BANDAR: Benchmarking Snippet Generation Algorithms for (RDF) Dataset Search

Xiaxia Wang, Gong Cheng, *Member, IEEE,* Jeff Z. Pan, Evgeny Kharlamov, and Yuzhong Qu

Abstract—The large volume of open data on the Web is expected to be reused and create value. Finding the right data to reuse is a non-trivial task addressed by the recent dataset search systems, which retrieve datasets relevant to a keyword query. An important component of such systems is snippet generation, extracting data from a retrieved dataset to exemplify its content and explain its relevance to the query. Snippet generation algorithms have emerged but were mainly evaluated by user studies. More efficient and reproducible evaluation methods are needed. To meet this challenge, in this article, we present a set of quality metrics for assessing the usefulness of a snippet from different perspectives, and we select and aggregate them into quality profiles for different stages of a dataset search process. Furthermore, we create a benchmark from thousands of collected real-world data needs and datasets, on which we apply the presented quality metrics and profiles to evaluate snippets generated by two existing algorithms and three adapted algorithms. The results, which are reproducible as they are automatically computed without human interaction, show the pros and cons of the tested algorithms and highlight directions for future research. The benchmark data is publicly available.

Index Terms—Snippet generation, RDF data, dataset search, benchmark.

1 Introduction

OPEN data is crucial to scientific research and Web application development. In recent years, we have witnessed an explosive growth of open data on the Web. By November 2020, the Web Data Commons project found triple-structured data embedded in 44% of the 34 million crawled pay-level domains [3], collectively contributing 86 billion triples. By January 2020, Google has indexed 25 million datasets in tabular or other formats [42]. Among such increasingly many and various datasets that are findable, accessible, interoperable, and reusable (FAIR) [55], there is great value, e.g., for improving machine learning such as transfer learning [35], zero-shot learning [8], [9], and their explanations [10]. Creating value from open data requires effective retrieval, sense-making, reuse of existing datasets. It has motivated the research and development of *dataset search* systems, ranging from specific systems such as LODAtlas [45] focused on datasets in the format of the Resource Description Framework (RDF) [18], [44], aka knowledge graphs, to the generic Google Dataset Search [4].

Current dataset search systems [7] match a user-submitted keyword query with the indexed metadata of datasets, and present in search results pages some metadata and data statistics to help the user decide the relevance of a retrieved dataset. This *metadata-centred* architecture can largely build upon the infrastructure of the general Web search—treating the metadata of a dataset as a webpage to index and search. However, its shortcomings are explicit. First, *the quality of metadata varies greatly* [4]. Some datasets lack metadata, while others have incomplete or inaccurate metadata fields. Second, *the data itself is completely ignored*, thereby limiting the search capability. Indeed, our recent analysis of real data needs [11] showed that 63.79% of keyword queries refer to elements that typically appear in the data rather than the metadata of a dataset, e.g., concrete entities and data values.

These limitations can be overcome by *incorporating data* into the pipeline of dataset search. Specifically, we can match a keyword query with indexed data, then generate and present a *snippet* in search results pages to show extracted data that best exemplifies the content of a retrieved dataset [13], [39] and explains how it is relevant to the query [11], [54]. This emerging problem of *snippet generation for dataset search* is our research focus. We should note the difference between snippet and summary [6]: a snippet is an extracted subset of data, while a summary often refers to an aggregated representation of data. To support the study of snippet generation, our work in this article is targeted on providing a convenient toolkit for evaluation, including evaluation metrics and benchmark data for comparing snippet generation algorithms. Previous research evaluated the effectiveness of such algorithms mainly by conducting a user study [11], [13]. In contrast to this expensive, time-consuming, and irreproducible way of evaluation, we aim at developing *computable quality metrics* for assessing the usefulness of a snippet in dataset search. Snippet generation algorithms will then be able to be automatically evaluated, and the experiments will be reproducible. Indeed, we apply the proposed quality metrics to evaluate and compare existing snippet generation algorithms based on a *benchmark we publish* called BANDAR, short for **B**enchm**A**rking s**N**ippet generation algorithms for **D**ataset se**AR**ch, which we create from collected real-world data needs and RDF datasets. Our

• *X. Wang, G. Cheng, and Y. Qu are with the State Key Laboratory for Novel Software Technology, Nanjing University, China. Jeff Z. Pan is with the School of Informatics at University of Edinburgh, UK. Evgeny Kharlamov is with the Department of Informatics, University of Oslo, Norway, and the Bosch Center for Artificial Intelligence, Robert Bosch GmbH, Germany.*
E-mail: xxwang@smail.nju.edu.cn, gcheng@nju.edu.cn, j.z.pan@ed.ac.uk, evgeny.kharlamov@ifi.uio.no, yzqu@nju.edu.cn

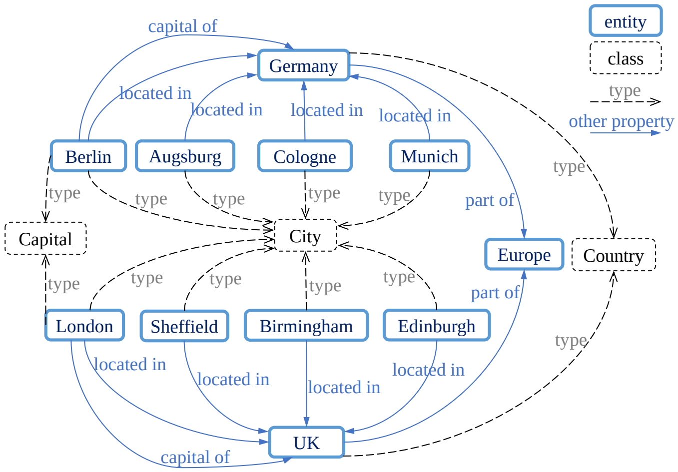

Fig. 1. An example RDF dataset.

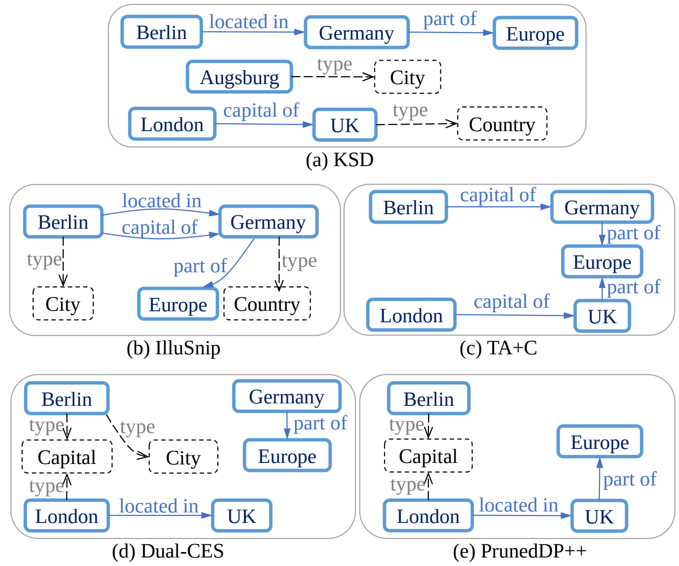

Fig. 2. Five different snippets under $k = 5$ for the dataset in Fig. 1.

benchmark data is available on GitHub.[1]

Our contributions are summarized as follows.

- We proposed six quality metrics for assessing the usefulness of a snippet in dataset search from three perspectives. Two metrics measure a snippet's representativeness of the schema-level and instance-level elements of the original dataset, two metrics measure the representativeness of description-level and link-level data patterns, and two metrics measure the query relevance of a snippet.
- For different stages of a typical dataset search process requiring different snippets to support different activities, including the search stage and the evaluate stage, we selected a different subset of suitable quality metrics to create a quality profile for each stage. For convenient comparison between snippets, we aggregated the selected metrics into a single metric representing overall usefulness for a stage.
- We created BANDAR, a publicly available benchmark for evaluating snippet generation algorithms for dataset search. We collected 2,067 keyword queries representing real-world data needs, and we indexed the data in 9,544 collected real-world RDF datasets. We paired each query with its most relevant datasets, and generated 13,429 query-dataset pairs for evaluating snippet generation.
- In addition to two existing algorithms for generating snippets for RDF datasets [13], [54], we re-implemented three adapted algorithms that were originally developed for ontology snippet generation [26], document summarization [48], and keyword search over graph data [37]. We employed all these algorithms to generate snippets for the query-dataset pairs in BANDAR.
- We used the proposed quality metrics and profiles to assess and compare the usefulness of the generated snippets in dataset search. The results revealed the strengths and weaknesses of the tested algorithms, and helped to select proper algorithms to be used for different stages of dataset search. The results also identified their common shortcomings and suggested research directions for future work.

1. https://github.com/nju-websoft/BANDAR

This article significantly extends our previous work [53] in five aspects. (1) We revised two metrics for query relevance and we added two new metrics for pattern representativeness. (2) We selected and aggregated quality metrics into quality profiles. (3) We added 9,233 real-world datasets from a new source. (4) We replaced synthetic queries with 2,067 queries representing real-world data needs, and we annotated and removed query keywords referring to metadata. (5) We added two new algorithms [48], [54] to evaluate.

The remainder of the article is organized as follows. Section 2 describes the proposed quality metrics and profiles. Section 3 presents the design of the created benchmark. Section 4 analyzes benchmark results. Section 5 discusses related work. Section 6 concludes the article.

2 Evaluation Framework

In this section, we firstly define necessary terms used in the article. Then we describe a set of metrics for evaluating the quality of a snippet. Finally we aggregate quality metrics into quality profiles for measuring the overall usefulness of a snippet for each stage of a typical dataset search process.

2.1 Preliminaries

In this work we focus on datasets in RDF format [18].

An *RDF term* is an Internationalized Resource Identifier (IRI), a blank node, or a literal. Let $\mathbf{I}$, $\mathbf{B}$, and $\mathbf{L}$ be the disjoint sets of all IRIs, blank nodes, and literals in RDF, respectively. An *RDF dataset*, or a *dataset* for short, is a non-empty set of subject-predicate-object triples $T \subseteq (\mathbf{I} \cup \mathbf{B}) \times \mathbf{I} \times (\mathbf{I} \cup \mathbf{B} \cup \mathbf{L})$. Each triple $t \in T$ consists of the subject t^s, predicate t^p, and object t^o, written as $t = \langle t^s,\ t^p,\ t^o \rangle$.

Each RDF term r in T has a *textual form* $\texttt{text}(r)$:

- if $r \in (\mathbf{I} \cup \mathbf{B})$ and there exists $t \in T$ such that $t^s = r$ and $t^p = \texttt{rdfs:label}$, then $\texttt{text}(r)$ will be the lexical form of the literal t^o;

- if $r \in \mathbf{I}$ but it is not associated with `rdfs:label`, then $\texttt{text}(r)$ will be the local name of r, i.e., the fragment component of the IRI;
- if $r \in \mathbf{L}$, then $\texttt{text}(r)$ will be the lexical form of r.

