Dense Re-Ranking with Weak Supervision for RDF Dataset Search

Qiaosheng Chen, Zixian Huang, Zhiyang Zhang, Weiqing Luo, Tengteng Lin, Qing Shi, and Gong Cheng(✉)

State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
{qschen,zixianhuang,zhiyangzhang,wqluo,tengtenglin, qingshi}@smail.nju.edu.cn, gcheng@nju.edu.cn

Abstract. Dataset search aims to find datasets that are relevant to a keyword query. Existing dataset search engines rely on conventional sparse retrieval models (e.g., BM25). Dense models (e.g., BERT-based) remain under-investigated for two reasons: the limited availability of labeled data for fine-tuning such a deep neural model, and its limited input capacity relative to the large size of a dataset. To fill the gap, in this paper, we study dense re-ranking for RDF dataset search. Our re-ranking model encodes the metadata of RDF datasets and also their actual RDF data—by extracting a small yet representative subset of data to accommodate large datasets. To address the insufficiency of training data, we adopt a coarse-to-fine tuning strategy where we warm up the model with weak supervision from a large set of automatically generated queries and relevance labels. Experiments on the ACORDAR test collection demonstrate the effectiveness of our approach, which considerably improves the retrieval accuracy of existing sparse models.

Keywords: Dataset search · Dense re-ranking · Data augmentation

1 Introduction

As data plays an increasingly crucial role in many domains, the capability to search for relevant datasets has become critical [5]. To satisfy this need, dataset search engines such as Google Dataset Search [2,3] have emerged. The Semantic Web community is particularly interested in RDF dataset search, and has also developed a few such solutions [6,24,30] and made benchmarking efforts [18].

Motivation. Existing RDF dataset search solutions employ conventional sparse models (e.g., BM25 [26]) to retrieve lexically relevant datasets, which cannot capture the semantic relationships between query and dataset. By contrast, building on the semantic matching capability of pre-trained language models (e.g., BERT [10]) to understand text, dense ranking models (e.g., DPR [12]) have achieved remarkable performance in document retrieval [33]. It inspires us to study *dense models for RDF dataset search* and investigate their effectiveness.

T. R. Payne et al. (Eds.): ISWC 2023, LNCS 14265, pp. 23–40, 2023.
https://doi.org/10.1007/978-3-031-47240-4_2

Challenges. Applying dense models to RDF dataset search is a nontrivial task. Indeed, we identify the following two challenges. Note that these difficulties also face dataset search in general, not limited to RDF dataset search.

- Unlike documents, (RDF) datasets are structured and commonly *very large*, e.g., containing thousands or millions of RDF triples. It remains unclear how to effectively feed such a huge amount of data into a dense ranking model which typically allows a maximum input length of only 512 tokens, and we should not simply drop the data but rely solely on the metadata of a dataset since this has been proven to hurt accuracy [6,18].
- Unlike document retrieval which is an established research task with many large test collections, (RDF) dataset search is relatively new and is now accompanied by only a few relatively small test collections [13,18]. The *limited labeled data* in these test collections is insufficient for tuning a dense ranking model having at least hundreds of millions of trainable parameters.

Our Work. We propose to study dense ranking models for RDF dataset search and address the above two challenges. Our approach adopts a popular retrieval-then-reranking architecture, and we use dense models in the re-ranking step. To feed the metadata and content of an RDF dataset into the model, we concatenate metadata fields as well as a small subset of RDF triples extracted from the data as a representative data sample. To tune the model, besides the limited labeled data provided by existing test collections, we adopt a coarse-to-fine tuning strategy and we propose two methods for automatically generating a large amount of possibly noisy labeled data to weakly supervise the model in the preliminary coarse-tuning phase. We refer to our approach as $\mathbf{DR}^2$, short for Dense Rdf Dataset Re-ranking. To summarize, our contributions include

- the first research attempt to adapt dense ranking models to RDF dataset search, by encoding representative RDF triples extracted from large datasets,
- two methods for automatically generating labeled data to coarse-tune the model, one based on distant supervision and the other based on self-training,
- experiments on a public test collection, empirically comparing a variety of triple extraction methods, dense ranking models, and tuning strategies.

Outline. The remainder of the paper is organized as follows. Section 2 introduces our retrieval-then-reranking approach for RDF dataset search. Section 3 details our coarse-to-fine tuning strategy. Section 4 presents evaluation results. Section 5 discusses related work. Section 6 concludes the paper with future work.

2 Dense Re-Ranking for RDF Dataset Search

In this section, we describe our retrieval-then-reranking approach for RDF dataset search. We begin with an overview of the approach. Then we detail its two major steps: compact document representation and dense re-ranking.

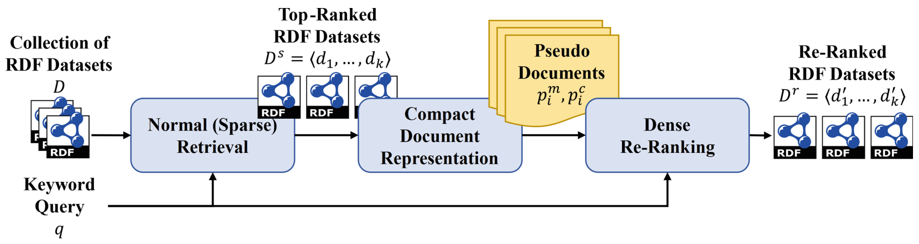

Collection of RDF Datasets
D
Keyword Query
q
Normal (Sparse) Retrieval
Top-Ranked RDF Datasets
$D^s = \langle d_1, \dots, d_k \rangle$
Compact Document Representation
Pseudo Documents
p_i^m, p_i^c
Dense Re-Ranking
Re-Ranked RDF Datasets
$D^r = \langle d'_1, \dots, d'_k \rangle$
RDF

Fig. 1. Our retrieval-then-reranking approach for RDF dataset search.

2.1 Overview

Figure 1 presents an overview of our approach for RDF dataset search. We follow best practice [17] to adopt a *retrieval-then-reranking* design. Specifically, given a keyword query q and a collection D of RDF datasets, the first step is to perform a *normal retrieval* by using a conventional off-the-shelf method for RDF dataset search to retrieve k top-ranked RDF datasets from D that are the most relevant to q, denoted by $D^s = \langle d_1, \dots, d_k \rangle$. For each retrieved RDF dataset $d_i \in D^s$, the second step is to construct its *compact document representation* to be fed into the downstream dense re-ranking model. We construct two pseudo documents in this step: p_i^m representing the metadata of d_i, and p_i^c representing the content of d_i, i.e., the actual RDF data in d_i. The last step is to employ a dense ranking model to *re-rank* each RDF dataset $d_i \in D^s$ based on the relevance of p_i^m and p_i^c to q, and output the re-ranked results denoted by $D^r = \langle d'_1, \dots, d'_k \rangle$.

