GraDaSE: Graph-Based Dataset Search with Examples

Jing He and **Mingyang Lv** and **Qing Shi** and **Gong Cheng**
State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
Correspondence: Gong Cheng (gcheng@nju.edu.cn)

Abstract

Dataset search is a specialized information retrieval task. In the emerging scenario of Dataset Search with Examples (DSE), the user submits a query and a few target datasets that are known to be relevant as examples. The retrieved datasets are expected to be relevant to the query and also similar to the target datasets. Distinguished from existing text-based retrievers, we propose a graph-based approach GraDaSE. Besides the textual metadata of the datasets, we identify their provenance-based and topic-based relationships to construct a graph, and jointly encode their structural and textual information for ranking candidate datasets. GraDaSE outperforms a variety of strong baselines on two test collections, including DataFinder-E that we construct.

1 Introduction

The continuous evolution of models is highly dependent on the availability and quality of the datasets, and finding the most suitable datasets is crucial for NLP research (Viswanathan et al., 2023). Beyond that, with the recent surge in AI advances and the proliferation of available datasets, *dataset search* has become an important research problem in multiple fields, including information retrieval and database (Altaf et al., 2019; Chapman et al., 2020; Irrera et al., 2024). Existing methods mainly address the task of *ad hoc dataset search*, which involves retrieving datasets that are relevant to a textual query (Färber and Leisinger, 2021; Kato et al., 2021). However, in real-world scenarios, dataset search presents in diverse and complex forms.

Research Problem Consider the following scenario: A researcher or a data journalist has known some relevant datasets, but needs more of this kind. For example, a junior researcher who evaluates multimodal recommendation models has used the MovieLens and TikTok datasets and is seeking one more such dataset. By querying Google Dataset Search (Brickley et al., 2019) with "multimodal recommendation", the MSD-A dataset for music recommendation is ranked higher than the Kwai dataset for short video recommendation, although Kwai better fits the need as it is more similar to MovieLens and TikTok, the researcher's *target datasets* that are known to be relevant. If they were included in the query as *examples* to guide the search, the ranking of the results would be improved. This emerging task is called *Dataset Search with Examples (DSE)*, adapting the well-known query-by-example paradigm to dataset search and extending the established task of similarity-based dataset discovery (Bogatu et al., 2020; Paton et al., 2024). This function can be integrated into current dataset search engines to improve search efficiency and relevance.

Limitations of Existing Work First, current DSE solutions, as evaluated in DSEBench (DSEBench, 2024), expand the query with the textual metadata of each target dataset in text-based retrieval. However, there are rich *relationships between datasets*, such as version updates and derivation, which are useful for DSE as they connect similar datasets, but such relationships may not be explicitly described in metadata and therefore be overlooked in retrieval.

Second, graphs have been used in conventional ad hoc dataset search, e.g., to represent citation relationships between datasets and papers (Altaf et al., 2019) to derive relevance from node embeddings. However, relationships incident with dataset are not limited to citation that is specific to the academic domain. *Domain-independent graph representation* is needed for DSE. Moreover, existing methods struggle to effectively capture *path-based multi-hop relationships* between datasets.

Third, to our knowledge, except for DSEBench, there is *a lack of test collection for DSE* despite the availability of many test collections for the conventional ad hoc dataset search (Kato et al., 2021; Lin et al., 2022; Chen et al., 2024c).

Our Approach To address the above limitations, we propose GraDaSE, short for **Gra**ph-based **Da**taset **S**earch with **E**xamples. We construct a dataset graph to represent *domain-independent provenance-based and topic-based relationships* between datasets, from which we obtain *contextualized path-based representations* of queries and candidate datasets that are biased toward the given target datasets. We jointly encode their graph structures and textual descriptions to rank candidate datasets. We evaluate our approach on DSEBench, an existing test collection for DSE containing government datasets, and on a *new test collection for DSE that we create* from DataFinder (Viswanathan et al., 2023) containing academic datasets.

In summary, our contributions include:

- a new graph-based method for DSE based on a generalized graph representation of domain-independent relationships between datasets,
- a new GNN model that computes novel target-biased representations of queries and datasets incorporating path structures, and
- a new test collection for DSE in a domain that is complementary to existing evaluation.

Our code and data are available at `https://github.com/nju-websoft/GraDaSE`.

2 Related Work

Dataset Search Most existing work on dataset search, including scientific dataset recommendation, essentially performs text-based retrieval (Chen et al., 2019; Färber and Leisinger, 2021). These approaches map the query and metadata of each dataset into the same vector space and then compute their similarity. For example, DataFinder (Viswanathan et al., 2023) trains SciBERT (Beltagy et al., 2019) on queries and datasets from Papers With Code. Google Dataset Search (Brickley et al., 2019), a popular dataset search engine, also indexes the metadata of the datasets for retrieval. *Such text-based retrieval overlooks the implicit and useful relationships between datasets.*

To address it, one line of work uses graph-based methods to capture, for example, the citation relationships of datasets (Altaf et al., 2019; Wang et al., 2022). *However, these domain-specific methods cannot be generalized to other domains.*

Search with Examples Currently, there are few attempts to address DSE. To our knowledge, DSEBench (DSEBench, 2024) is the only research effort and is focused on evaluating DSE with datasets that contain open government data. *There is still a lack of test collection in other domains. Moreover, the baseline methods evaluated in DSEBench are all text-based retrievers, overlooking the relationships between datasets.*

Beyond dataset search, other research fields have studied tasks similar to DSE. For example, personalized product search (Ai et al., 2019) takes a user's purchase history as examples to be included in the query. TEM (Bi et al., 2020) encodes a user's query history to achieve personalization. *However, it is difficult for such unstructured methods to effectively identify and exploit the implicit relationships between items.* Moreover, in product search, the number of examples is often large, easily exceeding dozens, while in DSE we expect the availability of a limited number of target datasets as examples.

Heterogeneous Graph Neural Networks Our approach is based on the heterogeneous graph neural network (HGNN). HGNNs have been shown to be more capable of capturing graph structure information than homogeneous GNNs, and they have been used successfully in information retrieval applications. SimpleHGN (Lv et al., 2021) and HINormer (Mao et al., 2023) are state-of-the-art HGNNs based on GAT (Veličković et al., 2018). *However, focusing on node-centric local structures, they on their own cannot effectively capture path-based multi-hop relationships between datasets in DSE, thus suffering from unsatisfying ranking accuracy as we will see in our experimental results.*

3 Task Formulation and Preliminaries

A **dataset** d is associated with its metadata, which is made up of three textual fields: title, description, and tags. Other fields are not considered in this paper. Following product-level dataset search engines such as Google Dataset Search (Brickley et al., 2019), we focus on the metadata of the datasets and do not rely on their data files so that our approach can easily be incorporated into existing systems.

