Leveraging Schema Labels to Enhance Dataset Search

Zhiyu Chen(✉), Haiyan Jia, Jeff Heflin, and Brian D. Davison

Lehigh University, Bethlehem, PA, USA
{zhc415,haiyan.jia}@lehigh.edu, {heflin,davison}@cse.lehigh.edu

Abstract. A search engine's ability to retrieve desirable datasets is important for data sharing and reuse. Existing dataset search engines typically rely on matching queries to dataset descriptions. However, a user may not have enough prior knowledge to write a query using terms that match with description text. We propose a novel schema label generation model which generates possible schema labels based on dataset table content. We incorporate the generated schema labels into a mixed ranking model which not only considers the relevance between the query and dataset metadata but also the similarity between the query and generated schema labels. To evaluate our method on real-world datasets, we create a new benchmark specifically for the dataset retrieval task. Experiments show that our approach can effectively improve the precision and NDCG scores of the dataset retrieval task compared with baseline methods. We also test on a collection of Wikipedia tables to show that the features generated from schema labels can improve the unsupervised and supervised web table retrieval task as well.

Keywords: Dataset search · Table retrieval · Text normalization · Data fusion

1 Introduction

Dataset retrieval is receiving more attention as people from different fields and domains start to rely on datasets for their work. There are many data portals with the purpose of effective and efficient data management and data sharing, such as data.gov[1], datahub[2] and data.world[3]. Most of those data portals use CKAN[4] as their backend. However, there are two problems of dataset search engines using such infrastructure: First, ranking performance relies on the quality of metadata of datasets, while many datasets lack high quality metadata; second, the information in the metadata may not satisfy the user's information need or help them solve their task [3]. A user may not know the organization of a

[1] https://www.data.gov/.
[2] http://datahub.io/.
[3] https://data.world/.
[4] https://docs.ckan.org/.

J. M. Jose et al. (Eds.): ECIR 2020, LNCS 12035, pp. 267–280, 2020.
https://doi.org/10.1007/978-3-030-45439-5_18

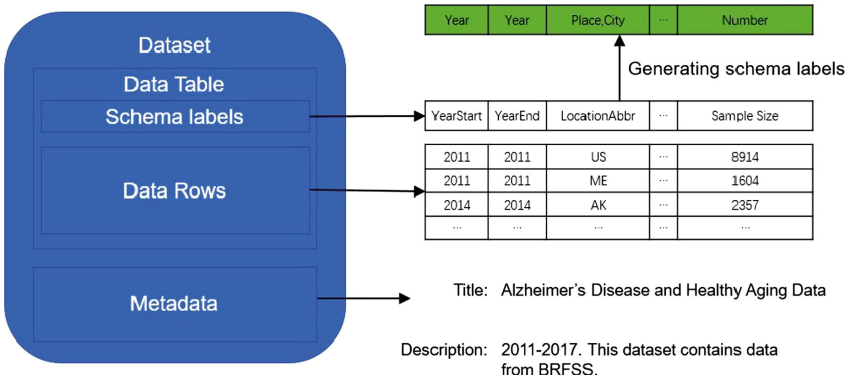

Fig. 1. The structure of a dataset. Metadata includes the title and any description. A trained schema label generator is used to generate additional schema labels (green part) from similar data tables. (Color figure online)

potentially relevant dataset, or the tags data publishers provide with a dataset. Such information can hardly be used for dataset ranking.

In this paper, we focus on the problem of dataset retrieval where dataset content is in tabular form, since tabular data is widely-used and easy to read and write. As illustrated in Fig. 1, a dataset consists of a data table (dataset content) and metadata. A data table usually has one header row, followed by one or more data rows. The header row consists of a list of **schema labels** (attribute names) whose actual values are stored in data rows. Metadata usually includes title and description of the dataset.