The retrieval model in the first step is out of our research focus. In the experiments we will use existing implementations provided in the literature [18]. In the following we will focus on the second and the third steps.

2.2 Compact Document Representation

An RDF dataset contains RDF data and typically has metadata description. Both metadata and RDF data are structured. They need to be linearized into pseudo documents so that they can be processed by the downstream dense re-ranking model. We call them *compact documents* because, relatively to the possibly large size of an RDF dataset (e.g., millions of RDF triples), the length of such a document has to be bounded to fit the maximum input length of the downstream dense model which is usually a small number (e.g., 512 tokens).

Metadata Document. For a retrieved RDF dataset $d_i \in D^s$, we construct its metadata document p_i^m, i.e., a pseudo document representing its metadata.

Specifically, recall that metadata commonly consists of a set of fields. Following [18], we choose four fields that should contain human-readable information and hence are used in the computation of query relevance: `title`, `description`, `tags`, and `author`. The values of these fields are concatenated into p_i^m as illustrated in Fig. 2, where `[CLS]` and `[SEP]` are standard separating tokens used

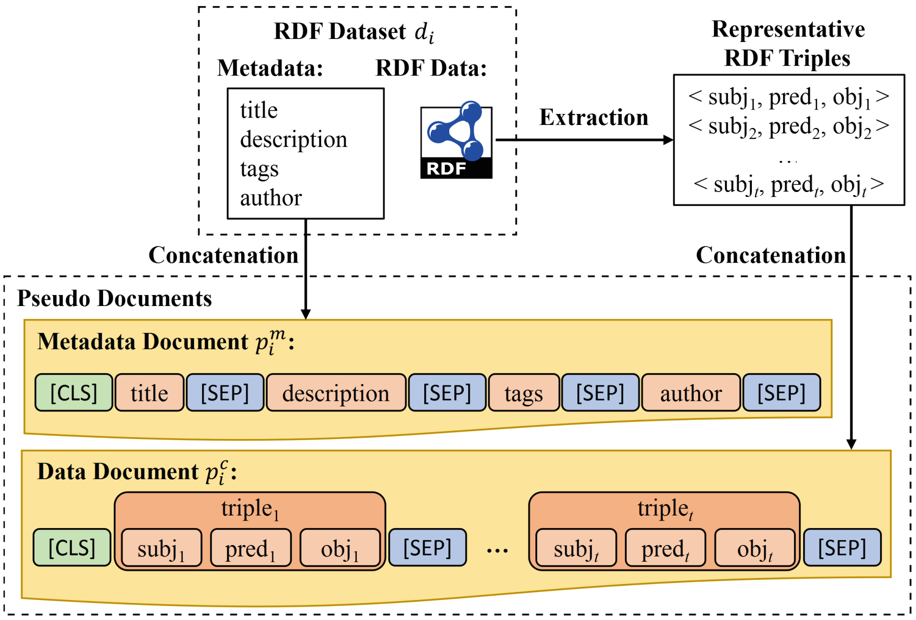

RDF Dataset d_i
Metadata:
RDF Data:
title
description
tags
author
RDF
Extraction
Representative RDF Triples
< subj$_1$, pred$_1$, obj$_1$ >
< subj$_2$, pred$_2$, obj$_2$ >
...
< subj$_t$, pred$_t$, obj$_t$ >
Concatenation
Concatenation
Pseudo Documents
Metadata Document p_i^m:
[CLS] title [SEP] description [SEP] tags [SEP] author [SEP]
Data Document p_i^c:
[CLS] triple$_1$ subj$_1$ pred$_1$ obj$_1$ [SEP] ... triple$_t$ subj$_t$ pred$_t$ obj$_t$ [SEP]

Fig. 2. Compact document representation for an RDF dataset.

in BERT-based models [10]. They can be replaced by their counterparts when other families of language models are used as a substitute for BERT.

Metadata is usually short enough to fit the maximum input length of a dense model. If exceeded, the metadata document will be truncated to the maximum input length in a normal way, i.e., its end will be cut off.

Data Document. For a retrieved RDF dataset $d_i \in D^s$, we construct its data document p_i^c, i.e., a pseudo document representing its RDF data content.

Specifically, recall that RDF data consists of a set of RDF triples. An RDF dataset may easily contain too many triples to fit the maximum input length of a dense model. Indeed, a median of 2k RDF triples was observed in the literature [18]. Instead of performing arbitrary truncation, we want to identify and keep the most important information in RDF data by extracting a representative subset of RDF triples. This resembles the research objective of *RDF dataset snippet generation* [29]. Therefore, we choose and implement two state-of-the-art solutions to this research problem: IlluSnip [8,19] and PCSG [28]. In a nutshell, IlluSnip computes a ranking of the RDF triples in an RDF dataset such that the top-ranked triples cover the most frequent classes, properties, and entities in the data. PCSG relies on entity description pattern (EDP) which is a set of classes and properties used to describe an entity in an RDF dataset. It selects a smallest number of subsets of RDF triples—each subset covering an EDP—such that the selected subsets of triples cover all the EDPs in the data. We further rank these subsets by the frequency of covered EDP.

From a possibly large RDF dataset d_i, subject to the maximum input length of a dense model, we extract the largest possible number of top-ranked RDF triples returned by IlluSnip, or the largest possible number of top-ranked subsets of triples returned by PCSG, depending on which algorithm is used. We will compare IlluSnip and PCSG in the experiments. We concatenate the human-readable forms of the subject, predicate, and object in each extracted RDF triple into p_i^c as illustrated in Fig. 2, separated by `[CLS]` and `[SEP]`. The human-readable form of an IRI or blank node refers to its `rdfs:label` (if available) or local name, and the human-readable form of a literal refers to its lexical form.

Moreover, a single RDF triple may occasionally be very long, e.g., containing a long literal. We need to find a trade-off between the number of extracted RDF triples and the maximum allowed length of a triple. Our implementation empirically truncates each RDF triple to at most 45 tokens because, by sampling RDF datasets used the literature [18], we find that more than 99% of the sampled RDF triples contain at most 45 tokens, i.e., truncation would be very rare in our setting so that the completeness of most triples could be guaranteed.

2.3 Dense Re-Ranking

Given a keyword query q and a set of retrieved k top-ranked RDF datasets D^s, we employ a dense ranking model to re-rank each dataset $d_i \in D^s$ based on the relevance of its metadata document p_i^m and data document p_i^c to q.

Specifically, we consider using dense models for re-ranking because, compared with normal sparse models which rely on *lexical features* for measuring query relevance, dense models are expected to extract *semantic features* from queries and documents to more accurately compute their relevance. We choose and adapt two dense ranking models, DPR [12] and ColBERT [14], because they have been widely used in information retrieval research. In a nutshell, DPR and ColBERT both based on BERT [10] perform sentence-level and token-level semantic matching, respectively. We will compare them in the experiments. They can also be substituted by other dense ranking models [33].