Given a textual query q, a set $T = \{t_1, ..., t_{k_t}\}$ of k_t target datasets (i.e., examples) that are known to be relevant to q, and a set $C = \{c_1, ..., c_{k_c}\}$ of k_c candidate datasets, the task of **DSE** aims to learn a ranking function f such that a higher ranking score $f(q, T, c)$ for a candidate dataset $c \in$

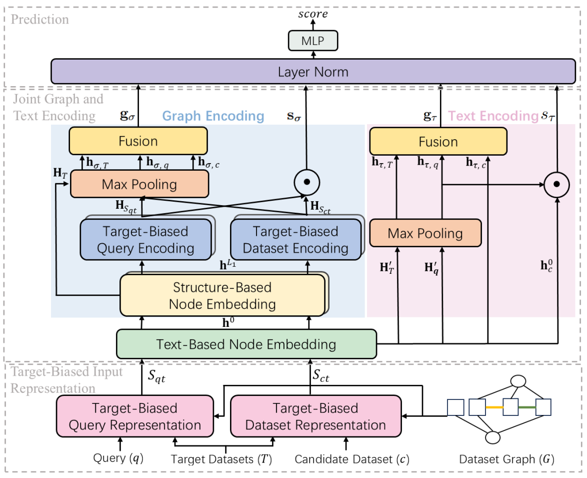

Figure 1: Model architecture of GraDaSE.

C indicates that it is more relevant to the user's search intent conveyed in the query q just like how the target datasets in T are relevant to q. We assume $T \neq \emptyset$, otherwise the task would reduce to the conventional task of ad hoc dataset search.

In a **graph** $G = (V, E, T_V, T_E)$, V is the set of nodes, E is the set of edges, T_V is the set of node types and T_E is the set of edge types. A heterogeneous graph contains multiple types of nodes or edges, i.e. $|T_V| + |T_E| > 2$. Each node $v \in V$ and each edge $e \in E$ are associated with a type given by $\phi : V \to T_V$ and $\psi : E \to T_E$, respectively.

4 The GraDaSE Approach

In GraDaSE, we first construct a generalized dataset graph G based on the textual metadata of all datasets considered to represent their various domain-independent relationships (Section 4.1). Then, as shown in Figure 1, contextualized by the graph structure of G, we generate a target-biased representation S_{qt} for the query q that is biased toward the target datasets T, and a target-biased representation S_{ct} for each candidate dataset $c \in C$ (Section 4.2). These representations incorporate path structures in G to capture the similarity between datasets that helps to indicate relevance. The structural information from the dataset graph and the textual information from the metadata in these representations are encoded holistically as $(\mathbf{g}_\sigma, \mathbf{s}_\sigma)$ and $(\mathbf{g}_\tau, s_\tau)$, respectively (Section 4.3), which then are fed together into a prediction layer to obtain the final ranking score for c (Section 4.4).

4.1 Dataset Graph

Beyond textual metadata, datasets exhibit rich but often implicit relationships between them such as version relationships resulting from dataset updates. These are useful for DSE by connecting target datasets to other relevant datasets. To exploit them, existing academic dataset discovery methods build a dataset citation graph (Altaf et al., 2019). However, this domain-specific method cannot be generalized to identify the diverse types of domain-independent relationships between datasets.

To address it, we construct a *dataset graph* G to represent the relationships between datasets derived from their *provenance* and *topics*. These *domain-independent relationships* increase the generalizability of our approach. The graph captures not only the dependencies among the datasets, but also their semantic and contextual similarities.

Provenance-based relationships of five types, inspired by Lin et al. (2024), are extracted from the textual metadata of the datasets. They are `Replica`, `Version`, `Subset`, `Derivation`, and `Variant`, where `Replica`, `Version`, `Derivation`, and `Variant` are represented by bidirectional edges in the dataset graph, while `Subset` corresponds to unidirectional edges with reverse edges labeled `Superset`. To determine whether and which relationship exists between two datasets according to their textual metadata, we train a classifier using the annotations published by Lin et al. (2024). In addition, we incorporate a heuristic method to improve the classification accuracy. We refer the reader to Appendix A.1 for more details.

Topic-based relationships are also useful for DSE. In the metadata of each dataset, its tags reflect its topics, so we include tags as nodes in the dataset graph by introducing two mutually reverse relationships between datasets and tags: `HasTag` and `IsTagOf`. An advantage of introducing these relationships is that a new dataset, even having different provenances from all previous datasets, can be connected to them through topic-based relationships. This allows the processing of new datasets to benefit from the graph structure.

To conclude, our heterogeneous dataset graph G defines $T_V = \{$`Dataset`, `Tag`$\}$ and $T_E = \{$`Replica`, `Version`, `Subset`, `Superset`, `Derivation`, `Variant`, `HasTag`, `IsTagOf`$\}$. An example dataset graph is illustrated in Figure 2.

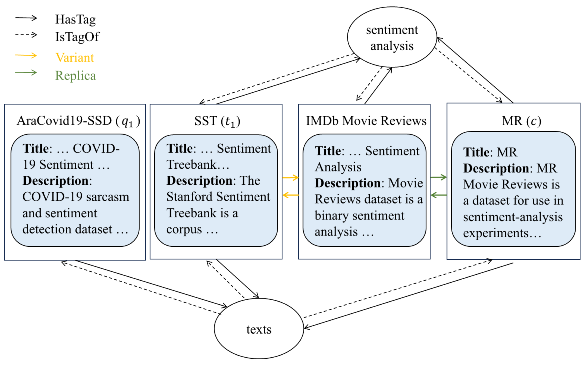
HasTag
IsTagOf
Variant
Replica
sentiment analysis
AraCovid19-SSD (q_1)
Title: … COVID-19 Sentiment …
Description: COVID-19 sarcasm and sentiment detection dataset …
SST (t_1)
Title: … Sentiment Treebank…
Description: The Stanford Sentiment Treebank is a corpus …
IMDb Movie Reviews
Title: … Sentiment Analysis
Description: Movie Reviews dataset is a binary sentiment analysis …
MR (c)
Title: MR
Description: MR Movie Reviews is a dataset for use in sentiment-analysis experiments…
texts

Figure 2: An example of dataset graph. Rectangles and ellipses represent datasets and tags, respectively.

4.2 Target-Biased Input Representation

With a dataset graph G representing dataset relationships, in the following, we will expand the query q and each candidate dataset $c \in C$ with node sequences S_{qt} and S_{ct}, respectively, providing their *contextualized representations biased toward the target datasets* T to capture the search intent implied by these examples.

Target-Biased Query Representation We map the query q to a set of k_q *query nodes* $Q = \{q_1, \dots, q_{k_q}\}$ in the dataset graph G as follows. Recall that, as illustrated in Figure 2, nodes in G represent either datasets having textual metadata or tags. We index these texts and use BM25 to retrieve the top-100 nodes, which are then reranked by a dense model, `bge-reranker-v2-m3` (Li et al., 2023; Chen et al., 2024a), to obtain k_q top-ranked query nodes. We concatenate them with k_t nodes representing the target datasets in $T = \{t_1, ..., t_{k_t}\}$ as a context into a node sequence $S_{qt} = [q_1 || \cdots || q_{k_q} || t_1 || \cdots || t_{k_t}]$. As analyzed in Appendix C.1, the order of the query nodes and target datasets in S_{qt} has insignificant influence on the performance of our approach.

