DUNKS: Chunking and Summarizing Large and Heterogeneous Data for Dataset Search

Qiaosheng Chen(✉), Xiao Zhou, Zhiyang Zhang, and Gong Cheng

State Key Lab for Novel Software Technology, Nanjing University, Nanjing, China
{qschen,xzhou,zhiyangzhang}@smail.nju.edu.cn, gcheng@nju.edu.cn

Abstract. With the vast influx of open data on the Web, dataset search has become a trending research problem which is crucial to data discovery and reuse. Existing methods for dataset search either employ only the unstructured metadata of datasets but ignore their actual data, or cater to structured data in a single format such as RDF despite the diverse formats of open data. In this paper, to address the magnitude of large datasets, we decompose RDF data into data chunks, and then, to accommodate big chunks to the limited input capacity of dense ranking models based on pre-trained language models, we propose a multi-chunk summarization method that extracts representative data from representative chunks. Moreover, to handle heterogeneous data formats beyond RDF, we transform other formats into chunks to be processed in a uniform way. Experiments on two test collections for dataset search demonstrate the effectiveness of our dense ranking over summarized data chunks.

1 Introduction

The rise of big data has opened a transformative era characterized by unprecedented access to a vast volume of information. This advancement has expedited the creation and publication of open datasets spanning diverse domains. Within this data-driven landscape, *dataset search engine* has emerged as an indispensable tool catering to the needs of researchers, analysts, and the general public, enabling them to find data encompassing fields such as science, politics, and encyclopedic knowledge [8]. The intricacies inherent to the composition of datasets, coupled with the unique requirements exhibited by users when seeking data, set dataset search apart from the realm of conventional document search and underscore the challenges associated with this underexplored task [22].

Motivation. Existing dataset search engines like Google Dataset Search [5] rely on matching keyword queries with the *unstructured metadata* of datasets. Recent studies have revealed the necessity of including the *actual structured data* in dataset search [11,29,44] since metadata-based search can hardly serve queries referring to specific data content. The Semantic Web community have accordingly developed data-based RDF dataset search engines such as LODAtlas [40] and CKGSE [48]. Still, data-based search remains a non-trivial undertaking due

G. Demartini et al. (Eds.): ISWC 2024, LNCS 15232, pp. 78–97, 2025.
https://doi.org/10.1007/978-3-031-77850-6_5

to the following two challenges. (1) *Data Magnitude*: Datasets typically surpass conventional documents in size, e.g., RDF datasets comprising millions of triples. Existing solutions predominantly apply sparse retrieval models like BM25 to the actual data [29], while potentially more effective dense ranking models grounded on pre-trained language models (PLMs) [53] tend to grapple with their input length limitation (e.g., 512 tokens) resulting in the under-utilization of only a small fraction of data such as a single extracted snippet of RDF data [10], thus significantly limiting the findability of large datasets. (2) *Data Heterogeneity*: Open datasets offer data in multiple formats. Existing methods, by contrast, are constrained to handling a single format such as tabular [11,44] or RDF data [10,29]. A more universal solution applicable to multiple data formats has not been explored in the field of dense ranking for dataset search.

Our Work. To address the challenges posed by data magnitude and heterogeneity, we introduce DUNKS, an approach for chunking and summarizing large multi-format data to facilitate applying dense ranking models to data-based dataset search. To address data magnitude, our method begins with a *chunking technique for RDF data decomposition*, followed by a novel *multi-chunk summarization method* that reduces both the number and size of data chunks by selecting representative triples from representative chunks in terms of reflecting the schema patterns of the original data. This summarization enhances ranking efficiency and ensures compatibility with the input length limitations of PLMs, so our method enables the use of PLM-based dense models to rank datasets over their summarized data chunks. To address data heterogeneity, we establish practical mapping rules that *transform various mainstream data formats into chunks*, enabling uniform processing across diverse datasets. With these challenges addressed, we adapt dense ranking models to data-based dataset search over multiple formats, and our extensive experiments demonstrate that DUNKS exhibits superior ranking accuracy on two public test collections.

Our core contributions are summarized as follows.

- To adapt dense ranking models to large RDF data, we design a novel pattern-oriented multi-chunk summarization method which compactly extracts representative subsets of chunked data. It outperforms existing methods for RDF data summarization in facilitating data-based dataset search.
- With practical rule-based data transformation, we extend our approach to heterogeneous formats beyond RDF. This effort, in the realm of data-based dataset search, represents a pioneering practice of dense ranking over multi-format data, and it achieves new state-of-the-art results.

2 Problem Statement

A dataset has the following components. Its *metadata* includes descriptive fields provided by the publisher, typically encompassing title, description, author, and other fields. The actual *data* is a set of data files, which we categorize into *structured data files* adhering to specific formats including graph data (RDF,

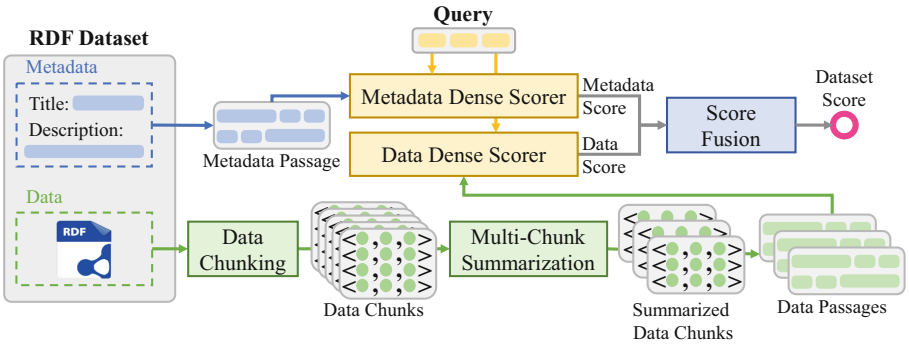


Fig. 1. Overview of our DUNKS approach for RDF dataset search.

OWL), tabular data (CSV, XLS), and key-value pair data (JSON, XML), and *unstructured data files* mainly consisting of text (TXT, PDF, DOC, HTML).

In this paper, we study *ad hoc dataset retrieval* [20,29], the most fundamental form of dataset search. It involves retrieving, from a collection D of datasets, a list of top-ranked datasets that are most relevant to a keyword query q, denoted by $\langle d_1, d_2, \ldots \rangle$. The primary task here is to compute the relevance score of each dataset $d \in D$ to the query q, relying on both the metadata and data of d.

3 DUNKS for RDF Dataset Search

3.1 Overview

Figure 1 outlines our DUNKS approach for RDF dataset search. Recall that an RDF dataset is anatomized into its metadata and the actual RDF data. Field values in the metadata (e.g., title, description) are directly concatenated into a textual passage, which is then fed into a PLM-based dense scorer.

