A Framework for Evaluating Snippet Generation for Dataset Search

Xiaxia Wang[1], Jinchi Chen[1], Shuxin Li[1], Gong Cheng[1(✉)], Jeff Z. Pan[2,3], Evgeny Kharlamov[4,5], and Yuzhong Qu[1]

[1] National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
xxwang1997@gmail.com, {jcchen,sxli}@smail.nju.edu.cn, {gcheng,yzqu}@nju.edu.cn
[2] Edinburgh Research Centre, Huawei, Edinburgh, UK
[3] Department of Computing Science, University of Aberdeen, Aberdeen, UK
jeff.z.pan@abdn.ac.uk
[4] Department of Informatics, University of Oslo, Oslo, Norway
evgeny.kharlamov@ifi.uio.no
[5] Bosch Center for Artificial Intelligence, Robert Bosch GmbH, Renningen, Germany
evgeny.kharlamov@de.bosch.com

Abstract. Reusing existing datasets is of considerable significance to researchers and developers. Dataset search engines help a user find relevant datasets for reuse. They can present a snippet for each retrieved dataset to explain its relevance to the user's data needs. This emerging problem of snippet generation for dataset search has not received much research attention. To provide a basis for future research, we introduce a framework for quantitatively evaluating the quality of a dataset snippet. The proposed metrics assess the extent to which a snippet matches the query intent and covers the main content of the dataset. To establish a baseline, we adapt four state-of-the-art methods from related fields to our problem, and perform an empirical evaluation based on real-world datasets and queries. We also conduct a user study to verify our findings. The results demonstrate the effectiveness of our evaluation framework, and suggest directions for future research.

Keywords: Snippet generation · Dataset search · Evaluation metric

1 Introduction

We are witnessing the rapid growth of open data on the Web, notably RDF, Linked Data and Knowledge Graphs [30]. Today, to develop a Web application, reusing existing datasets not only brings about productivity improvements and cost reductions, but also makes interoperability with other applications more achievable. However, there is a lack of tool support for conveniently finding datasets that match a developer's data needs. To address it, recent research efforts yielded *dataset search engines* like LODAtlas [32] and Google Dataset

C. Ghidini et al. (Eds.): ISWC 2019, LNCS 11778, pp. 680–697, 2019.
https://doi.org/10.1007/978-3-030-30793-6_39

Search [2]. They retrieve a list of datasets that are relevant to a keyword query by matching the query with the description in the metadata of each dataset.

These systems have made a promising start. Furthermore, a helpful dataset search engine should also explain why a retrieved dataset is relevant. A concise piece of information presented for each dataset in a search results page is broadly referred to as a *dataset summary*. It may help the user quickly identify a relevant dataset. Summaries presented in current dataset search engines, however, are mainly composed of some *metadata* about a dataset, such as provenance and license. Their utility in relevance judgment is limited, with users having to analyze each dataset in the search results to assess its relevance, which would be a time-consuming process.

To overcome the shortcoming of metadata, we study an emerging type of dataset summary called *dataset snippet*. For an RDF dataset retrieved by a keyword query, a dataset snippet is a size-constrained subset of RDF triples extracted from the dataset, being intended to exemplify the content of the dataset and to explain its relevance to the query. It differs from a *dataset profile* which represents a set of features describing attributes of the dataset [13]. It is also complementary to an *abstractive summary* which aggregates data into patterns and provides a high-level overview [4,8,38,39,45]. It is conceptually more similar to a snippet extracted from a webpage and presented in traditional Web search. However, little research attention has focused on this perspective.

As a preliminary effort along this way, we work towards establishing a framework for evaluating snippets generated for dataset search. That would provide a basis for future research, in terms of providing quantitative evaluation metrics and advising algorithm design. Existing evaluation metrics used in related fields such as snippet generation for ontologies [28] and documents [16] are mainly based on a human-created ground truth. However, an RDF dataset may contain millions of RDF triples, e.g., when it wrapped from a large database [18,19,23,33], or streaming data [24,25], or comes from a manufacturing environment [22,26,37] being much larger than an ontology schema or a document. It would be difficult, if not impossible, to manually identify the optimum snippet as the ground truth. Therefore, new evaluation metrics are needed.

To demonstrate the use of our evaluation framework, considering the lack of dedicated solutions to dataset snippets, we explore research efforts in related fields and adapt their methods to our problem. Using our framework, we analyze these methods and empirically evaluate them based on real-world datasets. We also carry out a user study to verify our findings and solicit comments to motivate future research.

To summarize, our contributions in this paper include

- a framework for evaluating snippets in dataset search, consisting of four metrics regarding how well a snippet covers a query and a dataset,
- an adaptation of four state-of-the-art methods selected from related fields to generate snippets for dataset search, as a baseline for future research, and

– an evaluation of the adapted methods using the proposed evaluation framework based on real-world datasets and queries, as well as a user study.

The remainder of the paper is organized as follows. Section 2 reviews related research. Section 3 describes our evaluation framework. Section 4 reports evaluation results. Section 5 presents a user study. Section 6 concludes the paper.

2 Related Work

Very little research attention has been given to the problem of snippet generation for dataset search. Therefore, in this section, we also review research efforts in related fields that can be adapted to the problem we study.

2.1 Snippets for RDF Datasets

In an early work [1], a snippet for an RDF document is generated to show how the document is relevant to a keyword query. Preference is given to RDF triples that describe central entities or contain query keywords. The proposed algorithm relies on manually defined ranking of predicates. In [12,36], an RDF dataset is compressed by keeping only a sample of triples in order to improve the performance of query processing while still serve query results as complete as possible. To this end, [36] samples triples that are central in the RDF graph and hence are likely to appear in the answers of typical SPARQL queries. By contrast, [12] iteratively expands the sample as needed to make it more precise. Completeness preserving summaries [15] help optimise distributed reasoning and querying.

In a recent work [7], an *illustrative snippet* is generated to exemplify the content of an RDF dataset. Snippet generation is formulated as a combinatorial optimization problem, aiming to find an optimum connected RDF subgraph such that it contains instantiation of the most frequently used classes and properties in the dataset and contains entities having the highest PageRank scores. An approximation algorithm is presented to solve this NP-hard problem. This kind of snippet can be used in dataset search, although it is not query-biased.