Target-Biased Dataset Representation A candidate dataset $c \in C$ that is relevant to the query q is expected to be similar to the target datasets T which are known to be relevant. We capture the similarity between c and T using the *paths* that connect them in the dataset graph G. As illustrated in Figure 2, the paths between c and the target dataset $t_1 \in T$ represent their common tags and multi-hop provenance-based relationships. We search for n_s shortest paths between c and T to capture their closest relationships. The worst-case time complexity of this search in a graph with n nodes and m edges is in $O(n_s m \sqrt{n})$ (Roditty and Zwick, 2012), and its actual run is very fast, as we will see in our experimental results. For each path, we take its node sequence. The set of n_s node sequences is denoted by S_{ct}. Sequences are bounded by a maximum length of l_s. Longer sequences are truncated.

4.3 Joint Graph and Text Encoding

With target-biased node sequence representations S_{qt} and S_{ct} for the query q and each candidate dataset $c \in C$, respectively, in the following, we will holistically encode their *graph-based structural information* into $(\mathbf{g}_\sigma, \mathbf{s}_\sigma)$ (Section 4.3.1) and *textual information* into $(\mathbf{g}_\tau, s_\tau)$ (Section 4.3.2).

Text-Based Node Embedding As a basis, we initially embed all nodes in the dataset graph G based on their textual information. Specifically, we use `stella_en_400M_v5` (Zhang et al., 2025) to embed the title and description for each `Dataset` node, and embed the tag text for each `Tag` node. To unify the embedding spaces for different types of nodes (Lv et al., 2021; Mao et al., 2023), for each node $v \in V$, its embedding $\mathbf{x}_v \in \mathbb{R}^d$ is fed into a linear layer to obtain a revised embedding:

$$\mathbf{h}_v^0 = \mathrm{MLP}(\mathbf{x}_v) \in \mathbb{R}^d \,, \tag{1}$$

where d is the dimension of the embeddings.

4.3.1 Graph Encoding

With the node sequences S_{qt} and S_{ct} where each node v has an initial text-based embedding $\mathbf{h}_v^0$, we encode their structural information in the dataset graph G into $(\mathbf{g}_\sigma, \mathbf{s}_\sigma)$ in the following steps.

Structure-Based Node Embedding To capture the structural information in G, we firstly employ a simple but effective heterogeneous graph encoder, SimpleHGN (Lv et al., 2021). This graph attention network *learns to weight different edge types* when attending to neighboring nodes, thus exploiting the semantics of the dataset relationships. Formally, for each node $v_i \in V$ in G, we enhance its text-based embedding $\mathbf{h}_{v_i}^0$ into a structure-based embedding $\mathbf{h}_{v_i}^{L_1} \in \mathbb{R}^d$ obtained from an L_1-layer SimpleHGN:

$$(\mathbf{h}_{v_1}^{L_1}, \dots, \mathbf{h}_{v_{|V|}}^{L_1}) = \mathrm{SimpleHGN}(\mathbf{h}_{v_1}^0, \dots, \mathbf{h}_{v_{|V|}}^0) \,. \tag{2}$$

Target-Biased Query and Dataset Encoding To encode a sequence of nodes, we use an efficient heterogeneous encoder provided by HINormer (Mao et al., 2023) that is capable of *learning to weight different node types* (i.e., `Dataset` or `Tag`) when attending to nodes in the sequence. Formally, for a node sequence $S \in \{S_{qt}\} \cup S_{ct}$ representing the query q or a candidate dataset $c \in C$, for each node $v_i \in S$, we enrich its structure-based embedding $\mathbf{h}_{v_i}^{L_1}$ with its context in the sequence into a target-biased embedding $\mathbf{h}_{v_i}^{L_2} \in \mathbb{R}^d$ obtained from an L_2-layer HINormer:

$$(\mathbf{h}_{v_1}^{L_2}, \dots, \mathbf{h}_{v_{|S|}}^{L_2}) = \mathrm{HINormer}(\mathbf{h}_{v_1}^{L_1}, \dots, \mathbf{h}_{v_{|S|}}^{L_1}) \,. \quad (3)$$

We use separate encoders for S_{qt} and the node sequences in S_{ct}. In particular, when encoding the node sequences in S_{ct}, since each of them represents a path where the node order is meaningful, we incorporate positional embeddings.

These target-biased embeddings of query nodes in S_{qt} and the candidate dataset in S_{ct} are combined into matrices to represent q and c, respectively:

$$\begin{aligned} \mathbf{H}_{S_{qt}} &= (\mathbf{h}_{q_1}^{L_2}; \dots; \mathbf{h}_{q_{k_q}}^{L_2}) \in \mathbb{R}^{d \times k_q} \,, \\ \mathbf{H}_{S_{ct}} &= (\mathbf{h}_{v_{c,1}}^{L_2}; \dots; \mathbf{h}_{v_{c,|S_{ct}|}}^{L_2}) \in \mathbb{R}^{d \times |S_{ct}|} \,, \end{aligned} \quad (4)$$

where $v_{c,i}$ represents the candidate dataset c in the i-th node sequence in S_{ct}.

Fusion We fuse the above embeddings into a holistic encoding $(\mathbf{g}_\sigma, \mathbf{s}_\sigma)$ of structural information. Specifically, we apply max-pooling to obtain graph-based embeddings for the query q, the target datasets T, and each candidate dataset $c \in C$:

$$\begin{aligned} \mathbf{g}_\sigma &= \mathrm{MLP}([\mathbf{h}_{\sigma,q} || \mathbf{h}_{\sigma,T} || \mathbf{h}_{\sigma,c}]) \in \mathbb{R}^d \,, \\ \mathbf{h}_{\sigma,q} &= \mathrm{MaxPooling}(\mathbf{H}_{S_{qt}}) \in \mathbb{R}^d \,, \\ \mathbf{h}_{\sigma,T} &= \mathrm{MaxPooling}(\mathbf{H}_T) \in \mathbb{R}^d \\ &\quad \text{where } \mathbf{H}_T = (\mathbf{h}_{t_1}^{L_1}; \dots; \mathbf{h}_{t_{k_t}}^{L_1}) \in \mathbb{R}^{d \times k_t} \,, \\ \mathbf{h}_{\sigma,c} &= \mathrm{MaxPooling}(\mathbf{H}_{S_{ct}}) \in \mathbb{R}^d \,. \end{aligned} \quad (5)$$

We also aggregate the similarity values of the dot product between all node embeddings in $\mathbf{H}_{S_{ct}}$ and $\mathbf{H}_{S_{qt}}$ as features characterizing the relevance of the candidate dataset c to the query q:

$$\mathbf{s}_\sigma = \mathrm{MLP}(\mathbf{H}_{S_{qt}}^T \cdot \mathbf{H}_{S_{ct}}) \in \mathbb{R}^{k_q} \,. \quad (6)$$