For addressing the magnitude of RDF data, we firstly decompose RDF data into entity-centered data chunks serving as the basic processing units for the subsequent steps (Sect. 3.2). Then we summarize the resulting set of chunks, with the objective of preserving representative chunks and representative triples within each chunk, while reducing both the number and size of chunks, respectively (Sect. 3.3). Summarized data chunks are converted into textual passages by concatenating their triple elements, and then fed into a dense scorer.

Finally, we employ both the metadata passage and data passages to perform dense scoring of the dataset (Sect. 3.4). This culmination yields a relevance score that properly merges metadata score and data score by a fusion function, providing a signal either for directly ranking datasets or for reranking the top-ranked datasets retrieved by an off-the-shelf model such as BM25.

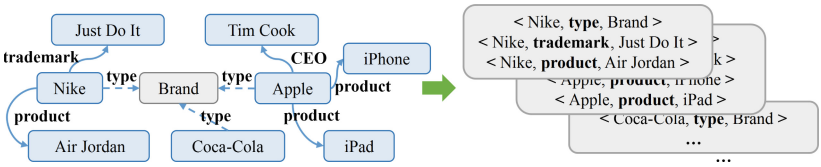


Fig. 2. An example of data chunking for RDF data.

3.2 Data Chunking

We decompose the RDF data in a dataset into *data chunks*, denoted by $\mathcal{C}$. Each chunk $C_i \in \mathcal{C}$ is a set $C_i = \{t_1, t_2, \ldots\}$ where each $t_j = \langle t_j^{\text{subj}}, t_j^{\text{pred}}, t_j^{\text{obj}} \rangle$ is a subject-predicate-object triple. All the triples $t_j \in C_i$ in a chunk describe a common *core entity* e such that $t_j^{\text{subj}} = e$ or $t_j^{\text{obj}} = e$.

Specifically, as illustrated in Fig. 2, we designate each instance-level non-literal node in the RDF graph as a core entity e, and convert its neighborhood into a chunk. Each triple t in the chunk takes e as t^{subj} (or t^{obj}), and represents an outgoing (or incoming) edge of e where the edge label becomes t^{pred} and the other endpoint becomes t^{obj} (or t^{subj}).

3.3 Multi-chunk Summarization

To reduce the scale of potentially many and large chunks $\mathcal{C}$ obtained in Sect. 3.2, we extract a compact yet representative *multi-chunk summary* denoted by $\mathcal{S}^*$, which preserves at most n chunks, and at most k triples in each chunk. These two parameters can be set empirically. Preserving more chunks might improve the findability of a large dataset but also influence the efficiency of the subsequent ranking model, so the choice of n depends on the available computational resources. The choice of k depends on the input length limitation of the dense ranking model since each time the model will be fed with a textual passage converted from all the triples preserved in a chunk.

Basic Idea. Given a set $\mathcal{C}$ of chunks, we perform a *two-stage process* for multi-chunk summarization. The first stage selects a subset of chunks $\mathcal{S} \subseteq \mathcal{C}$, and the second stage selects a subset of triples $C_i^* \subseteq C_i$ from each chunk $C_i \in \mathcal{S}$ to compose the final multi-chunk summary $\mathcal{S}^*$.

In both stages, we seek to maximize the resulting summary's *representativeness of the schema patterns in the original data* [46,47] because they provide a concise data overview which is useful for data-based dataset search. Accordingly, our goal is to generate a summary that covers the schema information in the original chunks as much as possible, so we formulate both stages as two different instances of the *weighted maximum coverage* (WMC) problem to be solved.

Formally, given a universe U of elements where each element $e \in U$ has a weight $\mathtt{w}(e)$, a collection $\mathcal{V}$ of subsets of U, and a number constraint l, a WMC instance requires selecting at most l of these subsets such that the weighted sum

of their collectively covered elements is maximized. For this NP-hard optimization problem, an approximate solution can be efficiently computed by a greedy algorithm [16], which guarantees an approximation ratio of $1 - \frac{1}{e}$.

Stage 1: Chunk Selection. The predicates in a chunk C_i, denoted by $\texttt{P}(C_i)$, are the elements that most directly signify the schema characteristics of C_i:

$$\texttt{P}(C_i) = \left\{ t^{\text{pred}} : t \in C_i \right\} . \tag{1}$$

Let T be the set of all triples in the original chunks $\mathcal{C}$:

$$T = \bigcup_{C_i \in \mathcal{C}} C_i . \tag{2}$$

The importance of a predicate $p \in \texttt{P}(C_i)$, denoted by $\texttt{pf}(p)$, is quantified by its triple-level *relative frequency* in T:

$$\texttt{pf}(p) = \frac{\left| \left\{ t \in T : t^{\text{pred}} = p \right\} \right|}{|T|} . \tag{3}$$

Beyond individual predicates, we also consider the holistic pattern reflected by a chunk, which provides a higher-order view of its schema characteristics. Specifically, inspired by [46], we define the *Entity Description Pattern* (EDP) of the core entity e in a chunk C_i, denoted by $\texttt{EDP}(C_i)$, as a tuple consisting of the forward predicates $\texttt{FP}(C_i)$ and backward predicates $\texttt{BP}(C_i)$ in C_i:

$$\begin{aligned} &\texttt{EDP}(C_i) = \langle \texttt{FP}(C_i), \texttt{BP}(C_i) \rangle , \\ &\texttt{FP}(C_i) = \left\{ p \in \texttt{P}(C_i) : \exists t \in T,\ t^{\text{subj}} = e,\ t^{\text{pred}} = p \right\} , \\ &\texttt{BP}(C_i) = \left\{ p \in \texttt{P}(C_i) : \exists t \in T,\ t^{\text{obj}} = e,\ t^{\text{pred}} = p \right\} . \end{aligned} \tag{4}$$

Similar to Eq. (3), chunk-level *relative frequency* is employed to quantify the importance of $\texttt{EDP}(C_i)$, denoted by

$$\texttt{ef}(\texttt{EDP}(C_i)) = \frac{|\{C_j \in \mathcal{C} : \texttt{EDP}(C_j) = \texttt{EDP}(C_i)\}|}{|\mathcal{C}|} . \tag{5}$$

By slightly abusing the notation, for all the predicates and EDPs in $\mathcal{C}$:

$$\texttt{P}(\mathcal{C}) = \bigcup_{C_i \in \mathcal{C}} \texttt{P}(C_i) , \quad \texttt{EDP}(\mathcal{C}) = \bigcup_{C_i \in \mathcal{C}} \{\texttt{EDP}(C_i)\} , \tag{6}$$

their union $\texttt{P}(\mathcal{C}) \cup \texttt{EDP}(\mathcal{C})$ comprises the universe of elements to be covered by selected chunks that exhibit these predicates and EDPs, where the weight of each predicate or EDP x, denoted by $\texttt{w}(x)$, is given by its relative frequency $\texttt{pf}(x)$ or $\texttt{ef}(x)$ representing its importance, respectively.