Let $\texttt{DenseRel}(\cdot, \cdot)$ be a dense ranking model, i.e., DPR or ColBERT, which computes the relevance of a document to a keyword query. We calculate the re-ranking score of an RDF dataset d_i by computing the relevance of its metadata document p_i^m to q and the relevance of its data document p_i^c to q, and then take their maximum value:

$$\texttt{Score}(d_i) = \max\{\texttt{DenseRel}(q, p_i^m),\ \texttt{DenseRel}(q, p_i^c)\}\,. \tag{1}$$

3 Coarse Tuning with Weak Supervision

Dense ranking models are supervised. Although they are based on pre-trained language models such as BERT, it is still expected to fine-tune them with task-specific training data to achieve better performance. However, RDF dataset search is a relatively new research problem. The labeled data provided by existing test collections such as [18] may be sufficient for testing but not sufficient for fine-tuning a deep neural model. Therefore, we propose to warm up our dense re-ranking model with *weak supervision*, i.e., with a large set of automatically generated labeled data which, however, possibly contains some noise.

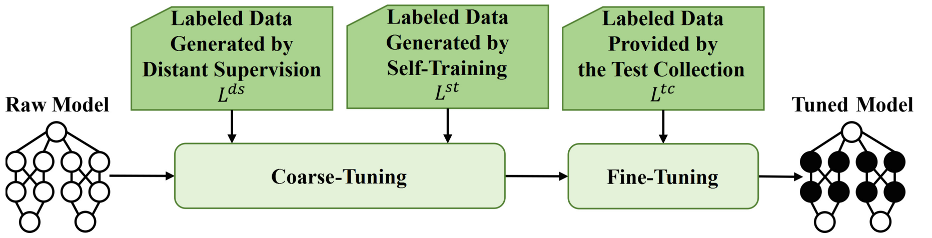

Labeled Data Generated by Distant Supervision L^{ds}
Labeled Data Generated by Self-Training L^{st}
Labeled Data Provided by the Test Collection L^{tc}
Raw Model
Coarse-Tuning
Fine-Tuning
Tuned Model

Fig. 3. Our coarse-to-fine strategy for tuning our dense re-ranking model.

In this section, we begin with an overview of our coarse-to-fine tuning strategy. Then we detail two methods for generating labeled data for coarse-tuning: one based on distant supervision, and the other based on self-training.

3.1 Overview

Figure 3 presents an overview of our strategy for tuning our dense re-ranking model. We adopt a *coarse-to-fine* design which has been popularly used in other tasks but not yet in RDF dataset search. Specifically, given a raw (i.e., untuned) model, in the first step, we employ a large set of automatically generated labeled data to tune the model. Such labeled data is generated by two methods: L^{ds} generated by distant supervision, and L^{st} generated by self-training. They are automatically generated and hence may contain noise, i.e., incorrect labels. Therefore, we refer to this step as *coarse-tuning*. In the second step, we employ the labeled data L^{tc} provided by the training and validation sets of a test collection for RDF dataset search to *fine-tune* the model in a normal way.

Each $L \in \{L^{ds}, L^{st}, L^{tc}\}$ is a set of *query-document-label triples* $\{\langle q, p_i, l\rangle\}$ where q is a keyword query, p_i is a pseudo document, i.e., the metadata document p_i^m or the data document p_i^c constructed for an RDF dataset d_i according to Sect. 2.2, and l is a Boolean label indicating whether d_i is relevant to q. Note that for L^{ds} and L^{st}, we will only focus on the generation of positive labels, i.e., $\langle q, p_i, \texttt{true}\rangle$. It is then common practice to automatically generate negative labels by randomly pairing keyword queries with pseudo documents. For example, DPR [12] is associated with such a built-in generator called in-batch negatives.

3.2 Coarse-Tuning Based on Distant Supervision

Our first method for generating labeled data is inspired by the concept of *distant supervision*, which was originally applied to the task of relation extraction [22]. The idea is that by treating the title of a dataset d_i as a query q, the metadata document p_i^c for d_i should be relevant to q. Therefore, although the availability of labeled data for training RDF dataset search is limited, it is relatively easy to collect metadata for a large number of datasets from the Web, and they are not even restricted to RDF datasets since only their metadata will be used. In this way, a large number of labeled data can be automatically generated.

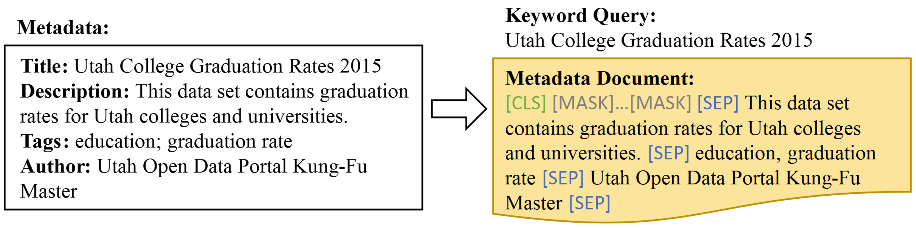

Metadata:
Title: Utah College Graduation Rates 2015
Description: This data set contains graduation rates for Utah colleges and universities.
Tags: education; graduation rate
Author: Utah Open Data Portal Kung-Fu Master
Keyword Query:
Utah College Graduation Rates 2015
Metadata Document:
[CLS] [MASK]...[MASK] [SEP] This data set contains graduation rates for Utah colleges and universities. [SEP] education, graduation rate [SEP] Utah Open Data Portal Kung-Fu Master [SEP]

Fig. 4. An example of generating labeled data by distant supervision.

Specifically, with the metadata of each collected dataset d_i, as illustrated in Fig. 4, we take d_i's title as a query q, and construct d_i's metadata document p_i^m according to Sect. 2.2. In particular, we *mask* the title field in p_i^m because otherwise the relevance of p_i^m to q would be too explicit to be useful when being used in tuning. We mask the title field by replacing each token in this field with `[MASK]` which is a standard masking token used in BERT-based models [10]. Finally, we add the triple $\langle q, p_i^m, \texttt{true}\rangle$ to L^{ds}.

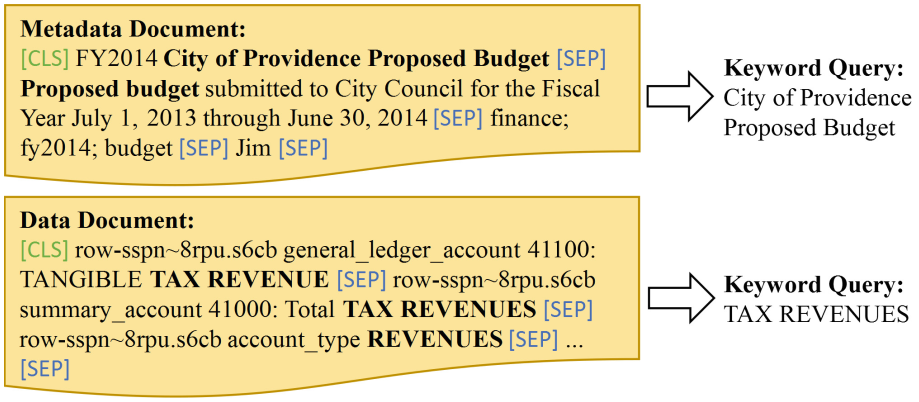
To use L^{ds} to coarse-tune our dense re-ranking model, we randomly split L^{ds} into 90% for training and 10% for validation.