4.3.2 Text Encoding

With query nodes $Q = \{q_1, \dots, q_{k_q}\}$, target datasets $T = \{t_1, ..., t_{k_t}\}$, and a candidate dataset $c \in C$ where each node v has an initial text-based embedding $\mathbf{h}_v^0$, we fuse these embeddings into a holistic encoding $(\mathbf{g}_\tau, s_\tau)$ of textual information. The fusion process basically resembles that for graph encoding described in the previous section:

$$\begin{aligned} \mathbf{g}_\tau &= \mathrm{MLP}([\mathbf{h}_{\tau,q} || \mathbf{h}_{\tau,T} || \mathbf{h}_{\tau,c}]) \in \mathbb{R}^d \,, \\ \mathbf{h}_{\tau,q} &= \mathrm{MaxPooling}(\mathbf{H}'_q) \in \mathbb{R}^d \\ &\quad \text{where } \mathbf{H}'_q = (\mathbf{h}_{q_1}^0; \dots; \mathbf{h}_{q_{k_q}}^0) \in \mathbb{R}^{d \times k_q} \,, \\ \mathbf{h}_{\tau,T} &= \mathrm{MaxPooling}(\mathbf{H}'_T) \in \mathbb{R}^d \\ &\quad \text{where } \mathbf{H}'_T = (\mathbf{h}_{t_1}^0; \dots; \mathbf{h}_{t_{k_t}}^0) \in \mathbb{R}^{d \times k_t} \,, \\ \mathbf{h}_{\tau,c} &= \mathbf{h}_c^0 \in \mathbb{R}^d \,. \end{aligned} \quad (7)$$

We also take the dot product similarity between $\mathbf{h}_c^0$ and $\mathbf{h}_{\tau,q}$ as a feature characterizing the relevance of the candidate dataset c to the query q:

$$s_\tau = \mathbf{h}_{\tau,q}^T \cdot \mathbf{h}_c^0 \in \mathbb{R} \,. \quad (8)$$

4.4 Prediction and Training

With graph encoding $(\mathbf{g}_\sigma, \mathbf{s}_\sigma)$ and text encoding $(\mathbf{g}_\tau, s_\tau)$, we feed all of them into a normalization layer followed by an MLP to obtain the final ranking score for each candidate dataset $c \in C$:

$$\mathit{score} = \mathrm{MLP}(\mathrm{LayerNorm}([\mathbf{g}_\sigma || \mathbf{s}_\sigma || \mathbf{g}_\tau || s_\tau])) \in \mathbb{R} \,. \quad (9)$$

We optimize the model by using the point-wise ranking strategy and the cross-entropy loss:

$$\begin{aligned} \mathcal{L} &= \sum_{(q,T,c) \in Y} \mathcal{L}(q, T, c) \\ &= \sum_{(q,T,c) \in Y} -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}) \,, \end{aligned} \quad (10)$$

where Y is the training set, $\hat{y}$ is the binary gold-standard label for the input (q, T, c) representing whether c is relevant to q and similar to T, and y is the predicted score for c.

5 Experimental Design

We evaluated our GraDaSE and examined the following three research questions that characterize our contributions in this paper.

- **RQ1**: Relative to the current methods for DSE that are all based on text, is our graph representation of dataset relationships useful?
- **RQ2**: Compared with the existing plain use of node embeddings, is our incorporation of path structures more effective for DSE?
- **RQ3**: Is it possible to create a new test collection for DSE not restricted to government datasets, preferably from NLP resources?

	Domain	# datasets	# q	avg. $\|T\|$
DataFinder-E	Academia	7,650	2,523	1.38
DSEBench	Government	46,615	5,840	1.00

Table 1: Statistics of test collections.

5.1 Test Collections

We evaluated with two test collections. As shown in Table 1, their characteristics are complementary.

DSEBench (DSEBench, 2024) was the only test collection available to evaluate DSE, which was adapted from NTCIR (Kato et al., 2021), a test collection for ad hoc dataset search constructed from real needs for government datasets. For each input (q, T, c), DSEBench provides annotated graded relevance of c to q and graded similarity between c and T. According to its suggested configuration, (q, T, c) was considered a positive example iff c's relevance to q and similarity with T were both annotated positive. Note that DSEBench limited itself to a single target dataset, i.e. $|T| = 1$. We used its predefined train-valid-test splits.

As a new test collection for DSE, we constructed, and published under Apache License 2.0, **DataFinder-E** by adapting DataFinder (Viswanathan et al., 2023), a test collection for ad hoc dataset search over academic datasets from the NLP community. *Distinguished from the government datasets in DSEBench, this new resource answered **RQ3**.* Specifically, for each query q in DataFinder, from its $k \geq 2$ annotated relevant datasets, we randomly chose $k - 1$ as T, i.e. $|T| \geq 1$, with the remaining one as our positive example c. We followed the train-test splits of the queries in DataFinder. For the training set, we took the original hard negatives in DataFinder as our negative examples. We retained 20% of the training set for validation.

In both test collections, for queries in training and validation sets, the candidate datasets C consisted of all annotated positives and negatives. For queries in test sets, we expanded each query with the metadata of the target datasets and used BM25 to retrieve top-20 datasets as C, where the datasets not annotated positive were regarded negative.

5.2 Baselines

We compared our GraDaSE with a wide range of baseline methods adapted from related tasks.

Text-Based Methods To answer RQ1, we compared with four representative text-based retrieval or reranking models that expand the query with the textual metadata of the target datasets. **BM25** is exactly the method mentioned above that we used to retrieve candidate datasets for test queries. In other words, all the methods below reranked these candidates. **BGE**, or `bge-reranker-v2-m3` (Li et al., 2023; Chen et al., 2024a), is a popular dense reranking model. **SciBERT** (Beltagy et al., 2019) is a pre-trained scientific language model that achieved state-of-the-art results on DataFinder (Viswanathan et al., 2023). For BGE and SciBERT, we implemented two variants: expanding the query with the textual metadata of the target datasets, denoted by (qt), or ignoring the target datasets, denoted by (q). **GLM**, or `glm-4-plus` (GLM et al., 2024), is a popular large language model (LLM) with strong capabilities to process long texts. Our prompt is given in Appendix B.1.

Personalized Product Search We compared with two popular methods adapted from a comparable task, personalized product search. **ZAM** (Ai et al., 2019) is based on attention. **TEM** (Bi et al., 2020) is a transformer-based embedding model. We used them to index the metadata fields of a dataset as the attributes of a product and employ target datasets as search or purchase history.