Schema labels, which represent high-level concepts, are underutilized if we directly score them with a user query. Consider the example in Fig. 1; the vocabulary of schema labels could be very different from other fields and user queries. "LocationAbbr", standing for "Location Abbreviation", is unlikely to appear in a user query so this dataset is less likely to be recalled. However, we can enhance this dataset by generating schema labels such as "place" and "city" appearing in other, similar datasets, which could provide a better soft-matching signal with respect to a user query, and therefore increase the chance that it can be recalled.

In this work, we first propose a new method for schema label generation. We learn latent feature representations of schema labels automatically by jointly decomposing the dataset-schema label interaction matrix and schema label-schema label interaction matrix. Then we propose a framework for enhancing dataset retrieval by schema label generation to address the problem that schema labels are not effectively used by existing dataset search engines. We create a new public benchmark[5] based on federal (U.S.) datasets and use it to demonstrate the effectiveness of our proposed framework for dataset retrieval. We additionally consider a web table retrieval task and demonstrate that the features generated from schema labels can be effective for supervised ranking.

[5] Available via https://github.com/Zhiyu-Chen/ECIR2020-dataset-search.

2 Related Work

Dataset search has become a new research field with new challenges. Chapman et al. [3] classify dataset search into *basic* and *constructive* dataset search. Basic dataset search returns a list of existing datasets based on a user's query, while constructive dataset search [5] generates datasets on-the-fly based on a user's needs and query. Google recently released a dataset search service[6]. Like many other data portals, their service relies on metadata of datasets, annotated on web pages using a standard defined by schema.org.

Other work on applications of Web tables is also related to our work. Cafarella et al. [2] proposed WebTables system which extract Web tables from top ranked pages by keyword search. Sekhavat et al. [13] proposed a probabilistic method that augments an existing knowledge base with facts from Web tables. Zhang et al. [16] developed generative probabilistic models to equip spreadsheets with smart assistance capabilities. Specifically, given a table, they recommend additional rows and column headings by leveraging the information from the Web tables. They also developed semantic matching features for table retrieval [17].

The techniques designed for Web table analysis could potentially be applied to dataset search. In our work, each dataset is associated with data in tabular form. Extracting useful information from tables such as entities and attribute names could help with the retrieval task. Trabelsi et al. [14] recently proposed custom embeddings for column headers based on multiple contexts for table retrieval, and found representing numerical cell values to be useful. Zhang et al. [16] proposed to use semantic concepts to represent queries and tables for ranking entity-focused tables. However, dataset search could be inherently more difficult since datasets do not need to be entity-focused.

3 Schema Label Enhanced Ranking

In this section, we introduce the framework of schema label enhanced dataset retrieval. As illustrated in Fig. 2, our framework has two stages: in the first stage, we first train a schema label generator with the method proposed in Sect. 3.1 and use it to generate additional schema labels for all the datasets; in the second stage, we use a mixed ranking model to combine the scores of schema labels and other fields for dataset ranking. In the following subsections, we present a detailed illustration of the two stages.

3.1 Schema Label Generation

We propose to improve dataset search by making use of generated schema labels, since these can be complementary to the original schema labels and especially valuable when they are otherwise absent from a dataset.

[6] https://toolbox.google.com/datasetsearch.

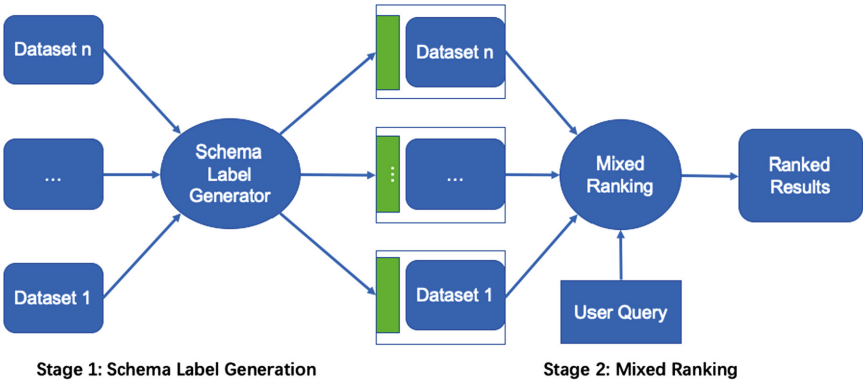

Fig. 2. The proposed schema label enhanced dataset retrieval framework. The green blocks indicate generated schema labels for different datasets. (Color figure online)