3.3 Coarse-Tuning Based on Self-training

Our second method for generating labeled data adopts a *self-training* design. The idea is to exploit both the labeled and unlabeled data in a test collection for RDF dataset search, by training a *document-to-query generator* on the labeled data and then applying it to generate a query q from the metadata document p_i^m or the data document p_i^c for each unlabeled RDF dataset d_i; these two documents should be relevant to q. Since unlabeled data is often in large amounts in a test collection, e.g., 80% of the RDF datasets in [18] are unlabeled (i.e., not involved in any query-document-label triple in L^{tc}), a large number of labeled data can be automatically generated in this way.

Specifically, recall that L^{tc} denotes the set of labeled data in the training and validation sets of a test collection for RDF dataset search. Let $L^{tc/t/m}, L^{tc/v/m} \subseteq$

L^{tc} be the subsets of labeled data in the training and validation sets, respectively, where query-document-label triples are about metadata documents. Let $L^{tc/t/c}, L^{tc/v/c} \subseteq L^{tc}$ be the subsets of labeled data in the training and validation sets, respectively, where query-document-label triples are about data documents. We separately train two document-to-query generators: $\mathcal{G}^m$ for metadata documents and $\mathcal{G}^c$ for data documents. We reduce document-to-query generation to a *text-to-text generation* task. We train $\mathcal{G}^m$ by employing $L^{tc/t/m}$ as the training set and $L^{tc/v/m}$ as the validation set to fine-tune a T5 model [25], which is a pre-trained text-to-text model. Model selection based on the validation set relies on the ROUGE score, i.e., the mean of ROUGE-1, ROUGE-2, and ROUGE-L, which are standard metrics for evaluating text generation. Then we apply the fine-tuned T5 model as $\mathcal{G}^m$ to the metadata document p_i^m constructed for each unlabeled RDF dataset d_i in the test collection to generate a query q, as illustrated in Fig. 5, and add the triple $\langle q, p_i^m, \texttt{true}\rangle$ to L^{st}. Analogously, we train $\mathcal{G}^c$ on $L^{tc/t/c}$ and $L^{tc/v/c}$, and apply it to the data documents of unlabeled RDF datasets to expand L^{st}.


Metadata Document:
[CLS] FY2014 **City of Providence Proposed Budget** [SEP] **Proposed budget** submitted to City Council for the Fiscal Year July 1, 2013 through June 30, 2014 [SEP] finance; fy2014; budget [SEP] Jim [SEP]

Keyword Query:
City of Providence Proposed Budget

Data Document:
[CLS] row-sspn~8rpu.s6cb general_ledger_account 41100: TANGIBLE **TAX REVENUE** [SEP] row-sspn~8rpu.s6cb summary_account 41000: Total **TAX REVENUES** [SEP] row-sspn~8rpu.s6cb account_type **REVENUES** [SEP] ... [SEP]

Keyword Query:
TAX REVENUES

Fig. 5. An example of generating labeled data by a document-to-query generator.

To use L^{st} to coarse-tune our dense re-ranking model, we use L^{st} as the training set and use the original validation set $L^{tc/v/m} \cup L^{tc/v/c}$ in the test collection as the validation set.

4 Evaluation

4.1 Test Collection

We conducted experiments on ACORDAR [18], the currently largest test collection for RDF dataset search, providing 493 queries and 10,671 labeled relevance judgments over 31,589 RDF datasets. Following [18], we conducted five-fold cross-validation using the official train-valid-test splits provided by ACORDAR.[1]

[1] https://github.com/nju-websoft/ACORDAR.

4.2 Labeled Data for Coarse-Tuning

We implemented the method for generating labeled data L^{ds} based on distant supervision described in Sect. 3.2 by collecting metadata of datasets from open data portals (ODPs). Specifically, to collect metadata for as many datasets as possible, inspired by [18], we found ODPs by looking up five large catalogues: CKAN,[2] DKAN,[3] DataPortals.org,[4] Open Data Portal Watch,[5] and Socrata.[6] We collected all the ODPs listed in these catalogues, and additionally took the Linked Open Data Cloud[7] into account as an ODP. We identified a total of 570 ODPs that were accessible at the time of experimentation. For each ODP, we used its API to download the metadata for all the datasets registered in the ODP. We successfully collected metadata for 704,370 datasets, but had to remove 354 due to their empty titles. We constructed metadata documents for the remaining 704,016 datasets to generate query-document-label triples as L^{ds}.

We implemented the method for generating labeled data L^{st} based on self-training described in Sect. 3.3 by exploiting the labeled and unlabeled data in ACORDAR. Specifically, we trained document-to-query generators on the training and validation sets in ACORDAR, and applied them to the metadata and data documents constructed for the 25,380 unlabeled RDF datasets in ACORDAR to generate queries and form query-document-label triples in L^{st}.

4.3 Implementation Details

Our document-to-query generators were implemented based on the T5-Base model.[8] We searched batch size in $\{8, 16\}$, learning rate in $\{1e{-}6, 5e{-}6, 1e{-}5\}$, and trained 10 epochs. We used the Adam optimizer [15]. We ran T5 on an NVIDIA GeForce RTX 3090 GPU with 24 GB memory.

For DPR and ColBERT in our dense re-ranking model, we used their open-source code.[9][10] For DPR, we trained 1 epoch in the coarse-tuning phase and 10 epochs in the fine-tuning phase, considering the different sizes of training data for different phases. We searched batch size in $\{2, 4\}$ and learning rate in $\{1e{-}5, 2e{-}5\}$. For ColBERT, we followed [14] to train 1 epoch in each phase. We searched batch size in $\{8, 16\}$ and learning rate in $\{1e{-}6, 3e{-}6, 7e{-}6\}$. We used the Adam optimizer. We ran DPR and ColBERT on eight NVIDIA Tesla V100 GPUs with 32 GB memory. Based on Faiss indexes,[11] the mean re-ranking time used by DPR and ColBERT for a query was 62 ms and 139 ms, respectively.

[2] https://ckan.org/.
[3] https://getdkan.org/.
[4] http://dataportals.org/.
[5] https://data.wu.ac.at/portalwatch/.
[6] https://dev.socrata.com/.
[7] http://cas.lod-cloud.net/.
[8] https://huggingface.co/t5-base.
[9] https://github.com/facebookresearch/DPR.
[10] https://github.com/stanford-futuredata/ColBERT.
[11] https://github.com/facebookresearch/faiss.