Graph-Based Methods To answer RQ2, we compared with the two graph-based methods used in our approach. **SimpleHGN** (Lv et al., 2021) is a GAT-based heterogeneous graph encoder that introduces edge-type attention and residual connections. **HINormer** (Mao et al., 2023) is a heterogeneous graph encoder based on graph transformer that incorporates node-type attention. We directly applied them to our dataset graph. With the node embeddings they generated, we concatenated the mean embedding of the query nodes, the mean embedding of the target datasets, and the embedding of each candidate dataset, and fed it to an MLP to obtain a ranking score for this candidate dataset.

5.3 Implementation Details

We conducted experiments on GeForce RTX 4090. Our implementation used PyTorch 2.4.0, HuggingFace Transformers 4.44.2, and DGL 2.4.0.

We set the dimension of the embeddings $d = 256$. For target-biased input representation, we set the number of query nodes $k_q = 5$, the number of shortest paths $n_s = 10$, and the maximum length of a node sequence $l_s = 10$. The effects of these hyperparameters will be analyzed in Section 6.3. In

Model	MAP@5	NDCG@5	P@5	MAP@10	NDCG@10	P@10
			DataFinder-E			
Text-Based Methods						
BM25	0.0787	0.1141	0.0444	0.0855	0.1306	0.0273
BGE (q)	0.0884	0.1047	0.0313	0.0990	0.1305	0.0236
BGE (qt)	0.1056	0.1465	0.0538	0.1099	0.1563	0.0298
SciBERT (q)	0.1508	0.1757	0.0502	0.1559	0.1878	0.0287
SciBERT (qt)	0.1084	0.1382	0.0458	0.1166	0.1582	0.0291
GLM	0.0800	0.1092	0.0400	0.0935	0.1413	0.0298
Personalized Product Search						
ZAM	0.0710	0.0901	0.0298	0.0774	0.1056	0.0196
TEM	0.0505	0.0709	0.0269	0.0570	0.0871	0.0185
Graph-Based Methods						
SimpleHGN	0.1529	0.1758	0.0489	0.1590	0.1906	0.0290
HINormer	0.1354	0.1537	0.0415	0.1431	0.1724	0.0265
Our Approach						
GraDaSE	**0.1973***	**0.2163***	**0.0545**	**0.2018***	**0.2272***	**0.0307**
			DSEBench			
Text-Based Methods						
BM25	0.0982	0.3364	0.3872	0.1739	0.3630	0.3660
BGE (q)	0.1346	0.4096	0.3986	0.2014	0.4029	0.3546
BGE (qt)	0.1384	0.4176	0.3986	0.2069	0.4129	0.3610
SciBERT (q)	0.1383	0.4242	0.4014	0.2012	0.4046	0.3468
SciBERT (qt)	0.1203	0.3845	0.4085	0.1929	0.3888	0.3652
GLM	0.1089	0.3590	0.4000	0.1856	0.3791	0.3716
Personalized Product Search						
ZAM	0.0667	0.2745	0.2596	0.1137	0.2918	0.2681
TEM	0.0546	0.2517	0.2525	0.1046	0.2806	0.2723
Graph-Based Methods						
SimpleHGN	0.1077	0.3565	0.3348	0.1618	0.3498	0.3071
HINormer	0.0747	0.2921	0.2908	0.1242	0.2989	0.2809
Our Approach						
GraDaSE	**0.1521***	**0.4524***	**0.4190**	**0.2260***	**0.4387***	**0.3757**

Table 2: Comparison with baselines. The best results are in bold, and the second best results are underlined. Stars (*) indicate that our approach significantly outperforms all the baselines ($p < 0.05$).

structure-based node embedding, for the number of layers, we set $L_1 = 3$ for DataFinder-E and $L_1 = 1$ for DSEBench. In target-biased query and dataset encoding, the number of layers was set to $L_2 = 3$ for both test collections. We used a dropout rate of 0.1 and a batch size of 128. For the learning rate, we selected $1e-4$ from $\{5e-4, 1e-4, 5e-5\}$ for DataFinder-E, and selected $5e-5$ from $\{1e-4, 5e-5, 1e-5\}$ for DSEBench.

5.4 Metrics

Following common practice, we used MAP@5, NDCG@5, Precision@5 (P@5), MAP@10, NDCG@10, and Precision@10 (P@10) to evaluate the ranking scores of candidate datasets computed by each method, averaged over all test queries.

6 Experimental Results

6.1 Comparison with Baselines

Table 2 shows the mean evaluation results of each method on DataFinder-E and DSEBench.

GraDaSE achieved new state-of-the-art results on both test collections. It significantly outperformed all baselines in MAP and NDCG according to the paired t-test at $p < 0.05$, demonstrating the efficacy of our approach in exploiting textual metadata and extracted dataset relationships.

Model	**DataFinder-E**			**DSEBench**		
	MAP@10	NDCG@10	P@10	MAP@10	NDCG@10	P@10
GraDaSE	**0.2018**	**0.2272**	**0.0307**	**0.2260**	**0.4387**	**0.3757**
w/o path	0.1604	0.1923	0.0292	0.1602	0.3477	0.3156
w/o $\mathbf{g}_\sigma$ and $\mathbf{s}_\sigma$	0.1898	0.2146	0.0293	0.2075	0.4182	0.3600
w/o $\mathbf{g}_\sigma$	0.1931	0.2185	0.0298	0.2098	0.4155	0.3539
w/o $\mathbf{s}_\sigma$	0.1960	0.2214	0.0301	0.2169	0.4265	0.3671
w/o $\mathbf{g}_\tau$ and s_τ	0.1927	0.2197	0.0303	0.2160	0.4267	0.3667

Table 3: Ablation study of GraDaSE.

Compared to text-based baselines, including personalized product search methods, GraDaSE exceeded by at least 4.65% in MAP@5 and at least 4.06% in NDCG@5 on DataFinder-E, by at least 1.37% in MAP@5 and at least 2.82% in NDCG@5 on DSEBench. *These results indicated the usefulness of our graph representation of dataset relationships for DSE tasks and answered **RQ1**.*

Among the baselines, the text-based and graph-based methods led on different test collections, possibly related to the different distributions of edge types in the dataset graphs compared in Appendix A.2. *This complementarity suggested using structural and textual information simultaneously to achieve more robust results in DSE tasks*, as demonstrated by the better performance of our GraDaSE.

6.2 Ablation Study

We investigated the usefulness of the key components of our GraDaSE. Table 3 shows the results.

GraDaSE encodes all nodes in the shortest paths between candidate and target datasets to capture their similarity. In its variant "w/o path" where we removed intermediate nodes and only kept the candidate and target datasets, the performance largely decreased and became close to that of SimpleHGN, a graph-based baseline applying node embeddings in a straightforward way. *This comparison answered **RQ2** with the demonstrated superiority of our inclusion of path structures for explicitly characterizing multi-hop relationships between datasets*, which are selectively attended to in our model.