It is possible that $\texttt{text}(r)$ is an empty string, e.g., when r is a blank node not associated with any `rdfs:label`.

An RDF dataset T can be represented as a graph $\mathtt{G}(T)$ where each triple $t = \langle t^s,\ t^p,\ t^o \rangle$ is represented as an edge directed from t^s to t^o and labeled with t^p. In Fig. 1 we illustrate the graph representation of an example RDF dataset, where for RDF terms we show their textual forms.

Given an integer size constraint k, a *snippet* for an RDF dataset T is a subset of triples $S \subseteq T$ satisfying $|S| \leq k$. A snippet S can be represented as a subgraph $\mathtt{G}(S)$ of $\mathtt{G}(T)$. In Fig. 2 we illustrate five different snippets under $k = 5$ for the dataset in Fig. 1. Consider the presenting capacity of a typical dataset search engine, k is usually set to a small integer in practice.

2.2 Quality Metrics

We present six metrics for evaluating the quality of a snippet. These metrics assess the usefulness of a snippet in dataset search from three perspectives: data representativeness, pattern representativeness, and query relevance.

2.2.1 Data Representativeness

A snippet serves as a preview of the data in a dataset. Therefore, a good snippet is expected to be representative of the central data elements. We divide data elements into two levels: the level of instances (i.e., entities) and of schema (i.e., classes and properties). We consider the emergent schema of a dataset, i.e., the actual schema used in the data, because a dataset may not explicitly specify the schema it uses or may not strictly conform to its specified schema [1].

Specifically, we distinguish between entities, classes, and properties. An *entity* is an instance-level RDF term represented by an IRI or a blank node. Entities are grouped by their types into *classes*. Entities are described by *properties*, which relate an entity to other RDF terms. Formally, the *emergent schema* of a dataset T consists of a set of classes $\mathtt{C}(T)$ and a set of properties $\mathtt{P}(T)$ that are used in T:

$$\begin{aligned} \mathtt{C}(T) &= \{r : \exists t \in T,\ t^p = \texttt{rdf:type} \text{ and } t^o = r\}\,, \\ \mathtt{P}(T) &= \{r : \exists t \in T,\ t^p = r\}\,. \end{aligned} \tag{1}$$

Entities described in T are assumed to be disjoint from schema-level elements:

$$\mathtt{E}(T) = \{r \in (\mathbf{I} \cup \mathbf{B}) : \exists t \in T,\ r \in \{t^s, t^o\} \text{ and } r \notin (\mathtt{C}(T) \cup \mathtt{P}(T))\}\,. \tag{2}$$

For example, in Fig. 1 we depict entities and classes as vertices of different styles.

Now we are ready to define two quality metrics that assess the data representativeness of a snippet at different levels: schema-level and entity-level representativeness.

Schema-Level Representativeness (SkmRep). A good snippet is expected to be representative of the central schema of the dataset. Since our schema emerges from data, we measure the importance of a schema-level element r by the number of times it is used in the data. Specifically, we compute the *relative frequency* of a class (CFreq) and the relative frequency of a property (PFreq) as follows:

$$\begin{aligned} \texttt{CFreq}(r) &= \frac{|\{t \in T : t^p = \texttt{rdf:type} \text{ and } t^o = r\}|}{|\{t \in T : t^p = \texttt{rdf:type}\}|}\,, \\ \texttt{PFreq}(r) &= \frac{|\{t \in T : t^p = r\}|}{|T|}\,. \end{aligned} \tag{3}$$

For example, in Fig. 1 we have

$$\begin{aligned} &\texttt{CFreq}(\text{City}) = \frac{8}{12},\ \texttt{CFreq}(\text{Capital}) = \frac{2}{12},\ \texttt{CFreq}(\text{Country}) = \frac{2}{12}, \\ &\texttt{PFreq}(\text{located in}) = \frac{8}{24},\ \texttt{PFreq}(\text{capital of}) = \frac{2}{24}, \\ &\texttt{PFreq}(\text{part of}) = \frac{2}{24},\ \texttt{PFreq}(\text{type}) = \frac{12}{24}. \end{aligned}$$

For a snippet S, we assess its *schema-level representativeness* (SkmRep) based on the total relative frequency of the classes and properties that are observed to be used in S, i.e., that are included in the emergent schema of S. We separately compute for classes and properties and then we integrate the results using harmonic mean (H):

$$\begin{aligned} &\texttt{SkmRep}(S) = \mathtt{H}(\sum_{r \in \mathtt{C}(S)} \texttt{CFreq}(r),\ \sum_{r \in \mathtt{P}(S)} \texttt{PFreq}(r))\,, \\ &\text{where } \mathtt{H}(x, y) = \frac{2xy}{x+y}\,. \end{aligned} \tag{4}$$

$\mathtt{C}(\cdot)$ and $\mathtt{P}(\cdot)$ are defined in Eq. (1). We choose harmonic mean because it is dominated by the minimum of its arguments, i.e., a snippet achieves high schema-level representativeness only if it uses both important classes and properties. Note that $\mathtt{P}(T) = \emptyset$ is impossible since $T \neq \emptyset$. However, if $\mathtt{C}(T) = \emptyset$ and hence the denominator of $\texttt{CFreq}(r)$ in Eq. (3) is zero, we will ignore classes in the computation of SkmRep:

$$\texttt{SkmRep}(S) = \sum_{r \in \mathtt{P}(S)} \texttt{PFreq}(r) \quad \text{if } \mathtt{C}(T) = \emptyset\,. \tag{5}$$

In Eq. (4) and Eq. (5), SkmRep is in the range of 0–1. For example, the snippet in Fig. 2(b) uses two classes (City and Country) and all the four properties that are used in the dataset. Its SkmRep is

$$\mathtt{H}(\frac{8}{12} + \frac{2}{12},\ \frac{8}{24} + \frac{2}{24} + \frac{2}{24} + \frac{12}{24}) = 0.91\,.$$

Entity-Level Representativeness (EntRep). A good snippet is also expected to be representative of the central entities in the dataset. There are many and various ways of measuring the importance of an entity [31]. We rely on the graph structure of $\mathtt{G}(T)$ and we compute the out-degree ($\mathtt{d}^+$) and in-degree ($\mathtt{d}^-$) of entity r to characterize its centrality:

$$\mathtt{d}^+(r) = |\{t \in T : t^s = r\}|\,, \quad \mathtt{d}^-(r) = |\{t \in T : t^o = r\}|\,. \tag{6}$$

We choose degree for efficiency and interpretability reasons. Degree can be inexpensively computed, and is easily understandable. Out-degree indicates the richness of the description of an entity, and in-degree indicates the influence of an entity on other entities. For example, in Fig. 1 we have

$$\begin{aligned} &\mathtt{d}^+(\text{Berlin}) = 4,\ \mathtt{d}^+(\text{Germany}) = 2,\ \mathtt{d}^+(\text{Europe}) = 0, \\ &\mathtt{d}^-(\text{Berlin}) = 0,\ \mathtt{d}^-(\text{Germany}) = 5,\ \mathtt{d}^-(\text{Europe}) = 2. \end{aligned}$$

Berlin and Germany are entities having the largest out-degree and in-degree in Fig. 1, respectively.

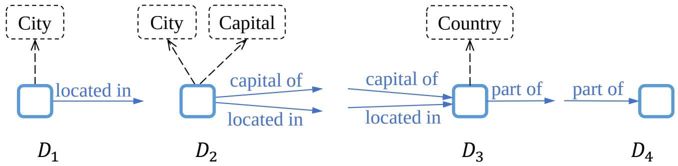

Fig. 3. Four EDPs for the entities described in Fig. 1.

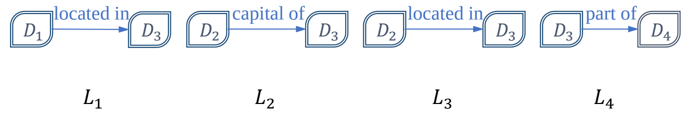

Fig. 4. Four LPs for the links in Fig. 1, based on EDPs depicted in Fig. 3.

For a snippet S, we assess its *entity-level representativeness* ($\mathtt{EntRep}$) based on the average normalized out-degree and in-degree of the entities that are described in S. Degree values are normalized firstly by logarithmizing each value because they usually follow a highly skewed power-law distribution in practice [25], and then by dividing each value by the largest value observed in T. We separately compute for out-degrees and in-degrees and then we integrate the results using harmonic mean ($\mathtt{H}$):

$$\mathtt{EntRep}(S) = \mathtt{H}\Big(\frac{1}{|\mathtt{E}(S)|} \cdot \sum_{r \in \mathtt{E}(S)} \frac{\log(\mathtt{d}^+(r)+1)}{\max_{r' \in \mathtt{E}(T)} \log(\mathtt{d}^+(r')+1)}, \frac{1}{|\mathtt{E}(S)|} \cdot \sum_{r \in \mathtt{E}(S)} \frac{\log(\mathtt{d}^-(r)+1)}{\max_{r' \in \mathtt{E}(T)} \log(\mathtt{d}^-(r')+1)}\Big), \tag{7}$$

where $\mathtt{E}(\cdot)$ is defined in Eq. (2). With harmonic mean, a snippet achieves high entity-level representativeness only if both its out-degree and in-degree are large. However, if the in-degrees of the entities in T are all zero (i.e., entities are never linked to each other) and hence the denominator of the last fraction in Eq. (7) is zero, we will ignore in-degrees in the computation of $\mathtt{EntRep}$:

$$\mathtt{EntRep}(S) = \frac{1}{|\mathtt{E}(S)|} \cdot \sum_{r \in \mathtt{E}(S)} \frac{\log(\mathtt{d}^+(r)+1)}{\max_{r' \in \mathtt{E}(T)} \log(\mathtt{d}^+(r')+1)} \quad \text{if } \forall r' \in \mathtt{E}(T),\ \mathtt{d}^-(r') = 0\,. \tag{8}$$

Theoretically, the largest out-degree observed in T can also be zero (i.e., no entities are described). We ignore such trivial datasets. In Eq. (7) and Eq. (8), $\mathtt{EntRep}$ is in the range of 0–1. For example, the snippet in Fig. 2(b) describes three entities (Berlin, Germany, Europe). Its $\mathtt{EntRep}$ is

$$\mathtt{H}\big(\tfrac{1}{3}(\tfrac{\log 5}{\log 5} + \tfrac{\log 3}{\log 5} + \tfrac{\log 1}{\log 5}),\ \tfrac{1}{3}(\tfrac{\log 1}{\log 6} + \tfrac{\log 6}{\log 6} + \tfrac{\log 3}{\log 6})\big) = 0.55\,.$$

2.2.2 *Pattern Representativeness*

As a preview of the data in a dataset, a good snippet is expected to show not only central data elements but also how data elements are regularly organized, i.e., central data patterns, which provide a fine-grained view of data. We focus on two types of data patterns: patterns for entity descriptions and for links between entities. We consider emergent patterns, i.e., the actual patterns observed in data.