4.4 Experimental Settings

We evaluated with the following settings of our approach.

For normal retrieval, we directly reused the 10 top-ranked RDF datasets outputted by each of the four sparse retrieval models provided by ACORDAR: TF-IDF, BM25, LMD, and FSDM. TF-IDF (Term Frequency—Inverse Document Frequency) is a weighting scheme giving large weights to locally frequent (i.e., within the current dataset) but globally infrequent (across all datasets) words; based on TF-IDF vector representations, datasets are ranked by their cosine similarity with the query. BM25 is a ranking function that combines term frequency, inverse document frequency, and document length normalization. LMD (Language Model with Dirichlet Smoothing) is a retrieval model that estimates the probability of generating a query given a document, incorporating smoothing techniques to handle unseen terms. FSDM (Fielded Sequential Dependence Model) is a retrieval model for structured document retrieval which considers term dependencies and optimizes document field weights.

For RDF triple extraction in compact document representation, we compared Illusnip and PCSG.

For the dense ranking model in re-ranking, we compared DPR and ColBERT.

For tuning the dense re-ranking model, we compared coarse-tuning based on distant supervision (denoted by `ds`), coarse-tuning based on self-training (denoted by `st`), and normal fine-tuning (denoted by `ft`).

4.5 Evaluation Metrics

Following [18], we used Normalized Discounted Cumulative Gain (NDCG) and Mean Average Precision (MAP). We calculated and reported the mean NDCG@5, NDCG@10, MAP@5, and MAP@10 scores over all the queries.

4.6 Evaluation Results

In the presented results tables, we highlight the best result in each setting **in bold**, and underline the second best result.

Effectiveness of Re-Ranking. As shown in Table 1, re-ranking brings improvements in all the settings, and the improvements in most settings are statistically significant. The best results are achieved with PCSG and ColBERT. Compared with the original sparse retrieval model, this dense re-ranking model raises NDCG@5 by 0.0218–0.0610 (4%–11%), NDCG@10 by 0.0088–0.0326 (1%–6%), MAP@5 by 0.0175–0.0476 (5%–17%), and MAP@10 by 0.0132–0.0363 (3%–9%). In particular, by re-ranking the outputs of FSDM with this model, we obtain the current state-of-the-art results on ACORDAR. The second best results are mostly achieved with IlluSnip and ColBERT. *These results demonstrate the effectiveness of our dense re-ranking for RDF dataset search.*

Table 1. Effectiveness of Re-Ranking (* indicating a significant improvement after re-ranking according to paired t-test under $p < 0.05$)

Retrieval	Re-Ranking	Tuning	NDCG@5	NDCG@10	MAP@5	MAP@10
TF-IDF	before re-ranking	-	0.5088	0.5452	0.2871	0.3976
	Illusnip+DPR	`ds+st+ft`	0.5579*	0.5720*	0.3295*	0.4319*
	Illusnip+ColBERT	`ds+st+ft`	<u>0.5610</u>*	<u>0.5749</u>*	<u>0.3329</u>*	**0.4339***
	PCSG+DPR	`ds+st+ft`	0.5521*	0.5704*	0.3234*	0.4280*
	PCSG+ColBERT	`ds+st+ft`	**0.5659***	**0.5753***	**0.3347***	**0.4339***
BM25	before re-ranking	-	0.5538	0.5877	0.3198	0.4358
	Illusnip+DPR	`ds+st+ft`	0.5888*	0.6082*	0.3481*	0.4592*
	Illusnip+ColBERT	`ds+st+ft`	<u>0.6028</u>*	<u>0.6136</u>*	<u>0.3553</u>*	<u>0.4623</u>*
	PCSG+DPR	`ds+st+ft`	0.5880*	0.6065*	0.3462*	0.4567*
	PCSG+ColBERT	`ds+st+ft`	**0.6079***	**0.6173***	**0.3625***	**0.4680***
LMD	before re-ranking	-	0.5465	0.5805	0.3266	0.4324
	Illusnip+DPR	`ds+st+ft`	0.5959*	0.6055*	0.3563*	0.4571*
	Illusnip+ColBERT	`ds+st+ft`	<u>0.5963</u>*	<u>0.6083</u>*	<u>0.3564</u>*	<u>0.4585</u>*
	PCSG+DPR	`ds+st+ft`	0.5908*	0.6003*	0.3498*	0.4509*
	PCSG+ColBERT	`ds+st+ft`	**0.6075***	**0.6131***	**0.3671***	**0.4654***
FSDM	before re-ranking	-	0.5932	0.6151	0.3592	0.4602
	Illusnip+DPR	`ds+st+ft`	0.6088	<u>0.6204</u>	<u>0.3709</u>	<u>0.4713</u>
	Illusnip+ColBERT	`ds+st+ft`	<u>0.6121</u>	0.6184	0.3677	0.4645
	PCSG+DPR	`ds+st+ft`	0.6061	0.6149	0.3655	0.4637
	PCSG+ColBERT	`ds+st+ft`	**0.6150**	**0.6239**	**0.3767**	**0.4734**

Effectiveness of Coarse-Tuning. For space reasons, Table 2 and Table 3 only show the results obtained with ColBERT. The results with DPR are similar.

As shown in Table 2, compared with fine-tuning (`ft`), adding coarse-tuning brings improvements in almost all the settings. In particular, such improvements are not an effect of longer training since naively increasing the number of epochs in the fine-tuning phase from 1 to 3 ($\texttt{ft}^3$) only brings marginal differences. In some settings, adding coarse-tuning based on distant supervision (`ds+ft`) or self-training (`st+ft`) brings larger improvements than adding both of them (`ds+st+ft`). However, the latter is more robust as it achieves the first or second best result in most settings. Interestingly, for IlluSnip-based re-ranking models, self-training (`st+ft`) generally brings larger improvements than distant supervision (`ds+ft`), whereas for PCSG-based re-ranking models, opposite results are observed. *These results demonstrate the effectiveness of our two coarse-tuning methods which complement fine-tuning and also complement each other.*

As shown in Table 3, coarse-tuning alone based on self-training without fine-tuning (`st`) is generally comparable with fine-tuning (`ft`), *which demonstrates the quality of our augmented labeled data via self-training.* However, there are noticeable gaps between fine-tuning (`ft`) and coarse-tuning alone based on distant supervision without fine-tuning (`ds`), *which shows difficulty in dense re-ranking for RDF dataset search solved as a zero-shot learning task.*

