Content-Based Open Knowledge Graph Search: A Preliminary Study with OpenKG.CN

Xiaxia Wang, Tengteng Lin, Weiqing Luo, Gong Cheng, and Yuzhong Qu

State Key Laboratory for Novel Software Technology, Nanjing University, China
{xxwang,tengtenglin,wqluo}@smail.nju.edu.cn, {gcheng,yzqu}@nju.edu.cn

Abstract. Users rely on open data portals and search engines to find open knowledge graphs (KGs). However, existing systems only provide metadata-based KG search but ignore the contents of KGs, i.e., triples. In this paper, we present one of the first content-based search engines for open KGs. Our system CKGSE supports keyword-based KG search, KG snippet generation, KG profiling and browsing, all computed over KGs' (large) contents rather than their (small) metadata. We implement a prototype with Chinese KGs crawled from OpenKG.CN and we report some preliminary results about the practicability of such a system.

Keywords: Knowledge Graph · Search Engine · Snippet · Profiling.

1 Introduction

Open data, especially knowledge graphs (KGs), plays an important role in scientific research and application development. These days, increasingly more research efforts for constructing reusable KGs bring about a large number of KG resources available on the Web, which motivates the development of data sharing platforms. Open data portals such as European Data Portal have made a promising start. Users are allowed to freely publicize their own KGs, or search and download published KGs from others. Recent research efforts have been paid to assist users to conveniently find the KG they want. Thus various systems have been developed, ranging from general search engines such as Google Dataset Search, to KG-centric systems like LODAtlas [14]. OpenKG.CN is a popular platform for Chinese open KGs, providing a keyword search service.

Limitations of Existing Work. The above systems provide search services based on *metadata*, i.e., meta-level data descriptions attached to a KG, such as authorship and provenance. This leads to two weaknesses. **(W1)** They cannot handle queries with keywords referring to the *content* (i.e., triples) of the target KG, such as a class, property, or entity. Queries of this kind are common in practice. Indeed, over 60% of KG queries contain keywords that refer to KG content [4]. **(W2)** Metadata cannot provide close-up views of the underlying KG content. Its utility for relevance judgment in search is limited [16].

Our Work. To address the above two limitations, *content-based KG search* is needed but the *practicability* of such a new paradigm is unknown. In this paper, we present CKGSE, short for **C**hinese **KG** **S**earch **E**ngine, as our preliminary effort to build a content-based search system for open KGs. CKGSE has four components: *KG crawling and storage* for managing KGs and their metadata, *content-based keyword search* for parsing queries and retrieving relevant KGs based on their contents, *content-based snippet generation* for extracting a sub-KG from a KG to justify its query relevance, and *content profiling and browsing* for providing detailed information about a KG including abstractive and extractive summaries and a way to explore its content. To evaluate the practicability of CKGSE, we implement a prototype[1] based on Chinese KGs collected from OpenKG.CN. Our contributions are summarized as follows.

- We develop one of the first content-based search engines for open KGs.
- We report experimental results about the practicability of such a system.

Outline. Section 2 discusses related work. Section 3 and Section 4 describe an overview of CKGSE and its detailed implementation, respectively. Section 5 presents experimental results. Section 6 concludes the paper with future work.

2 Related Work

2.1 Open KG Portals and Search Engines

Today, hundreds of open data portals are available [13]. They manage collected resources with metadata using a vocabulary such as DCAT.[2] Google Dataset Search aims at navigating users to the original page of each data resource. It only indexes the metadata of each resource without considering its content. Current search engines for KGs are also based on metadata, e.g., LODAtlas [14].

To overcome the limitations of metadata-based systems, our CKGSE takes KG content into consideration. To support keyword search over KG content, complementary to metadata index, CKGSE also indexes the content of each KG. To facilitate relevance judgment of search results, CKGSE generates a snippet for each KG extracted from its content. To provide a close-up view for each KG, apart from its metadata, CKGSE offers content-based summaries and an exploration mechanism for easier comprehension.