Specifically, an entity is described in a subset of triples of T using schema-level elements (i.e., classes and properties). The *entity description pattern* (EDP) for an entity $r \in \mathtt{E}(T)$, denoted by $\mathtt{EDP}_T(r)$, consists of a set of classes $\mathtt{EDP}^c_T(r)$, a set of forward properties $\mathtt{EDP}^f_T(r)$, and a set of backward properties $\mathtt{EDP}^b_T(r)$ that are used to describe r in T:

$$\begin{aligned} &\mathtt{EDP}_T(r) = \langle \mathtt{EDP}^c_T(r),\ \mathtt{EDP}^f_T(r),\ \mathtt{EDP}^b_T(r) \rangle\,, \text{ where} \\ &\mathtt{EDP}^c_T(r) = \{r' \in \mathtt{C}(T) : \exists t \in T,\ t^s = r,\ t^p = \mathtt{rdf{:}type},\ t^o = r'\}\,, \\ &\mathtt{EDP}^f_T(r) = \{r' \in (\mathtt{P}(T) \setminus \{\mathtt{rdf{:}type}\}) : \exists t \in T,\ t^s = r,\ t^p = r'\}\,, \\ &\mathtt{EDP}^b_T(r) = \{r' \in (\mathtt{P}(T) \setminus \{\mathtt{rdf{:}type}\}) : \exists t \in T,\ t^p = r',\ t^o = r\}\,, \end{aligned} \tag{9}$$

where $\mathtt{C}(\cdot)$ and $\mathtt{P}(\cdot)$ are defined in Eq. (1). Our definition of EDP extends [6], [57] where backward properties are not considered. For example, in Fig. 3 we depict all the unique EDPs for the entities described in Fig. 1.

Moreover, entities are linked to each other by properties in T. Let $t \in T$ be a triple representing a link between two entities, i.e., $t^s, t^o \in \mathtt{E}(T)$. The *link pattern* (LP) for t, denoted by $\mathtt{LP}_T(t)$, consists of the EDP for t^s, the property t^p, and the EDP for t^o in T:

$$\mathtt{LP}_T(t) = \langle \mathtt{EDP}_T(t^s),\ t^p,\ \mathtt{EDP}_T(t^o) \rangle\,. \tag{10}$$

For example, in Fig. 4 we depict all the unique LPs for the links in Fig. 1.

Now we are ready to define two quality metrics that assess the pattern representativeness of a snippet at different levels: description-level and link-level representativeness.

Description-Level Representativeness ($\mathtt{DescRep}$). A good snippet is expected to be representative of the central EDPs in the dataset. Since our EDPs emerge from the data, we measure the importance of EDP D by the number of times it is observed in the data. Specifically, we compute the *relative frequency* of an EDP ($\mathtt{DFreq}$) as follows:

$$\mathtt{DFreq}(D) = \frac{|\{r \in \mathtt{E}(T) : \mathtt{EDP}_T(r) = D\}|}{|\mathtt{E}(T)|}\,, \tag{11}$$

where $\mathtt{E}(\cdot)$ is defined in Eq. (2). For example, for the EDPs in Fig. 3 we have

$$\mathtt{DFreq}(D_1) = \frac{6}{11},\ \mathtt{DFreq}(D_2) = \frac{2}{11},$$
$$\mathtt{DFreq}(D_3) = \frac{2}{11},\ \mathtt{DFreq}(D_4) = \frac{1}{11}\,.$$

For a snippet S, we assess its *description-level representativeness* ($\mathtt{DescRep}$) based on the total relative frequency of the EDPs that are preserved in S, i.e., that are observed on entity descriptions preserved in S:

$$\begin{aligned} &\mathtt{DescRep}(S) = \sum_{D \in \mathtt{EDPS}(S)} \mathtt{DFreq}(D)\,, \\ &\text{where } \mathtt{EDPS}(S) = \{D : \exists r \in \mathtt{E}(S),\ D = \mathtt{EDP}_S(r) = \mathtt{EDP}_T(r)\}\,. \end{aligned} \tag{12}$$

Note that to preserve $\mathtt{EDP}_T(r)$ in S, i.e., $\mathtt{EDP}_S(r) = \mathtt{EDP}_T(r)$, if r is described by some property in more than one triple in T, then S only needs to include one of these triples. $\mathtt{DescRep}$ is in the range of 0–1. For example, the snippet in Fig. 2(b) preserves the EDPs for two entities: D_3 for

Germany and D_4 for Europe in Fig. 3, but it fails to preserve the EDP for Berlin. Its DescRep is

$$\frac{2}{11} + \frac{1}{11} = 0.27\,.$$

Link-Level Representativeness (LinkRep). A good snippet is also expected to be representative of the central LPs in the dataset. We measure the importance of LP L by the number of times it is observed in the data. Specifically, we compute the *relative frequency* of a LP (LFreq) as follows:

$$\texttt{LFreq}(L) = \frac{|\{t \in T : \texttt{LP}_T(t) = L\}|}{|\{t \in T : t^s, t^o \in \texttt{E}(T)\}|}\,. \tag{13}$$

For example, for the LPs in Fig. 4 we have

$$\texttt{LFreq}(L_1) = \frac{6}{12},\ \texttt{LFreq}(L_2) = \frac{2}{12},$$
$$\texttt{LFreq}(L_3) = \frac{2}{12},\ \texttt{LFreq}(L_4) = \frac{2}{12}\,.$$

For a snippet S, we assess its *link-level representativeness* (LinkRep) based on the total relative frequency of the LPs that are preserved in S, i.e., that are observed on links and linked entity descriptions preserved in S:

$$\texttt{LinkRep}(S) = \sum_{L \in \texttt{LPS}(S)} \texttt{LFreq}(L)\,,$$
$$\text{where } \texttt{LPS}(S) = \{L : \exists t \in S,\ L = \texttt{LP}_S(t) = \texttt{LP}_T(t)\}\,. \tag{14}$$

LinkRep is in the range of 0–1. For example, the snippet in Fig. 2(b) preserves the LP for one triple: L_4 in Fig. 4 for triple $\langle$Germany, part of, Europe$\rangle$, but it fails to preserve the LP for triple $\langle$Berlin, located in, Germany$\rangle$ because the EDP for Berlin is not preserved. Its LinkRep is

$$\frac{2}{12} = 0.17\,.$$

2.2.3 Query Relevance

In the application of dataset search, for a dataset retrieved for a query, a good snippet is expected to reveal how the dataset is relevant to the query. We focus on keyword queries as they have been widely supported by existing search engines. We analyze query relevance at two levels: the level of single keywords and of the entire query.

Specifically, a *keyword query* $Q = \langle q_1, \ldots, q_g \rangle$ is a sequence of g keywords. We assume an indicator function hit that, for each keyword q and each RDF term r, returns whether r matches q. This function can be implemented in various ways. For our experiments we follow a standard information retrieval process to transform text(r), the textual form of r, into a sequence of words and perform case-insensitive word stem matching:

$$\texttt{hit}(q, r) = \begin{cases} 1 & \text{if } q \text{ and a word in } \texttt{text}(r) \text{ have the same stem}\,, \\ 0 & \text{otherwise}\,. \end{cases} \tag{15}$$

Note that changing hit may affect evaluation results.

Now we are ready to define two quality metrics that assess the query relevance of a snippet at two levels: keyword-level and query-level relevance.

Keyword-Level Relevance (KwRel). A good snippet is expected to match as many keywords in the query as possible. We compute the set of keywords in Q that are matched in T; they form the largest possible subset of Q that can be matched in a snippet:

$$\texttt{Kws}(T) = \{q \in Q : \exists t \in T,\ \exists r \in \{t^s, t^p, t^o\},\ \texttt{hit}(q, r) = 1\}\,. \tag{16}$$

For a snippet S, we assess its *keyword-level relevance* (KwRel) based on the proportion of keywords that are matched in S:

$$\texttt{KwRel}(S) = \frac{|\texttt{Kws}(S)|}{|\texttt{Kws}(T)|}\,. \tag{17}$$

KwRel is in the range of 0–1. However, if $\texttt{Kws}(T) = \emptyset$ and hence both the numerator and the denominator of KwRel in Eq. (17) are zero, we will leave KwRel undefined. This trivial case should not occur in practice because a search engine would not retrieve such T for Q. For example, for keyword query *london berlin europe*, all the three keywords are matched in the dataset in Fig. 1. The snippet in Fig. 2(b) matches two keywords in the query (*berlin* and *europe*). Its KwRel is

$$\frac{2}{3} = 0.67\,.$$

Note that our definition of KwRel in Eq. (17) generalizes its old version in our conference paper [53] where the experiments were limited to datasets that could match *all* the query keywords and thus the denominator of KwRel was $|Q|$ rather than $|\texttt{Kws}(T)|$. Now, the extended definition of KwRel is also suitable for evaluating a snippet for a dataset that only matches a proper subset of query keywords, e.g., in a search engine that retrieves datasets using OR as the default Boolean operator between keywords in a query.

Query-Level Relevance (QryRel). To reveal query relevance, a good snippet is expected to not only match each keyword in the query but also capture their connections as per the query. In a query which is a sequence of keywords, a pair of consecutive keywords probably refer to the same concept or to two related concepts. A captured connection between two consecutive keywords is represented as a path in the graph representation of a snippet, i.e., the two keywords are in the same connected component of the graph. This goes beyond the scope of conventional information retrieval evaluation. To realize it, we partition T into a disjoint union of subsets $\texttt{CC}(T) = \{T_1, \ldots, T_w\}$; $\texttt{G}(T_i)$ for $1 \leq i \leq w$, the graph representation of T_i, is a unique connected component of $\texttt{G}(T)$. We compute the set of ordered pairs of consecutive keywords in $Q = \langle q_1, \ldots, q_g \rangle$ that are connected in T; they form the largest possible subset of pairs of consecutive keywords that can be connected in a snippet:

$$\texttt{Kwp}(T) = \{\langle q_j, q_{j+1} \rangle \in Q \times Q : \exists T_i \in \texttt{CC}(T),\ q_j, q_{j+1} \in \texttt{Kws}(T_i)\}\,, \tag{18}$$

where $\texttt{Kws}(\cdot)$ is defined in Eq. (16).