For each candidate chunk $C_i \in \mathcal{C}$, the predicates in C_i and the EDP of the core entity in C_i are collectively denoted by $\texttt{Cov}(C_i)$:

$$\texttt{Cov}(C_i) = \texttt{P}(C_i) \cup \{\texttt{EDP}(C_i)\} . \tag{7}$$

Table 1. Formulation of chunk selection and triple selection as WMC instances.

WMC	Chunk Selection	Triple Selection
universe U of elements	$\mathtt{P}(\mathcal{C}) \cup \mathtt{EDP}(\mathcal{C})$	$\mathtt{P}(C_i)$
element weight $\mathtt{w}(e)$	$\mathtt{w}(x) = \begin{cases} \mathtt{pf}(x), x \in \mathtt{P}(\mathcal{C}) \\ \mathtt{ef}(x), x \in \mathtt{EDP}(\mathcal{C}) \end{cases}$	$\mathtt{w}(p) = \mathtt{pf}(p)$
collection $\mathcal{V}$ of subsets	$\{\mathtt{Cov}(C_i) : C_i \in \mathcal{C}\}$	$\left\{\{t^{\text{pred}}\} : t \in C_i\right\}$
number constraint l	n	k

It represents the subset of elements that are covered by selecting C_i.

By solving the above WMC instance, we essentially *retain the most frequent predicates and EDPs over the original chunks* with $l = n$ selected representative chunks $\mathcal{S} \subseteq \mathcal{C}$. The full formulation is summarized in Table 1.

Stage 2: Triple Selection. For each selected chunk $C_i \in \mathcal{S}$, we reduce its size also by formulating and solving a WMC instance to *retain as many predicates in the original chunk as possible* with $l = k$ selected representative triples $C_i^* \subseteq C_i$. This ensures *the maximum retention of the EDP of the core entity* in C_i. The full formulation is summarized in Table 1.

Specifically, here, the universe of elements to be covered by selected triples is the set of all predicates in C_i, i.e., $\mathtt{P}(C_i)$, where the weight of each predicate p, denoted by $\mathtt{w}(p)$, is given by its relative frequency $\mathtt{pf}(p)$ representing its importance according to Eq. (3). For each candidate triple $t \in C_i$, the singleton set $\{t^{\text{pred}}\}$ represents the set of elements that are covered by selecting t.

In addition, when a chunk contains multiple triples having the same predicate, we employ the relative frequency of predicate value to break ties. Formally, the triple-level *relative frequency* of a value v of predicate p is given by

$$\mathtt{vf}_p(v) = \frac{\left|\left\{t \in T : t^{\text{pred}} = p \wedge \left(t^{\text{subj}} = v \vee t^{\text{obj}} = v\right)\right\}\right|}{\left|\{t \in T : t^{\text{pred}} = p\}\right|}. \tag{8}$$

When triples share the same predicate, those with values of higher relative frequency are prioritized for selection in the greedy algorithm for WMC.

3.4 Dataset Scoring

Now we are ready to convert each dataset into a set of textual passages.

- All the metadata fields are concatenated into a metadata passage p^{meta}.
- For each RDF data file, having computed its multi-chunk summary $\mathcal{S}^*$, each summarized chunk $C_i^* \in \mathcal{S}^*$ is converted into a passage p_i^{data} by concatenating the human-readable forms (e.g., `rdfs:label`, local name, lexical form of literal) of the subject, predicate, and object in each triple $t \in C_i^*$ in the order of triple selection in the greedy algorithm. With the union of these passages from RDF data, we obtain a set of data passages $P^{\text{data}} = \{p_i^{\text{data}} : C_i^* \in \mathcal{S}^*\}$.

Given a keyword query q and a collection D of datasets to be (re)ranked, we employ a *metadata dense scorer* $\texttt{Rel}^{\text{meta}}(\cdot,\cdot)$ and a *data dense scorer* $\texttt{Rel}^{\text{data}}(\cdot,\cdot)$ to compute the relevance score of each candidate dataset $d \in D$ based on its metadata and data, respectively:

$$\begin{aligned} \texttt{score}^{\text{meta}}(q,d) &= \texttt{Rel}^{\text{meta}}(q, p^{\text{meta}}) \,, \\ \texttt{score}^{\text{data}}(q,d) &= \max_{p_i^{\text{data}} \in P^{\text{data}}} \left\{ \texttt{Rel}^{\text{data}}(q, p_i^{\text{data}}) \right\} \,, \end{aligned} \tag{9}$$

where we follow common practice in multi-passage retrieval to perform MaxP [13] which takes the highest-scored data passage to represent all the data passages.

Finally, we utilize a *normalization strategy* $\varphi(\cdot)$ (e.g., min-max normalization) and a *fusion algorithm* $\phi(\cdot)$ (e.g., weighted sum) to combine $\texttt{score}^{\text{meta}}(q,d)$ and $\texttt{score}^{\text{data}}(q,d)$ as the final score of d:

$$\texttt{score}(q,d) = \phi(\varphi(\texttt{score}^{\text{meta}}(q,d)), \varphi(\texttt{score}^{\text{data}}(q,d))) \,. \tag{10}$$

The concrete implementation of the dense scorers $\texttt{score}^{\text{meta}}$ and $\texttt{score}^{\text{data}}$ (i.e., dense ranking models), the normalization strategy φ, and the fusion algorithm ϕ used in our experiments will be described in Sect. 5.

4 Extension to Multi-format Dataset Search

4.1 Overview

To handle heterogeneously formatted data, we extend our approach to other mainstream data formats including unstructured and structured data.

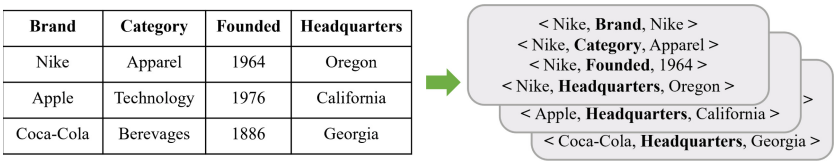
For unstructured data, we remove non-textual data such as images. Aligning with the common practice in the field of document retrieval, we segment the remaining text into passages of fixed length (500 tokens in our implementation). These passages are directly fed as data passages into the data dense scorer.

For structured data other than RDF, we define mapping rules (Sect. 4.2) to transform it into chunks to be processed in the same way as RDF data.