GraDaSE jointly encodes graph and text, with our main contribution on the graph structure. In its variant "w/o $\mathbf{g}_\sigma$ and $\mathbf{s}_\sigma$" where the graph encoding was removed, we observed notable performance declines, even if only removing either of them, i.e., in the variants "w/o $\mathbf{g}_\sigma$" and "w/o $\mathbf{s}_\sigma$". *These results supported the effectiveness of our design of the graph encoding module for DSE and strengthened our answer to **RQ1**.* Text encoding in GraDaSE was also helpful, as evidenced by the decreased

	Value	DataFinder-E			DSEBench		
		MAP@10	NDCG@10	P@10	MAP@10	NDCG@10	P@10
k_q	1	**0.2027**	**0.2286**	**0.0309**	**0.2261**	**0.4408**	**0.3766**
	2	**0.2027**	0.2280	0.0306	0.2259	0.4399	0.3763
	5*	0.2018	0.2272	0.0307	0.2260	0.4387	0.3757
	10	0.1927	0.2180	0.0297	0.2212	0.4343	0.3765
n_s	5	0.1996	0.2246	0.0303	0.2213	0.4325	0.3718
	10*	**0.2018**	**0.2272**	**0.0307**	**0.2260**	**0.4387**	**0.3757**
	15	0.1989	0.2234	0.0300	0.2204	0.4307	0.3685
	20	0.1983	0.2230	0.0301	0.2106	0.4203	0.3657
l_s	5	**0.2026**	**0.2273**	0.0305	**0.2358**	**0.4509**	**0.3817**
	10*	0.2018	0.2272	**0.0307**	0.2260	0.4387	0.3757
	15	0.2005	0.2255	0.0303	0.2205	0.4311	0.3712
	20	0.1973	0.2230	0.0303	0.2184	0.4329	0.3721

Table 4: Hyperparameter analysis of GraDaSE. Stars (*) indicate default values used in our main experiment.

performance of its variant "w/o $\mathbf{g}_{\mathcal{T}}$ and $s_{\mathcal{T}}$", which *confirmed the benefit of jointly encoding structural and textual information for DSE*. In addition, the performance of this variant was much higher than the baselines based on graphs in Table 2, indicating the effectiveness of our graph encoding compared to existing GNNs.

6.3 Hyperparameter Analysis

In GraDaSE, target-biased input representation is dependent on three hyperparameters: k_q representing the number of top-ranked nodes mapped to from a query, n_s representing the number of encoded shortest paths between each candidate dataset and the target datasets, and l_s representing the maximum length of an encoded node sequence (i.e. path). Table 4 presents our hyperparameter analysis and its impact on model performance.

For k_q, we intuitively set $k_q = 5$ in our main experiment, while GraDaSE performed even better when $k_q < 5$. Performance peaked at $k_q = 1$ on both test collections, showing that the inclusion of more query nodes introduced noisy information.

For n_s, our default value $n_s = 10$ representing a moderate choice appeared reasonable since the overall performance decreased with a lower value of this hyperparameter possibly due to the loss of crucial paths, and also decreased with a higher value for noise paths. This speculation was confirmed by a detailed analysis in Appendix C.2 where we reported the performance on different numbers of target datasets. As in our future work, a better strategy would be to dynamically adjust n_s according to the number of target datasets.

For l_s, better results were obtained when setting $l_s = 5$ or $l_s = 10$ but not to higher values where a lot of padding was required and therefore influenced performance. In fact, we observed that the average length of the shortest paths was around 6.

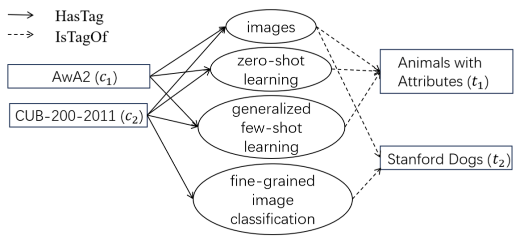
HasTag
IsTagOf
images
zero-shot learning
generalized few-shot learning
fine-grained image classification
AwA2 (c_1)
CUB-200-2011 (c_2)
Animals with Attributes (t_1)
Stanford Dogs (t_2)

Figure 3: An error case for the query "fine-grained image classification".

6.4 Error Analysis

We analyzed 50 error cases sampled from DataFinder-E, which were classified into three types of error: *misclassified relationships between datasets* (10%), *misleading provenance-based relationships* (12%), and *misleading topic-based relationships* (90%). Some cases were caused by multiple types of error, so their sum exceeded 100%.

For the most common type of error, misleading topic-based relationships, Figure 3 illustrates an error case. For the query "fine-grained image classification", among the two candidate datasets, c_2 is relevant to the query, while c_1 is irrelevant because it is not for a "fine-grained" setting. However, given two target datasets t_1 and t_2, since they share many tags with both c_1 and c_2, the target-biased representations for c_1 and c_2 become almost indistinguishable in our model, although the similarity captured by these paths is trivial or irrelevant to the query. In other words, in this case, the dataset graph provides misleading information for the relevance judgment. As a result, GraDaSE mistakenly predicted that c_1 is relevant. Note that this error was caused not by the quality of metadata but by our method's failure to distinguish between more informative and less informative metadata tags in the dataset graph. It inspired us to filter out trivial tags or irrelevant paths in future work to improve accuracy while preserving topic-based relationships to alleviate cold start issues.

6.5 Scalability Analysis

In GraDaSE, target-biased dataset representation searches for n_s shortest paths between each candidate dataset and the target datasets. In Appendix C.3, we analyzed its scalability. For $n_s \leq 20$, each search took less than 10 ms on DataFinder-E and less than 100 ms on DSEBench, demonstrating acceptable efficiency and practicability.

7 Conclusion

In this work, we propose GraDaSE, a graph-based approach to the emerging DSE task, and we contribute DataFinder-E, a new test collection for this task. GraDaSE complements existing text-based methods by mining provenance-based and topic-based relationships between datasets from their textual metadata to construct a domain-independent dataset graph, where path structures are employed to generate target-biased representations to capture query relevance and dataset similarity. GraDaSE significantly outperforms strong text-based and graph-based baselines on DataFinder-E and an existing test collection, DSEBench. Our work provides a promising tool for researchers and data journalists to more easily find suitable datasets.

In future work, we plan to enhance GraDaSE to address the shortcomings found in the experiments, including filtering tags and paths and dynamically adjusting the number of encoded shortest paths.

Limitations

Our work has the following limitations that are orthogonal to our research contributions but deserve exploration in future work. First, we follow existing product-level dataset search engines to only use the textual metadata of the datasets and do not rely on their data files because the actual data may be difficult to access or process due to their magnitude and heterogeneity. However, the quality of metadata in real scenarios may be unsatisfactory and the content of the data files has proven to be useful for ad hoc dataset search (Lin et al., 2022; Chen et al., 2024b; Zhou et al., 2025). Data files may also help to more accurately capture the similarity between candidate and target datasets in DSE. Second, we assume that the target datasets are truly relevant to the query, but in practice they may contain insufficient or inaccurate examples. Both our approach and the test collections can be extended to cover such more general settings.