We treat schema label generation as a multi-label classification problem. Let $L = \{l_1, l_2, ..., l_k\}$ denote the labels appearing in all datasets and $D = \{(\mathbf{x}^i, \mathbf{y}^i)|1 \leq i \leq n\}$ denote the training set. Here, for each training sample $(\mathbf{x}^i, \mathbf{y}^i)$, $\mathbf{x}^i$ is a d-dimensional feature vector of column i which can be calculated from data rows [4] or learned from matrix factorization proposed later in this section. $\mathbf{y}^i$ is k-dimensional vector $[y_1^i, y_2^i, ..., y_k^i]$ and $y_j^i = 1$ only if x_i is relevant to label l_j, otherwise $y_j^i = 0$. Our objective is to learn a function that models $P(l|x_i)$, $(l \in L)$. To generate m schema labels for column i, we can select the top m labels L_m by:

$$L_m = \arg\max_{l \in L_m \subseteq L} P(l|x_i)$$

We could also generate schema labels by selecting a probability threshold θ:

$$L_m = \{l \in L|P(l|x_i) \geq \theta\}$$

In practice, we could first generate the top m schema labels and filter out those results with a probability lower than the threshold.

Chen et al. [4] proposed to predict schema labels based on curated features of data values. Instead of designing curated features for schema labels, we consider learning their representations in an automated manner. Inspired by collaborative filtering methods in recommender systems, we model each dataset as a user and each schema label as an item. Then a dataset with a schema label can be considered as positive feedback between a user and an item. By exploiting the user-item co-occurrences and item-item co-occurrences, we can learn the latent representations of schema labels. In the following, we show how to construct a preference matrix in the context of schema label generation and how to learn the schema label features.

Preference Matrix Construction. With m data tables and n unique schema labels, we can construct a dataset-column preference matrix $M^{m \times n}$, where M_{up} is 1 if dataset u contains schema label p.

Matrix Factorization. MF [7] decomposes M into the product of $U^{m\times k}$ and $P^{k\times n}$ where $k < min(m, n)$. U^T can be denoted as $(\alpha_1, ..., \alpha_u ..., \alpha_m)$ where $\alpha_u \in R^k$ represents the latent factor vector of dataset u. Similarly, P^T can be denoted as $(\beta_1, ..., \beta_p ..., \beta_n)$ where $\beta_p \in R^k$ represents the latent factor vector of schema label p. Since the preference matrix actually models the implicit feedback, MF optimizes the following objective function:

$$\mathcal{L}_{mf} = \sum_{u,p} c_{up}(M_{up} - \alpha_u^T \beta_p)^2 + \lambda_\alpha \sum_u \|\alpha_u\|^2 + \lambda_\beta \sum_p \|\beta_p\|^2 \quad (1)$$

where c_{up} is a hyperparameter tuned to balance the non-zero and zero values since M is a sparse matrix. λ_α and λ_β are regularization parameters that adjust the importance of regularization terms $\sum_u \|\alpha_u\|^2$ and $\sum_p \|\beta_p\|^2$.

Label Embedding. Recently, word embedding techniques (e.g., word2vec [11]) have been valuable in natural language processing tasks. Given a sequence of words, a low-dimensional continuous representation called word embedding can be learned for each word. Word2vec's skip-gram model with negative sampling (SGNS) is equivalent to implicitly factorizing a word-context matrix, whose cells are the pointwise mutual information (PMI) of the respective word and context pairs, shifted by a global constant [9]. The PMI between word i and its context word j is defined as:

$$PMI(i, j) = log \frac{P(i, j)}{P(i) \times P(j)} = log \frac{\#(i, j) \times |D|}{\sum_j \#(i, j) \times \sum_i \#(i, j)}$$

where $\#(i, j)$ is the number of times word j appears in the context window of word i and $|D|$ is the total number of word-context pairs. Then, a shifted positive PMI (SPPMI) of word i and word j is calculated as:

$$SSPMI(i, j) = max\{PMI(i, j) - log(k), 0\} \quad (2)$$

where k is the number of negative samples of SGNS. Given a corpus, matrix M^{SPPMI} can be constructed based on Eq. (2) and factorizing it is equivalent to performing SGNS.