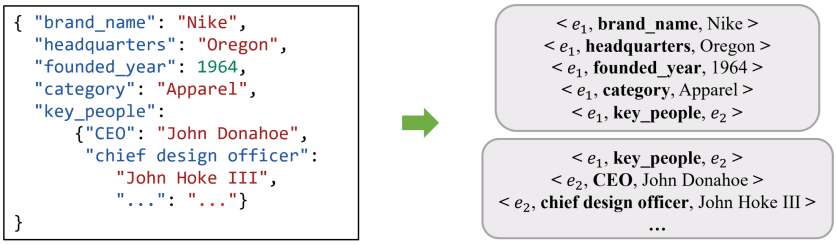
4.2 Unified Data Chunking

We define mapping rules for tabular data and key-value pair data because these formats together with RDF and unstructured textual data constitute more than 80% of real-world open data [4], with the remainder primarily consisting of non-textual formats such as images and videos. Figure 3 shows examples of mappings from tabular data and key-value pair data to chunks. Our mapping rules take into account the unique feature of each data format. They are empirically defined to be practical for the majority of open data according to our observation.

Tabular Data. For a tabular data file, each row (not including the row of column headers) is transformed into a chunk. We assume that the first cell in the row represents the core entity e in the chunk. For every other non-empty cell



(a) Mapping from tabular data to chunks.



(b) Mapping from key-value pair data to chunks.

Fig. 3. Examples of mappings from heterogeneously formatted data to chunks.

in the row, we create a triple $t = \langle t^{\text{subj}}, t^{\text{pred}}, t^{\text{obj}} \rangle$ with e as t^{subj}, the column header as t^{pred}, and the cell content as t^{obj}.

Key-Value Pair Data. Since a key-value pair data file lacks a natural core entity, we firstly create a core entity e, and transform all the top-level $\langle key, value \rangle$ pairs into a chunk where e serves as t^{subj}, key as t^{pred}, and $value$ as t^{obj} of a triple t. Note that some key-value pair data formats such as JSON support nesting, i.e., $value$ itself may consist of a set of key-value pairs. In this case, the $value$ is represented as an entity e' which not only appears as the object of a triple in the current chunk, but also becomes the core entity of a new chunk containing triples representing the nested key-value pairs. This transformation process for nested key-value pairs is recursive.

5 Experiments on RDF Datasets

5.1 Experimental Setup

Test Collection. We used **ACORDAR** [29],[1] a test collection for data-based ad hoc RDF dataset retrieval. It includes 31,589 RDF datasets, 493 queries consisting of an average of 3.9 words, and 10,671 annotated query-dataset relevance judgments. We followed its five-fold cross-validation setup, each fold using 60% of the queries for training, 20% for validation, and 20% for testing.

First-Stage Retrieval Results. Following [10], we evaluated reranking performance, i.e., we applied our approach to rerank the top-10 results of first-stage retrieval returned by BM25. We directly reused the retrieval results of weighted

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

Table 2. Results of different normalization strategies and fusion algorithms on the validation set of ACORDAR.

Normalization Strategy	NDCG@5	NDCG@10	Fusion Algorithm	NDCG@5	NDCG@10
Min-Max	**0.6036**	**0.6163**	Mixed	**0.6036**	**0.6163**
Max	0.6035	0.6147	Min	0.5815	0.6066
ZMUV	0.6035	0.6160	Max	0.5942	0.6109
Rank	0.5981	0.6125	Sum	0.5967	0.6118
Borda	0.5987	0.6128	RRF	0.5884	0.6059

BM25 provided by ACORDAR, which were obtained by searching over both metadata fields and data elements (i.e., RDF resources). We also experimented with other first-stage retrievers like FSDM and drew conclusions similar to those with BM25, so we omitted those results due to space limitations.

Evaluation Metrics. Following [29], we used Normalized Discounted Cumulative Gain (**NDCG**) and Mean Average Precision (**MAP**) at positions 5, 10.

5.2 Participating Methods

Baselines. We compared with **FSDM** [54], the leading sparse ranking model on ACORDAR [29], and DR^2 [10] which held the currently best results on ACORDAR by reranking the results of BM25 with ColBERT [21] fed with extracted snippets of RDF data and tuned with distant supervision and self-training.

Our Approach. For the dense scorers in our approach, we adapted three ranking models: **BGE**-en-large-v1.5 [49],[2] an open-sourced dual-encoder model exhibiting state-of-the-art performance on the MTEB benchmark [33]; **BGE-reranker**-large,[3] a cross-encoder version of BGE-en-large-v1.5; and **monoBERT** (large) [37],[4] a popular cross-encoder model pre-trained on MS-MARCO [36]. We also followed [43] to use a large language model (LLM), **GLM-4** [52],[5] to zero-shot rerank datasets based on their metadata and data passages.

5.3 Implementation Details

Model Fine-Tuning. When fine-tuning dense ranking models on a labeled dataset with multiple data passages, we followed [19] to choose the passage having the highest BM25 score as the representative for the dataset during training. Regarding the hyperparameters, we searched batch size in $\{4, 8\}$ for BGEs, and in $\{8, 16\}$ for monoBERT. For all these models we consistently searched learning rate in {1e-5, 5e-5, 1e-4} and set 5 epochs. We used Tesla V100 GPUs for fine-tuning and GeForce RTX 3090 GPUs for inference.

[2] https://huggingface.co/BAAI/bge-large-en-v1.5.
[3] https://huggingface.co/BAAI/bge-reranker-large.
[4] https://huggingface.co/castorini/monobert-large-msmarco.
[5] https://open.bigmodel.cn/dev/api#glm-4.

Table 3. Main results on ACORDAR, with * indicating a significant improvement after reranking ($p < 0.05$).

First-Stage Retrieval	Reranking	NDCG@5	NDCG@10	MAP@5	MAP@10
FSDM	-	0.5932	0.6151	0.3592	0.4602
BM25	-	0.5538	0.5877	0.3198	0.4358
BM25	DR^2	0.6079*	0.6173*	0.3625*	0.4680*
BM25	DUNKS + BGE	**0.6306***	**0.6322***	0.3763*	0.4803*
BM25	DUNKS + BGE-reranker	0.6303*	0.6309*	**0.3766***	**0.4804***
BM25	DUNKS + monoBERT	0.6193*	0.6233*	0.3671*	0.4729*
BM25	DUNKS + GLM-4	0.6150*	0.6198*	0.3596*	0.4656*

Table 4. Results of ablation study on ACORDAR.