References

Qingyao Ai, Daniel N. Hill, S. V. N. Vishwanathan, and W. Bruce Croft. 2019. A zero attention model for personalized product search. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 379–388, New York, NY, USA. Association for Computing Machinery.

Basmah Altaf, Uchenna Akujuobi, Lu Yu, and Xiangliang Zhang. 2019. Dataset recommendation via variational graph autoencoder. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 11–20.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: Pretrained language model for scientific text. In *EMNLP*.

Keping Bi, Qingyao Ai, and W. Bruce Croft. 2020. A transformer-based embedding model for personalized product search. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 1521–1524, New York, NY, USA. Association for Computing Machinery.

Alex Bogatu, Alvaro A. A. Fernandes, Norman W. Paton, and Nikolaos Konstantinou. 2020. Dataset discovery in data lakes. In *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*, pages 709–720. IEEE.

Dan Brickley, Matthew Burgess, and Natasha Noy. 2019. Google dataset search: Building a search engine for datasets in an open web ecosystem. In *The World Wide Web Conference*, WWW '19, page 1365–1375, New York, NY, USA. Association for Computing Machinery.

Adriane Chapman, Elena Simperl, Laura Koesten, George Konstantinidis, Luis-Daniel Ibáñez, Emilia Kacprzak, and Paul Groth. 2020. Dataset search: a survey. *VLDB J.*, 29(1):251–272.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024a. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *Preprint*, arXiv:2402.03216.

Qiaosheng Chen, Jiageng Chen, Xiao Zhou, and Gong Cheng. 2024b. Enhancing dataset search with compact data snippets. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pages 1093–1103. ACM.

Qiaosheng Chen, Weiqing Luo, Zixian Huang, Tengteng Lin, Xiaxia Wang, Ahmet Soylu, Basil Ell, Baifan Zhou, Evgeny Kharlamov, and Gong Cheng. 2024c. ACORDAR 2.0: A test collection for ad hoc dataset retrieval with densely pooled datasets and question-style queries. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pages 303–312. ACM.

Yujun Chen, Yuanhong Wang, Yutao Zhang, Juhua Pu, and Xiangliang Zhang. 2019. Amender: An attentive and aggregate multi-layered network for dataset recommendation. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 988–993.

DSEBench. 2024. A test collection for dataset search with examples. `https://github.com/nju-websoft/DSEBench`.

Michael Färber and Ann-Kathrin Leisinger. 2021. Recommending datasets for scientific problem descriptions. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, page 3014–3018, New York, NY, USA. Association for Computing Machinery.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Mingdao Liu, Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao, Shuxun Yang, Weng Lam Tam, Wenyi Zhao, Xiao Liu, Xiao Xia, Xiaohan Zhang, Xiaotao Gu, Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yifan An, Yifan Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang, Zhen Yang, Zhengxiao Du, Zhenyu Hou, and Zihan Wang. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *Preprint*, arXiv:2406.12793.

Ornella Irrera, Matteo Lissandrini, Daniele Dell'Aglio, and Gianmaria Silvello. 2024. Reproducibility and analysis of scientific dataset recommendation methods. In *Proceedings of the 18th ACM Conference on Recommender Systems*, RecSys '24, page 570–579, New York, NY, USA. Association for Computing Machinery.

Makoto P. Kato, Hiroaki Ohshima, Ying-Hsang Liu, and Hsin-Liang Chen. 2021. A test collection for ad-hoc dataset retrieval. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 2450–2456, New York, NY, USA. Association for Computing Machinery.

Chaofan Li, Zheng Liu, Shitao Xiao, and Yingxia Shao. 2023. Making large language models a better foundation for dense retrieval. *Preprint*, arXiv:2312.15503.

Kate Lin, Tarfah Alrashed, and Natasha Noy. 2024. Relationships are complicated! an analysis of relationships between datasets on the web. In *The Semantic Web – ISWC 2024*, pages 47–66, Cham. Springer Nature Switzerland.

Tengteng Lin, Qiaosheng Chen, Gong Cheng, Ahmet Soylu, Basil Ell, Ruoqi Zhao, Qing Shi, Xiaxia Wang, Yu Gu, and Evgeny Kharlamov. 2022. ACORDAR: A test collection for ad hoc content-based (RDF) dataset retrieval. In *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pages 2981–2991. ACM.

Qingsong Lv, Ming Ding, Qiang Liu, Yuxiang Chen, Wenzheng Feng, Siming He, Chang Zhou, Jianguo Jiang, Yuxiao Dong, and Jie Tang. 2021. Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 1150–1160, New York, NY, USA. Association for Computing Machinery.

Qiheng Mao, Zemin Liu, Chenghao Liu, and Jianling Sun. 2023. Hinormer: Representation learning on heterogeneous information networks with graph transformer. In *Proceedings of the ACM Web Conference 2023*, WWW '23, page 599–610, New York, NY, USA. Association for Computing Machinery.

Norman W. Paton, Jiaoyan Chen, and Zhenyu Wu. 2024. Dataset discovery and exploration: A survey. *ACM Comput. Surv.*, 56(4):102:1–102:37.

Liam Roditty and Uri Zwick. 2012. Replacement paths and k simple shortest paths in unweighted directed graphs. *ACM Trans. Algorithms*, 8(4).

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. *Preprint*, arXiv:1710.10903.

Vijay Viswanathan, Luyu Gao, Tongshuang Wu, Pengfei Liu, and Graham Neubig. 2023. DataFinder: Scientific dataset recommendation from natural language descriptions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10288–10303, Toronto, Canada. Association for Computational Linguistics.

Xu Wang, Frank van Harmelen, Michael Cochez, and Zhisheng Huang. 2022. Scientific item recommendation using a citation network. In *Knowledge Science, Engineering and Management: 15th International Conference, KSEM 2022, Singapore, August 6–8, 2022, Proceedings, Part II*, page 469–484, Berlin, Heidelberg. Springer-Verlag.

Dun Zhang, Jiacheng Li, Ziyang Zeng, and Fulong Wang. 2025. Jasper and stella: distillation of sota embedding models. *Preprint*, arXiv:2412.19048.

Wenjia Zhang, Lin Gui, Rob Procter, and Yulan He. 2024. Multi-layer ranking with large language models for news source recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 2537–2542, New York, NY, USA. Association for Computing Machinery.

Xiao Zhou, Qiaosheng Chen, Jiageng Chen, and Gong Cheng. 2025. μds: Multi-objective data snippet extraction for dataset search. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2025, Padua, Italy, July 13-18, 2025*, pages 116–126. ACM.

A Details of Dataset Graphs

A.1 Construction Method

To train a dataset relationship classifier, we use labeled relationships between Web datasets.[1] For each positive example, that is, a pair of datasets having a known relationship, we sample a negative example by replacing either dataset in the pair with a random one. We divide all the examples into 70% for training, 15% for validation, and 15% for testing. Then we train a Gradient Boosting Decision Tree (GBDT) classifier (Lin et al., 2024).

