Virtual Document-based Methods for Keyword Search on RDF Graphs

Dennis Dosso
Department of Information Engineering, University of Padua
Padua, Italy
dosso@dei.unipd.it

Gianmaria Silvello
Department of Information Engineering, University of Padua
Padua, Italy
silvello@dei.unipd.it

ABSTRACT

In recent years, RDF datasets emerged as the *de-facto* standard for publishing data on the web. SPARQL, the structured query language to interrogate RDF dataset, is however hard to use for non-expert users due to its syntax. Keyword Search, on the other hand, is an intuitive query paradigm to which users are today accustomed to. In this paper, we discuss the recent research about Keyword Search on RDF datasets with virtual document-based approaches and the future directions in the creation of virtual documents in order to improve the quality of the retrieval.

CCS CONCEPTS

• **Information systems → Retrieval models and ranking**; *Information retrieval*; *Document representation*; *Evaluation of retrieval results*;

KEYWORDS

RDF datasets, keyword search, virtual documents

1 INTRODUCTION AND RELATED WORKS

In recent years The *Resource Description Framework* (RDF) has become the *de facto* standard for the Linked Data paradigm and the publication of information in the Web of Data. An RDF dataset is a set of triples composed by subject, predicate, and object. This set can also be seen as a directed labeled graph. Every node is labeled with an IRI or a string called Literal (only when object) while the edges are labeled with an IRI. RDF graphs enable flexible manipulation of data, their enrichment, their discovery and reuse across different applications. RDF datasets on the Web today contain typically thousands of millions of edges [9].

To interrogate these databases we use the structured language SPARQL. This language is complex due to its syntax and the necessity to know the structure of the graph to build the query. This is a hindrance for non-expert users like doctors or other specialists that do not have the time or the will to learn the language. In the following, we always consider a particular type of SPARQL queries: the `construct` query type, which returns a subgraph.

Keyword Search is a simpler paradigm that can enable users to easily access data, overcoming SPARQL complexity. Keyword Search is a best effort approach based on bag of words query. The result is a ranking of potential answers ordered following their relevance to the user query. Since we are working with `construct` SPARQL query, the output of a keyword query is a ranking of answer subgraphs.

Keyword Search has been thoroughly studied in the contest of structured datasets such as relational databases and Knowledge Bases. Good reviews about this topic are [1] and [10]. However, as pointed out in [3], no prototype has led to a transition from proof-of-concept implementations into fully deployed systems.

In this work, we discuss a particular subclass of keyword search systems: the one based on the *virtual document-based approach*. The *associated virtual document* of an RDF graph is the textual bag of words derived from the extraction of words from the resources composing the graph. In [4] we studied already existing state-of-the-art methods and developed new strategies that leverage on the virtual documents and showed how they are able to overcome the limitations of effectiveness and efficiency highlighted in [2] and [3]. In the following, we discuss the virtual document-based approach and some systems that make use of it and we highlight new possible directions for the improvement of the approach.

2 THE VIRTUAL DOCUMENT-BASED APPROACHES

Among the keyword search system based on virtual documents we count SLM [5], MRF-KS [7] and SUMM [6]. In [4] we propose other two systems: TSA+BM25 and TSA+VDP. As we show, the two systems overcome limitations presented by the baselines. SLM ranks its answers using an adapted language model. These are produced by concatenating triples that contain keywords. MRF-KS creates answer graphs in the form of trees whose leaves are nodes containing keywords and uses a Markov Random Field (MRF) function [8] for the ranking. SUMM also creates answer trees with a particular focus on the efficiency, using homomorphism among graphs and indexes to speed up the computation on-line. TSA+BM25 creates a subset of subgraphs extracted from the main database and builds one virtual document from each of them. It indexes these documents and performs the ranking with the BM25 ranking model. The final ranking of graphs is based on the ranking produced over the corresponding virtual documents by BM25. TSA+VDP improves the ranking of TSA+BM25. In particular, firstly it applies a pruning heuristic. It proceeds inward, removing all the triples that do not contain at least one keyword, starting from the nodes with highest distance from the source node. Then the system applies a second Markov Random Field ranking function to the new collection of graphs. This function presents two factors, which take into accounts unigrams

IIR 2019, September 16–18, 2019, Padova, Italy

Table 1: Performances of the different algorithms. † indicates the systems in the top performing group with $\alpha < 0.01$. The best system is in bold.

Dataset	Systems	tb-DCG	time (sec)	memory (MB)
LinkedMDB 1M	TSA+BM25	0.201±0.02	**39.64±01.90**†	13.19±0.41
	TSA+VDP	**0.490±0.04**†	318.78±21.60	21.06±1.18
	SLM	0.011±0.00	39.90±08.69†	**0.82±0.22**
	MRF-KS	0.400±0.03†	285.22±30.10	0.99±0.09
	SUMM	0.106±0.01	429.52±37.17	20.54±1.20
LUBM 1M	TSA+BM25	**0.284±0.06**†	**35.28±20.36**†	18.59±9.67
	TSA+VDP	0.243±0.05†	406.64±90.74	19.44±9.69
	SLM	0.048±0.00†	61.14±28.84†	**1.71±0.94**
	MRF-KS	0.090±0.03†	304.86±61.54†	2.00±0.40
	SUMM	0.024±0.01	526.14±65.67	11.11±1.41
LinkedMDB 7M	TSA+BM25	0.171±0.01	**87.52±12.86**†	t **23.24±00.55**
	TSA+VDP	**0.429±0.04**†	425.30±42.26	45.88±03.22
	SUMM	0.049±0.01	741.16±25.78	37.03±00.76
LUBM 10M	TSA+BM25	**0.281±0.07**†	**29.57±8.27**†	**26.42±14.72**
	TSA+VDP	0.234±0.06†	37.42±10.14†	26.71±14.72
	SUMM	0.053±0.00	794.29±79.99	10.93±01.60

and bigrams, and also weights the distance of a keyword from the center of the graph. In this way TSA+VDP also takes into account the graph structure of the answer.