For a snippet S, we assess its *query-level relevance* (QryRel) based on the proportion of ordered pairs of consecutive keywords that are connected in S:

$$\texttt{QryRel}(S) = \frac{|\texttt{Kwp}(S)|}{|\texttt{Kwp}(T)|}\,. \tag{19}$$

If $\texttt{Kwp}(T) = \emptyset$ and hence both the numerator and denominator of QryRel in Eq. (19) are zero, we cannot leave QryRel undefined because this case frequently occurs in practice. Instead, we will reduce query-level relevance QryRel to keyword-level relevance KwRel since no pairs of consecutive keywords can be connected by any snippet:

$$\texttt{QryRel}(S) = \texttt{KwRel}(S) \quad \text{if } \texttt{Kwp}(T) = \emptyset\,. \tag{20}$$

In Eq. (19) and Eq. (20), QryRel is in the range of 0–1. For example, for keyword query *london berlin europe*, the snippet in Fig. 2(b) captures a connection between one ordered pair of consecutive keywords in the query (*berlin* and *europe*). Its QryRel is

$$\frac{1}{2} = 0.50\,.$$

Note that our definition of QryRel in Eq. (19) is different from its old version in our conference paper [53] where *all* pairs of keywords in the query were used. Now, the new definition of QryRel is focused on ordered pairs of consecutive keywords. This modification is reasonable as in many cases a user may not have a particular interest in the long-distance dependencies between nonconsecutive keywords in the query, although in some cases such connections may be needed. However, the new definition is more cost-effective, reducing the computation complexity from $\binom{|Q|}{2}$ pairs (i.e., quadratic) to $(|Q|-1)$ pairs (i.e., linear). As we will see in the experiments, the two definitions of QryRel are strongly correlated in practice.

2.3 Quality Profiles

We have presented six metrics for evaluating the quality of a snippet. They may be incomplete and can be extended by future research. However, it is both impossible and unnecessary for a snippet to exhibit high quality in all these aspects. Indeed, search is a complex process, consisting of multiple stages and involving various activities [34]. Dataset search is no exception. In different stages of a dataset search process, different kinds of snippets are needed to support different search activities, where the usefulness of a snippet is thus to be assessed from different perspectives. Therefore, we select and aggregate the proposed quality metrics in different ways into different *quality profiles*, and measure the *overall quality* of a snippet *for each stage* of dataset search.

In [33], the process of dataset search is divided into five "pillars" or stages, and a dataset search system is expected to focus on two of these stages: *search* and *evaluate*. This is consistent with the implementation of current systems [4], [45]. Specifically, in the search stage, a user submits a keyword query, and the system retrieves a list of top-ranked datasets and presents the list in a search results page. The user quickly browses the list to identify datasets that are probably relevant to the query. In the evaluate stage, the user clicks a dataset and opens a new page where the system provides detailed information about the clicked dataset for the user to evaluate and decide whether to use it. Following the analysis in [33], for each of these two stages we select a subset of suitable quality metrics to create a quality profile. Note that new stages may be supported by future systems, and new profiles can be created accordingly.

Quality Profile for the Search Stage (QS). In this stage, the user wants to quickly scan through and filter a list of datasets, and the primary concern is *query relevance*. Snippets here would be expected to help the user filter out datasets that are not relevant to the query. If multiple datasets contain relevant data, the user may give priority to datasets where the central data elements are relevant. This decision can be made with the assistance of snippets featuring high *data representativeness* by exemplifying the central schema and entities in a dataset. Therefore, we select four quality metrics to form a quality profile for the search stage: query relevance (KwRel and QryRel) and data representativeness (SkmRep and EntRep).

To assess the usefulness of a snippet S for the search stage, we rely on not all but only the above selected quality metrics. For convenient comparison between different snippets, we can also aggregate the selected metrics into a single metric representing the *overall quality for the search stage* (QS):

$$\texttt{QS}(S) = \frac{1}{4}(\texttt{KwRel}(S)+\texttt{QryRel}(S)+\texttt{SkmRep}(S)+\texttt{EntRep}(S))\,. \tag{21}$$

QS is in the range of 0–1. For example, for the snippet in Fig. 2(b), its QS is

$$\frac{1}{4}(0.67+0.50+0.91+0.55) = 0.66\,.$$

One can extend Eq. (21) to a weighted sum, but setting proper weights may be difficult and is outside our focus.

Quality Profile for the Evaluate Stage (QE). In this stage, the user wants to decide the usefulness of a dataset and hence needs to more carefully examine the data. Snippets here would be expected to provide a representative preview of data. Knowing the central elements and patterns in the data is beneficial to the understanding and sensemaking of data, and can help the user decide whether this dataset contains the right data that the user is seeking. This decision can be made with the assistance of snippets featuring high *data representativeness* and high *pattern representativeness*. Therefore, we select four quality metrics to form a quality profile for the evaluate stage: data representativeness (SkmRep and EntRep) and pattern representativeness (DescRep and LinkRep).

To assess the usefulness of a snippet S for the evaluate stage, we only use the above selected quality metrics. Again, we can aggregate the selected metrics into a single metric representing the *overall quality for the evaluate stage* (QE):

$$\texttt{QE}(S) = \frac{1}{4}(\texttt{SkmRep}(S)+\texttt{EntRep}(S)+\texttt{DescRep}(S)+\texttt{LinkRep}(S))\,. \tag{22}$$

QE is in the range of 0–1. For example, for the snippet in Fig. 2(b), its QE is

$$\frac{1}{4}(0.91+0.55+0.27+0.17) = 0.48\,.$$

One can also extend Eq. (22) to a weighted sum.

3 Benchmark Design

In this section, we present the design of our benchmark. We firstly introduce the datasets and queries we used. Then we describe the algorithms selected to be benchmarked. Finally we present experiment settings.

3.1 Datasets and Queries

We used real-world RDF datasets and we derived keyword queries from real-world data needs. We combined each query with the most relevant datasets, for which snippets were to be generated by benchmarked algorithms.

TABLE 1
Statistics about Collected RDF Datasets

Sources	#datasets	#triples		#classes		#properties		#entities		#EDPs		#LPs	
		Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
DataHub	311	275,885	20,968,879	19	2,030	39	3,982	54,567	5,399,234	1,005	270,224	1,490	156,722
Data.gov	9,233	7,131	6,343,524	1	2	21	545	411	273,774	11	500	15	1,103

TABLE 2
Statistics about Collected Keyword Queries

Sources	#queries		#words in a query		
	All	With data words	Min	Mean	Max
WWW-18	449	399	2	9	18
CIKM-19	1,498	843	2	9	21
ECIR-20	120	114	3	8	21

TABLE 3
Dataset Distribution of Generated Q-D Pairs

Sources	#Q-D pairs	#distinct datasets
DataHub	3,677	124
Data.gov	9,752	1,804
Total	13,429	1,928

TABLE 4
Query Distribution of Generated Q-D Pairs

Sources	#Q-D pairs	#distinct queries
WWW-18	3,981	399
CIKM-19	8,308	832
ECIR-20	1,140	114
Total	13,429	1,345

3.1.1 RDF Datasets

We collected 9,544 real-world RDF datasets from two well-known data portals.

- **DataHub**: We used the CKAN API to retrieve the metadata of all the 11,462 datasets indexed by DataHub.[2] For 1,262 datasets we found dump files in RDF format, namely N-Triples, RDF/XML, or Turtle. We successfully downloaded and used Apache Jena v3.8.0 to parse 311 RDF datasets.
- **Data.gov**: We followed the same procedure to collect from Data.gov.[3] We retrieved the metadata of 230,579 datasets, found RDF dump files for 11,846 datasets, and successfully processed 9,233 RDF datasets.

In Table 1 we show some basic statistics about the collected RDF datasets. In general, compared with DataHub, we collected more but smaller datasets from Data.gov.

3.1.2 Keyword Queries

We collected 2,067 keyword queries representing real-world data needs from three published datasets. We could not find other real queries for datasets at the time of experiments.

- **WWW-18** [32]: The authors published 449 queries[4] that were manually generated from data requests to data.gov.uk.
- **CIKM-19** [11]: The authors published 1,498 queries[5] that were manually generated from data needs posted on Stack Overflow, Open Data Stack Exchange, and Reddit.
- **ECIR-20** [12]: The authors published 120 queries[6] for a set of dataset search tasks.

In Table 2 we show some basic statistics about the collected queries. An example query is

Datasets about social media usage by country and age from Google . (23)

For each query, we removed stop words, and we manually annotated and filtered the remaining words according to the query annotation scheme we presented in [11]. We distinguished between words to be matched with the data and with the metadata of a dataset.

- **Data words** referred to classes, properties, entities, and data values that should appear in the data of a dataset, e.g., *country* and *age* in Eq. (23).
- **Metadata words** referred to the name, format, language, accessibility, provenance, and statistics about a dataset that should appear in the metadata rather than the data of a dataset, e.g., *Google* in Eq. (23).

Metadata words were removed from queries because we focused on snippets generated from data rather than metadata. Otherwise, data might mistakenly match metadata words and cause some algorithms to generate undesirable query-biased snippets which would distort evaluation results. As shown in Table 2, most queries contained data words. The mean and maximum of the number of data words in a query were 3.29 and 15, respectively.

3.1.3 Query-Dataset Pairs

We combined each query with up to 10 most relevant datasets. Specifically, we used Apache Lucene v7.5.0 to index the data in each dataset as a *pesudo document*. Each triple was transformed into a sentence in the document by concatenating the textual forms of the subject, predicate, and object. Search was performed using OR as the default Boolean operator between words in a query. Words were lowercased and stemmed before matching. Search results were ranked by the default scoring function in Lucene.

We generated 13,429 query-dataset pairs, or *Q-D pairs* for short. In Table 3 and Table 4, we show their dataset distribution and query distribution, respectively. These Q-D pairs were diverse, involving 1,928 distinct datasets and 1,345 distinct queries from different sources.

2. https://old.datahub.io/
3. https://www.data.gov/
4. https://github.com/chabrowa/data-requests-query-dataset
5. http://ws.nju.edu.cn/datasetsearch/query-cikm2019/
6. https://github.com/Zhiyu-Chen/ECIR2020-dataset-search

3.2 Participating Algorithms

To the best of our knowledge, there were only a few algorithms for generating snippets for RDF datasets [2], [13], [39], [54]. We excluded [2] because this algorithm required manual definition of property rankings; it was impracticable to define for 1,928 datasets used in our experiments. To widen the scope of our experiments, we also adapted state-of-the-art algorithms for several related problems to our problem [26], [37], [48]. We sought to configure each algorithm according to the setting recommended by its authors since such a "standard" setting would likely be followed by its users. However, some algorithms in the evaluation might have performed better in other settings.

3.2.1 *KSD*

KSD [54] represents the state of the art in generating query-biased snippets for RDF datasets.

KSD formulates the selection of triples as a weighted maximum coverage (WMC) problem, where elements to be covered include: query keywords weighted evenly, classes and properties weighted by their relative frequencies, and entities weighted by the harmonic mean of their normalized out-degrees and in-degrees. A triple t is regarded as a set that covers keywords matched in t, classes and properties used in t, and entities described in t. The goal is to choose at most k triples that maximize the total weight of covered elements. The WMC problem is solved by a greedy algorithm which, in each iteration, chooses a triple that contains the largest weight of uncovered elements.

We held the source code of KSD. All its parameters were set to the values suggested in [54].