2.2 KG Profiling

KG profiling is to interpret a KG according to various features [7, 3, 9]. In [7], by reviewing the literature over the past two decades, profiling features are divided into seven categories, most of which are metadata-oriented such as Provenance and Licensing. The Statistical category includes numbers and distributions of

[1] http://ws.nju.edu.cn/CKGSE
[2] https://www.w3.org/TR/vocab-dcat-2/

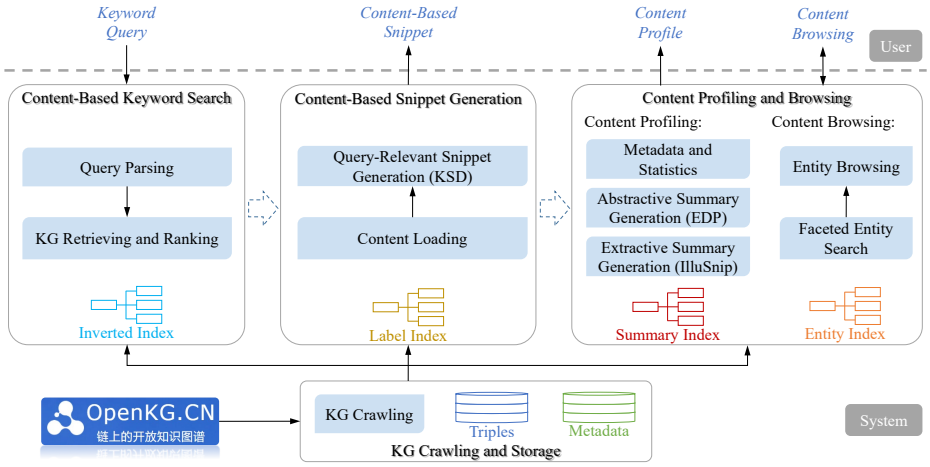

Fig. 1: Overview of CKGSE.

data elements such as the number of classes and properties instantiated in a KG [1]. The General category includes methods for choosing representative data elements, such as structural summaries [2, 15, 8, 6] and pattern mining [20, 21].

Benefited from these fruitful research efforts, our CKGSE incorporates several profiling techniques. In addition to metadata and statistics, CKGSE mines frequent entity description patterns (EDPs) in a KG as its abstractive summary [19, 18], and illustrates its content with an extractive summary [5, 10].

3 System Overview

An overview of CKGSE is presented in Fig. 1, including four main components. KG Crawling and Storage crawls, parses, and stores KGs and their metadata. Content-Based Keyword Search parses keyword queries and retrieves and ranks KGs. For each top-ranked KG, Content-Based Snippet Generation extracts a query-relevant sub-KG as a snippet presented in the search results page. For each KG selected by the user, Content Profiling and Browsing presents its profiled detailed information and supports exploring its content.

KG Crawling and Storage. As an offline process, KG Crawling retrieves all available metadata from OpenKG.CN through its CKAN API. Following the links in metadata, the dump files of all KGs are downloaded, parsed into triples, and stored in a local database. Four indexes are then built to support efficient online computation in downstream components, which will be detailed in Section 4.

Content-Based Keyword Search. Given an input (Chinese) keyword query, it is parsed by segmenting into words and removing stop words. The parsed query is executed over an inverted index to retrieve top-ranked KGs according to a relevance score function involving multiple fields of metadata and content.

Content-Based Snippet Generation. For each top-ranked KG, its content is loaded into memory from which a query-relevant snippet is online generated. The generation method, KSD [17], considers content representativeness and query relevance, and is supported by a label index. The output is an optimal snippet containing k top-ranked triples where k is a pre-defined size limit. It is presented as a node-link diagram in the search results page.

Content Profiling and Browsing. For each selected KG, its offline indexed profile is presented, including its metadata, statistics, and two content summaries. The abstractive summary contains the most frequent entity description patterns (EDPs) [19, 18] in the KG, and the extractive summary generated by IlluSnip [5, 10] contains an optimal sub-KG of k' triples in terms of content representativeness where k' is a pre-defined size limit. All entities in the KG can be explored in a faceted manner supported by an entity index, i.e., by filtering entities by their classes and properties. All triples describing a filtered entity can be browsed.

4 System Implementation

In this section we detail the implementation of each component of CKGSE.