2.2 Snippets for Ontology Schemas

An *ontology snippet* distills the most important information from an ontology schema and forms an abridged version [42,43]. Existing methods often represent an ontology schema as a graph, and apply some centrality-based measures to identify the most important terms or axioms as an ontology snippet [34,35]. It is possible to adapt these methods to generate snippets for an RDF dataset because it can be viewed as an RDF graph to process.

We give particular attention to methods that are capable of generating *query-biased snippets for ontology search* [3,5,6,17,31]. An ontology schema is often represented as a graph where nodes represent terms and edges represent axioms

associating terms [17,44]. In a state-of-the-art approach [17], such a graph is decomposed into a set of maximal radius-bounded connected subgraphs, which in turn are reduced to tree-structured sub-snippets. A greedy algorithm is performed to select and merge an optimum set of sub-snippets, in terms of compactness and query relevance.

2.3 Keyword Search on Graphs

Keyword search on a graph is to find an optimum connected subgraph that contains all the keywords in a query [9,41]. An optimum subgraph has the smallest total edge weight [11,21,29], or a variant of this property [27]. As each keyword can match a set of nodes in a graph, the problem is formulated as a *group Steiner tree (GST) problem*. This kind of subgraph can be used as a query-biased snippet for an RDF dataset viewed as an RDF graph. However, the problem is NP-hard and is difficult to solve. Many algorithms perform not well on large graphs [10].

A state-of-the-art algorithm for the GST problem is PrunedDP++ [29]. The algorithm progressively refines feasible solutions based on dynamic programming with an A*-search strategy. In dynamic programming, optimal-tree decomposition and conditional tree merging techniques are proposed to prune unpromising states. For A*-search, several lower-bounding techniques are used.

2.4 Snippets for Documents

A *document snippet* consists of salient sentences selected from the original document [16]. To adapt such a method to our problem, we could replace the three elements of an RDF triple with their textual forms. The triple becomes a pseudo sentence, and an RDF dataset is transformed into a set of sentences to process.

Among existing solutions, *unsupervised query-biased methods* [40] are closer to our problem setting because, at this stage, training data for dataset search is not available. The CES method [14] is among the state-of-the-art in this line of work. It formulates sentence selection as an optimization problem and solves it using the cross-entropy method. Preference is given to diversified long sentences that are relevant to a query.

3 Evaluation Framework

In this section, we firstly define some terms used in the paper, and then propose a framework for evaluating snippets generated for dataset search. Our framework, consisting of four metrics characterizing different aspects of a dataset snippet, will be used in later sections to evaluate selected methods reviewed in Sect. 2.

3.1 Preliminaries

Datasets vary in their formats. Search queries have various types. This paper is focused on keyword queries over RDF datasets because this combination is common. We will consider other data formats and query types in future work.

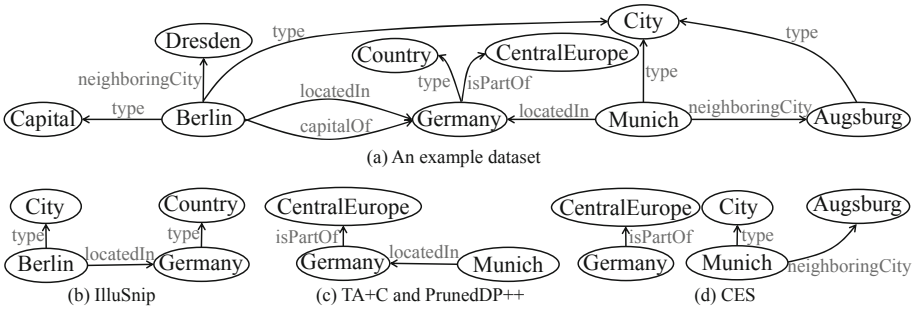

Fig. 1. (a) An example dataset and (b)(c)(d) three of its snippets generated by different methods w.r.t. the query *munich europe*.

Definition 1 (RDF Dataset). *An RDF dataset, or a dataset for short, is a set of n RDF triples denoted by $T = \{t_1, \ldots, t_n\}$. Each $t_i \in T$ is a subject-predicate-object triple denoted by $\langle t_i^s, t_i^p, t_i^o \rangle$.*

In RDF, t_i^{s}, t_i^{p}, and t_i^{o} can be IRIs, blank nodes, or literals, which are collectively known as RDF terms. An "RDF term" and the "resource" it denotes are used interchangeably in the paper.

Definition 2 (Keyword Query). *A keyword query, or a query for short, is a set of m keywords denoted by $Q = \{q_1, \ldots, q_m\}$.*

A snippet of a dataset is a size-constrained subset of triples extracted from the dataset. The extraction should consider the query.

Definition 3 (Dataset Snippet). *Given a positive integer k, a snippet of a dataset T is denoted by S subject to $S \subseteq T$ and $|S| \leq k$.*

An RDF dataset T can be viewed as an RDF graph denoted by $\mathtt{G}(T)$. Each triple $\langle t^{\mathrm{s}}, t^{\mathrm{p}}, t^{\mathrm{o}} \rangle \in T$ is represented as a directed edge labeled with t^{p} from node t^{s} to node t^{o} in $\mathtt{G}(T)$. Analogously, a snippet S is a subgraph denoted by $\mathtt{G}(S)$. In Fig. 1 we illustrate three snippets for an example dataset w.r.t. a query.

3.2 Evaluation Metrics

To assess the quality of a snippet w.r.t. a query, we propose four quantitative metrics: `coKyw`, `coCnx`, `coSkm`, and `coDat`. Recall that a snippet is generated to exemplify the content of a dataset and to explain its relevance to the query. So a good snippet should, on the one hand, match the query intent (`coKyw`, `coCnx`) and, on the other hand, cover the main content of the dataset (`coSkm`, `coDat`). Our metrics are open source[1].

[1] http://ws.nju.edu.cn/datasetsearch/evaluation-iswc2019/metrics.zip.

Coverage of Query Keywords (coKyw). Keywords in a query express a user's data needs. A good snippet should cover as many keywords as possible, to show how a dataset is plainly relevant to the query.