Ranking Model	Data Processing	NDCG@5	NDCG@10	MAP@5	MAP@10	Summarization Time (s)
BGE	w/o data	0.6261	0.6294	0.3743	0.4786	-
	DUNKS	**0.6306**	**0.6322**	**0.3763**	**0.4803**	0.345
	IlluSnip	0.6305	0.6291	0.3745	0.4768	0.929
	PCSG	0.6234	0.6269	0.3718	0.4762	1.702
BGE-reranker	w/o data	0.6133	0.6200	0.3607	0.4672	-
	DUNKS	**0.6303**	**0.6309**	**0.3766**	**0.4804**	0.345
	IlluSnip	0.6154	0.6215	0.3618	0.4684	0.929
	PCSG	0.6135	0.6206	0.3606	0.4674	1.702

Score Normalization and Fusion. We used the ranx library [2,3][6] for normalizing and fusing metadata-based and data-based relevance scores in Eq. (10). Among various implementations of normalization and fusion provided by ranx, we finally selected Min-Max normalization and Mixed fusion because they achieved the best results on the validation set, as shown in Table 2. The coefficients in score fusion were tuned on the validation set in 0.1 increments using NDCG@5 as the optimization objective.

Parameter Settings in Summarization. To balance ranking accuracy with time cost, we set $n = 100$ to select at most 100 chunks from each dataset. Regarding the maximum number of triples retained in each chunk, we set $k = 20$ to exploit our PLMs' input length limitation of 512 tokens. With these settings, our summarization retained an average of 5,700 out of 72,301 tokens per dataset. We will compare different settings of these parameters in Sect. 6.4.

5.4 Experimental Results

Main Results: Effectiveness of Dataset Reranking. As shown in Table 3, *our reranking approach exhibits significant improvements over first-*

[6] https://github.com/AmenRa/ranx.

stage retrieval of RDF datasets. The three fine-tuned dense reranking models consistently boost NDCG@5 by 0.0655–0.0768 (12%–14%), NDCG@10 by 0.0356–0.0445 (6%–8%), MAP@5 by 0.0473–0.0568 (15%–18%), and MAP@10 by 0.0371–0.0446 (9%–10%). They also noticeably surpass the state-of-the-art FSDM and DR^2. For example, our DUNKS with BGE outperforms DR^2 by 0.0227 (4%) in NDCG@5. Our LLM-based zero-shot reranking with GLM-4 also significantly improves over first-stage retrieval, though lower than fine-tuned dense reranking.

Ablation Study: Comparison with Metadata-Only Methods. As shown in Table 4, we conducted an ablation study comparing our approach with its variant that ignores data passages (denoted by w/o data), i.e., only using the metadata passage in reranking. After excluding data passages, NDCG and MAP scores fall consistently. For example, for BGE-reranker, its NDCG@5 decreases by 0.017 (3%), which confirms *the usefulness of the actual data in dataset search*.

Ablation Study: Comparison with Other RDF Data Summarization Methods. To show the effectiveness of our multi-chunk summarization, we conducted an ablation study to compare it with other RDF data summarization methods. Specifically, we implemented two variants of our approach where we replaced data chunking and multi-chunk summarization in our approach with IlluSnip [12,30] or PCSG [46], the two RDF data snippet generation methods used by DR^2 [10] to extract a representative subset of triples. For a fair comparison with our approach, we configured IlluSnip to also extract $k = 20$ triples, while PCSG is non-parametric and extracts as many triples as it wants. As shown in Table 4, DUNKS exhibits higher ranking accuracy and also shorter processing time than both variants. The results suggest *the effectiveness and efficiency of our RDF data chunking and summarization method*.

6 Experiments on Multi-format Datasets

6.1 Experimental Setup

Test Collection. We used **NTCIR-E** [20], the English version of the test collection created for the Data Search task in NTCIR-15.[7] It comprises 46,615 datasets with 92,930 data files in a variety of formats (e.g., PDF, XML, CSV, text, JSON, Excel, RDF) sourced from open government data portals, and 192 queries with an average length of 4.6 words, including 96 for training and 96 for testing; we further split its training set to 76 for training and 20 for validation. It provides 10,536 annotated query-dataset relevance judgments.

First-Stage Retrieval Results. Same as the experiments on ACORDAR, we evaluated by reranking the results of BM25 on NTCIR-E. We followed [20][8] to reproduce the results of BM25 using Pyserini [27],[9] by retrieving over a concatenation of the title and description of each dataset.

[7] https://ntcir.datasearch.jp/data_search_1/.
[8] https://github.com/mpkato/ntcir-datasearch.
[9] https://github.com/castorini/pyserini.

Table 5. Main results on NTCIR-E, with * indicating a significant improvement after reranking ($p < 0.05$).

First-Stage Retrieval	Reranking	NDCG@3	NDCG@5	NDCG@10	NERR@3	NERR@5	NERR@10
KSU	-	0.204	0.231	0.255	0.229	0.257	0.276
NII	-	0.233	0.237	0.248	0.251	0.264	0.278
BM25	-	0.219	0.225	0.238	0.235	0.250	0.264
BM25	ANCE	0.311*	0.302*	0.279*	0.338*	0.344*	0.347*
BM25	BM25+	0.244	0.247	0.256	0.263	0.274	0.288
BM25	Dunks + BGE	0.312*	0.303*	0.287*	0.347*	0.353*	0.357*
BM25	Dunks + BGE-reranker	0.301*	0.293*	0.279*	0.334*	0.340*	0.346*
BM25	Dunks + monoBERT	0.289*	0.272	0.265	0.313*	0.316*	0.324*
BM25	Dunks + GLM-4	**0.339***	**0.314***	**0.296***	**0.375***	**0.373***	**0.377***

Evaluation Metrics. Following [20], we used **NDCG** and Normalized Expected Reciprocal Rank (**NERR**) at positions 3, 5, 10.

6.2 Participating Methods

Baselines. We compared with **KSU** [38] and **NII** [35] which produced the best runs during the shared task [20], and compared with the current state-of-the-art performance achieved by reranking the results of BM25 with **ANCE** [50] reported in a recent empirical study [32]. Besides, since the above-mentioned first-stage retrieval on NTCIR-E only used metadata, for a fair comparison with our approach, we implemented a variant of our approach as another baseline, denoted by **BM25+**, where we replaced our dense scorers with BM25 and disabled multi-chuck summarization since BM25 could directly handle all the chunks.

Our Approach. Our approach was implemented based on the same dense ranking models and LLM as in the experiments on ACORDAR.

6.3 Implementation Details

Data File Parsing. For NTCIR-E, we successfully parsed the data files in 42,522 datasets (91.2%). Parsing failures include encrypted PDF files, ill-formed CSV/JSON/XML/RDF files, etc. Our mapping rules support processing over 90% of the successfully parsed data files. The remaining data files were ignored.

Other settings are the same as those described in Sect. 5.3. After summarization, we retained an average of 21,500 out of 2,054,194 tokens per dataset.

6.4 Experimental Results

Main Results: Effectiveness of Dataset Reranking. As shown in Table 5, all our three fine-tuned dense reranking models significantly improve the metrics after reranking. They raise NDCG@3 by 0.070–0.093 (24%–42%), NDCG@5

Table 6. Results of ablation study on NTCIR-E.