Table 2. Effectiveness of Coarse-Tuning in Complementing Fine-Tuning

Retrieval	Re-Ranking	Tuning	NDCG@5	NDCG@10	MAP@5	MAP@10
TF-IDF	Illusnip+ColBERT	`ft`	0.5530	0.5696	0.3287	0.4307
		`ft`^{3}	0.5558	0.5703	0.3320	0.4320
		`ds+ft`	**0.5632**	**0.5754**	0.3343	0.4343
		`st+ft`	0.5625	0.5746	**0.3368**	**0.4355**
		`ds+st+ft`	0.5610	0.5749	0.3329	0.4339
	PCSG+ColBERT	`ft`	0.5621	0.5738	0.3372	0.4356
		`ft`^{3}	0.5578	0.5716	0.3345	0.4336
		`ds+ft`	**0.5688**	**0.5784**	**0.3379**	**0.4368**
		`st+ft`	0.5566	0.5687	0.3294	0.4297
		`ds+st+ft`	0.5659	0.5753	0.3347	0.4339
BM25	Illusnip+ColBERT	`ft`	0.5976	0.6113	0.3522	0.4603
		`ft`^{3}	0.6027	0.6137	0.3581	0.4640
		`ds+ft`	0.6022	0.6124	0.3544	0.4605
		`st+ft`	**0.6081**	**0.6163**	**0.3602**	**0.4658**
		`ds+st+ft`	0.6028	0.6136	0.3553	0.4623
	PCSG+ColBERT	`ft`	0.6045	0.6163	0.3602	0.4671
		`ft`^{3}	0.5961	0.6099	0.3534	0.4608
		`ds+ft`	**0.6103**	**0.6195**	**0.3636**	**0.4694**
		`st+ft`	0.6047	0.6154	0.3608	0.4672
		`ds+st+ft`	0.6079	0.6173	0.3625	0.4680
LMD	Illusnip+ColBERT	`ft`	0.5918	0.6037	0.3514	0.4544
		`ft`^{3}	0.5978	0.6068	0.3574	0.4574
		`ds+ft`	0.5989	0.6064	0.3582	0.4569
		`st+ft`	**0.6096**	**0.6121**	**0.3666**	**0.4637**
		`ds+st+ft`	0.5963	0.6083	0.3565	0.4585
	PCSG+ColBERT	`ft`	0.6029	0.6077	0.3596	0.4578
		`ft`^{3}	0.5933	0.6051	0.3562	0.4572
		`ds+ft`	0.6067	0.6122	0.3656	0.4636
		`st+ft`	0.6035	0.6104	0.3642	0.4625
		`ds+st+ft`	**0.6075**	**0.6131**	**0.3671**	**0.4654**
FSDM	Illusnip+ColBERT	`ft`	0.5981	0.6119	0.3615	0.4608
		`ft`^{3}	0.6058	**0.6187**	0.3695	**0.4673**
		`ds+ft`	0.6030	0.6134	0.3610	0.4592
		`st+ft`	0.6090	0.6182	**0.3704**	**0.4673**
		`ds+st+ft`	**0.6121**	0.6184	0.3677	0.4645
	PCSG+ColBERT	`ft`	**0.6152**	0.6195	0.3766	0.4695
		`ft`^{3}	0.6064	0.6181	0.3690	0.4662
		`ds+ft`	0.6133	0.6223	0.3732	0.4703
		`st+ft`	0.6128	0.6224	0.3755	0.4725
		`ds+st+ft`	0.6150	**0.6239**	**0.3767**	**0.4734**

Table 3. Effectiveness of Coarse-Tuning in Replacing Fine-Tuning

Retrieval	Re-Ranking	Tuning	NDCG@5	NDCG@10	MAP@5	MAP@10
TF-IDF	Illusnip+ColBERT	ft	0.5530	0.5696	**0.3287**	**0.4307**
		ds	0.4928	0.5344	0.2757	0.3878
		st	**0.5589**	**0.5708**	0.3276	0.4284
	PCSG+ColBERT	ft	**0.5621**	**0.5738**	**0.3372**	**0.4356**
		ds	0.5095	0.5416	0.2844	0.3937
		st	0.5538	0.5680	0.3209	0.4236
BM25	Illusnip+ColBERT	ft	0.5976	**0.6113**	**0.3522**	**0.4603**
		ds	0.5376	0.5752	0.3031	0.4211
		st	**0.5998**	0.6108	0.3517	0.4597
	PCSG+ColBERT	ft	**0.6045**	**0.6163**	**0.3602**	**0.4671**
		ds	0.5495	0.5849	0.3087	0.4275
		st	0.5948	0.6115	0.3484	0.4594
LMD	Illusnip+ColBERT	ft	0.5918	0.6037	0.3514	0.4544
		ds	0.5332	0.5692	0.2993	0.4115
		st	**0.6001**	**0.6072**	**0.3551**	**0.4552**
	PCSG+ColBERT	ft	**0.6029**	0.6077	**0.3596**	**0.4578**
		ds	0.5477	0.5805	0.3128	0.4243
		st	0.5980	**0.6087**	0.3541	0.4560
FSDM	Illusnip+ColBERT	ft	0.5981	0.6119	0.3615	0.4608
		ds	0.5380	0.5807	0.3103	0.4221
		st	**0.6033**	**0.6168**	**0.3667**	**0.4645**
	PCSG+ColBERT	ft	**0.6152**	0.6195	**0.3766**	**0.4695**
		ds	0.5505	0.5878	0.3158	0.4269
		st	0.6079	**0.6212**	0.3666	0.4670

Table 4. Comparison between Triple Extraction Methods

Retrieval	Re-Ranking	NDCG@5	NDCG@10	MAP@5	MAP@10
TF-IDF	Illusnip+∗	**0.5594**	**0.5734**	**0.3312**	**0.4329**
	PCSG+∗	0.5590	0.5729	0.3290	0.4309
BM25	Illusnip+∗	0.5958	0.6109	0.3517	0.4608
	PCSG+∗	**0.5979**	**0.6119**	**0.3543**	**0.4624**
LMD	Illusnip+∗	0.5961	**0.6069**	0.3564	0.4578
	PCSG+∗	**0.5992**	0.6067	**0.3585**	**0.4582**
FSDM	Illusnip+∗	0.6105	**0.6194**	0.3693	0.4679
	PCSG+∗	**0.6106**	**0.6194**	**0.3711**	**0.4686**

Comparison Between Triple Extraction Methods. Table 4 aggregates the results in Table 1 by IlluSnip and PCSG. There is no clear winner between them. In fact, according to Table 1, IlluSnip outperforms PCSG when accompanying DPR, whereas opposite results are observed when accompanying ColBERT, suggesting that extracting representative RDF triples in the context of dataset search deserves to be further studied in the future.