Specifically, let $\texttt{Text}(r)$ be a set of textual forms of a resource r. For r denoted by an IRI, $\texttt{Text}(r)$ include

- the lexical forms of r's *human-readable names* (if any), i.e., literal values of r's `rdfs:label` property, and
- r's *local name*, i.e., the fragment component of r's IRI (if its exists) or the last segment of the path component of the IRI.

For r denoted by a blank node, $\texttt{Text}(r)$ only include the lexical forms of r's human-readable names (if any). For r denoted by a literal, $\texttt{Text}(r)$ only include the lexical form of the literal.

A resource r *covers* a keyword q if any textual form in $\texttt{Text}(r)$ contains a *match* for q. Our implementation considers keyword matching, which can be extended to semantic matching in future work. A triple t *covers* a keyword q, denoted by $t \prec q$, if r covers q for any $r \in \{t^{\mathrm{s}}, t^{\mathrm{p}}, t^{\mathrm{o}}\}$. For a snippet S, its coverage of keywords in a query Q is the proportion of covered keywords:

$$\texttt{coKyw}(S) = \frac{1}{|Q|} \cdot |\{q \in Q : \exists t \in S,\ t \prec q\}| \,. \tag{1}$$

For example, Fig. 1(c) and (d) cover all the query keywords, so $\texttt{coKyw} = 1$. None of the keywords are covered by Fig. 1(b), so $\texttt{coKyw} = 0$.

Coverage of Connections between Query Keywords (coCnx). Keywords in a query are not independent but often refer to a set of related concepts which collectively represent a query intent. To show how a dataset is relevant to the query and its underlying intent, a good snippet should cover not only query keywords but also their connections captured by the dataset.

Specifically, for a snippet S, consider its RDF graph $\texttt{G}(S)$. Query keywords can be covered by nodes or edges of $\texttt{G}(S)$. For convenience, we obtain a *subdivision* of $\texttt{G}(S)$, by subdividing every edge labeled with t^{p} from node t^{s} to node t^{o} into two unlabeled undirected edges: one between t^{s} and t^{p}, and the other between t^{p} and t^{o}. The resulting graph is denoted by $\texttt{SD}(\texttt{G}(S))$. A snippet S *covers* the connection between two keywords $q_i, q_j \in Q$, denoted by $S \prec (q_i, q_j)$, if there is a path in $\texttt{SD}(\texttt{G}(S))$ that connects two nodes: one covering q_i and the other covering q_j. For S, its coverage of connections between keywords in Q is the proportion of covered connections between unordered pairs of keywords:

$$\texttt{coCnx}(S) = \begin{cases} \frac{1}{\binom{|Q|}{2}} \cdot |\{\{q_i, q_j\} \subseteq Q : q_i \neq q_j \text{ and } S \prec (q_i, q_j)\}| & \text{if } |Q| > 1\,, \\ \texttt{coKyw}(S) & \text{if } |Q| = 1\,. \end{cases} \tag{2}$$

When there is only one keyword, coCnx is meaningless and we set it to coKyw.

For example, Fig. 1(c) covers the connection between the two query keywords, so $\texttt{coCnx} = 1$. By contrast, although Fig. 1(d) covers all the keywords, it fails to cover their connections, so $\texttt{coCnx} = 0$.

Coverage of Data Schema (coSkm). Snippets are expected to not only interpret query relevance but also offer a representative preview of a dataset. In particular, the RDF schema of a dataset is important to users. A good snippet should cover as many classes and properties used in the dataset as possible, to exemplify which types of things and facts a user can obtain from the dataset.

Specifically, a snippet S *covers* a class or a property if S contains its instantiation. Let $\texttt{Cls}(S)$ and $\texttt{Prp}(S)$ be the set of classes and the set of properties instantiated in S, respectively:

$$\begin{aligned} \texttt{Cls}(S) &= \{c : \exists t \in S,\ t^{\mathrm{p}} = \texttt{rdf:type} \text{ and } t^{\mathrm{o}} = c\}\,, \\ \texttt{Prp}(S) &= \{p : \exists t \in S,\ t^{\mathrm{p}} = p\}\,. \end{aligned} \tag{3}$$

Classes and properties that are used more often in a dataset are more representative. The relative frequency of a class c observed in a dataset T is

$$\texttt{frqCls}(c) = \frac{|\{t \in T :\ t^{\mathrm{p}} = \texttt{rdf:type} \text{ and } t^{\mathrm{o}} = c\}|}{|\{t \in T :\ t^{\mathrm{p}} = \texttt{rdf:type}\}|}\,. \tag{4}$$

Analogously, the relative frequency of a property p observed in T is

$$\texttt{frqPrp}(p) = \frac{|\{t \in T :\ t^{\mathrm{p}} = p\}|}{|T|}\,. \tag{5}$$

For a snippet S, its coverage of the schema of T is related to: (a) the total relative frequency of the covered classes, and (b) the total relative frequency of the covered properties. We calculate the harmonic mean (hm) of the two:

$$\begin{aligned} \texttt{coSkm}(S) &= \texttt{hm}\Big(\sum_{c \in \texttt{Cls}(S)} \texttt{frqCls}(c),\ \sum_{p \in \texttt{Prp}(S)} \texttt{frqPrp}(p)\Big)\,, \\ \texttt{hm}(x, y) &= \frac{2xy}{x + y}\,. \end{aligned} \tag{6}$$

For example, Fig. 1(b) covers a frequent class (City) and a frequent property (locatedIn) in the dataset, so its coSkm score is higher than that of Fig. 1(c) which covers only properties but not classes.

Coverage of Data (coDat). Classes and properties high relative frequency are central elements in the schema used in a dataset. Complementary to them, a good snippet should also cover central elements at the data level (i.e., central entities), to show the key content of the dataset.