4.1 KG Crawling and Storage

This component crawls and stores metadata and KGs from OpenKG.CN.

KG Crawling. Through the CKAN API, the metadata of all available resources are retrieved from OpneKG.CN, including KGs and non-KG resources which are not our focus. Among these resources, 40 are identified as KGs in formats such as RDF/XML, Turtle, and JSON-LD. Following the links in metadata, KG Crawling first downloads KG dump files, and then parses them into triples using Apache Jena 3.8.0. Metadata and triples are stored in a MySQL database.

Metadata. The metadata of all KGs is stored in a table, where each row represents a KG, and each column represents a field of metadata, such as `Title` and `Author`. Observe that incorrect/incomplete field values are common in practice since they are freely submitted by KG publishers.

Triples. The triples of each KG are stored in a table whose columns contain the subject, predicate and object of each triple. Moreover, each IRI in a KG is labeled by a human-readable textual form, including its local name, the value of its `rdfs:label` property, and the value of its associated literals. These labels will be used in downstream components.

4.2 Content-Based Keyword Search

Supported by an inverted index, this component parses the keyword query and retrieves a ranked list of relevant KGs.

Inverted Index. An inverted index is created for keyword-based KG search and has three fields: `Content`, `Title`, and `Other metadata`. For KG content which is not considered in existing systems, each triple is transformed into text by concatenating the labels of its subject, predicate, and object. The textual forms of all triples in a KG compose the field `Content`. Among metadata, the title of each KG is individually indexed in the field `Title` for its importance in KG search, while other metadata is combined into the field `Other metadata`.

Query Parsing. The keyword search component of CKGSE is developed over Apache Lucene 7.5.0, which provides only basic word segmentation modules incompetent for parsing Chinese queries. To address this problem, we incorporate the IK analyzer[3] which is an open-source tool for Chinese word segmentation.

KG Retrieving and Ranking. Receiving the parsed query from Query Parsing as a list of keywords, it is searched over the inverted index. Searches are performed using OR as the default boolean operator between keywords. A multi-field query parser provided by Apache Lucene is used to retrieve and rank the search results over the three fields. The relevance score function adopted by CKGSE, at the time of writing, is BM25 as default provided by Lucene, which could be replaced by more advanced functions for KGs in future work.

4.3 Content-Based Snippet Generation

To facilitate relevance judgement of KGs, this component outputs a snippet for each top-ranked KG to justify its query relevance and illustrate its main content. This unique feature distinguishes CKGSE from existing systems.

Label Index. For each KG, a label index maps each word to the IRIs whose textual forms contain the word. It is used to compute the query relevance of each triple, supporting query-relevant snippet generation. The query relevance of a triple is measured by the number of keywords contained by its textual form, i.e., contained by the textual form of its subject, predicate, or object.

Content Loading. Before snippet generation, for each KG in the search results, its triples are loaded into memory from the local triple store. During the loading process, the query relevance of each triple is computed with the support of the Label Index, as well as some scores measuring content representativeness such as the relative frequency of the class or property instantiated in the triple.

Query-Relevant Snippet Generation. KSD [17] formulates the computation of an optimal snippet for a KG as a weighted maximum coverage problem, where each triple is regarded as a set of elements including query keywords, classes, properties, and entities in the triple. Each element is assigned a weight of importance, e.g., relative frequency. KSD aims at selecting at most k triples maximizing the

[3] https://code.google.com/archive/p/ik-analyzer/

total weight of covered elements. By applying a greedy strategy, it selects a triple containing the largest weight of uncovered elements in each iteration until reaching the size limit k or all elements are covered. We set $k = 10$. An example snippet is shown in Fig. 6. We refer the reader to [17] for details about KSD.

4.4 Content Profiling

This component presents an offline computed profile for each selected KG including metadata, statistics, an abstractive summary, and an extractive summary.

Metadata and Statistics. For each selected KG, its metadata is loaded from the database and presented to the user as a table. Each row contains a key-value pair as shown in Fig. 7. Statistics provide a high-level overview of the selected KG, including basic counts such as the number of triples, entities, the distributions of instantiated classes and properties presented in pie charts as shown in Fig. 8, and top-ranked entities by PageRank showing the central content of the KG.