Table 1 reports some of our results on four databases that we used: LinkedMDB, of circa 7 millions of triples, and a reduced version called here LinkedMDB 1M of one million triples and two versions of LUBM of 1M and 10M of triples respectively. For LinkedMDB the results are computed on average over 50 topics we created by hand. For LUBM we used the 14 queries available on the website of the database. For every topic, we create one SPARQL query that enables us to extract an RDF subgraph. This subgraph works as a Ground Truth (GT). Every system should return answer graphs that are as close as possible to the GT. Every triple in the GT is considered as a *relevant triple*. tb-DCG is a new evaluation metric defined by us which rewards the number of relevant triples in the answer, its position in the ranking and the quantity of noise (non-relevant triples) in the graph.

Only three systems, TSA+BM25, TSA+VDP, and SUMM are able to scale to the bigger databases. SLM is not able to scale since it performs too many operations online, and thus the bigger the database the bigger the execution time. MRF-KS relies too much on a Dijkstra-based exploration of the graph in its offline phase, which does not allow to scale to bigger dimensions

TSA+VDP is the top performing system when we work on the real database LinkedMDB, while TSA+BM25 is the top one on LUBM 1M. TSA+BM25 obtains the lowest execution time thanks to its BM25-based strategy, while TSA+VDP performs a trade-off between time and effectiveness. Similarly, for LinkedMDB 7M TSA+VDP obtains the highest result of tb-DCG among the three systems, while TSA+BM25 is the best one in LUBM 10M.

3 FUTURE DIRECTIONS

Systems based on virtual documents are able to scale to bigger datasets thanks to the help of the virtual document nature of the subgraphs, that enables them to use adapted state of the art IR methods to improve efficiency.

However, the creation of a virtual document is still quite unsophisticated in its execution: the document is built extracting words from the IRIs and Literals of the graph. Often, these words are extracted from the last part of the *path* of an IRI. However, often there is more information contained inside an IRI that can be used to build more structured and useful virtual documents.

For example, an RDF triple that can be found in LinkedMDB is *<http://data.linkedmdb.org/resource/director/8469, http://data.linkedmdb.org/resource/movie/director_name, "Quentin Tarantino">*, which has corresponding virtual document: *"8469 director name Quentin Tarantino"*. The number "8469" is not useful for a human reader, since it is simply a serial inside the database. Taken alone, the single word does not contain any useful information. However, the subject's IRI also contains the word "resource", signaling that this is an entity, and "director", signaling that this entity is a director. While this information is somehow already contained in the IRI of the predicate of this triple, this may not always be the case. Thus, the whole content of the path of the IRI can be used to improve the information contained in the virtual documents.

A possibility is to divide a virtual document in fields, for example *metadata* and *content*. The content field contains the bag of words document, while the metadata field can contain the metadata derived from the IRIs, like, in this case, the word "director", highlighting the fact that the document is talking about a director. It is also often the case that words like "director" or "actor" in databases like LinkedMDB are quite frequent. This can be a problem for models like BM25, that can rank a non-relevant document high in the ranking simply because a single keyword is repeated many times. Using the metadata field can help the algorithm to avoid the repetition of the same word many times if it is associated with the same entity, thus creating more human-friendly documents.

Acknowledgments This work is partially supported by the Computational Data Citation (CDC-STARS) project of the University of Padua.

REFERENCES

[1] H. Bast, B. Buchhold, and H. Haussmann. 2016. Semantic Search on Text and Knowledge Bases. *Foundations and Trends in Information Retrieval* 10, 2-3 (2016), 119–271.
[2] J. Coffman and A. C. Weaver. 2010. A framework for evaluating database keyword search strategies. In *Proc. of the 19th ACM International Conference on Information and knowledge management*. ACM Press, 729–738.
[3] J. Coffman and A. C. Weaver. 2014. An Empirical Performance Evaluation of Relational Keyword Search Systems. *IEEE Transactions on Knowledge and Data Engineering* 26, 1 (2014), 30–42.
[4] D. Dosso and G. Silvello. 2019. A Scalable Virtual Document-Based Keyword Search System for RDF Datasets. In *Proceedings of the 42nd International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR*.
[5] S. Elbassuoni and R. Blanco. 2011. Keyword Search over RDF Graphs. In *Proc. of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011,*. ACM Press, New York, USA, 237–242.
[6] W. Le, F. Li, A. Kementsietsidis, and S. Duan. 2014. Scalable Keyword Search on Large RDF Data. *IEEE Trans. Knowl. Data Eng.* 26, 11 (2014), 2774–2788.
[7] Y. Mass and Y. Sagiv. 2016. Virtual Documents and Answer Priors in Keyword Search over Data Graphs. In *Proc. of the Workshops of the EDBT/ICDT 2016 Joint Conference (CEUR Workshop Proceedings)*, Vol. 1558. CEUR-WS.org.
[8] D. Metzler and W. B. Croft. 2005. A Markov random field model for term dependencies. In *SIGIR 2005: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 472–479.
[9] S. Sahu, A. Mhedhbi, S. Salihoglu, J. Lin, and M. T. Özsu. 2017. The Ubiquity of Large Graphs and Surprising Challenges of Graph Processing. *PVLDB* 11, 4 (2017), 420–431.
[10] H. Wang and C. C. Aggarwal. 2010. A Survey of Algorithms for Keyword Search on Graph Data. In *Managing and Mining Graph Data*. Springer, 249–273.