A Document-based RDF Keyword Search System: Query-by-Query Analysis

Dennis Dosso[1 [0000−0001−7307−4607]] and Gianmaria Silvello[1 [0000−0003−4970−4554]]

Department of Information Engineering, University of Padua
{dosso, silvello}@dei.unipd.it

Abstract. RDF datasets are today used more and more for a great variety of applications mainly due to their flexibility. However, accessing these data via the SPARQL query language can be cumbersome and frustrating for end-users accustomed to Web-based search engines. In this context, KS is becoming a key methodology to overcome access and search issues. In this paper, we further dig on our previous work on the state-of-the-art system for keyword search on RDF by giving more insights on the quality of answers produced and its behavior with different classes of queries.

Keywords: RDF · Keyword Search · Virtual Documents

1 Introduction

In the last decade, the Web of Data emerged as one of the principal means to access, share and re-use structured data on the Web [12]. RDF has become the *de facto* standard for the publication of information on the Web [14] because it eases the enrichment and discovery of data as well as their interoperability.

In the past years, we have seen a significant increase in the number of knowledge bases published as large RDF datasets, such as DBpedia , Wikidata , and OpenPHACTS . The use of RDF is growing also for the representation and management of enterprise data with heterogeneous and very large graphs, often containing millions of edges [14].

RDF datasets are typically queried through the SPARQL structured language [13], which is often difficult to use for non-expert users due to its syntax and the required knowledge of the underlying structure of the database and the utilized IRIs. This language is not intuitive for end-users and it does not allow the natural expression of their information needs based on keywords [17].

https://wiki.dbpedia.org/

https://www.wikidata.org/wiki/Wikidata:Main_Page

https://www.openphacts.org/

Keyword search is a simpler paradigm that enables users to express their information need through keywords [10]. Keyword-based methods have gained importance over time both in research and in industry as a means to facilitate access to structured data. The result of a keyword query is a *ranking* of potential answers, ordered depending on their relevance to the information need.

In this paper, we introduce the TSA+BM25 and TSA+VDP keyword search systems, first published in [6] and the new evaluation methodology presented in [5].

The new **contributions** of this paper are: (i) a study of the *quality* of the answers produced by state-of-the-art systems in keyword search on RDF: SLM, MRF-KS, SUMM, TSA+BM25, and TSA+VDP; (ii) a study of the behavior of the two most performing systems, MRF-KS and TSA+VDP, on different *classes* of queries. Each query is composed of a `CONSTRUCT` SPARQL query and a corresponding "equivalent" keyword query. These classes differ both in terms of syntax (the SPARQL pattern used to write the query) and in their semantics (the information need expressed by the query). We analyze how the syntax and the semantic impact the performances of the system at hand.

The paper is organized as follows: Section 2 summarizes the related works; Section 3 briefly describes our methods, the datasets, the evaluation metrics and the classes of queries we defined; in Section 4 we present the experiments; finally, Section 5 presents the conclusions.

2 Related Work

In recent years keyword search received a lot of attention in the research community, both in the contest of Relational Databases (RDB) and Graph DB. Good reviews about this topic are [2], [18] (for a focus on RDB) and [16] (for a focus on graph data).

Keyword Search systems may be categorized into three main classes: (i) *schema-based* (e.g. DISCOVER [1]), which uses the schema of the database to produce a structured query; (ii) *graph-based* (e.g. BANKS [3]), which explores the graph (the graph induced by the foreign keys, if we are in a relational database); (iii) *document-based* (first described in [10]), stemming from the graph-based, this class is based on the creation of textual documents, also called *virtual documents*, from the words contained in the graphs. It also uses state-of-the-art IR methods for the indexing and ranking of the answers. Thanks to this strategy, the systems belonging to this class showed to scale to real-world scenarios.

In this paper we compare our system TSA+VDP [6] to the following state-of-the-art keyword search systems:

1. *Structured Language Model* (SLM) [7] is among the few native RDF search systems. It starts the searching process from the triples that match at least one of the query keywords and then produces answers by seeking connected components in the graph. SLM is based on virtual documents and employs IR models (statistical language models) for retrieval. SLM returns a ranking of RDF subgraphs to the user and not virtual text documents as in [8, 15].

2. *Markov Random Fields-Keyword Search* (MRF-KS)[11] is based on a combination of graph and virtual-document-based approaches. This system has proven to be the most effective one on the shared benchmark in [4]. MRF-KS was originally thought to work on RDBs, even though in principle it can be used for any kind of graph data. We redesigned it to work on RDF and use it as a competing baseline.
3. SUMM [9] is a native RDF search system. The core idea behind this system is to produce a first partition of the whole graph through a BFS-based greedy algorithm. The partitions are edge-disjointed but are connected one to the other through nodes called portal nodes. Once given the keyword query, a backtracking algorithm similar to BANKS [3] is deployed to find connected subgraphs containing all the keywords. On these subgraphs, a different backtracking algorithm similar to BLINKS [8] is used to produce tree-shaped answer graphs whose leaves are nodes containing one keyword each. The systems use indexes and particular stopping conditions to speed-up the execution.