Ranking Model	Data Processing	NDCG@3	NDCG@5	NDCG@10	NERR@3	NERR@5	NERR@10
BGE	Dunks	**0.312**	**0.303**	**0.287**	**0.347**	**0.353**	**0.357**
	Dunks w/o data	0.300	0.297	0.283	0.333	0.342	0.347
BGE-reranker	Dunks	**0.301**	**0.293**	**0.279**	**0.334**	**0.340**	**0.346**
	Dunks w/o data	0.298	0.289	0.277	0.330	0.337	0.343

Table 7. Mean offline processing time (s) for datasets in NTCIR-E.

	#Triples: (1e2, 1e3]	(1e3, 1e4]	(1e4, 1e5]	(1e5, 1e6]
Data File Parsing	0.05	0.53	1.12	5.19
Data Chunking	0.00	0.53	7.39	68.32
Multi-Chunk Summarization	0.00	0.13	2.33	26.16
Total	0.05	1.19	10.84	99.67

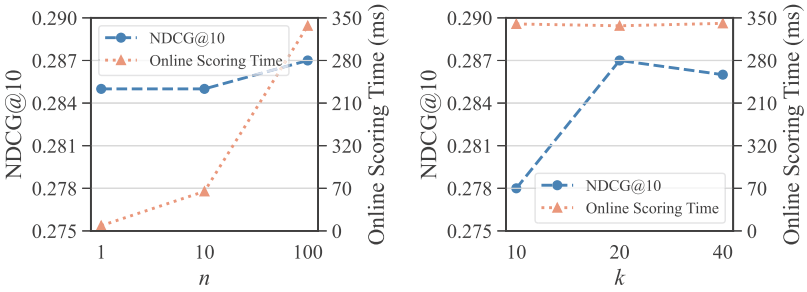
by 0.047–0.078 (17%–35%), and NDCG@10 by 0.027–0.049 (10%–21%). In particular, BGE exceeds the state-of-the-art ANCE. BM25+ which feeds all the chunks into BM25 for reranking also shows better results after reranking, but the improvements are not significant. It demonstrates *the unparalleled effectiveness of dense reranking of datasets over their multi-format data enabled by our approach*, which unifies and summarizes data to fit the processing capacity of PLMs. Our LLM-based zero-shot reranking with GLM-4 achieves the best results.

Ablation Study: Comparison with Metadata-Only Methods. Again, we conducted an ablation study of the metadata-only variant of our approach. Note that the above experiment with BM25+ has partially showed the usefulness of incorporating multi-format data into dataset reranking via our unified data chunking, even using a sparse ranking model. Further, as shown in Table 6, in the metadata-only configuration (w/o data), all metrics decrease. The decreases here are not as large as those in Table 4, which is not surprising because the construction of ACORDAR gives particular attention to the actual data, e.g., by creating data-oriented queries, while NTCIR-E is focused on metadata.

Running Time. Although our data chunking and multi-chunk summarization can be precomputed offline, their actual running time and scalability are interesting considering the potential magnitude of data in practice. To this end, we grouped datasets in NTCIR-E by the size of their transformed data, i.e., the total number of triples, disregarding small datasets containing less than 100 triples. For the datasets in each size range, we calculated their mean offline processing time, including data file parsing, data chunking, and multi-chunk summarization. As shown in Table 7, on an Intel Xeon E7-4820 CPU (2.00GHz), the mean processing time for a dataset is below 100 s even for million-scale data, and the processing time increases almost linearly in data size, representing *the*

practically acceptable time cost and satisfying scalability of our approach. Further investigation showed that most processing time was used for data chunking.

The online reranking time is dominated by the dense ranking model computing relevance scores for each candidate dataset. For example, with BGE on a GeForce RTX 3090 GPU, the mean time for scoring a dataset is 0.34 s. This is acceptable for practical applications which can score multiple datasets in parallel. Unsurprisingly, feeding data passages into a dense scorer, our approach is slower than metadata-only methods, e.g., 0.12 s used by ANCE.



(a) Comparison among different n (b) Comparison among different k

Fig. 4. Influence of n and k on DUNKS + BGE on NTCIR-E.

Influence of Parameters. Recall that in our approach, n bounds the number of chunks, and k bounds the number of triples in each chunk. They can be tuned to achieve different trade-offs between ranking accuracy and time cost. In all the above experiments, they were fixed to $n = 100$ and $k = 20$. Below we compare the results obtained in different settings of these parameters.

As shown in Fig. 4(a), by increasing n from 1 to 100, since more chunks are considered in reranking, we observe a small rise in NDCG, but at the cost of a sharper increase in online scoring time. Therefore, for practical applications, we would recommend choosing n from the range of $[10, 100]$ but not larger.

In Fig. 4(b), by increasing k from 10 to 20, since more triples are retained in each chunk, we observe an improvement in NDCG with fluctuating online scoring time. By further increasing k to 40, both NDCG and scoring time fluctuate, thus suggesting choosing k from the range of $[20, 40]$ for practical applications.

7 Related Work

7.1 Dataset Search

Dataset search has garnered significant research attention, leading to various methods and systems [8]. For instance, Google developed a dataset search engine that utilizes metadata for keyword-based retrieval of Web datasets [5]. Luo et

al. [32] investigated knowledge-enhanced methods for metadata-based dataset search. Silva et al. [42] leveraged LLMs to generate training data for dataset search by synthesizing queries from metadata. However, as revealed in [34,41], *metadata faces quality issues and barely represents the actual data adequately, which is insufficient for supporting dataset search.*

Accordingly, recent studies have highlighted the importance of the actual data in enhancing users' dataset search experience [9]. Ota et al. [39] proposed a data-driven approach to perform domain discovery for tabular datasets. Chen et al. [11] focused on improving retrieval models by generating schema labels from tabular datasets. Auctus [6] is a dataset search engine which indexes data summaries of tabular datasets and presents data samples. StruBERT [44] is a structure-aware BERT model proposed for table search tasks, which captures the cross-matching signals of rows and columns. For RDF datasets, LODAtlas [40] and CKGSE [48] utilize summary visualizations to provide users with an efficient way of browsing and analyzing large RDF data. ACORDAR [29] has served as a data-based ad hoc retrieval test collection for RDF datasets, of which the query creation and relevance judgment heavily relied on the actual RDF data. DR^2 [10] employs a data snippet extracted by IlluSnip [12] or PCSG [46] to represent RDF data in dense reranking, and it augments training data via distant supervision and self-training. Compared with DR^2, *our approach incorporates a novel multi-chunk summarization method leading to higher ranking accuracy in RDF dataset search.* Besides, through unified data chunking, *our approach is applicable to multiple data formats, while all the above efforts cater to a single format.*

7.2 Data Summarization

Extractive and aggregative approaches constitute major proportions of existing data summarization solutions [55]. A large number of aggregative approaches abstract data into high-level representations that can be used to restore the original data [7]. *Since data restoration is not the main purpose of dataset search, our work is focused on extracting a representative subset of data to fit the input capacity of PLM-based dense ranking models.* Closer to our approach are the studies on snippet extraction [45] and entity summarization [31].