A schema label exists in the context of other schema labels. Therefore, we perform word embedding techniques to learn the latent representations of schema labels. However, we do not consider the order of schema labels. Therefore, given a schema label, all other schema labels which come from the same data table are considered as its context. With the constructed SSPMI matrix of co-occurring schema labels, we are able to decompose it to learn the latent representations of schema labels.

Joint Learning of Schema Label Representations. Schema label representations learned from MF capture the interactive information between datasets and schema labels, while the word2vec style representations explain the co-occurrence relationships of schema labels. We use the CoFactor model [10] to

jointly learn schema label representations from both dataset-label interaction and label-label interaction:

$$\begin{aligned}\mathcal{L} = & \overbrace{\sum_{u,p} c_{up}(M_{up} - \alpha_u^T \beta_p)^2}^{MF} \\ & + \overbrace{\sum_{M_{pi}^{SPPMI} \neq 0} (M_{pi}^{SPPMI} - \beta_p^T \gamma_i - b_p - c_i)^2}^{schema\ label\ embedding} \\ & + \lambda_\alpha \sum_u \|\alpha_u\|^2 + \lambda_\beta \sum_p \|\beta_p\|^2 + \lambda_\gamma \sum_i \|\gamma_i\|^2\end{aligned} \tag{3}$$

From the objective function we can see the schema label representation β_p is shared between MF and schema label embedding. γ_i is the latent representation of context embedding. b_p and c_i are the schema label embedding bias and context embedding bias, respectively. The last line of Eq. 3 incorporates regularization terms with different λ controlling their effects. We use the vector-wise ALS algorithm [15] to optimize the parameters.

Schema Label Generation. After obtaining the jointly learned representations of schema labels, we can use them as features for schema label generation. In this paper, we use the concatenation of schema label representations introduced here and the curated features proposed by Chen et al. [4] to construct each x^i. Any multi-label classification models can be used to train the schema label generator and in this paper we choose Random Forest.

3.2 The Mixture Ranking Model

Based on the schema label generation method proposed above, we index the generated schema labels for each dataset. Now, each dataset has the following fields: metadata, data rows, schema labels and generated schema labels. A straightforward way to rank datasets is to use traditional ranking methods for documents.

Zhang and Balog [17] represent tables as single field documents or multifield documents for table retrieval task. For *single field document representation*, a dataset is treated as a single document by concatenating the text from all the fields. Then traditional methods such as BM25 can be used to score the dataset. For *multifield document representation*, each field is scored independently against the query and a weighted sum is used for ranking.

In our **Schema Label Mixed Ranking (SLMR)** model, we score schema labels differently from other fields. The focus of our work is to learn how schema labels, data rows and other metadata may differently influence dataset retrieval performance. Note that, for simplicity, we consider the other metadata (title and description) as a single text field, since title and description are homogeneous compared with schema labels and data rows. Therefore, we have the following

scoring function for a dataset D:

$$score(q, D) = \sum_{i \in \{text, data\}} w_i \times score_{text}(q, F_i) + w_l \times score_l(q, F_l) \quad (4)$$

where F_{text} denotes the concatenation of title and description, F_{data} denotes the data table, and F_l denotes the generated schema labels. Each field has a corresponding weights. F_{text} and F_{data} have the same scoring function $score_{text}$ while F_l has a different scoring function $score_l$. For F_{text} and F_{data}, we can use a standard scoring function for normal documents. In the experiments, we use BM25 as $score_{text}$.