Table 5. Comparison between Dense Ranking Models

Retrieval	Re-Ranking	NDCG@5	NDCG@10	MAP@5	MAP@10
TF-IDF	∗+DPR	0.5550	0.5712	0.3264	0.4299
	∗+ColBERT	**0.5659**	**0.5753**	**0.3347**	**0.4339**
BM25	∗+DPR	0.5884	0.6074	0.3471	0.4580
	∗+ColBERT	**0.6053**	**0.6154**	**0.3589**	**0.4652**
LMD	∗+DPR	0.5934	0.6029	0.3531	0.4540
	∗+ColBERT	**0.6019**	**0.6107**	**0.3618**	**0.4620**
FSDM	∗+DPR	0.6074	0.6177	0.3682	0.4675
	∗+ColBERT	**0.6136**	**0.6211**	**0.3722**	**0.4690**

Table 6. An Example of Top-Ranked RDF Datasets Before and After Re-Ranking (**bold**: highly relevant; underlined: partially relevant)

Keyword Query: nitrogen reduction plan in MaryLand		
	Top-Ranked RDF Datasets by TF-IDF	Re-Ranked by PCSG+ColBERT
1	[ID-46561] Plan Review	[ID-42421] **Percent of required nitrogen reduction achieved: Line Chart**
2	[ID-11683] Plan Review	[ID-41757] Chesapeake Bay Pollution Loads - Nitrogen
3	[ID-86273] January Water Reduction Chart	[ID-03531] Chesapeake Bay Pollution Loads - Nitrogen
4	[ID-08199] Class Size Reduction Projects	[ID-86273] January Water Reduction Chart
5	[ID-03531] Chesapeake Bay Pollution Loads - Nitrogen	[ID-40742] Watershed Contaminant Reduction Index
6	[ID-41757] Chesapeake Bay Pollution Loads - Nitrogen	[ID-08199] Class Size Reduction Projects
7	[ID-07248] 2019 NYC Open Data Plan: Removed Datasets	[ID-79232] Open Publishing Plan Dataset
8	[ID-79232] Open Publishing Plan Dataset	[ID-46561] Plan Review
9	[ID-40742] Watershed Contaminant Reduction Index	[ID-11683] Plan Review
10	[ID-42421] **Percent of required nitrogen reduction achieved: Line Chart**	[ID-07248] 2019 NYC Open Data Plan: Removed Datasets

Comparison Between Dense Ranking Models. Table 5 aggregates the results in Table 1 by DPR and ColBERT. ColBERT consistently outperforms DPR in all the settings, showing its relative suitability for RDF dataset search.

4.7 Case Study

Table 6 illustrates and compares the top-ranked RDF datasets before and after re-ranking for the keyword query "nitrogen reduction plan in MaryLand" which is sampled from our experiments. Before re-ranking, the four top-ranked datasets retrieved by a sparse model (i.e., TF-IDF) are actually irrelevant to the query, although in their metadata, the query keyword "plan" or "reduction" has a misleadingly very high lexical frequency. After re-ranking by our dense model (i.e., PCSG+ColBERT), these datasets fall noticeably, while the three relevant datasets rise to the top. It is expected since they exhibit better semantic matching with the query, which is satisfyingly captured by the dense model.

5 Related Work

5.1 Dataset Search

Researchers have explored various principles and methods for dataset search [5]. For example, Koesten et al. [16] studied the user behavior of seeking structured data on the Web, based on interviews with participants and analyses of search logs. Google Dataset Search [3] is a dataset search engine providing keyword search over reconciled metadata of datasets discovered on the Web.

The Semantic Web community is interested in RDF dataset search, and has developed several prototype systems. LODAtlas [24] allows users to search for RDF datasets and browse a retrieved dataset through a summary visualization. CKGSE [30] supports full-text search over RDF datasets and presents extracted data snippets and summaries. All these systems employ a sparse retrieval model such as BM25. Moreover, Lin et al. [18] constructed the ACORDAR test collection for RDF dataset search and evaluated a set of retrieval methods—all based on sparse models. By contrast, our work is distinguished by *studying dense ranking models* for RDF dataset search and addressing the encountered challenges.

5.2 Dense Ranking

Benefiting from the progress of pre-trained language models like BERT [10], dense ranking models have exhibited higher accuracy than sparse models in document retrieval [12,14]. However, one factor that restricts the application of dense models to boarder tasks is their limited input length of 512 tokens. ANCE [31] addressed it by splitting a document into segments and then pooling segment-level scores, where the semantic dependency among segments was ignored. SeDR [7] used segment interaction to capture document-level representations, but only extended the maximum input length to 2,048 tokens. This capacity still cannot fit the possibly large size of an RDF dataset. Differently, we addressed this challenge by *extracting a subset of representative RDF triples*.

Another factor affecting the performance of dense models is the availability of labeled data for training. Indeed, as reported in [12], accuracy dropped largely after reducing training samples from 59k to 1k. To alleviate this challenge, [11] fine-tuned on a large set of out-of-domain labeled data such as MS MARCO [23], and then transferred the model to the target domain. However, existing labeled data is mainly for document retrieval which differs greatly from RDF dataset search, and our preliminary experiments have confirmed this concern. Therefore, we chose a different direction partially inspired by self-training [21], i.e., we adopted a coarse-to-fine tuning strategy and devised two methods for *generating in-domain (i.e., task-specific) labeled data* for coarse tuning.

6 Conclusion and Future Work

Our exploration of applying dense ranking models to RDF dataset search has brought an improvement of up to 11% in NDCG@5 and 17% in MAP@5 compared with conventional sparse retrieval, considerably pushing the state of the art on the ACORDAR test collection. It represents an encouraging start to connect the task of RDF dataset search with recent advances in pre-trained language models and dense text retrieval. Our empirical findings are expected to expand the understanding of dense dataset search as a promising research pathway, and our code and generated labeled data are shared to facilitate future studies.

As for future work, we identify the following three research directions. First, for compact document representation, it remains unknown whether our selected IlluSnip or PCSG is most suitable for sampling RDF data in the context of dataset search. There are other snippet extraction and data summarization methods [4,9,20,27,29] which deserve to be investigated, and a new specialized method may be more helpful. Besides, beyond simple concatenation, a better way of verbalizing RDF data into a document may also be helpful. Second, contrastive learning is known to be useful for enhancing dense ranking models [31]. However, to boost training, it relies on high-quality (i.e., hard) negative samples [32]. Their generation in the scenario of dataset search is still an open problem. Third, we plan to extend our approach to a more general setting of dataset search going beyond RDF datasets. One major challenge to be overcome is how to encode different formats of data in a universal way. A possible solution is to convert all types of data into graphs [1].

Supplemental Material Statement: Source code for all the dense re-ranking models, their outputs, and all the generated labeled data are available from GitHub at https://github.com/nju-websoft/DR2.

Acknowledgments. This work was supported by the NSFC (62072224).