Abstractive Summary Generation. Inspired by pattern mining techniques for KG profiling, CKGSE incorporates frequent entity description pattern (EDP) [19, 18] into the KG profile page. As each entity in the KG is described by a set of schema-level elements, i.e., classes and properties. EDPs retain the common data patterns shared among entities in the KG. Thus frequent EDPs can be regarded as an abstractive summary of the KG. Each of them consists of a set of classes, forward properties, and backward properties that describe an entity. In the profile page, each EDP is presented as a node-link diagram as in Fig. 9.

Extractive Summary Generation. Complementary to EDPs providing abstractive schema-level summaries, CKGSE also provides extractive summaries to exemplify the KG content. IlluSnip [5, 10] formulates the selection of triples as a combinatorial optimization problem, aiming at computing an optimal connected sub-KG containing the most frequently instantiated classes, properties, and the most important entities with the highest PageRank scores. Such a sub-KG can be viewed as an extractive summary of the KG. A greedy algorithm is applied to generate a sub-KG containing at most k' triples. We set $k' = 20$. The result is presented as a node-link diagram on the profile page as shown in Fig. 10.

Summary Index. Offline computed abstractive and extractive summaries are stored in a summary index.

4.5 Content Browsing

Beyond the profile page, CKGSE also enables the user to interactively explore the entities in the selected KG.

Entity Index. An entity index is created to support efficient filtering of entities. It consists of two parts, mapping classes and properties to their instance entities, respectively. The index is implemented with Lucene.

Table 1: Statistics of the KGs.

Triples		Classes		Properties		Entities		Dump Files (MB)	
median	max	median	max	median	max	median	max	median	max
68,399	151,976,069	7	20,096	23	40,077	93,268	303,952,138	23	28,929

Table 2: Run-time of offline computation (hours).

Parsing	Inverted Index	Label Index	Statistics	Summary Index	Entity Index
20.27	5.10	2.52	2.97	31.38	33.21

Table 3: Disk use (MB).

Dump Files	Triple Store	Inverted Index	Label Index	Summary Index	Entity Index
54,133	35,756	1,816	9,708	165	1,129

Faceted Entity Search. As shown in Fig. 11, users are provided with a panel to choose classes and properties instantiated in the KG. The selected classes and properties are used as a filter for entities with AND as the boolean operator. Filtered entities are presented as a list. Each of them can be further browsed.

Entity Browsing. For each of the filtered entities, CKGSE retrieves all the triples describing it and presents them as a node-link diagram. Intuitively it shows the neighborhood of this entity in the original KG. By switching between entities, users are able to explore the KG according to her/his own interest.

5 Experiments

We implemented a prototype of CKGSE on an Intel Xeon E7-4820 (2GHz) with 100GB memory for JVM. Our experiments were focused on the practicability of CKGSE. We also presented a case study comparing CKGSE with the current version of OpenKG.CN to show the usefulness of the unique features of CKGSE.

5.1 Practicability Analysis

KG Crawling and Storage. Table 1 shows statistics about the 40 KGs crawled from OpenKG.CN. They vary greatly in size and schema. Some are very large.

As shown in Table 2, parsing all the 40 KGs affordably ended within 1 day, where 10% (4/40) were large and parsed in more than 1 hour. Observe that the time for parsing the largest KG[4] reached 8 hours.

Table 3 shows the disk use. The total size of the triple store and all the indexes is smaller than that of the original dump files, showing practicability. Fig. 2 presents the disk use of each KG in ascending order. Compared with dump files, the indexes have relatively small sizes and are affordable.

[4] http://www.openkg.cn/dataset/zhishi-me-dump

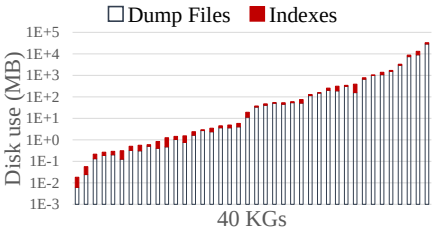

Fig. 2: Disk use of each KG.