Specifically, let $\texttt{d}^+(r)$ and $\texttt{d}^-(r)$ be the out-degree and in-degree of a resource r in an RDF graph $\texttt{G}(T)$, respectively:

$$\begin{aligned} \texttt{d}^+(r) &= |\{t \in T : t^{\mathrm{s}} = r\}|\,, \\ \texttt{d}^-(r) &= |\{t \in T : t^{\mathrm{o}} = r\}|\,. \end{aligned} \tag{7}$$

Out-degree characterizes the richness of the description of a resource, and in-degree characterizes popularity. They suggest the centrality of a resource from

Table 1. Overview of selected methods and their alignment with evaluation metrics.

		coKyw	coCnx	coSkm	coDat
IlluSnip [7]	(illustrative dataset snippet)			✓	✓
TA+C [17]	(query-biased ontology snippet)	✓	✓		
PrunedDP++ [29]	(GST for keyword search)	✓	✓		
CES [14]	(query-biased document snippet)	✓		✓	✓

different aspects. For a snippet S, its coverage of a dataset T at the data level is related to: (a) the mean normalized out-degree of the constituent entities, and (b) the mean normalized in-degree of the constituent entities. We calculate the harmonic mean of the two:

$$\texttt{coDat}(S) = \texttt{hm}\Big(\frac{1}{|\texttt{Ent}(S)|}\cdot\sum_{e\in\texttt{Ent}(S)}\frac{\log(\texttt{d}^+(e)+1)}{\max_{e'\in\texttt{Ent}(T)}\log(\texttt{d}^+(e')+1)},\ \frac{1}{|\texttt{Ent}(S)|}\cdot\sum_{e\in\texttt{Ent}(S)}\frac{\log(\texttt{d}^-(e)+1)}{\max_{e'\in\texttt{Ent}(T)}\log(\texttt{d}^-(e')+1)}\Big), \tag{8}$$

$$\texttt{Ent}(X) = \{r : \exists t \in X,\ r \in \{t^{\mathrm{s}}, t^{\mathrm{o}}\},\ r \notin \texttt{Cls}(T),\ \text{and } r \text{ is not a literal.}\},$$

where $\texttt{Cls}(T)$ is the set of all classes instantiated in T defined in Eq. (3), $\texttt{Ent}(S)$ is the set of all entities (i.e., non-literal resources at the data level) that appear in S, and $\texttt{Ent}(T)$ is the set of all entities that appear in T. Degree is normalized by the maximum value observed in the dataset. Considering that degree usually follows a highly skewed power-law distribution in practice, normalization is performed on a logarithmic scale.

For example, Fig. 1(b) is focused on `Germany`, which is a central entity in the dataset, so its `coDat` score is higher than that of Fig. 1(c) and (d) which contain more of subordinate entities.

4 Evaluation

In Sect. 2, each subsection reviews methods in a related research field that can be adapted to generate snippets for dataset search. The second paragraph of each subsection identifies a state-of-the-art method from each field that is suitable for our context: [7,17,29] and [14]. In this section, we evaluate these methods using the evaluation framework proposed in Sect. 3. We first analyze whether and how the components of these methods are aligned with each evaluation metric. Then we perform an extensive empirical evaluation based on real-world datasets.

4.1 Analysis of Selected Methods

Table 1 presents an overview of the selected methods and whether they have components that are conceptually similar to each evaluation metric. All the

methods have been detailed in Sect. 2. In the following we focus on how their components are aligned with each evaluation metric.

Illustrative Dataset Snippet. Dataset snippets generated by existing methods reviewed in Sect. 2.1 can be used in dataset search without adaptation. The method we choose, IlluSnip [7], generates an illustrative snippet for an RDF dataset by extracting a connected subgraph to exemplify the content of the dataset. This intended use is very close to our problem.

IlluSnip explicitly considers a snippet's coverage of a dataset. Giving priority to the most frequent classes and properties, a snippet is likely to show a high coverage of data schema (`coSkm`). Besides, IlluSnip computes the centrality of an entity by PageRank, which positively correlates with in-degree. Therefore, a snippet containing such central entities may also have a reasonably high coverage of data (`coDat`), which is jointly measured by in-degree and out-degree.

However, IlluSnip is not query biased. A snippet it generates may not contain any keyword in a query, and hence its coverage of query keywords (`coKyw`) and the connections thereof (`coCnx`) can be very low.

For example, Fig. 1(b) illustrates a snippet generated by IlluSnip.

Query-Biased Ontology Snippet. Query-biased snippets for ontology search reviewed in Sect. 2.2 are useful for deciding the relevance of an ontology schema to a query. It is similar to our intent to support judging the relevance of a dataset. The method we choose, TA+C [17], extracts a query-biased subgraph from the RDF graph representation of an ontology schema. This method can be directly used to generate snippets for RDF datasets without adaptation.

TA+C explicitly considers a snippet's coverage of a query. It greedily adds query-biased sub-snippets into a snippet, giving preference to those containing more query keywords. A sub-snippet is a radius-bounded connected subgraph. Therefore, the resulting snippet has the potential to establish a high coverage of query keywords (`coKyw`) and their connections (`coCnx`), especially when keywords are closely located in the dataset.

On the other hand, coverage of dataset (`coSkm` and `coDat`) is not of concern to this query-centered method.

For example, Fig. 1(c) illustrates a snippet generated by TA+C.

GST for Keyword Search. Methods for keyword search on graphs reviewed in Sect. 2.3 find a GST, which is a connected subgraph where nodes contain all the query keywords. These methods can be straightforwardly applied to generate snippets for RDF datasets by computing a GST. The method we choose, PrunedDP++ [29], is one of the most efficient algorithms for the GST problem.

PrunedDP++ has two possible outputs. Either it finds a GST that covers all the query keywords (`coKyw`) and connections between all pairs of them (`coCnx`), or such a GST does not exist. In the latter case, PrunedDP++ returns empty results. So it is conceptually similar to TA+C but appears more "aggressive".

Coverage of dataset (`coSkm` and `coDat`) is not the focus of PrunedDP++. Nevertheless, these factors may be partially addressed by properly defining edge weights. Weighting is orthogonal to the design of PrunedDP++.

For example, Fig. 1(c) illustrates a snippet generated by PrunedDP++.

Query-Biased Document Snippet. Query-biased methods for generating document snippets reviewed in Sect. 2.4 can be adapted to generate snippets for RDF datasets, by replacing resources in a triple with their textual forms (e.g., labels of IRI-identified resources, lexical forms of literals) to obtain a pseudo sentence. The method we choose, CES [14], generates a query-biased snippet by selecting a subset of sentences (i.e., triples in our context). This unsupervised method fits current dataset search, for which training data is in shortage.