Due to the existence of a large number of non-dictionary words in schema labels [4] that would otherwise be outside of the vocabulary of a word-based embedding, we represent schema labels and query terms using fastText [1] in $score_l$, since such word embeddings are calculated from character n-grams instead of terms. To score the schema labels with respect to a query, we use the negative Word Mover's Distance (WMD) [8]. WMD measures the dissimilarity between two text documents as the minimum amount of distance that the word embeddings of one document need to "travel" to reach the word embeddings of another document. So $score_l(q, F_l) = -wmd(fasttext(q), fasttext(F_l))$ reflects the semantic similarity between a query and schema labels.

4 Data Collection

Here we describe how we construct the new benchmark for dataset retrieval in detail. We collected 2417 resources published by the U.S. federal government from Data.gov which cover a variety of topics. Each resource includes one or more CSV format data tables and corresponding metadata. Each CSV table is treated as a single dataset and we use the resource-level metadata to annotate each dataset.

4.1 Task Creation and Query Collection

We created six tasks in which each describes a separate information need to find one or more datasets. For each, we have a statement about the information need which describes what datasets are considered as relevant. We additionally verified for each task the existence of at least one relevant dataset. The dataset is public available[7].

We used Amazon Mechanical Turk[8] to obtain diverse queries for these tasks from real users. Every annotator was presented with the task descriptions and asked to provide a query for each created task. To avoid the impact of task order on the quality of annotations, we randomly shuffled the order of tasks for each annotator. We paid one dollar for each completed annotation job and 20 queries were collected for each task. Every collected query was manually examined and obviously unrelated queries were excluded from the collection.

[7] Available from https://github.com/Zhiyu-Chen/ECIR2020-dataset-search.
[8] https://www.mturk.com/.

Table 1. For each task, the number of pairs assigned to each relevance label.

Task #	Off topic	Poor	Good	Excellent
1	1006	34	37	64
2	164	248	585	308
3	300	270	456	153
4	246	324	660	289
5	162	246	355	198
6	181	303	614	367

4.2 Relevance Assessments

For each task and each suggested query, we used traditional ranking functions to score single field representations of each dataset and collect the top 100 results. The following ranking models were used: BM25, TF-IDF, Language model based on Jelinek-Mercer smoothing, and Language Model with Dirichlet smoothing. We also used each model with two different representations: the concatenation of all fields of the dataset and the concatenation of title and description. This leads to eight baselines for the pooled results.

Then, the collected task-dataset pairs were annotated for relevance using the crowdsourcing service provided by Figure Eight[9]. We did not annotate the *query*-dataset pairs because the goal of dataset retrieval is to find relevant datasets with respect to a *task* which represents the real information need.

Annotators were presented with the task title, description and link to the data table. Each task-dataset pair was judged on a four point scale: 0 (off topic), 1 (poor), 2 (good), and 3 (excellent).[10] Every annotator was paid 10 cents per task-dataset judgement.

Every single task-dataset pair was judged by three annotators and we take the majority vote as the relevance label. If no majority agreement is achieved, we take the average of the scores as the final label. The statistics of annotation results is shown in Table 1.

[9] https://www.figure-eight.com/.

[10] The following labeling guidance was provided to annotators: *a dataset is off topic if the information does not satisfy the information need, and should not be listed in the search results from a search engine; a dataset is poor if a search engine were to include this in the search results, but it should not be listed at the top; a dataset is good if you would expect this dataset to be included in the search results from a search engine; a dataset is excellent if you would expect this dataset ranked near the top of the search results from a search engine.*

Table 2. NDCG@k and Precision@k of different models on dataset retrieval. The superscript + shows statistically significant improvements for our SLMR model over other single and multifield document ranking models. T means title, D means description, DT means data table, G means generated schema labels.