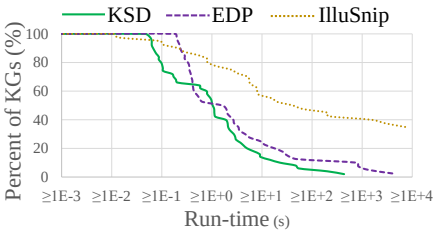

Fig. 3: Cumulative distribution of run-time.

Content-Based Keyword Search. CKGSE spent 5 hours to build an inverted index for all the 40 KGs to support keyword search over both metadata and KG contents (Table 2). Note that we did not build in parallel which would otherwise be much faster. The index only takes 1.8GB, being small and affordable (Table 3).

Content-Based Snippet Generation. CKGSE spent 2.5 hours to construct a label index to support query-relevant snippet generation (Table 2). About half of the time, i.e., 1.1 hours, was spent on the largest KG. The index takes 9.7GB (Table 3) where about half is for the largest KG.

To evaluate the performance of online snippet generation using KSD, we created 10 keyword queries containing 1-5 keywords and retrieved top-5 KGs with each query. We recorded the run-time of generating a snippet for each of these 50 KGs. Fig. 3 shows the cumulative distribution of run-rime over all these KGs. The median run-time is only 1 second, while for 12% (6/50) the run-time exceeded 10 seconds. Further optimization of KSD is needed for large KGs.

Content Profiling. CKGSE spent 3 hours to prepare statistics (Table 2), including about 2 hours for computing PageRank to identify central entities.

The summary index only uses 165MB (Table 3) but its computation spent 31 hours (Table 2). Most time was for generating extractive summaries using IlluSnip. We used an anytime version of IlluSnip [10] and we allowed 2 hours for generating a snippet. If needed, one could set a smaller time bound to trade snippet quality for generation time. In our experiments, the median run-time of IlluSnip was 31 seconds. By comparison, generating abstractive summaries (i.e., EDP) was much faster, being comparable with KSD as shown in Fig. 3.

Content Browsing. CKGSE spent 33 hours to create an entity index to support faceted entity search (Table 2), although the index was as small as 1.1GB upon completion (Table 3). Observe that there is much room for improving the performance of our trivial implementation of this index, e.g., using better index structures and/or faster algorithms.

(a) OpenKG.CN (b) CKGSE

Fig. 4: Search results pages for the query “哈利波特 人物关系”.

(a) OpenKG.CN (b) CKGSE

Fig. 5: Search results pages for the query “格兰芬多 人物关系”.

5.2 Case Study

We compare CKGSE with the current version of OpenKG.CN by a case study.

Keyword Search. Fig. 4 shows the search results pages returned by OpenKG.CN and CKGSE for the query “哈利波特 人物关系”. Both systems successfully find the target KG as both keywords in the query match the metadata of this KG.

However, for the query “格兰芬多 人物关系” where “格兰芬多” refers to an entity in the target KG, only CKGSE finds this KG as shown in Fig. 5, thanks to its content-based keyword search whose inverted index covers both metadata and KG content, distinguishing CKGSE from existing systems.

Snippet Generation. In the search results pages, while OpenKG.CN and other existing systems only show some metadata about each top-ranked KG, CKGSE further computes and presents an extracted sub-KG, as shown in Fig. 6. The generation of this snippet is biased toward the keyword query, e.g., containing the

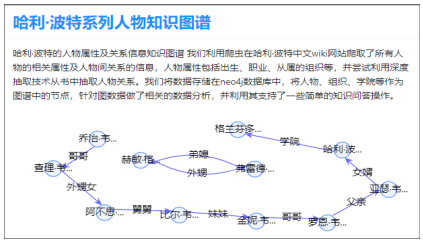

Fig. 6: Snippet (KSD).

Fig. 7: Metadata.

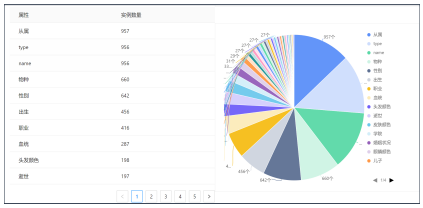

Fig. 8: Statistics.