References

1. Anadiotis, A.G., et al.: Graph integration of structured, semistructured and unstructured data for data journalism. Inf. Syst. **104**, 101846 (2022). https://doi.org/10.1016/j.is.2021.101846
2. Benjelloun, O., Chen, S., Noy, N.F.: Google dataset search by the numbers. In: ISWC 2020, vol. 12507, pp. 667–682 (2020). https://doi.org/10.1007/978-3-030-62466-8_41

3. Brickley, D., Burgess, M., Noy, N.F.: Google dataset search: building a search engine for datasets in an open Web ecosystem. In: WWW 2019, pp. 1365–1375 (2019). https://doi.org/10.1145/3308558.3313685
4. Cebiric, S., Goasdoué, F., Kondylakis, H., Kotzinos, D., Manolescu, I., Troullinou, G., Zneika, M.: Summarizing semantic graphs: a survey. VLDB J. **28**(3), 295–327 (2019). https://doi.org/10.1007/s00778-018-0528-3
5. Chapman, A., Simperl, E., Koesten, L., Konstantinidis, G., Ibáñez, L., Kacprzak, E., Groth, P.: Dataset search: a survey. VLDB J. **29**(1), 251–272 (2020). https://doi.org/10.1007/s00778-019-00564-x
6. Chen, J., Wang, X., Cheng, G., Kharlamov, E., Qu, Y.: Towards more usable dataset search: From query characterization to snippet generation. In: CIKM 2019, pp. 2445–2448 (2019). https://doi.org/10.1145/3357384.3358096
7. Chen, J., Chen, Q., Li, D., Huang, Y.: Sedr: segment representation learning for long documents dense retrieval. CoRR abs/2211.10841 (2022). https://doi.org/10.48550/arXiv.2211.10841
8. Cheng, G., Jin, C., Ding, W., Xu, D., Qu, Y.: Generating illustrative snippets for open data on the Web. In: WSDM 2017, pp. 151–159 (2017). https://doi.org/10.1145/3018661.3018670
9. Cheng, G., Jin, C., Qu, Y.: HIEDS: a generic and efficient approach to hierarchical dataset summarization. In: IJCAI 2016, pp. 3705–3711 (2016)
10. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: NAACL-HLT 2019, vol. 1, pp. 4171–4186 (2019). https://doi.org/10.18653/v1/n19-1423
11. Izacard, G., et al.: Unsupervised dense information retrieval with contrastive learning. CoRR abs/2112.09118 (2021). 10.48550/arXiv.2112.09118
12. Karpukhin, V., et al.: Dense passage retrieval for open-domain question answering. In: EMNLP 2020, pp. 6769–6781 (2020). https://doi.org/10.18653/v1/2020.emnlp-main.550
13. Kato, M.P., Ohshima, H., Liu, Y., Chen, H.: A test collection for ad-hoc dataset retrieval. In: SIGIR 2021, pp. 2450–2456 (2021). https://doi.org/10.1145/3404835.3463261
14. Khattab, O., Zaharia, M.: ColBERT: efficient and effective passage search via contextualized late interaction over BERT. In: SIGIR 2020, pp. 39–48 (2020). https://doi.org/10.1145/3397271.3401075
15. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: ICLR 2015 (2015)
16. Koesten, L.M., Kacprzak, E., Tennison, J.F.A., Simperl, E.: The trials and tribulations of working with structured data - a study on information seeking behaviour. In: CHI 2017, pp. 1277–1289 (2017). https://doi.org/10.1145/3025453.3025838
17. Lin, J., Nogueira, R.F., Yates, A.: Pretrained Transformers for Text Ranking: BERT and Beyond. Synthesis Lectures on Human Language Technologies, Morgan & Claypool Publishers, San Rafael (2021). https://doi.org/10.2200/S01123ED1V01Y202108HLT053
18. Lin, T., et al.: ACORDAR: a test collection for ad hoc content-based (RDF) dataset retrieval. In: SIGIR 2022, pp. 2981–2991 (2022). https://doi.org/10.1145/3477495.3531729
19. Liu, D., Cheng, G., Liu, Q., Qu, Y.: Fast and practical snippet generation for RDF datasets. ACM Trans. Web **13**(4), 19:1–19:38 (2019). https://doi.org/10.1145/3365575

20. Liu, Q., Cheng, G., Gunaratna, K., Qu, Y.: Entity summarization: state of the art and future challenges. J. Web Semant. **69**, 100647 (2021). https://doi.org/10.1016/j.websem.2021.100647
21. Luo, H., Li, S., Gao, M., Yu, S., Glass, J.R.: Cooperative self-training of machine reading comprehension. In: NAACL 2022, pp. 244–257 (2022). https://doi.org/10.18653/v1/2022.naacl-main.18
22. Mintz, M., Bills, S., Snow, R., Jurafsky, D.: Distant supervision for relation extraction without labeled data. In: ACL 2009, pp. 1003–1011 (2009)
23. Nguyen, T., et al.: MS MARCO: a human generated machine reading comprehension dataset. In: CoCo 2016, vol. 1773 (2016)
24. Pietriga, E., et al.: Browsing linked data catalogs with LODAtlas. In: ISWC 2018, pp. 137–153 (2018). https://doi.org/10.1007/978-3-030-00668-6_9
25. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. **21**, 140:1-140:67 (2020)
26. Robertson, S., Zaragoza, H.: The probabilistic relevance framework: Bm25 and beyond. Found. Trends Inf. Retr. **3**(4), 333–389 (2009). https://doi.org/10.1561/1500000019
27. Wang, X., Cheng, G., Kharlamov, E.: Towards multi-facet snippets for dataset search. In: PROFILES & SEMEX 2019, pp. 1–6 (2019)
28. Wang, X., et al.: PCSG: pattern-coverage snippet generation for RDF datasets. In: ISWC 2021, pp. 3–20 (2021). https://doi.org/10.1007/978-3-030-88361-4_1
29. Wang, X., Cheng, G., Pan, J.Z., Kharlamov, E., Qu, Y.: BANDAR: benchmarking snippet generation algorithms for (RDF) dataset search. IEEE Trans. Knowl. Data Eng. **35**(2), 1227–1241 (2023). https://doi.org/10.1109/TKDE.2021.3095309
30. Wang, X., Lin, T., Luo, W., Cheng, G., Qu, Y.: CKGSE: a prototype search engine for chinese knowledge graphs. Data Intell. **4**(1), 41–65 (2022). https://doi.org/10.1162/dint_a_00118
31. Xiong, L., et al.: Approximate nearest neighbor negative contrastive learning for dense text retrieval. In: ICLR 2021 (2021). https://openreview.net/forum?id=zeFrfgyZln
32. Zhan, J., Mao, J., Liu, Y., Guo, J., Zhang, M., Ma, S.: Optimizing dense retrieval model training with hard negatives. In: SIGIR 2021, pp. 1503–1512 (2021). https://doi.org/10.1145/3404835.3462880
33. Zhao, W.X., Liu, J., Ren, R., Wen, J.: Dense text retrieval based on pretrained language models: a survey. CoRR abs/2211.14876 (2022). https://doi.org/10.48550/arXiv.2211.14876