3.2.2 *IlluSnip*

IlluSnip [13], [39] generates a snippet for an RDF dataset to illustrate its main content.

IlluSnip formulates the selection of triples as a Maximum-weight-and-coverage Connected Graph (MwcCG) problem, where elements to be covered include: classes and properties weighted by their relative frequencies, and entities weighted by their normalized PageRank scores. A triple t is regarded as a set that covers classes and properties used in t, and entities described in t. The goal is to choose at most k triples that maximize the total weight of covered elements, subject to the constraint that the graph representation of the selected triples is a connected graph. The MwcCG problem is solved by a multi-start greedy algorithm which greedily constructs a solution starting from each triple and, in each iteration of a construction process, chooses a triple that contains the largest weight of uncovered elements.

We re-implemented IlluSnip. All its parameters were set to the values suggested in [13].

3.2.3 *TA+C*

TA+C [26] represents the state of the art in generating query-biased snippets for ontologies. It processes an ontology as an RDF graph, i.e., the graph representation of an RDF dataset, and hence it can be applied to our problem.

TA+C transforms an RDF graph into a term association graph (TAG) where each vertex represents an RDF term, and each edge represents a set of RDF sentences connecting two RDF terms. An RDF sentence is either a triple not containing any blank nodes or a maximal set of triples containing common blank nodes. RDF sentences at the schema level and at the instance level are assigned different weights. The weight of an edge is the total weight of the RDF sentences it represents. TAG is then decomposed into maximal radius-bounded subgraphs, from which min-weight group Steiner trees (GSTs) that cover all possible query keywords are extracted as sub-snippets. Finally, sub-snippets are greedily selected and merged into a snippet containing at most k RDF terms where, in each iteration, a sub-snippet that has the smallest weight and covers the most uncovered query keywords is chosen. Note that TA+C constrains the size of a snippet in terms of the number of RDF terms. It is inappropriate to constrain it in terms of the number of triples because, for example, TA+C may produce a snippet containing isolated RDF terms that match query keywords but do not appear in any triples in the snippet.

We re-implemented TA+C. All its parameters were set to the values suggested in [26].

3.2.4 *Dual-CES*

Dual-CES [48] represents the state of the art in unsupervisedly generating query-biased snippets for documents, while supervised algorithms currently could not apply to our problem due to the lack of training data. To adapt Dual-CES to our problem, we transformed each RDF dataset into a pesudo document as described in Section 3.1.3.

Dual-CES is an extension of the CES approach [21]. It performs two-step Monte-Carlo sampling to iteratively select subsets of sentences and return the optimal subset, containing at most k sentences. Preference is given to long sentences that are close to the centroid of the document (in the first step) and are relevant to the query (in the second step). Query relevance is computed based on term-frequency vectors and unigram language models.

We re-implemented Dual-CES. Most of its parameters were set to the values suggested in [21], [48]. We modified some parameters due to the relatively very large size of an RDF dataset compared with a document. In the preliminary step of sentence pruning, we considered top-$(1000 \cdot k)$ sentences, instead of only top-150 sentences in [48], to trade runtime for summaries of higher quality. However, sampling was reduced from 10,000 times in [21] to 1,000 times in our implementation, because otherwise Dual-CES would frequently reach timeout. The first step of sampling was configured to output at most $(6 \cdot k)$ sentences.

3.2.5 *PrunedDP++*

PrunedDP++ [37] is a popular algorithm for keyword search over graph data. It extracts a min-weight group Steiner tree (GST) from graph data such as an RDF graph so that it can be applied to our problem.

A min-weight GST is a tree of minimum weight that covers all the query keywords. To compute a min-weight GST, PrunedDP++ optimizes a dynamic programming algorithm [19] and performs progressive A*-search. Note that the size of a min-weight GST is not bounded, i.e., the size of a snippet generated by PrunedDP++ is not constrainable.

TABLE 5
Statistics over 13,429 Q-D Pairs

		Non-empty snippets	Empty snippets	Timeout
KSD	$k = 20$	13,429	0	0
	$k = 40$	13,429	0	0
IlluSnip	$k = 20$	13,429	0	2,140
	$k = 40$	13,429	0	2,277
TA+C	$k = 20$	11,164	2,265	488
	$k = 40$	11,162	2,267	490
Dual-CES	$k = 20$	13,429	0	629
	$k = 40$	13,429	0	556
PrunedDP++	$k = 20$	9,819	3,610	297
	$k = 40$	9,819	3,610	297

Besides, PrunedDP++ will return an empty result if the query keywords are not connected in the graph.

We re-implemented PrunedDP++. In [37] it was unclear how to weight edges. We followed [19] to define the weight of an edge as the normalized degrees of its endpoints.

3.2.6 Running Example

In Fig. 2 we show the snippets generated by the above five algorithms for the example RDF dataset in Fig. 1 w.r.t. keyword query *london berlin europe*.

3.3 Experiment Settings

For the size constraint k, we experimented with two values: $k = 20$ and $k = 40$. Small snippets ($k = 20$) could be used in the search stage of dataset search, to be compactly presented in a search results page and help users quickly identify relevant datasets. Large snippets ($k = 40$) could be used in the evaluate stage of dataset search, to provide more detailed information for dataset evaluation.

Our experiments were performed on an Intel Xeon E7-4820 with 80GB memory for the JVM. As a preprocessing step, we materialized inverted indexes for efficient keyword matching in KSD, TA+C, Dual-CES, and PrunedDP++. For TA+C, we also indexed all the maximal 1-radius subgraphs in each dataset for efficient extraction of sub-snippets.

For generating a single snippet we set a timeout of 1,000 seconds. It was the longest time our computational resources could afford for the experiments, and it should already be very long for generating a snippet in practice. When timeout was reached, the runtime would be defined as 1,000 seconds, and the generating process would be terminated. In that case, for algorithms that iteratively generated better snippets (IlluSnip and Dual-CES), the best snippet found at timeout was returned. For other algorithms (KSD, TA+C, and PrunedDP++), timeout indicated failure.

4 Results and Analysis

We ran each algorithm on each of the 13,429 Q-D pairs. As shown in Table 5, KSD successfully generated non-empty snippets for all the Q-D pairs without reaching timeout. IlluSnip and Dual-CES also consistently generated non-empty snippets but they reached timeout for 2,140–2,277 (16–17%) and 556–629 (4–5%) Q-D pairs, respectively. In these cases, the returned snippets might not be the optimal ones that these algorithms could find given unlimited time. TA+C and PrunedDP++ generated empty snippets for 2,265–2,267 (17%) and 3,610 (27%) Q-D pairs, respectively. In addition to timeout failure, reasons for these empty snippets included: failing to precompute an index for a dataset in 12 hours (TA+C), failing to find matching vertices for every query keyword (TA+C and PrunedDP++), and failing to find any GST to connect all the query keywords (PrunedDP++). Therefore, the use of TA+C and PrunedDP++ would be limited in practice.

The values of quality metrics reported in the following were computed over non-empty snippets.

4.1 Quality Metrics

4.1.1 Overall Results

Table 6 shows the mean values of quality metrics computed over all the non-empty snippets.

KSD achieved fairly high data representativeness (`SkmRep` and `EntRep`) and query relevance (`KwRel`). Its mean values of these quality metrics were close to the highest values achieved by other algorithms. Indeed, KSD was designed to keep a balance between these aspects. It was implemented to optimize the coverage of frequent classes, properties, central entities, and query keywords. The absolute values of `EntRep`, 0.1938–0.2297, were not high because KSD selected entities with high out-degrees *or* in-degrees, whereas `EntRep` would be high only if out-degree *and* in-degree were both high. Query-level relevance was not as high as keyword-level relevance ($0.5 <$ `QryRel` < 0.6) because connectivity was ignored in the design of KSD.

IlluSnip showed the highest data representativeness (`SkmRep` > 0.75 and `EntRep` > 0.25). It was not surprising because IlluSnip was exactly designed to optimize the coverage of frequent classes, properties, and central entities. However, IlluSnip also showed the lowest query relevance (`KwRel` < 0.4 and `QryRel` < 0.3) since it generated query-unbiased snippets. The design of IlluSnip did not consider query relevance.

TA+C exhibited very high keyword-level relevance (`KwRel` > 0.9) by greedily selecting sub-snippets to cover more query keywords. Query-level relevance was not as high as keyword-level relevance ($0.4 <$ `QryRel` < 0.5) because sub-snippets were extracted from radius-bounded subgraphs which could not capture long-distance connections between keywords. TA+C exhibited the lowest data representativeness (`SkmRep` < 0.05 and `EntRep` < 0.05), which was not the focus of its design. Increasing k from 20 to 40 did not noticeably increase its data representative or query relevance, different from some other algorithms.

Dual-CES seemed to have achieved a better trade-off than TA+C. Its keyword-level relevance (`KwRel`) was close to TA+C, but its data representativeness (`SkmRep` and `EntRep`) was much higher because Dual-CES preferred central sentences (i.e., triples) which often used frequent classes, properties, and described central entities. However, this trade-off was not as good as the trade-off achieved by KSD, where data representativeness was notably higher but query relevance was still close.

TABLE 6
Mean Values of Quality Metrics over Non-Empty Snippets

		`SkmRep`	`EntRep`	`DescRep`	`LinkRep`	`KwRel`	`QryRel`	`QryRel` (non-trivial)
KSD	$k = 20$	0.6404	0.2297	0.0873	0.0014	0.8624	0.5412	0.4059
	$k = 40$	0.7097	0.1938	0.0949	0.0015	0.8805	0.5555	0.4228
IlluSnip	$k = 20$	0.7580	**0.3093**	0.2088	0.0171	0.3120	0.2267	0.1097
	$k = 40$	**0.8570**	0.2622	**0.2655**	**0.0228**	0.3821	0.2779	0.1606
TA+C	$k = 20$	0.0153	0.0298	0.0045	0.0000	0.9395	0.4630	0.3525
	$k = 40$	0.0175	0.0308	0.0045	0.0000	0.9395	0.4690	0.3597
Dual-CES	$k = 20$	0.2320	0.1060	0.0400	0.0009	0.8896	0.6486	0.5407
	$k = 40$	0.4729	0.1144	0.0743	0.0022	0.9098	0.7089	0.6198
PrunedDP++	$k = 20$	0.1018	0.1212	0.0383	0.0000	**1.0000**	**1.0000**	**1.0000**
	$k = 40$	0.1018	0.1212	0.0383	0.0000	**1.0000**	**1.0000**	**1.0000**

PrunedDP++ showed perfect query relevance (`KwRel` = 1 and `QryRel` = 1) because every non-empty snippet it generated was guaranteed to cover and connect all the query keywords. Its data representativeness (`SkmRep` and `EntRep`) was low as they were not considered in the design of PrunedDP++. However, its entity-level representativeness (`EntRep`) was higher than TA+C because query keywords were often connected by paths passing through central entities.