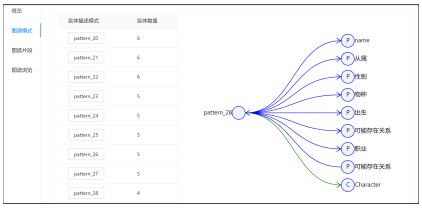

Fig. 9: Abstractive summary (EDP).

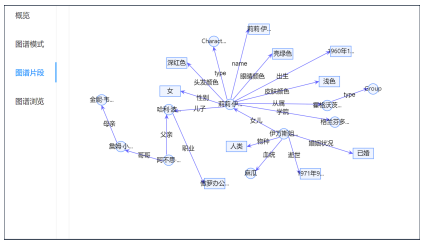

Fig. 10: Extractive summary (IlluSnip).

Fig. 11: Content browsing.

"格兰芬多" entity mentioned in the query. Therefore, it can help the user quickly judge the relevance of the underlying KG to the query even before browsing its content which could be a time-consuming process.

Profiling and Browsing. When a KG is selected, both OpenKG.CN and CKGSE show its metadata in a profile page, as shown in Fig. 7. CKGSE further presents some statistics about the content of the KG, such as the distribution of all properties instantiated in the KG visualized as a pie chart shown in Fig. 8. Such statistics provide the user with a brief overview of the KG content.

Moreover, CKGSE provides two content summaries of the KG. The abstractive summary in Fig. 9 presents the most frequent EDPs in the KG to show how entities in the KG are described, i.e., by which combinations of classes and properties. The extractive summary in Fig. 10 presents an extracted sub-KG which is different from the snippet in the search results page. The sub-KG here

is query-independent but illustrates the most frequent classes and properties in the KG with a few concrete entities and triples. Compared with metadata and statistics, our content summaries provide a distinguishing closer-up view of the KG content, thus assisting the user in comprehending the KG and further judging its relevance before downloading it.

Last but not least, in Fig. 11 CKGSE allows the user to interactively browse entities in the KG. The user can select classes and properties to filter entities. For each filtered entity, all its triples are visualized as a node-link diagram. With this simple yet effective browsing interface, for many users they do not need any other tools for KG browsing but can easily investigate the KG content.

6 Conclusion

We presented CKGSE, one of the first content-based search engines for open KGs. By incorporating triples into the inverted index, CKGSE can handle queries referring to the content of the target KG. Apart from metadata and statistics, CKGSE provides content snippets, summaries, and browsing capabilities to comprehensively assist users in relevance judgement. We implemented a prototype with KGs crawled from OpenKG.CN, and our preliminary experimental results demonstrated the practicability of such a new paradigm for KG search.

Our experiments also showed some shortcomings of CKGSE which we will address in future work. We will particularly focus on improving the efficiency of processing large KGs, and will improve the efficiency of browsing large KGs by using entity summarization techniques [12, 11]. We also plan to conduct a user study to assess the usefulness and usability of our system.

Acknowledgements

This work was supported by the NSFC (62072224).

References

1. Auer, S., Demter, J., Martin, M., Lehmann, J.: LODStats - an extensible framework for high-performance dataset analytics. In: EKAW 2012. pp. 353–362 (2012). https://doi.org/10.1007/978-3-642-33876-2_31
2. Cebiric, S., Goasdoué, F., Kondylakis, H., Kotzinos, D., Manolescu, I., Troullinou, G., Zneika, M.: Summarizing semantic graphs: a survey. VLDB J. **28**(3), 295–327 (2019). https://doi.org/10.1007/s00778-018-0528-3
3. Chapman, A., Simperl, E., Koesten, L., Konstantinidis, G., Ibáñez, L., Kacprzak, E., Groth, P.: Dataset search: a survey. VLDB J. **29**(1), 251–272 (2020). https://doi.org/10.1007/s00778-019-00564-x
4. Chen, J., Wang, X., Cheng, G., Kharlamov, E., Qu, Y.: Towards more usable dataset search: From query characterization to snippet generation. In: CIKM 2019. pp. 2445–2448 (2019). https://doi.org/10.1145/3357384.3358096