CES tends to select diversified triples that are relevant to a query, so it is likely to achieve a high coverage of query keywords (`coKyw`). CES also computes the cosine similarity between the term frequency—inverse document frequency (TF-IDF) vectors of the document (i.e., RDF dataset) and a snippet. This feature measures to what extent the snippet covers the main content of the dataset. It increases the possibility of including frequent classes, properties, and entities, and hence may improve a snippet's coverage of dataset (`coSkm` and `coDat`).

As a side effect of diversification, triples in a snippet are usually disparate. Connections between query keywords (`coCnx`) can hardly be observed.

For example, Fig. 1(d) illustrates a snippet generated by CES.

4.2 Empirical Evaluation

We used the proposed framework to evaluate the above selected methods. All the experiments were performed on an Intel Xeon E7-4820 (2.00 GHz) with 80GB memory for the JVM. Our implementation of these methods is open source[2].

Datasets and Queries. We retrieved the metadata of 11,462 datasets from DataHub[3] using CKAN's API. Among 1,262 RDF datasets that provided Turtle, RDF/XML, or N-Triples dump files, we downloaded and parsed 311 datasets using Apache Jena. The others were excluded due to download or parse errors.

We used two kinds of queries: real queries and artificial queries.

Real Queries. We used crowdsourced natural language queries[4] that were originally submitted to data.gov.uk for datasets [20]. They were transformed into keyword queries by removing stop words using Apache Lucene.

Artificial Queries. To have more queries, we leveraged the DMOZ open directory[5] to imitate possible data needs. For each $i = 1 \ldots 4$, we constructed a group of queries denoted by DMOZ-i. A query in DMOZ-i consisted of the names of

[2] http://ws.nju.edu.cn/datasetsearch/evaluation-iswc2019/baselines.zip.
[3] https://old.datahub.io/.
[4] https://github.com/chabrowa/data-requests-query-dataset.
[5] http://dmoz-odp.org/.

Table 2. Statistics about query-dataset (Q-D) pairs.

	#Q-D pairs	#keywords in Q		#triples in D		#classes in D		#properties in D	
		mean	max	mean	max	mean	max	mean	max
data.gov.uk	42	2.88	8	116,822	2,203,699	13	129	47	357
DMOZ-1	88	1.25	3	137,257	2,203,699	30	2,030	66	3,982
DMOZ-2	84	2.33	5	151,104	2,203,699	10	129	34	357
DMOZ-3	87	3.66	6	164,714	2,203,699	13	153	43	357
DMOZ-4	86	5.02	8	219,844	2,203,699	13	129	46	357

i random sub-categories of a random top-level category in DMOZ. Such closely related concepts had a reasonable chance to be fully covered by some dataset.

To conclude, we had five groups of queries: data.gov.uk, DMOZ-1, DMOZ-2, DMOZ-3, and DMOZ-4. For each group, we randomly retained 100 queries such that each query could be paired with a dataset that covered all the query keywords. These 500 query-dataset pairs were used in our evaluation. We required a dataset to cover all the query keywords in order to make sense of the experiment results. Otherwise, a low score of `coKyw` would be ambiguous: reflecting the poor quality of the snippet, and/or the irrelevance of the dataset.

Configuration of Methods. We detail their configuration in the following.

Size of Snippet. Following [7], we configured IlluSnip and CES to generate a snippet containing at most 20 RDF triples (i.e., $k = 20$). For TA+C, it would be inappropriate to bound the number of triples because the snippets it generated could contain isolated nodes. So we bounded it to output a snippet whose graph representation contained at most 20 nodes. For PrunedDP++, the size of its output was automatically determined but not configurable.

Weights and Parameters. For TA+C [17], edge weights were defined as in the original paper. For PrunedDP++ [29], its authors did not specify how to weight edges. Our weighting followed [11]—the predecessor of PrunedDP++. For CES [14], it had several parameters. Most of them were set to the values used in the original paper. However, due to the large size of RDF dataset, the sampling step in CES was performed 1,000 times (instead of 10,000 times in [14]) in consideration of memory use.

Preprocessing. We built inverted indexes for efficient keyword mapping in TA+C, PrunedDP++, and CES. For TA+C, following its original implementation [17], we precomputed and materialized all the maximal 1-radius subgraphs.

Timeout. After preprocessing, we set a timeout of one hour for each method to generate a snippet. The generating process would be terminated when reaching timeout. In that case, the runtime would be defined to be one hour. For IlluSnip and CES which iteratively found better snippets, the best snippet at timeout would be returned. For TA+C and PrunedDP++, timeout indicated failure.

Table 3. Average scores of evaluation metrics on all the query-dataset pairs.

	coKyw	coCnx	coSkm	coDat
IlluSnip	0.1000	0.0540	0.6820	0.3850
TA+C	0.9590	0.4703	0.0425	0.0915
PrunedDP++	1	1	0.0898	0.2133
CES	0.9006	0.3926	0.3668	0.2684

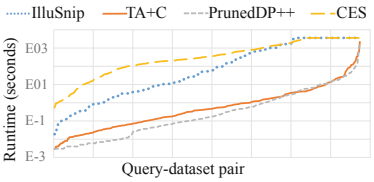

Fig. 2. Runtime on each query-data set pair, in ascending order.

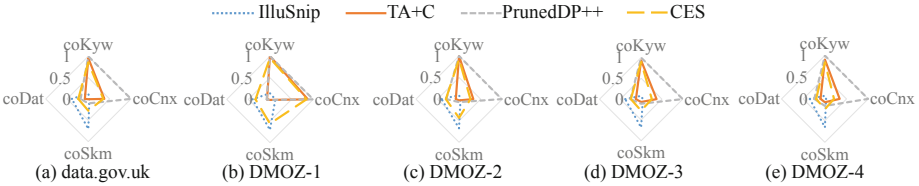

Fig. 3. Average scores of evaluation metrics on each group of query-dataset pairs.

Evaluation Results. Out of the 500 query-dataset pairs, 113 pairs were not included in our results for one of the following reasons.

- PrunedDP++ did not find any GST to connect all the query keywords, and hence generated an empty snippet.
- TA+C and PrunedDP++ were forced to terminate due to timeout.
- TA+C did not complete preprocessing after twelve hours.