Method	Used fields	NDCG@5	@10	@20	@50	P@5	@10	@20	@50
SDR	T+D	0.8920	0.8490	0.8222	0.8121	0.4122	0.3652	0.3452	0.3585
SDR	DT	0.7378	0.7036	0.6964	0.7107	0.2856	0.2974	0.2931	0.3122
SDR	T+D+DT	0.8435	0.7954	0.7763	0.7785	0.2574	0.2870	0.3170	0.3357
MDR	T+D+DT	0.9285	0.8874	0.8683	0.8631	0.4086	0.3612	**0.4026**	0.3767
SLMR	T+D+G	**0.9293**$^{+}$	**0.8898**	**0.8722**$^{+}$	**0.8662**	**0.5000**$^{+}$	**0.4388**$^{+}$	0.4000	0.3761
SLMR	T+D+DT+G	0.9169	0.8808	0.8680	0.8555	**0.5000**$^{+}$	0.4345^{+}	0.4013	**0.3783**

5 Evaluation

5.1 Evaluation Metrics

We evaluate dataset retrieval performance over a range of metrics: Precision at k and Normalized Discounted Cumulative Gain (NDCG) at k [6]. To test the significance of differences between model performances, we use paired t-tests with significance at the $p = 0.01$ level.

5.2 Baselines

We first present the baseline retrieval methods.

Single-Field Document Ranking (SDR). A dataset is considered as a single document. We use BM25 to score the concatenation of title and description, the text of the data table and the concatenation of all of them. By comparing the three results, we can learn about field level importance for dataset retrieval. Parameters are chosen by grid search.

Multifield Document Ranking (MDR). By setting $w_l = 0$, Eq. (4) degenerates to the Mixture of Language Models [12]. BM25 is also used here as $score_{text}()$ in order to have a fair comparison with other methods. To optimize field weights, we use coordinate ascent. Finally, smoothing parameters are optimized in the same manner as single-field document ranking.

5.3 Experimental Results

In this section, we examine the following research questions:

Q1 Does data table content help in dataset retrieval?
Q2 Do generated schema labels help in dataset retrieval?
Q3 Which fields are most important for the dataset retrieval task?

We first obtain features of schema labels as described in Sect. 3.1 and the number of latent factors is set to 40. Then we train a Random Forest with the learned schema label features. The scikit-learn implementation of Random Forest[11] is used with default parameters except the number of trees is set to 25. In practice, we could choose any multi-label classifier. For each column, we select the top 10 generated schema labels and filter those with probability lower than 0.5. For each dataset, we index the generated schema labels as an additional field. Table 2 summarizes the NDCG at k and Precision at k of different models. Note that, for Schema Label Mixed Ranking (SLMR), we trained three different models and the weights of used fields were forced to be non-zero in order to study the proposed research questions. The weights of used fields for multifield document representation are also set non-zero when optimizing the parameters.

From the results of single-field document ranking, we can see that only utilizing the data table for ranking leads to the worst performance. Scoring on the concatenation of title and description achieved the best results, which indicates that title and description are more important than the data table for ranking a dataset (**Q3**). Treating all fields of a dataset as a single-field document provides performance between the previous two models. This result is expected since the length of data tables are usually much larger than titles and descriptions, and therefore dominate the table representation.

By comparing the results of single-field and multifield document ranking, we observe that the combination of the scores of data table, title and description could improve NDCG@k. Though NDCG@k decreases when k increases, the relative improvement against single-field document ranking are more significant. In contrast, for Precision@5, Precision@10, single-field document ranking performs better than multifield document ranking, though the differences are small. So for **Q1**, under the setting of multifield document ranking, the content of the data table could help NDCG, but not help Precision of dataset retrieval results.

Without scoring data tables, our proposed schema label mixed ranking approach achieves the highest NDCG on all the rank cut-offs, which indicates that the generated schema labels can be useful to improve the NDCG of dataset retrieval results (**Q2**). Though Precision@20 of multifield document ranking are higher than our proposed model, the difference is no more than 0.4% ($p\_value > 0.9$). Significantly, our model outperforms by 21.3% for Precision@5 ($\frac{0.5-0.4122}{0.4122}$) and by 20.1% for Precision@10 ($\frac{0.4388-0.3652}{0.3652}$) than the best baseline methods ($p\_value < 0.01$). Whether data tables are scored or not, Precision@k is not significantly different for schema label mixed ranking. Therefore, under the setting of schema label mixed ranking, data tables make little contribution in this scenario (**Q1**). One possible reason could be that data tables collected from data.gov contain large quantities of numerical values and will rarely be used to match user queries.