Our preliminary analysis of the test results reveals that the classifier tends to misclassify `Replica` as `Variant`. To fix this issue, we incorporate a heuristic based on the string Levenshtein distance between dataset titles to refine the classification results. As a result, the classifier achieves an accuracy greater than 95% on the test set.

With this classifier, for each dataset in the test collections used in our experiments, we retrieve its top-20 nearest neighbors obtained based on the cosine similarity between their metadata representations encoded by `stella_en_400M_v5` (Zhang et al., 2025), and then use the classifier to predict a relationship (or no relationship) between this dataset and each of its neighboring datasets.

A.2 Statistics

For each test collection in the experiments, we constructed a dataset graph over all datasets. As shown in Table 5, the dataset graph for DSEBench is larger and denser mainly for `Variant` and tag-based edges. In contrast, the dataset graph for DataFinder-E contains more `Replica` edges.

	DataFinder-E	DSEBench
Nodes	10,950	107,310
`Dataset`	7,650	46,615
`Tag`	3,300	60,695
Edges	81,388	2,779,185
`Replica`	38,826	2,764
`Version`	48	60
`Subset`	683	1,268
`Superset`	683	1,268
`Derivation`	366	359
`Variant`	4,234	1,299,986
`HasTag`	18,274	736,740
`IsTagOf`	18,274	736,740

Table 5: Statistics of dataset graphs.

[1]https://figshare.com/articles/dataset/Metadata_for_Datasets_and_Relationships/22790810

B Details of Experimental Design

B.1 Prompt for GLM

We followed Zhang et al. (2024) to randomly divide 20 candidate datasets into 5 groups and prompted GLM to rerank each group. We repeated this process 10 times. The ranking score of a candidate dataset was given by its frequency of appearing in the top half of a reranked group. We used the following prompt to rerank a group according to the query and the target datasets.

> You are a relevance ranker specializing in reranking candidate datasets based on a keyword query and some target datasets' description.
>
> You are given a keyword query and some target datasets. Based on these inputs, a search system has already retrieved a set of candidate datasets. Your task is to rerank these candidate datasets so that those most relevant to the input are listed first. Each dataset has a unique ID and some descriptive fields (Title, Description, Tags).
>
> Please rank all the candidate datasets from most to least relevant to the keyword query and the target dataset. The output should be a list of IDs in the format: '[ID_1, ID_2, ID_3, ID_4]' without any additional words or explanations.
>
> Here is an example: {*example*}
>
> Now the inputs are: {*query*}. Target Datasets: {*target_datasets*}
>
> The candidate datasets (separated by 'separator') are: {*candidate_datasets*}
>
> Now please provide the ranked list of candidate dataset IDs in the specified format: [ID_1, ID_2, ID_3, ID_4]. Don't output other words.

C Additional Experiments

C.1 Order of Query Nodes and Target Datasets in S_{qt}

In GraDaSE, we concatenate query nodes with target datasets to form S_{qt} as a target-biased query representation. As shown in Table 6, by changing the order of the query nodes and target datasets in S_{qt}, the influence on the performance of the approach on both test collections is insignificant according to the paired t-test at $p > 0.2$.

S_{qt}	MAP@5	NDCG@5	P@5	MAP@10	NDCG@10	P@10
DataFinder-E						
$[q_1 \| \cdots \| q_{k_q} \| t_1 \| \cdots \| t_{k_t}]$	0.1973	0.2163	0.0545	0.2018	0.2272	0.0307
$[t_1 \| \cdots \| t_{k_t} \| q_1 \| \cdots \| q_{k_q}]$	0.1967	0.2167	0.0553	0.2006	0.2266	0.0308
DSEBench						
$[q_1 \| \cdots \| q_{k_q} \| t_1 \| \cdots \| t_{k_t}]$	0.1521	0.4524	0.4190	0.2260	0.4387	0.3757
$[t_1 \| \cdots \| t_{k_t} \| q_1 \| \cdots \| q_{k_q}]$	0.1510	0.4560	0.4235	0.2226	0.4351	0.3713

Table 6: Experimental results of GraDaSE with different orders of query nodes and target datasets in S_{qt}.

C.2 Different Numbers of Target Datasets

Table 7 shows the mean evaluation results of GraDaSE on different numbers of target datasets (i.e., $|T|$) in DataFinder-E. Recall that we set the number of encoded shortest paths between each candidate dataset and the target datasets to $n_s = 10$. The best results were obtained when $|T| = 3$. For smaller $|T|$, the performance decreased slightly possibly due to excessive noise paths. For larger $|T|$, the performance decreased noticeably, indicating that $n_s = 10$ seemed too small to adequately capture the rich connections to a relatively large set of target datasets. In practical scenarios, users are typically not expected to provide many examples, so this default value may be reasonable, although a generally better strategy would be to dynamically adjust n_s according to user input, which is left for our future work.

$\|T\|$	MAP@5	NDCG@5	P@5	MAP@10	NDCG@10	P@10
1	0.2076	0.2229	0.0538	0.2139	0.2379	0.0315
2	0.2202	0.2440	0.0629	0.2223	0.2496	0.0333
3	0.2379	0.2595	0.0648	0.2448	0.2762	0.0376
4	0.0873	0.0977	0.0256	0.0891	0.1025	0.0144
≥ 5	0.0364	0.0502	0.0187	0.0418	0.0632	0.0133

Table 7: Experimental results of GraDaSE on different numbers of target datasets in DataFinder-E.

C.3 Scalability Analysis

To analyze the scalability of target-biased dataset representation in GraDaSE, for different values of n_s, we measured the mean running time of this search of n_s shortest paths on dataset graphs of different sizes. We conducted this experiment on Xeon Gold 6326. Figure 4 presents the results.

Not surprisingly, running time increased as n_s increased. However, even for $n_s = 20$, each search took an average of less than 10 milliseconds on DataFinder-E which contains 10K nodes and 81K edges, and less than 100 milliseconds on DSEBench which contains 107K nodes and 2.7M edges. *This acceptable efficiency demonstrated the practicability of shortest path search in GraDaSE.* It has the potential to scale to larger values of n_s and larger dataset graphs.

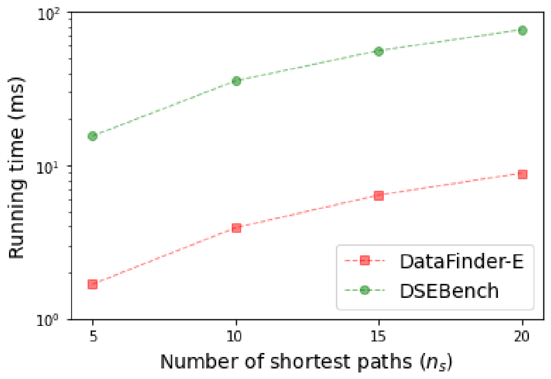

Figure 4: Scalability analysis of shortest path search in GraDaSE.