5. Cheng, G., Jin, C., Ding, W., Xu, D., Qu, Y.: Generating illustrative snippets for open data on the web. In: WSDM 2017. pp. 151–159 (2017). https://doi.org/10.1145/3018661.3018670
6. Cheng, G., Jin, C., Qu, Y.: HIEDS: A generic and efficient approach to hierarchical dataset summarization. In: IJCAI 2016. pp. 3705–3711 (2016)
7. Ellefi, M.B., Bellahsene, Z., Breslin, J.G., Demidova, E., Dietze, S., Szymanski, J., Todorov, K.: RDF dataset profiling - a survey of features, methods, vocabularies and applications. Semant. Web **9**(5), 677–705 (2018). https://doi.org/10.3233/SW-180294
8. Khatchadourian, S., Consens, M.P.: ExpLOD: summary-based exploration of interlinking and RDF usage in the linked open data cloud. In: ESWC 2010, Part II. pp. 272–287 (2010). https://doi.org/10.1007/978-3-642-13489-0_19
9. Koesten, L., Simperl, E., Blount, T., Kacprzak, E., Tennison, J.: Everything you always wanted to know about a dataset: Studies in data summarisation. Int. J. Hum. Comput. Stud. **135** (2020). https://doi.org/10.1016/j.ijhcs.2019.10.004
10. Liu, D., Cheng, G., Liu, Q., Qu, Y.: Fast and practical snippet generation for RDF datasets. ACM Trans. Web **13**(4), 19:1–19:38 (2019). https://doi.org/10.1145/3365575
11. Liu, Q., Chen, Y., Cheng, G., Kharlamov, E., Li, J., Qu, Y.: Entity summarization with user feedback. In: ESWC 2020. pp. 376–392 (2020). https://doi.org/10.1007/978-3-030-49461-2_22
12. Liu, Q., Cheng, G., Gunaratna, K., Qu, Y.: Entity summarization: State of the art and future challenges. J. Web Semant. **69**, 100647 (2021). https://doi.org/10.1016/j.websem.2021.100647
13. Neumaier, S., Umbrich, J., Polleres, A.: Automated quality assessment of metadata across open data portals. ACM J. Data Inf. Qual. **8**(1), 2:1–2:29 (2016). https://doi.org/10.1145/2964909
14. Pietriga, E., Gözükan, H., Appert, C., Destandau, M., Cebiric, S., Goasdoué, F., Manolescu, I.: Browsing linked data catalogs with LODAtlas. In: ISWC 2018, Part II. pp. 137–153 (2018). https://doi.org/10.1007/978-3-030-00668-6_9
15. 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). https://doi.org/10.1109/TKDE.2018.2807442
16. Wang, X., Chen, J., Li, S., Cheng, G., Pan, J.Z., Kharlamov, E., Qu, Y.: A framework for evaluating snippet generation for dataset search. In: ISWC 2019, Part I. pp. 680–697 (2019). https://doi.org/10.1007/978-3-030-30793-6_39
17. Wang, X., Cheng, G., Kharlamov, E.: Towards multi-facet snippets for dataset search. In: PROFLILES & SemEx 2019. pp. 1–6 (2019)
18. Wang, X., Cheng, G., Lin, T., Xu, J., Pan, J.Z., Kharlamov, E., Qu, Y.: PCSG: pattern-coverage snippet generation for RDF datasets. In: ISWC 2021 (2021)
19. Wang, X., Cheng, G., Pan, J.Z., Kharlamov, E., Qu, Y.: BANDAR: benchmarking snippet generation algorithms for (RDF) dataset search. IEEE Trans. Knowl. Data Eng. (2021)
20. Zneika, M., Lucchese, C., Vodislav, D., Kotzinos, D.: RDF graph summarization based on approximate patterns. In: ISIP 2015. vol. 622, pp. 69–87 (2015). https://doi.org/10.1007/978-3-319-43862-7_4
21. Zneika, M., Lucchese, C., Vodislav, D., Kotzinos, D.: Summarizing linked data RDF graphs using approximate graph pattern mining. In: EDBT 2016. pp. 684–685 (2016). https://doi.org/10.5441/002/edbt.2016.86