If a schema label mixed ranking model scores only on titles and descriptions ($w_l = 0$), it is equivalent to single-field ranking model scoring on titles and

[11] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifer.html.

Table 3. Supervised ranking results on table retrieval.

Method	NDCG@5	@10	@15	@20
STR [16]	0.6366	0.6571	0.663	0.6632
Schema Label Features	0.4489	0.5201	0.534	0.5347
STR + Schema Label Feat	**0.6530**	**0.6728**	**0.6789**	**0.6761**

descriptions. Therefore, we can compare the results in first and fifth rows in Table 2. With generated schema labels, the ranking model can have a higher performance on dataset retrieval task (**Q2**).

5.4 Schema Label Generation Enhanced Search for Web Tables

The task of dataset search is similar to Web table search since both tasks use table structure to represent data. The difference is that a large amount of Web tables are entity focused and contain many named entities that can be linked to a knowledge base. However, our datasets collected from the data.gov data portal contain few useful entities in the table. Therefore, a lot of methods designed for Web table ranking cannot be applied to dataset search. The semantic table retrieval (STR) method proposed by Zhang and Balog [16] relies on features from knowledge bases (bag of entities) which are not generally available for the scenario of dataset search. However, the schema label generation based method can be applied to table search. Thus, we performed additional experiments to show the performance of our method for the table search scenario.

We first generate schema labels for the table corpus shared by Zhang and Balog [16] using the method proposed in Sect. 3.1. Then we append five additional features to their proposed features[12] based on schema labels. Each feature is one type of semantic similarity between query and schema labels. Four features are calculated using the measurement proposed by Zhang and Balog (one early fusion feature, three late fusion features) and the last feature is the negative of Word Mover's Distance. Finally, like Zhang and Balog, we use Random Forest to perform pointwise regression and the final reported results are averaged over five runs of 5-fold cross-validation and shown in Table 3.

We can see that schema label features along cannot outperform STR. But combining them results in improvement. However, by calculating the normalized feature importance measured in terms of Gini score, we find that for STR with schema label features, WMD based measurement contributes the most among all the semantic features. Thus it demonstrates that the schema labels can be valuable for the table retrieval task as well.

Notably, in this table corpus, many tables lack much table content but contain rich text descriptions, which could be unfair for schema label generation-based methods. While for dataset search, each table has values but may lack high quality dataset descriptions. We believe that our schema label generation method

[12] https://github.com/iai-group/www2018-table/tree/master/feature.

can outperform STR in the scenario where text descriptions provide less useful information than the table itself.

Table 4. Unsupervised ranking results on table retrieval.

Used fields	NDCG@5	@10	@15	@20
text	0.3724	0.3891	0.4009	0.4178
text + data table	0.3901	0.4042	0.4422	0.4686
text + data table + generated labels	**0.4006**	**0.4118**	**0.4495**	**0.4766**
text + data table + original labels	0.3930	0.4055	0.4457	0.4709
text + original labels	0.3785	0.3934	0.4110	0.4283
text + generated labels	0.3808	0.3955	0.4064	0.4197

We also show unsupervised ranking results with Eq. 4 in Table 4. Unlike Zhang and Balog [16], we consider page title, section title and caption as a single text field, in order to reduce the number of hyperparameters (field weights). The results show that generated labels are more effective than original labels for table ranking. It is unsurprising because generated labels often include not only original labels but also additional labels that can benefit the ranking model. We also notice that including the data table field achieves better results than not scoring it, which is contrary to the results of dataset ranking. It is also expected since WikiTables are entity-focused and include a lot of text information while data tables from data.gov include more numeric values.