Query-level relevance would be reduced to keyword-level relevance if no pairs of consecutive keywords could be connected by any snippet, as shown in Eq. (20). The above-mentioned mean values of `QryRel` might have been distorted because for $3,199$ out of the 13,429 Q-D pairs (24%), `QryRel` was trivially reduced to `KwRel`. We re-computed the mean value of `QryRel` over the $10,230$ non-trivial Q-D pairs. As shown in Table 6, whereas PrunedDP++ still achieved perfect results, the differences between algorithms became more noticeable. We also calculated the Pearson correlation coefficient between `QryRel` and its old version [53]. The result of 0.969 suggested their strong correlation, thus demonstrating the cost-effectiveness of the new version.

Pattern representativeness was not considered in any participating algorithm. KSD, TA+C, Dual-CES, and PrunedDP++ achieved very low values of `DescRep` < 0.1 and `LinkRep` < 0.1. IlluSnip exhibited a bit higher description-level representativeness (`DescRep` > 0.2) because it was optimized to use frequent classes and properties and, more importantly, the graph representations of the generated snippets were connected graphs. This feature helped to raise the possibility of preserving EDPs.

To conclude, none of the five participating algorithms could lead on all the six quality metrics. IlluSnip shows the highest data and pattern representativeness, followed by KSD, but there is much room for improving pattern representativeness. PrunedDP++ leads on query relevance, followed by Dual-CES, TA+C, and KSD.

4.1.2 Result Breakdown

Figure 5(a) depicts the mean values of quality metrics computed over all queries as a radar chart. The results are broken down into queries from different sources in Figs. 5(b)–(d). The results observed over different sources were generally consistent. One exception was IlluSnip. Its query relevance (`KwRel` and `QryRel`) over ECIR-20 in Fig. 5(d) were notably lower than the results over other sources. Among 120 queries from this source, 52 queries (43%) contained temporal words, and 95 queries (79%) contained geospatial words. Such words were often matched with literals describing different entities. These literal vertices were at least 3 hops away from each other in the graph representation of an RDF dataset, and could hardly be covered by a single query-unbiased connected subgraph generated by IlluSnip.

The results are broken down into queries containing different numbers of keywords in Figs. 5(e)–(h). Most of these constituent results were similar to the overall results in Fig. 5(a). However, in Fig. 5(e), query-level relevance (`QryRel`) was slightly exaggerated for most algorithms because `QryRel` was always reduced to `KwRel` for queries containing a single keyword. When the number of keywords was increased from Fig. 5(e) to Fig. 5(h), we observed small decreases of `KwRel` and `QryRel` for KSD and Dual-CES. Query relevance was not the unique factor considered in these algorithms. Therefore, the more keywords a query contained, the more keywords and their connections these algorithms failed to cover. This phenomenon was not observed for TA+C and PrunedDP++. Keyword-level relevance was the primary concern of these algorithms. They always tried to cover as many query keywords as possible.

TABLE 7
Mean Overall Quality over Non-Empty Snippets

	`QS` ($k = 20$)	`QE` ($k = 40$)
KSD	**0.5684**	0.2500
IlluSnip	0.4015	**0.3519**
TA+C	0.3619	0.0132
Dual-CES	0.4691	0.1659
PrunedDP++	0.5558	0.0653

4.2 Quality Profiles

In Section 2.3 we selected and aggregated quality metrics into two quality profiles for the search stage and the evaluate stage of dataset search. Table 7 shows the mean overall quality computed over all the non-empty snippets.

For the **search stage**, we evaluated small snippets ($k = 20$) which could be compactly presented in a search results page to help users quickly identify relevant datasets. KSD and PrunedDP++ exhibited the highest overall quality for this stage (`QS` > 0.55). However, we would like to remind that PrunedDP++ could not generate non-empty snippets for 27% of the Q-D pairs. Therefore, KSD appeared to be a better and more reliable solution. IlluSnip and TA+C showed relatively low quality for this stage due to ignoring query relevance and data representativeness, respectively.

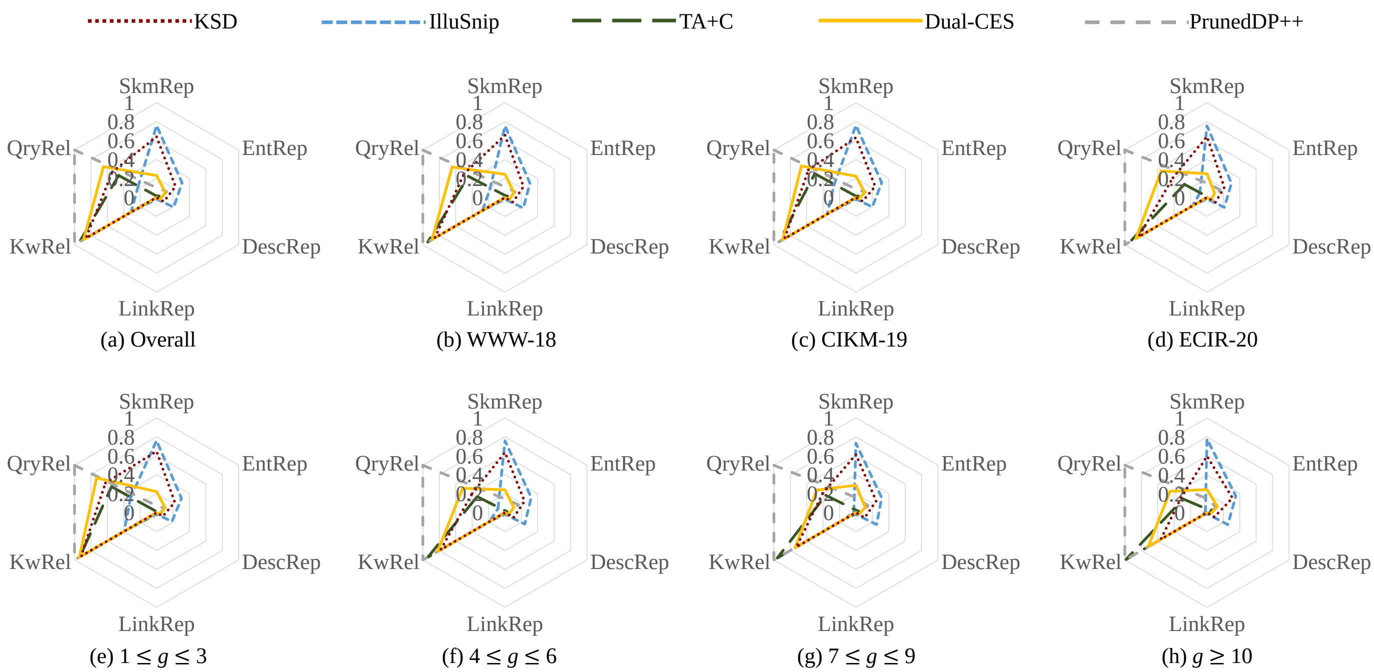

Fig. 5. Mean values of quality metrics computed over: (a) all queries, (b)–(d) queries from different sources, and (e)–(h) queries containing different numbers of keywords (i.e., g).

TABLE 8
Statistics about Preprocessing Time (s)

	Precomputed data	Median	Max
KSD	Inverted index	0.17	40
IlluSnip	–	–	–
TA+C	Inverted index and maximal 1-radius subgraphs	5.50	40,307
Dual-CES	Inverted index	0.24	52
PrunedDP++	Inverted index	0.23	59

For the **evaluate stage**, we evaluated large snippets ($k = 40$) providing more detailed information for assessing the usefulness of a dataset. IlluSnip exhibited the highest overall quality for this stage ($\texttt{QE} > 0.35$), which was not surprising as it achieved the highest data and pattern representativeness. However, this level of overall quality was not satisfactory and called for future research on pattern representativeness. Among the other algorithms, KSD exhibited higher quality ($\texttt{QE} > 0.25$) than TA+C, Dual-CES, and PrunedDP++ ($\texttt{QE} < 0.20$) where data and pattern representativeness were not their focus of design.

To conclude, among the five participating algorithms, KSD is more balanced and suitable for both stages. PrunedDP++ would also be a good choice for the search stage when it could generate non-empty snippets. IlluSnip is the top selection for the evaluate stage.

4.3 Runtime

Table 8 presents the median and maximum runtime of preprocessing in each algorithm for each Q-D pair under $k = 20$. TA+C used much more time than other algorithms due to the indexing of maximal 1-radius subgraphs.

Figure 6 depicts the distribution of runtime (excluding preprocessing) of each algorithm for each Q-D pair on a logarithmic scale. KSD was the only algorithm that never reached timeout. KSD, PrunedDP++, and TA+C used less than 1 second to generate a snippet for 92%, 88%, and 81% of the Q-D pairs, respectively, showing promising performance for practical use. IlluSnip and Dual-CES used at least 10 seconds for most Q-D pairs, but the runtime of IlluSnip as a query-independent offline algorithm seemed acceptable.

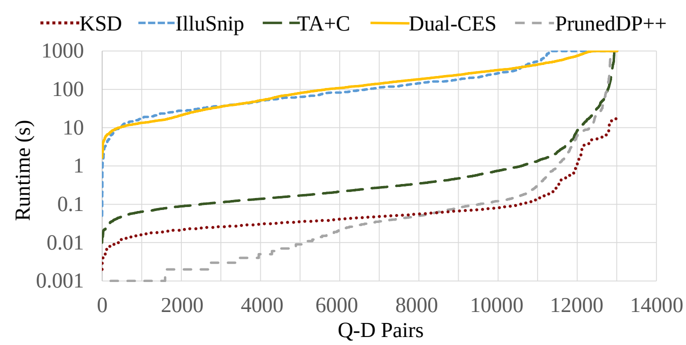

Fig. 6. Distribution of runtime for each Q-D pair.

5 Related Work

5.1 Generating Snippets for RDF Data

RDF datasets in the early ages are small and serialized into documents. To enhance **RDF document** search systems such as Sindice [43], in a pioneering work [2], triples in an RDF document describing entities that match the keyword query or occupy a central position in the document are ranked and selected into the snippet for the document. Triple ranking in this work employs predefined importance of properties, and hence can hardly be applied to the Web scale.

Recent methods can handle **large RDF datasets**. IlluSnip [13] generates a snippet to illustrate the main content of an RDF dataset. It tends to select triples where frequent classes and properties are used and central entities are described. Efficient implementations of this approach are presented in [39]. Query relevance is not considered in IlluSnip, but is incorporated into KSD [54]. This extended approach also selects triples that cover query keywords. IlluSnip and KSD were evaluated using earlier versions of a subset of quality metrics proposed in this article. IlluSnip was also evaluated by a user study, which is relatively expensive, time-consuming, and not reproducible.

Entity summarization [40] is a task focused on generating snippets for a particular kind of RDF data: triples describing an entity or several related entities. In this task, triples are selected to best characterize an entity and comprise an entity card presented in downstream applications [15], [27], [36], helping to distinguish between similar entities for quick comparison [16], [17], or to connect a set of mentioned entities for enriching a document [28], [30], [38]. The core techniques used for this entity-centred task, such as measuring the informativeness (i.e., rarity) of a triple, are fundamentally different from those needed for our problem.