We reported evaluation results on the remaining 387 pairs where every method successfully generated a non-empty snippet before timeout. Table 2 characterizes these queries and datasets. They are available online[6].

Note that IlluSnip and CES were configured to generate a snippet containing at most 20 triples, and they selected 19.68 and 19.89 triples on average, respectively. By comparison, for PrunedDP++ the size of its output was automatically determined, and the mean number of triples in the experiment was only 4.60. This may affect the evaluation results. Besides, TA+C and PrunedDP++ sometimes produced isolated nodes instead of triples. The query keywords covered by these nodes were considered in the computation of `coKyw` and `coCnx`.

Table 3 presents the average score of each evaluation metric achieved by each method on all the query-dataset pairs. Compared with Table 1, a higher score was generally observed when a metric was conceptually considered in the components of a method. We concluded that the results of our empirical evaluation were basically consistent with our analysis in Sect. 4.1. Figure 3 depicts the scores on each group of query-dataset pairs using radar charts.

IlluSnip achieved much higher scores of `coSkm` and `coDat` than other methods. It was not surprising because covering the schema and data of a dataset was

[6] http://ws.nju.edu.cn/datasetsearch/evaluation-iswc2019/query-dataset-pairs.zip.

central to the design of IlluSnip. However, there were still notable gaps between the achieved scores (coSkm = 0.6820 and coDat = 0.3850) and their upper bound (i.e., 1), because IlluSnip was constrained to output a size-bounded connected subgraph. The coverage of such a subgraph was limited. On the other hand, all the other three methods were query-biased, whereas IlluSnip was not. Its very low scores of coKyw = 0.1000 and coCnx = 0.0540 suggested that the snippets generated by IlluSnip usually failed to cover queries.

TA+C was opposite in scores to IlluSnip. Coverage of dataset was not the focus of its design. The lowest scores of coSkm = 0.0425 and coDat = 0.0915 were observed on this method. By contrast, opting for connected subgraphs containing more query keywords, it achieved a fairly high score of coKyw = 0.9590. However, connections between query keywords were not captured well, because radius-bounded connected subgraph was incapable of covering long-distance connections. As shown in Fig. 3, actually the overall score of coCnx = 0.4703 was even exaggerated by the case of DMOZ-1, where most queries comprised only one keyword and hence coCnx was trivially defined to be coKyw according to Eq. (2). In other cases, coCnx was not high.

PrunedDP++ could not find any GST to connect all the query keywords on 86 query-dataset pairs, which had been excluded from our results. On the remaining pairs, not surprisingly, its coverage of query keywords (coKyw = 1) and their connections (coCnx = 1) was perfect. In a GST, query keywords were often connected via paths that passed through hub nodes in a dataset. Involving such high-degree nodes, a GST's coverage of data (coDat = 0.2133) was considerably higher than that of TA+C (coDat = 0.0915). However, similar to TA+C, a GST's coverage of data schema was limited (coSkm = 0.0898).

CES appeared to be a more balanced method, as visualized in Fig. 3. Towards generating a query-biased and diversified snippet, its coverage of query keywords (coKyw = 0.9006) was close to TA+C and PrunedDP++, and its coverage of dataset (coSkm = 0.3668 and coDat = 0.2684) was notably better. However, similar to TA+C, its coverage of connections between query keywords was not satisfying because selecting diversified triples usually led to a fragmented snippet. The overall score of coCnx = 0.3926 was exaggerated by the case of DMOZ-1.

Runtime. We also evaluated the runtime of each method because fast generation of snippets is an expectation of search engine users. Figure 2 depicts, on a logarithmic scale, the runtime of each method used for generating a snippet for each of the 387 query-dataset pairs. Runtime was mainly related to the number of triples in a dataset.

PrunedDP++ was generally the fastest method, with a median runtime of 0.16 s. It rarely reached timeout, and it completed computation in less than one second in 68% of the cases. TA+C was also reasonably fast, with a median runtime of 0.43 s. These two methods showed promising performance for practical use. By contrast, IlluSnip and CES reached timeout in 22% and 18% of the cases, respectively. They often spent tens or hundreds of seconds generating a snippet.

Fortunately, IlluSnip was not query-biased, and hence could be used to generate snippets offline.

5 User Study

We recruited 20 students majoring in computer science to assess the quality of snippets generated by different methods. All the participants had the experience in working with RDF datasets. The results could be compared with the above evaluation results, to verify the effectiveness of our proposed evaluation metrics.

Design. From fifty random candidates, each participant chose 5 datasets with interest according to their metadata. For each dataset, the participant had access to a list of classes and properties used in the dataset to help understanding. The participant was required to describe some data needs that could be fulfilled by the dataset, and then repeatedly rephrase the needs as a keyword query until all of IlluSnip, TA+C, and PrunedDP++ could generate a non-empty snippet. For reasonable response time, CES was excluded from user study, and datasets containing more than one million triples were not used. Following [7,17], we visualized a snippet (which was an RDF graph) as a node-link diagram. The participant rated its usefulness in relevance judgment on a scale of 1–5, and commented its strengths and weaknesses.

Results. Table 4 summarizes the responses from participants about snippets for a total of $20 \cdot 5 = 100$ datasets. IlluSnip received a higher mean rating than TA+C and PrunedDP++. Repeated measures ANOVA (rANOVA) indicated that their differences were statistically significant ($p < 0.01$). LSD post-hoc tests ($p < 0.01$) suggested that IlluSnip was more helpful to users than TA+C and PrunedDP++, whereas the difference between TA+C and PrunedDP++ was not statistically significant.

Figure 4 shows the mean score of each evaluation metric, grouped by user ratings. For each evaluation metric we excluded the results of some methods when their scores were hardly distinguishable (all close to 1) because those methods had components that were conceptually similar to the metrics (cf. Table 1). The scores of all the four metrics generally increased as user ratings increased. The observed positive correlation demonstrated the effectiveness of our evaluation framework. Exceptions were the notable falls of `coSkm` and `coDat` at the end, where very few (<10) snippets were rated 5 so that the scores at this point might not be significant.