6 Conclusion

In this paper, we have proposed a schema label enhanced ranking framework for dataset retrieval. The framework has two stages: in the first stage, a schema label generator is trained to generate additional schema labels for each dataset column; in the second stage, given a user query, datasets are ranked by their original fields together with generated schema labels. Schema label generation is treated as a multi-label classification task in which each column of a dataset is associated with multiple schema labels. Instead of using hand-curated features, we learn the latent feature representations of schema labels by a CoFactor model in which the dataset-schema label interactions and schema label-schema label interactions are captured. With the schema label mixed ranking model, the traditional ranking scores for text fields (title, description, data rows) and word embedding-based scores for generated schema labels can be used to rank the datasets.

We created a new benchmark to evaluate the performance of dataset retrieval. The experimental results demonstrate our proposed framework can effectively improve the performance on the dataset retrieval task. It achieved the highest NDCG on all the rank cut-offs compared with all baseline methods. We

also apply our method to the web table retrieval task which is similar to dataset search and find that the features generated from schema labels can help in supervised ranking as well.

Acknowledgment. This material is based upon work supported by the National Science Foundation under Grant No. IIS-1816325.

References

1. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Trans. Assoc. Comput. Linguist. **5**, 135–146 (2017)
2. Cafarella, M.J., Halevy, A., Wang, D.Z., Wu, E., Zhang, Y.: Webtables: exploring the power of tables on the web. Proc. VLDB Endow. **1**(1), 538–549 (2008)
3. Chapman, A., et al.: Dataset search: a survey. arXiv preprint arXiv:1901.00735 (2019)
4. Chen, Z., Jia, H., Heflin, J., Davison, B.D.: Generating schema labels through dataset content analysis. In: Companion of the The Web Conference 2018, pp. 1515–1522. International World Wide Web Conferences Steering Committee (2018)
5. Gentile, A.L., Kirstein, S., Paulheim, H., Bizer, C.: Extending RapidMiner with data search and integration capabilities. In: Sack, H., Rizzo, G., Steinmetz, N., Mladenić, D., Auer, S., Lange, C. (eds.) ESWC 2016. LNCS, vol. 9989, pp. 167–171. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47602-5_33
6. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of IR techniques. ACM Trans. Inf. Syst. (TOIS) **20**(4), 422–446 (2002)
7. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009)
8. Kusner, M., Sun, Y., Kolkin, N., Weinberger, K.: From word embeddings to document distances. In: International Conference on Machine Learning, pp. 957–966 (2015)
9. Levy, O., Goldberg, Y.: Neural word embedding as implicit matrix factorization. In: Advances in Neural Information Processing Systems, pp. 2177–2185 (2014)
10. Liang, D., Altosaar, J., Charlin, L., Blei, D.M.: Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In: Proceedings of the 10th ACM Conference on Recommender Systems, pp. 59–66. ACM (2016)
11. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
12. Ogilvie, P., Callan, J.: Combining document representations for known-item search. In: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval, pp. 143–150. ACM (2003)
13. Sekhavat, Y.A., Di Paolo, F., Barbosa, D., Merialdo, P.: Knowledge base augmentation using tabular data. In: LDOW (2014)
14. Trabelsi, M., Davison, B., Jeff, H.: Improved table retrieval using multiple context embeddings for attributes. In: Proceedings of IEEE Big Data 2019. IEEE (2019)
15. Yu, H.-F., Hsieh, C.-J., Si, S., Dhillon, I.S.: Parallel matrix factorization for recommender systems. Knowl. Inf. Syst. **41**(3), 793–819 (2013). https://doi.org/10.1007/s10115-013-0682-2

16. Zhang, S., Balog, K.: Entitables: smart assistance for entity-focused tables. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2017, pp. 255–264, ACM, New York (2017). https://doi.org/10.1145/3077136.3080796
17. Zhang, S., Balog, K.: Ad hoc table retrieval using semantic similarity. In: Proceedings of the 2018 World Wide Web Conference, WWW 2018, pp. 1553–1562, Republic and Canton of Geneva, Switzerland (2018). https://doi.org/10.1145/3178876.3186067