Snippet extraction aims to succinctly exemplify the content of a large dataset or elucidate its relevance to a query [45]. SubTab [1] extracts a diverse sub-table that captures prominent association rules in the original table. IlluSnip [12] formulates a maximum-weight-and-coverage connected graph problem and extracts an optimal connected subgraph that covers the most important classes and properties in the original RDF data. PCSG [46] mines schema patterns from RDF data and extracts a tree that covers these patterns by solving a group Steiner tree problem. Our multi-chunk summarization also aims to reflect the schema patterns in the original data and it incorporates the notion of EDP from PCSG. *It has the capability to jointly extract multiple data chunks*, thus retaining more information useful for dataset search than a single snippet extracted by IlluSnip or PCSG. Since all the triples in a chunk cohesively describe the same core entity and are naturally connected, *we formulate a weighted maximum coverage*

problem for triple selection, which can be more efficiently solved than those concerning graph connectivity formulated in IlluSnip and PCSG. An extra benefit of such a cohesive chunk is that, when fed into a PLM-based dense ranking model, *its meaning can arguably be more effectively captured by the PLM than a semantically divergent snippet.* These advantages make our multi-chunk summarization empirically superior to IlluSnip and PCSG in dense ranking of RDF datasets.

Entity summarization is to extract a small subset of triples from RDF data to present the main characteristics of an entity [31]. Pertaining to the concept of measuring summary representativeness in entity summarization, triple selection in our multi-chunk summarization also takes into account factors including predicate frequency and value frequency [15,23,51]. However, *beyond triple selection, our approach further incorporates the selection of chunks (i.e., core entities), which is not considered in entity summarization where a target entity is given.*

7.3 Document Ranking

As Transformer-based PLMs such as BERT [14] have been applied to passage retrieval and demonstrated promising performance, a series of document retrieval models using dense encoding have emerged [28,53]. However, the input length of PLMs limits the effectiveness of long document retrieval. To handle long documents, existing methods commonly segment text into short passages [18,24]. MaxP [13] is a standard method, processing passages individually and selecting the most relevant passage as the representative of the document in ranking. Subsequent efforts [17,25,26] select key passages through a cascade architecture to optimize ranking accuracy. Our motivation for chunking large structured data comes from this line of research. *In contrast to documents which have inherent paragraph segmentation, chunking structured data to fit PLMs has its unique challenges and has not been thoroughly investigated. Our work represents a first step towards search-oriented chunking of large multi-format data.*

8 Conclusion

Our work represents a significant move from metadata-based to data-based dataset search. As opposed to handling only metadata or a single format of data in current systems, our Dunks approach addresses the heterogeneity and magnitude of open data with chunking and multi-chunk summarization, and, for the first time, enables to perform dense ranking of datasets over their multi-format data in a unified manner. Its value is not limited to achieving new state-of-the-art ranking accuracy on two test collections for dataset search. Our summarized data chunks also have the potential to be presented with search results to strengthen the interpretability of the dataset search process, which will be our future work.

Limitations. Our approach has the following limitations. First, the parameters in our multi-chunk summarization are empirically fixed. For the convenience of practitioners, it would be helpful to study dynamic parameter configuration. Second, our unified data chunking relies on predefined mapping rules which have

been observed to be practical for most real open data. Still, some assumptions underlying these rules (e.g., the first cell in each row represents an entity) may not hold for all data and need to be improved. Third, our approach is most suitable for large and heterogeneously formatted datasets that are fully accessible. If data is inaccessible, or small enough to be fed directly into PLMs, or available in a single format, the usefulness of our approach will be limited.

Acknowledgements. This work was supported by the NSFC (62072224) and the CIPSC-SMP-Zhipu.AI Large Model Cross-Disciplinary Fund.

Supplemental Material Statement. Source code is available from GitHub at https://github.com/nju-websoft/DUNKS.

References

1. Amsterdamer, Y., Davidson, S.B., Milo, T., Razmadze, K., Somech, A.: Selecting sub-tables for data exploration. In: ICDE 2023, pp. 2496–2509 (2023). https://doi.org/10.1109/ICDE55515.2023.00192
2. Bassani, E.: ranx: A blazing-fast python library for ranking evaluation and comparison. In: ECIR 2022, pp. 259–264 (2022). https://doi.org/10.1007/978-3-030-99739-7_30
3. Bassani, E., Romelli, L.: ranx.fuse: A python library for metasearch. In: CIKM 2022, pp. 4808–4812 (2022). https://doi.org/10.1145/3511808.3557207
4. Benjelloun, O., Chen, S., Noy, N.F.: Google dataset search by the numbers. In: ISWC 2020, pp. 667–682 (2020). https://doi.org/10.1007/978-3-030-62466-8_41
5. 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
6. Castelo, S., Rampin, R., Santos, A.S.R., Bessa, A., Chirigati, F., Freire, J.: Auctus: a dataset search engine for data discovery and augmentation. Proc. VLDB Endow. **14**(12), 2791–2794 (2021). https://doi.org/10.14778/3476311.3476346
7. Cebiric, S., et al.: Summarizing semantic graphs: a survey. VLDB J. **28**(3), 295–327 (2019). https://doi.org/10.1007/S00778-018-0528-3
8. 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
9. 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
10. Chen, Q., Huang, Z., Zhang, Z., Luo, W., Lin, T., Shi, Q., Cheng, G.: Dense reranking with weak supervision for RDF dataset search. In: ISWC 2023, pp. 23–40 (2023). https://doi.org/10.1007/978-3-031-47240-4_2
11. Chen, Z., Jia, H., Heflin, J., Davison, B.D.: Leveraging schema labels to enhance dataset search. In: ECIR 2020, pp. 267–280 (2020). https://doi.org/10.1007/978-3-030-45439-5_18
12. 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