We analyzed participants' comments. For IlluSnip, 15 participants (75%) complimented the connectivity of its results which facilitated understanding, and 13 participants (65%) referred to the richness and diversity of the content, which accorded well with its high coverage of data schema. Not surprisingly, 16 participants (80%) criticized its weak relevance to the query. For TA+C, 15 participants (75%) appreciated its query relevance, but 19 participants (95%)

Table 4. Human-rated usefulness of snippets (1–5) in relevance judgment.

Mean ± standard deviation		
IlluSnip	TA+C	PrunedDP++
3.10 ± 1.28	2.36 ± 1.29	1.92 ± 1.19
rANOVA (p-value): 0.00264		
LSD post-hoc ($p < 0.01$): IlluSnip > TA+C, PrunedDP++		

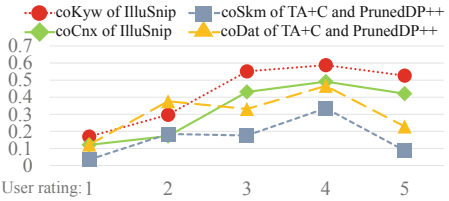

Fig. 4. Correlation between evaluation metrics and user ratings.

complained that its results sometimes contained many isolated nodes. It happened frequently when a query contained only one keyword. Although these nodes covered the query keyword, they were not associated with any further description, which dissatisfied 12 participants (60%). For PrunedDP++, similar feedback was received from 17 participants (85%) for some cases, but in other cases, 15 participants (75%) commented its high coverage of query keywords and the paths between them, which facilitated the comprehension of their connections. Besides, 8 participants (40%) favored the conciseness of its results.

Participants were invited to a post-experiment interview. Whereas they confirmed the usefulness of snippets, they generally believed that snippet could not replace but complement abstractive summary with statistics. Some participants suggested implementing interactive (e.g., hierarchical and zoomable) snippets for user exploration, which could be a future direction of research.

Discussion. Participants' ratings, comments, and the results of our proposed evaluation metrics were generally consistent with each other. The results justified the appropriateness of our framework to evaluating snippets in dataset search.

From the participants' comments, we concluded that a good dataset snippet should, on the one hand, cover query keywords and their connections to make sense of the underlying query intent. TA+C and PrunedDP++ were focused on this aspect. On the other hand, it should provide rich and diverse description about matched resources and triples to make sense of the dataset content. This was overlooked by TA+C and PrunedDP++, and it suggested a difference between snippet generation and keyword search. A trade-off between informativeness and compactness should be considered. IlluSnip showed promising results along this way. However, none of the three methods fulfilled these requirements completely, and hence their usefulness scores were far from perfection.

6 Conclusion

To promote research on the emerging problem of snippet generation for dataset search, we have proposed an evaluation framework for assessing the quality of dataset snippets. With our metrics, methods proposed in the future can be evaluated more easily. Our framework relies on neither ground-truth snippets which

are difficult to create, nor human efforts in user study which are inefficient and expensive. Evaluation can be automated offline. This in turn will be beneficial to the rapid development and deployment of snippets for dataset search engines.

Our evaluation results reveal the shortcomings of state-of-the-art methods adapted from related fields, which are also verified by a user study. None of the evaluated methods address all the considered aspects. It inspires us to put forward new methods for generating dataset snippets with more comprehensive features. Efficiency and scalability of methods are also important factors. Storage will also be a concern because a dataset search engine may have to store and index each dataset for snippet generation.

Our work has the following limitations. First, our evaluation framework may not be comprehensive. It can partially assess the quality of a dataset snippet, but still is not ready to completely replace user study. There may be other useful metrics, such as distinctiveness, readability, and coherence, which we will study in future work. Second, our evaluation metrics are implemented specifically for RDF datasets. To extend the range of application of our framework, more generalized implementation for other data formats needs to be explored.

Acknowledgements. This work was supported in part by the National Key R&D Program of China under Grant 2018YFB1005100, in part by the NSFC under Grant 61572247, and in part by the SIRIUS Centre, Norwegian Research Council project number 237898. Cheng was funded by the Six Talent Peaks Program of Jiangsu Province under Grant RJFW-011.

References

1. Bai, X., Delbru, R., Tummarello, G.: RDF snippets for semantic web search engines. In: Meersman, R., Tari, Z. (eds.) OTM 2008. LNCS, vol. 5332, pp. 1304–1318. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-88873-4_27
2. Brickley, D., Burgess, M., Noy, N.F.: Google dataset search: building a search engine for datasets in an open web ecosystem. In: WWW, pp. 1365–1375 (2019)
3. Butt, A.S., Haller, A., Xie, L.: Dwrank: learning concept ranking for ontology search. Semant. Web **7**(4), 447–461 (2016)
4. Cebiric, S., Goasdoué, F., Manolescu, I.: Query-oriented summarization of RDF graphs. PVLDB **8**(12), 2012–2015 (2015)
5. Cheng, G., Ge, W., Qu, Y.: Generating summaries for ontology search. In: WWW (Companion Volume), pp. 27–28 (2011)
6. Cheng, G., Ji, F., Luo, S., Ge, W., Qu, Y.: Biprank: ranking and summarizing RDF vocabulary descriptions. In: JIST, pp. 226–241 (2011)
7. Cheng, G., Jin, C., Ding, W., Xu, D., Qu, Y.: Generating illustrative snippets for open data on the web. In: WSDM, pp. 151–159 (2017)
8. Cheng, G., Jin, C., Qu, Y.: HIEDS: a generic and efficient approach to hierarchical dataset summarization. In: IJCAI, pp. 3705–3711 (2016)
9. Cheng, G., Kharlamov, E.: Towards a semantic keyword search over industrial knowledge graphs (extended abstract). In: IEEE BigData, pp. 1698–1700 (2017)
10. Coffman, J., Weaver, A.C.: An empirical performance evaluation of relational keyword search techniques. IEEE Trans. Knowl. Data Eng. **26**(1), 30–42 (2014)

