragina and Giovanni Manzini. 2000. Opportunistic Data Structures with Applications. In *Proceedings 41st annual symposium on foundations of computer science*. IEEE, 390–398.

[13] Luciano Floridi and Massimo Chiriatti. 2020. GPT-3: Its Nature, Scope, Limits, and Consequences. *Minds and Machines* 30, 4 (2020), 681–694.

[14] Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, and Alfio Gliozzo. 2021. Robust Retrieval Augmented Generation for Zero-shot Slot Filling. In *EMNLP 2021*. 1939–1949.

[15] Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2022. PPT: Pre-trained Prompt Tuning for Few-shot Learning. In *ACL*.

[16] Zhaochen Guo and Denilson Barbosa. 2018. Robust Named Entity Disambiguation with Random Walks. *Semantic Web* 9, 4 (2018), 459–479.

[17] Han He and Jinho D Choi. 2021. The Stem Cell Hypothesis: Dilemma Behind Multi-task Learning with Transformer Encoders. *arXiv preprint arXiv:2109.06939* (2021).

[18] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[19] 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 *Proceedings of the 2011 conference on empirical methods in natural language processing*. 782–792.

[20] Gautier Izacard and Édouard Grave. 2021. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. In *EACL*. 874–880.

[21] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2022. Survey of Hallucination in Natural Language Generation. *arXiv preprint arXiv:2202.03629* (2022).

[22] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *ACL*.

[23] 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 2020*. 6769–6781.

[24] 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.

[25] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.).

[26] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics* 7 (2019), 453–466.

[27] Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-Shot Relation Extraction via Reading Comprehension. In *CoNLL 2017*. 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, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented Generation for Knowledge-intensive NLP Tasks. *NeurIPS 2020* 33 (2020), 9459–9474.

[30] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *arXiv preprint arXiv:2107.13586* (2021).

[31] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. GPT Understands, Too. *arXiv preprint arXiv:2103.10385* (2021).

[32] 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 2021*. 1098–1111.

[33] 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.

[34] 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 *NAACL 2021*. Association for Computational Linguistics, Online, 2523–2544.

[35] 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.

[36] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval* 3 (2009), 333–389. Issue 4.

[37] Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M. Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Févry, Jason Alan Fries, Ryan Teehan, Stella Biderman, Leo Gao, Tali Bers, Thomas Wolf, and Alexander M. Rush. 2022. Multitask Prompted Training Enables Zero-Shot Task Generalization. In *ICLR*.

[38] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. *Advances in neural information processing systems* 27 (2014).

[39] Alon Talmor and Jonathan Berant. 2019. MultiQA: An Empirical Investigation of Generalization and Transfer in Reading Comprehension. *arXiv preprint arXiv:1905.13453* (2019).

[40] Yi Tay, Vinh Q Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, Tal Schuster, William W. Cohen, and Donald Metzler. 2022. Transformer Memory as a Differentiable Search Index. *arXiv preprint arXiv:2202.06991* (2022).

[41] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: A Large-scale Dataset for Fact Extraction and VERification. In *NAACL: Human Language Technologies, Volume 1 (Long Papers)*. 809–819.

[42] Yujing Wang, Yingyan Hou, Haonan Wang, Ziming Miao, Shibin Wu, Hao Sun, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, Zheng Liu, Xing Xie, Hao Allen Sun, Weiwei Deng, Qi Zhang, and Mao Yang. 2022. A Neural Corpus Indexer for Document Retrieval. *arXiv preprint arXiv:2206.02743* (2022).

[43] Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned Language Models are Zero-shot Learners. *arXiv preprint arXiv:2109.01652* (2021).

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

[45] Shicheng Xu, Liang Pang, Huawei Shen, and Xueqi Cheng. 2022. Improving Multi-task Generalization Ability for Neural Text Matching via Prompt Learning. *arXiv preprint arXiv:2204.02725* (2022).

[46] 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.

[47] 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).