13. Dai, Z., Callan, J.: Deeper text understanding for IR with contextual neural language modeling. In: SIGIR 2019, pp. 985–988 (2019). https://doi.org/10.1145/3331184.3331303
14. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: NAACL 2019, pp. 4171–4186 (2019). https://doi.org/10.18653/v1/n19-1423
15. Gunaratna, K., Thirunarayan, K., Sheth, A.P., Cheng, G.: Gleaning types for literals in RDF triples with application to entity summarization. In: ESWC 2016, pp. 85–100 (2016). https://doi.org/10.1007/978-3-319-34129-3_6
16. Hochbaum, D.S.: Approximating covering and packing problems: set cover, vertex cover, independent set, and related problems, pp. 94–143. PWS Publishing Co., USA (1996)
17. Hofstätter, S., Mitra, B., Zamani, H., Craswell, N., Hanbury, A.: Intra-document cascading: Learning to select passages for neural document ranking. In: SIGIR 2021, pp. 1349–1358 (2021). https://doi.org/10.1145/3404835.3462889
18. Hofstätter, S., Zamani, H., Mitra, B., Craswell, N., Hanbury, A.: Local self-attention over long text for efficient document retrieval. In: SIGIR 2020, pp. 2021–2024 (2020). https://doi.org/10.1145/3397271.3401224
19. Karpukhin, V., Oguz, B., Min, S., Lewis, P.S.H., Wu, L., Edunov, S., Chen, D., Yih, W.: 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
20. 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
21. 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
22. 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
23. Kroll, H., Nagel, D., Balke, W.T.: Bafrec: balancing frequency and rarity for entity characterization in open linked data. In: EYRE 2018 (2018)
24. Li, C., Yates, A., MacAvaney, S., He, B., Sun, Y.: PARADE: passage representation aggregation for document reranking. CoRR abs/2008.09093 (2020)
25. Li, M., Gaussier, É.: KeyBLD: selecting key blocks with local pre-ranking for long document information retrieval. In: SIGIR 2021, pp. 2207–2211 (2021). https://doi.org/10.1145/3404835.3463083
26. Li, M., Popa, D.N., Chagnon, J., Cinar, Y.G., Gaussier, É.: The power of selecting key blocks with local pre-ranking for long document information retrieval. ACM Trans. Inf. Syst. **41**(3), 73:1–73:35 (2023). https://doi.org/10.1145/3568394
27. Lin, J., Ma, X., Lin, S., Yang, J., Pradeep, R., Nogueira, R.F.: Pyserini: a python toolkit for reproducible information retrieval research with sparse and dense representations. In: SIGIR 2021, pp. 2356–2362 (2021). https://doi.org/10.1145/3404835.3463238
28. Lin, J., Nogueira, R.F., Yates, A.: Pretrained Transformers for Text Ranking: BERT and Beyond. Morgan & Claypool Publishers (2021). https://doi.org/10.2200/S01123ED1V01Y202108HLT053
29. 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

30. 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
31. 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
32. Luo, W., Chen, Q., Zhang, Z., Huang, Z., Cheng, G.: An empirical investigation of implicit and explicit knowledge-enhanced methods for ad hoc dataset retrieval. In: Findings of EMNLP 2023, pp. 14349–14360 (2023). https://doi.org/10.18653/V1/2023.FINDINGS-EMNLP.957
33. Muennighoff, N., Tazi, N., Magne, L., Reimers, N.: MTEB: massive text embedding benchmark. In: EACL 2023, pp. 2006–2029 (2023). https://doi.org/10.18653/V1/2023.EACL-MAIN.148
34. Neumaier, S., Umbrich, J., Polleres, A.: Automated quality assessment of metadata across open data portals. ACM J. Data Inf. Qual. **8**(1), 2:1–2:29 (2016). https://doi.org/10.1145/2964909
35. Nguyen, P., et al.: Nii table linker at the ntcir-15 data search task: Re-ranking with pre-trained contextualized embeddings, data content, entity-centric, and cluster-based approaches. In: NTCIR 2020 (2020)
36. Nguyen, T., et al.: MS MARCO: a human generated machine reading comprehension dataset. In: Workshop on Cognitive Computation (NIPS 2016), vol. 1773 (2016)
37. Nogueira, R.F., Cho, K.: Passage re-ranking with BERT. CoRR abs/1901.04085 (2019)
38. Okamoto, T., Miyamori, H.: Ksu systems at the ntcir-15 data search task. In: NTCIR 2020 (2020)
39. Ota, M., Mueller, H., Freire, J., Srivastava, D.: Data-driven domain discovery for structured datasets. Proc. VLDB Endow. **13**(7), 953–965 (2020). doi:https://doi.org/10.14778/3384345.3384346
40. Pietriga, E., Gözükan, H., Appert, C., Destandau, M., Cebiric, S., Goasdoué, F., Manolescu, I.: Browsing linked data catalogs with lodatlas. In: ISWC 2018, pp. 137–153 (2018). https://doi.org/10.1007/978-3-030-00668-6_9
41. Quarati, A.: Open government data: Usage trends and metadata quality. J. Inf. Sci., 1–24 (2021). https://doi.org/10.1177/01655515211027775
42. Silva, L., Barbosa, L.: Improving dense retrieval models with LLM augmented data for dataset search. Knowl. Based Syst. **294**, 111740 (2024). https://doi.org/10.1016/j.knosys.2024.111740
43. Sun, W., et al.: Is ChatGPT good at search? investigating large language models as re-ranking agents. In: EMNLP 2023, pp. 14918–14937 (2023). https://doi.org/10.18653/V1/2023.EMNLP-MAIN.923
44. Trabelsi, M., Chen, Z., Zhang, S., Davison, B.D., Heflin, J.: Strubert: structure-aware BERT for table search and matching. In: WWW 2022, pp. 442–451 (2022). doi:https://doi.org/10.1145/3485447.3511972
45. Wang, X., Cheng, G.: A survey on extractive knowledge graph summarization: applications, approaches, evaluation, and future directions. In: IJCAI 2024 (2024)
46. 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
47. 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

48. 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
49. Xiao, S., Liu, Z., Zhang, P., Muennighof, N.: C-pack: packaged resources to advance general Chinese embedding (2023). https://doi.org/10.48550/ARXIV.2309.07597
50. Xiong, L., et al.: Approximate nearest neighbor negative contrastive learning for dense text retrieval. In: ICLR 2021 (2021)
51. Yang, E., Hao, F., Yang, Y., Maio, C.D., Nasridinov, A., Min, G., Yang, L.T.: Incremental entity summarization with formal concept analysis. IEEE Trans. Serv. Comput. **15**(6), 3289–3303 (2022). https://doi.org/10.1109/TSC.2021.3090276
52. Zeng, A., et al.: ChatGLM: a family of large language models from GLM-130B to GLM-4 all tools. CoRR abs/2406.12793 (2024). https://doi.org/10.48550/ARXIV.2406.12793
53. 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
54. Zhiltsov, N., Kotov, A., Nikolaev, F.: Fielded sequential dependence model for ad-hoc entity retrieval in the Web of data. In: SIGIR 2015, pp. 253–262 (2015). https://doi.org/10.1145/2766462.2767756
55. Zneika, M., Vodislav, D., Kotzinos, D.: Quality metrics for RDF graph summarization. Semantic Web **10**(3), 555–584 (2019). https://doi.org/10.3233/SW-190346