11. Ding, B., Yu, J.X., Wang, S., Qin, L., Zhang, X., Lin, X.: Finding top-k min-cost connected trees in databases. In: ICDE, pp. 836–845 (2007)
12. Dolby, J., et al.: Scalable semantic retrieval through summarization and refinement. In: AAAI, pp. 299–304 (2007)
13. Ellefi, M.B., et al.: RDF dataset profiling - a survey of features, methods, vocabularies and applications. Semant. Web **9**(5), 677–705 (2018)
14. Feigenblat, G., Roitman, H., Boni, O., Konopnicki, D.: Unsupervised query-focused multi-document summarization using the cross entropy method. In: SIGIR, pp. 961–964 (2017)
15. Fkoue, A., Meneguzzi, F., Sensoy, M., Pan, J.Z.: Querying linked ontological data through distributed summarization. In: AAAI (2012)
16. Gambhir, M., Gupta, V.: Recent automatic text summarization techniques: a survey. Artif. Intell. Rev. **47**(1), 1–66 (2017)
17. Ge, W., Cheng, G., Li, H., Qu, Y.: Incorporating compactness to generate term-association view snippets for ontology search. Inf. Process. Manag. **49**(2), 513–528 (2013)
18. Horrocks, I., Giese, M., Kharlamov, E., Waaler, A.: Using semantic technology to tame the data variety challenge. IEEE Internet Comput. **20**(6), 62–66 (2016)
19. Jiménez-Ruiz, E., et al.: BootOX: practical mapping of RDBs to OWL 2. In: Arenas, M., et al. (eds.) ISWC 2015. LNCS, vol. 9367, pp. 113–132. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25010-6_7
20. Kacprzak, E., Koesten, L., Ibáñez, L.D., Blount, T., Tennison, J., Simperl, E.: Characterising dataset search - an analysis of search logs and data requests. J. Web Semant. **55**, 37–55 (2019)
21. Kasneci, G., Ramanath, M., Sozio, M., Suchanek, F.M., Weikum, G.: STAR: steiner-tree approximation in relationship graphs. In: ICDE, pp. 868–879 (2009)
22. Kharlamov, E., et al.: Capturing industrial information models with ontologies and constraints. In: Groth, P., et al. (eds.) ISWC 2016. LNCS, vol. 9982, pp. 325–343. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46547-0_30
23. Kharlamov, E., et al.: Ontology Based Data Access in Statoil. J. Web Semant. **44**, 3–36 (2017)
24. Kharlamov, E., et al.: An ontology-mediated analytics-aware approach to support monitoring and diagnostics of static and streaming data. J. Web Semant. **56**, 30–55 (2019)
25. Kharlamov, E., et al.: Semantic access to streaming and static data at Siemens. J. Web Semant. **44**, 54–74 (2017)
26. Kharlamov, E., Mehdi, G., Savković, O., Xiao, G., Kalayci, E.G., Roshchin, M.: Semantically-enhanced rule-based diagnostics for industrial internet of things: the SDRL language and case study for siemens trains and turbines. J. Web Semant. **56**, 11–29 (2019)
27. Le, W., Li, F., Kementsietsidis, A., Duan, S.: Scalable keyword search on large RDF data. IEEE Trans. Knowl. Data Eng. **26**(11), 2774–2788 (2014)
28. Li, N., Motta, E., d'Aquin, M.: Ontology summarization: an analysis and an evaluation. In: IWEST (2010)
29. Li, R., Qin, L., Yu, J.X., Mao, R.: Efficient and progressive group steiner tree search. In: SIGMOD, pp. 91–106 (2016)
30. Pan, J., Vetere, G., Gomez-Perez, J., Wu, H. (eds.): Exploiting Linked Data and Knowledge Graphs for Large Organisations. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-319-45654-6
31. Penin, T., Wang, H., Tran, T., Yu, Y.: Snippet generation for semantic web search engines. In: ASWC, pp. 493–507 (2008)

32. Pietriga, E., et al.: Browsing linked data catalogs with LODAtlas. In: Vrandečić, D., et al. (eds.) ISWC 2018. LNCS, vol. 11137, pp. 137–153. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00668-6_9
33. Pinkel, C., et al.: RODI: benchmarking relational-to-ontology mapping generation quality. Semant. Web **9**(1), 25–52 (2018)
34. Pouriyeh, S., et al.: Graph-based methods for ontology summarization: A survey. In: AIKE, pp. 85–92 (2018)
35. Pouriyeh, S., et al.: Ontology summarization: graph-based methods and beyond. Int. J. Semant. Comput. **13**(2), 259–283 (2019)
36. Rietveld, L., Hoekstra, R., Schlobach, S., Guéret, C.: Structural properties as proxy for semantic relevance in RDF graph sampling. In: Mika, P., et al. (eds.) ISWC 2014. LNCS, vol. 8797, pp. 81–96. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11915-1_6
37. Ringsquandl, M., et al.: Event-enhanced learning for KG completion. In: ESWC, pp. 541–559 (2018)
38. Song, Q., Wu, Y., Lin, P., Dong, X., Sun, H.: Mining summaries for knowledge graph search. IEEE Trans. Knowl. Data Eng. **30**(10), 1887–1900 (2018)
39. Troullinou, G., Kondylakis, H., Stefanidis, K., Plexousakis, D.: Exploring RDFS KBs Using summaries. In: Vrandečić, D., et al. (eds.) ISWC 2018. LNCS, vol. 11136, pp. 268–284. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00671-6_16
40. Turpin, A., Tsegay, Y., Hawking, D., Williams, H.E.: Fast generation of result snippets in web search. In: SIGIR, pp. 127–134 (2007)
41. Wang, H., Aggarwal, C.C.: A survey of algorithms for keyword search on graph data. In: Managing and Mining Graph Data, pp. 249–273. Springer, Boston (2010). https://doi.org/10.1007/978-1-4419-6045-0_8
42. Zhang, X., Cheng, G., Ge, W., Qu, Y.: Summarizing vocabularies in the global semantic web. J. Comput. Sci. Technol. **24**(1), 165–174 (2009)
43. Zhang, X., Cheng, G., Qu, Y.: Ontology summarization based on rdf sentence graph. In: WWW, pp. 707–716 (2007)
44. Zhang, X., Li, H., Qu, Y.: Finding important vocabulary within ontology. In: ASWC, pp. 106–112 (2006)
45. Zneika, M., Vodislav, D., Kotzinos, D.: Quality metrics for RDF graph summarization. Semant. Web **10**(3), 555–584 (2019)