Apart from human-oriented snippets, there are also data samples generated **for machine use**. In [20], [22], a subset of data is sampled to replace the original data, so that reasoning can be performed more efficiently. A similar goal is pursued in [47], [49] where a subset of triples is sampled to realize more efficient query answering while maximizing answer coverage. In distributed settings, triples are sampled to support source selection [23] or data statistics estimation [29]. These completeness-preserving data samples are much larger than our snippets as they will be used for different purposes, thus using different techniques such as hashing and query execution plan analysis.

5.2 Summarizing RDF Data

To summarize the content of an RDF dataset, a snippet can be viewed as an extractive summary, whereas a large body of research has been focused on aggregating the data into a high-level representation [6]. Such an **aggregated summary** is used to (approximately) restore the original data. For example, entities can be aggregated according to EDP-based similarity [41], [56] or (multi-hop) neighborhood similarity [51], [52]. Aggregation can be hierarchical [14]. A trade-off between the accuracy of restoration and the size of an aggregated summary is to be achieved [5]. Extraction and aggregation are complementary paradigms for data summarization. Restoration is not the purpose of our snippet.

To assess the quality of an aggregated summary, a set of **evaluation metrics** are presented in [57]. Some of these metrics rely on the existence of a gold-standard summary, which is not needed in our evaluation framework. Others are conceptually similar to our quality metrics for assessing data representativeness, but they are used to evaluate aggregated summaries rather than snippets. Moreover, we assess pattern representativeness and query relevance which are not considered in [57]. It would be interesting to adapt our metrics to evaluating aggregated summaries.

5.3 Other Related Techniques

We have identified several related problems and in the experiments we adapted their state-of-the-art solutions to our problem.

An ontology, used as the schema of an RDF dataset, can be too large to be easily browsed. **Snippet generation for ontologies** has attracted widespread research and found application in ontology search systems. Existing methods mainly process some graph representation of an ontology and employ various graph centralities to extract a subgraph [46]. They are immediately applicable to our problem, processing the graph representation of an RDF dataset. The algorithm we chose to adapt, TA+C [26], represents the state of the art in generating query-biased ontology snippets.

Document summarization is an established research topic [24]. A document snippet, typically consisting of a few sentences selected from the original document, is used in Web search systems. By transforming triples in an RDF dataset into sentences of a pesudo document, methods for document snippet generation can be applied to our problem. Although many existing methods are supervised, we adapted a state-of-the-art unsupervised query-biased algorithm [48] as we lack labeled RDF data for training.

Recent methods for **keyword search over graph data** compute a min-weight GST that spans all the query keywords [37], [50]. Applying these methods to the graph representation of an RDF dataset, as we did in the experiments with PrunedDP++ [37], a computed GST can be used as a snippet for the dataset.

All the above adapted algorithms are primarily concerned with query relevance. In our experiments they generated highly relevant snippets that appeared suitable for the search stage of data search. Their less promising results for the evaluate stage were not surprising as they were not optimized towards representativeness of the original data.

6 Conclusion and Future Work

Snippet generation is a key component of dataset search. We created BANDAR, a public benchmark for evaluating snippet generation algorithms for dataset search. It consists of 13,429 query-dataset pairs generated from thousands of collected keyword queries and RDF datasets. We used BANDAR to evaluate and compare five algorithms for snippet generation, based on six quality metrics and two aggregated quality profiles. This evaluation framework supports inexpensive and reproducible experiments without involving human experts or users in the loop. Our evaluation results showed that KSD and IlluSnip are relatively suitable for the search stage and the evaluate stage of a typical dataset search process, respectively. However, none of the tested algorithms achieved satisfactory pattern representativeness, which has not been considered in current algorithms and calls for future research. BANDAR is public and can be used to evaluate future algorithms. It helps to foster researching and developing dataset search systems.

As for future work, first, observe that our syntactic metrics rely on explicit data while RDF (Schema) entailment may deduce implicit data. We plan to incorporate semantics into evaluation by either precomputing the deductive closure of data or developing semantics-aware metrics. Second, while we used real datasets, it would also be interesting to consider synthetic datasets with controllable parameters for a systematic evaluation. Third, we want to adapt our evaluation framework to datasets in other formats, in particular tables. However, it would depend on the concrete form of snippet for tables. One universal adaptation might be mapping tables into RDF data (e.g., mapping tables and columns into classes and properties, respectively) and applying our metrics to the mapped data. Fourth, to address the limitations of existing snippets shown in the evaluation, we will study novel algorithms incorporating pattern representativeness. It might be an idea to extend IlluSnip or KSD to cover not only classes and properties but also patterns.

References

[1] Z. Abedjan, J. Lorey, and F. Naumann, "Reconciling ontologies with the web of data," in *Proc. 21st ACM Int'l Conf. Information and Knowledge Management (CIKM '12)*, Oct.–Nov. 2012, pp. 1532–1536.

[2] X. Bai, R. Delbru, and G. Tummarello, "RDF snippets for semantic web search engines," in *Proc. OTM 2008 Confederated Int'l Conf. (OTM '08), Part II*, vol. 5332, Nov. 2008, pp. 1304–1318.

[3] C. Bizer, R. Meusel, and A. Primpeli, "Web data commons - microdata, rdfa, json-ld, and microformat data sets," Nov. 2020. [Online]. Available: http://webdatacommons.org/structureddata/#results-2020-1

[4] D. Brickley, M. Burgess, and N. F. Noy, "Google dataset search: Building a search engine for datasets in an open web ecosystem," in *Proc. 28th World Wide Web Conf. (WWW '19)*, May 2019, pp. 1365–1375.

[5] S. Campinas, R. Delbru, and G. Tummarello, "Efficiency and precision trade-offs in graph summary algorithms," in *Proc. 17th Int'l Database Engineering & Applications Symposium (IDEAS '13)*, Oct. 2013, pp. 38–47.

[6] S. Cebiric, F. Goasdoué, H. Kondylakis, D. Kotzinos, I. Manolescu, G. Troullinou, and M. Zneika, "Summarizing semantic graphs: a survey," *VLDB Journal*, vol. 28, no. 3, pp. 295–327, 2019.

[7] A. Chapman, E. Simperl, L. Koesten, G. Konstantinidis, L. D. Ibáñez, E. Kacprzak, and P. Groth, "Dataset search: a survey," *VLDB Journal*, vol. 29, no. 1, pp. 251–272, 2020.

[8] J. Chen, Y. Geng, Z. Chen, I. Horrocks, J. Z. Pan, and H. Chen, "Knowledge-aware zero-shot learning: Survey and perspective," in *Proc. 30th Int'l Joint Conf. Artificial Intelligence (IJCAI '21)*, 2021.

[9] J. Chen, F. Lécué, Y. Geng, J. Z. Pan, and H. Chen, "Ontology-guided semantic composition for zero-shot learning," in *Proc. 17th Int'l Conf. Principles of Knowledge Representation and Reasoning (KR '20)*, 2020, pp. 850–854.

[10] J. Chen, F. Lécué, J. Z. Pan, I. Horrocks, and H. Chen, "Knowledge-based transfer learning explanation," in *Proc. 16th Int'l Conf. Principles of Knowledge Representation and Reasoning (KR '20)*, 2018, pp. 349–358.

[11] J. Chen, X. Wang, G. Cheng, E. Kharlamov, and Y. Qu, "Towards more usable dataset search: From query characterization to snippet generation," in *Proc. 28th ACM Int'l Conf. Information and Knowledge Management (CIKM '19)*, Nov. 2019, pp. 2445–2448.

[12] Z. Chen, H. Jia, J. Heflin, and B. D. Davison, "Leveraging schema labels to enhance dataset search," *CoRR*, vol. abs/2001.10112, 2020.

[13] G. Cheng, C. Jin, W. Ding, D. Xu, and Y. Qu, "Generating illustrative snippets for open data on the web," in *Proc. 10th ACM Int'l Conf. Web Search and Data Mining (WSDM '17)*, Feb. 2017, pp. 151–159.

[14] G. Cheng, C. Jin, and Y. Qu, "HIEDS: A generic and efficient approach to hierarchical dataset summarization," in *Proc. 25th Int'l Joint Conf. Artificial Intelligence (IJCAI '16)*, Jul. 2016, pp. 3705–3711.

[15] G. Cheng, T. Tran, and Y. Qu, "RELIN: relatedness and informativeness-based centrality for entity summarization," in *Proc. 10th Int'l Semantic Web Conf. (ISWC '11), Part I*, Oct. 2011, pp. 114–129.

[16] G. Cheng, D. Xu, and Y. Qu, "C3D+P: A summarization method for interactive entity resolution," *Journal of Web Semantics*, vol. 35, pp. 203–213, 2015.

[17] G. Cheng, D. Xu, and Y. Qu, "Summarizing entity descriptions for effective and efficient human-centered entity linking," in *Proc. 24th Int'l Conf. World Wide Web (WWW '15)*, May 2015, pp. 184–194.

[18] R. Cyganiak, D. Wood, and M. Lanthaler, "RDF 1.1 concepts and abstract syntax," Feb. 2014. [Online]. Available: https://www.w3.org/TR/rdf11-concepts/

[19] B. Ding, J. X. Yu, S. Wang, L. Qin, X. Zhang, and X. Lin, "Finding top-k min-cost connected trees in databases," in *Proc. 23rd Int'l Conf. Data Engineering (ICDE '07)*, Apr. 2007, pp. 836–845.

[20] J. Dolby, A. Fokoue, A. Kalyanpur, A. Kershenbaum, E. Schonberg, K. Srinivas, and L. Ma, "Scalable semantic retrieval through summarization and refinement," in *Proc. 22nd AAAI Conf. Artificial Intelligence (AAAI '07)*, Jul. 2007, pp. 299–304.

[21] G. Feigenblat, H. Roitman, O. Boni, and D. Konopnicki, "Unsupervised query-focused multi-document summarization using the cross entropy method," in *Proc. 40th Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '17)*, Aug. 2017, pp. 961–964.

[22] A. Fokoue, A. Kershenbaum, L. Ma, E. Schonberg, and K. Srinivas, "The summary abox: Cutting ontologies down to size," in *Proc. 5th Int'l Semantic Web Conf. (ISWC '06)*, Nov. 2006, pp. 343–356.

[23] A. Fokoue, F. Meneguzzi, M. Sensoy, and J. Z. Pan, "Querying linked ontological data through distributed summarization," in *Proc. 26th AAAI Conf. Artificial Intelligence (AAAI '12)*, Jul. 2012.

[24] M. Gambhir and V. Gupta, "Recent automatic text summarization techniques: a survey," *Artificial Intelligence Review*, vol. 47, no. 1, pp. 1–66, 2017.

[25] W. Ge, J. Chen, W. Hu, and Y. Qu, "Object link structure in the semantic web," in *Proc. 7th Extended Semantic Web Conf. (ESWC '10), Part II*, 2010, pp. 257–271.

[26] W. Ge, G. Cheng, H. Li, and Y. Qu, "Incorporating compactness to generate term-association view snippets for ontology search," *Information Processing and Management*, vol. 49, no. 2, pp. 513–528, 2013.

[27] K. Gunaratna, K. Thirunarayan, and A. P. Sheth, "FACES: diversity-aware entity summarization using incremental hierarchical conceptual clustering," in *Proc. 29th AAAI Conf. Artificial Intelligence (AAAI '15)*, Jan. 2015, pp. 116–122.

[28] K. Gunaratna, A. H. Yazdavar, K. Thirunarayan, A. P. Sheth, and G. Cheng, "Relatedness-based multi-entity summarization," in *Proc. 26th Int'l Joint Conf. Artificial Intelligence (IJCAI '17)*, Aug. 2017, pp. 1060–1066.

[29] L. Heling and M. Acosta, "Estimating characteristic sets for RDF dataset profiles based on sampling," in *Proc. 17th Extended Semant. Web Conf. (ESWC '20)*, 2020, pp. 157–175.

[30] Z. Huang, S. Li, G. Cheng, E. Kharlamov, and Y. Qu, "Micron: Making sense of news via relationship subgraphs," in *Proc. 28th ACM Int'l Conf. Information and Knowledge Management (CIKM '19)*, Nov. 2019, pp. 2901–2904.

[31] V. Jindal, S. Bawa, and S. Batra, "A review of ranking approaches for semantic search on web," *Information Processing and Management*, vol. 50, no. 2, pp. 416–425, 2014.

[32] E. Kacprzak, L. Koesten, J. Tennison, and E. Simperl, "Characterising dataset search queries," in *Proc. 27th World Wide Web Conf. (WWW '18)*, Apr. 2018, pp. 1485–1488.

[33] L. M. Koesten, E. Kacprzak, J. F. A. Tennison, and E. Simperl, "The trials and tribulations of working with structured data - a study on information seeking behaviour," in *Proc. 2017 CHI Conf. Hum. Factors Comput. Syst. (CHI '17)*, 2017, pp. 1277–1289.

[34] C. C. Kuhlthau, "Inside the search process: Information seeking from the user's perspective," *Journal of the American Society for Information Science*, vol. 42, no. 5, pp. 361–371, 1991.

[35] F. Lécué, J. Chen, J. Z. Pan, and H. Chen, "Augmenting transfer learning with semantic reasoning," in *Proc. 28th Int'l Joint Conf. Artificial Intelligence (IJCAI '20)*, 2019, pp. 1779–1785.

[36] J. Li, G. Cheng, Q. Liu, W. Zhang, E. Kharlamov, K. Gunaratna, and H. Chen, "Neural entity summarization with joint encoding and weak supervision," in *Proc. 29th Int'l Joint Conf. Artificial Intelligence (IJCAI '20)*, Jul. 2020, pp. 1644–1650.

[37] R. Li, L. Qin, J. X. Yu, and R. Mao, "Efficient and progressive group Steiner tree search," in *Proc. 2016 Int'l Conf. Management of Data (SIGMOD '16)*, Jun.–Jul. 2016, pp. 91–106.

[38] S. Li, Z. Huang, G. Cheng, E. Kharlamov, and K. Gunaratna, "Enriching documents with compact, representative, relevant knowledge graphs," in *Proc. 29th Int'l Joint Conf. Artificial Intelligence (IJCAI '20)*, Jul. 2020, pp. 1748–1754.

[39] D. Liu, G. Cheng, Q. Liu, and Y. Qu, "Fast and practical snippet generation for RDF datasets," *ACM Transactions on the Web*, vol. 13, no. 4, pp. 19:1–19:38, 2019.

[40] Q. Liu, G. Cheng, K. Gunaratna, and Y. Qu, "Entity summarization: State of the art and future challenges," *CoRR*, vol. abs/1910.08252, 2019.

[41] C. Lucchese, S. Orlando, and R. Perego, "A unifying framework for mining approximate top-k binary patterns," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 12, pp. 2900–2913, 2014.

[42] N. Noy, "Discovering millions of datasets on the web," Jan. 2020. [Online]. Available: https://www.blog.google/products/search/discovering-millions-datasets-web/

[43] E. Oren, R. Delbru, M. Catasta, R. Cyganiak, H. Stenzhorn, and G. Tummarello, "Sindice.com: a document-oriented lookup index for open linked data," *International Journal of Metadata, Semantics and Ontologies*, vol. 3, no. 1, pp. 37–52, 2008.

[44] J. Z. Pan, "Resource description framework," in *Handbook on Ontologies*. Springer, 2009.

[45] E. Pietriga, H. Gözükan, C. Appert, M. Destandau, S. Cebiric, F. Goasdoué, and I. Manolescu, "Browsing linked data catalogs with LODAtlas," in *Proc. 17th Int'l Semantic Web Conf. (ISWC '18), Part II*, Oct. 2018, pp. 137–153.

[46] S. Pouriyeh, M. Allahyari, Q. Liu, G. Cheng, H. R. Arabnia, M. Atzori, F. G. Mohammadi, and K. Kochut, "Ontology summarization: Graph-based methods and beyond," *International Journal of Semantic Computing*, vol. 13, no. 2, pp. 259–283, 2019.

[47] L. Rietveld, R. Hoekstra, S. Schlobach, and C. Guéret, "Structural properties as proxy for semantic relevance in RDF graph sampling," in *Proc. 13th Int'l Semantic Web Conf. (ISWC '14), Part II*, vol. 8797, Oct. 2014, pp. 81–96.

[48] H. Roitman, G. Feigenblat, D. Cohen, O. Boni, and D. Konopnicki, "Unsupervised dual-cascade learning with pseudo-feedback distillation for query-focused extractive summarization," in *Proc. The Web Conference 2020 (WWW '20)*, Apr. 2020, pp. 2577–2584.

[49] T. Safavi, C. Belth, L. Faber, D. Mottin, E. Müller, and D. Koutra, "Personalized knowledge graph summarization: From the cloud to your pocket," in *Proc. 2019 IEEE Int'l Conf. Data Mining (ICDM '19)*, Nov. 2019, pp. 528–537.

[50] Y. Shi, G. Cheng, and E. Kharlamov, "Keyword search over knowledge graphs via static and dynamic hub labelings," in *Proc. The Web Conf. (WWW '20)*, Apr. 2020, pp. 235–245.

[51] Q. Song, Y. Wu, P. Lin, X. Dong, and H. Sun, "Mining summaries for knowledge graph search," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 10, pp. 1887–1900, 2018.

[52] Y. Tian, R. A. Hankins, and J. M. Patel, "Efficient aggregation for graph summarization," in *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '08)*, Jun. 2008, pp. 567–580.

[53] X. Wang, J. Chen, S. Li, G. Cheng, J. Z. Pan, E. Kharlamov, and Y. Qu, "A framework for evaluating snippet generation for dataset search," in *Proc. 18th Int'l Semantic Web Conf. (ISWC '19), Part I*, Oct. 2019, pp. 680–697.

[54] X. Wang, G. Cheng, and E. Kharlamov, "Towards multi-facet snippets for dataset search," in *Joint Proc. 6th Int'l Workshop Dataset PROFILing and Search (PROFILES '19) & 1st Workshop Semantic Explainability (SemEx '19)*, Oct. 2019, pp. 1–6.

[55] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, J. Bouwman, A. J. Brookes, T. Clark, M. Crosas, I. Dillo, O. Dumon, S. Edmunds, C. T. Evelo, R. Finkers, A. Gonzalez-Beltran, A. J. G. Gray, P. Groth, C. Goble, J. S. Grethe, J. Heringa, P. A. C. 't Hoen, R. Hooft, T. Kuhn, R. Kok, J. Kok, S. J. Lusher, M. E. Martone, A. Mons, A. L. Packer, B. Persson, P. Rocca-Serra, M. Roos, R. van Schaik, S.-A. Sansone, E. Schultes, T. Sengstag, T. Slater, G. Strawn, M. A. Swertz, M. Thompson, J. van der Lei, E. van Mulligen, J. Velterop, A. Waagmeester, P. Wittenburg, K. Wolstencroft, J. Zhao, and B. Mons, "The FAIR guiding principles for scientific data management and stewardship," *Scientific Data*, vol. 3, p. 160018, 2016.

[56] M. Zneika, C. Lucchese, D. Vodislav, and D. Kotzinos, "RDF graph summarization based on approximate patterns," in *Proc. 10th Int'l Workshop Information Search, Integration, and Personalization (ISIP '15)*, vol. 622, Oct. 2015, pp. 69–87.

[57] M. Zneika, D. Vodislav, and D. Kotzinos, "Quality metrics for RDF graph summarization," *Semantic Web*, vol. 10, no. 3, pp. 555–584, 2019.

Xiaxia Wang is an M.S. student at the Department of Computer Science and Technology, Nanjing University. She received the B.S. degree in Information and Computing Science from Nanjing University of Aeronautics and Astronautics in 2019. Her current research interests include data summarization and semantic search. She has published papers in ISWC and CIKM.

Gong Cheng is an Associate Professor at the Department of Computer Science and Technology, Nanjing University. He received the Ph.D. degree in computer software and theory from Southeast University in 2010. His research interests include semantic search, data summarization, and question answering. He has published more than 50 papers in major venues in these areas such as *TKDE*, *TWEB*, WWW, AAAI, IJCAI, and ISWC. He is a member of the IEEE.

Jeff Z. Pan is a Reader in Knowledge Graph in the School of Informatics at the University of Edinburgh. He received his Ph.D. degree in computer science in the University of Manchester in 2004. His research interests include knowledge based representation, reasoning and learning, as well as their applications in e.g. search and question answering. He is a chair of the Knowledge Graph group at the Alan Turing Institute. He is a Programme Chair of ISWC2020 and an Associate Editor of *JWS*.

Evgeny Kharlamov is an Associate Professor at the University of Oslo and a Research Scientist at the Bosch Centre for Artificial Intelligence. He received his Ph.D. degree in computer science at the Free University of Bozen-Bolzano in 2011. His research interests include artificial intelligence and semantic technologies with applications in industry 4.0. He has published more than 70 papers in major venues in these areas such as *TKDE*, *JWS*, PVLDB, AAAI, IJCAI, CIKM and ISWC.

Yuzhong Qu is a Professor at the Department of Computer Science and Technology, Nanjing University. He got the Ph.D. in Computer Software from Nanjing University in 1995, M.S. in Mathematics from Fudan University in 1988, and B.S. in Mathematics from Fudan University in 1985. His research interests include Semantic Web, Question Answering, and novel software technology for the Web. He has published more than 80 papers in major venues in these areas such as WWW, ISWC, and *JWS*.