2.1 TSA+BM25 and TSA+VDP

To overcome the limitations of the previous state-of-the-art systems, in [6] we proposed two new keyword search systems, based on the virtual document approach, one the refinement of the other. These systems are composed of an off-line and an on-line phase:

Off-line phase The off-line phase is common to both systems and is called **Topological Semantical Aggregator** (TSA), a virtual document-based approach that produces documents by aggregating RDF triples containing close concepts. TSA mimics the "clustering hypothesis" defined in IR which dictates that triples characterized by similar concepts will cluster together.

The algorithm takes an RDF graph as input and creates subgraphs without considering the user query. The idea is to build a subgraph around a single "topic" which characterizes the graph semantics. To do so, the algorithm starts randomly from a node that presents an out-degree higher of an empirically-defined threshold. From that node, the algorithm starts a Breadth-First Search-like exploration that visits only nodes that present in and out-degree between certain thresholds, limiting the exploration inside a radius τ. All these parameters are also empirically-defined. In this way, a subgraph is extracted from the bigger database. Once the exploration stops, the algorithm marks the utilized nodes as visited and proceeds randomly to the next unvisited node with high out-degree. Nodes may be visited more than once if they present a high in-degree, and in this case, they are shared by many graphs. The rationale is that these nodes may contain information belonging to more topics.

The result of TSA is a collection of subgraphs covering the whole database. They are then converted to their corresponding virtual documents through the extraction and concatenation of the words contained in the IRIs and Literals in their nodes and edges. We then index these documents with Terrier.

On-line phase Given a keyword query, our first system, **TSA+BM25**, applies the ranking function BM25 to the collection of virtual documents, and obtains a ranking of graphs. This system then looks for overlapping graphs in the ranking produced. Graphs that present a percentage of shared triples beyond an empirically-found threshold are merged. BM25 is used a second time to produce a new ranking on these merged graphs, which is returned to the user. This system is rather quick, since it uses state-of-the-art implementations of BM25, but does not leverage the information contained in the topology of the graphs.

Our second system is called **TSA+VDP** (VDP stands for **Virtual Document Pruning**). The system then considers the result of the set-based union of these ranking produced by TSA+BM25, which is called *query graph* E. This graph is not necessarily connected.

TSA+VDP then starts a BFS from every subject node within E. This BFS is also limited to a radius τ', and produces subgraphs. Of these subgraphs, only the ones containing all the keywords are kept, producing a new collection, called the *best candidates* collection. The rationale is that these selected graphs will contain, with high probability, triples that are relevant to the user. Since the BFS exploration could also have introduced many non-relevant triples, the next step is a pruning algorithm over these candidate graphs. The pruning proceeds inward, removing all the triples that do not contain at least one keyword, starting from the nodes with the highest distance from the source node.

These pruned graphs are finally ranked using a Markov Random Field scoring function inspired by the one used by MRF-KS. The final ranking is the answer returned to the user.

3 Methods

3.1 Preliminaries

While SPARQL enables the user to obtain an *exact answer* to his/her information need, it is not always the most viable approach [10]. SPARQL is a structured language with a complex syntax, which also requires complete knowledge of the graph being queried. A knowledge that the standard user (one who is not familiar with the concepts of RDF graphs) may not possess.

Keyword Search is a different paradigm that can help this kind of user to retrieve her information without SPARQL. In this approach, queries are an array $< w_1, \cdots, w_n >$ of keywords. The output is a list of *answers*, ordered depending on their relevance to the information need expressed by the query. This ordering is defined by a ranking function. It is important to note that while SPARQL returns an exact answer (even an empty one is that is the case), Keyword Search is a *best effort* approach.

The nature of the answers provided by the keyword search system depends on the task. In this paper, we consider a user who has an information need that can be formulated with a SPARQL `CONSTRUCT` query, which he/she is not able to formulate. If that query existed, the answer graph can be considered as the

set of all and only the triples of the database that the user wants to see. We consider this set of triples as a *ground truth* (GT). The triples of the GT are called *relevant triples*.

Our task is to produce a ranking that contains as many relevant triples as possible in answer graphs ranked as highly as possible. These graphs should also not contain too many non-relevant (noisy) triples. A good system, therefore, produces a list of (preferably not-overlapping) subgraphs extracted from the database such that the subgraphs at the top of the list contain many relevant triples.

3.2 Evaluation

If a Keyword Search System returns a list of answers, we first of all need to understand when one of these graphs is *relevant*. We follow the same approach of [6], where a *signal-to-noise ratio* is defined for each answer graph.

Definition 1. *Given a topic $t_k \in T$, a ranking R_k and the ground-truth graph GT_{t_k}, we define the* Signal-to-Noise Ratio *(SNR) of $G_i \in R_k$ as*

$$SNR(G_i) = \frac{|(G_i \cap GT_{t_k}) \setminus S|}{|G_i|} \tag{1}$$

where S is the union set of all the relevant triples in $G_j \in R_k, \forall j \in [1, i[$ such that G_j is relevant.

Given this definition, we say that a graph is relevant when its SNR is bigger or equal to a threshold value $\lambda \in [0, 1]$. Note how the SNR keeps into consideration the relevant triples *already seen* in the ranking, avoiding to count them twice. We can say that a *quality answer* is a graph that has a high SNR.

We call λ the *relevance parameter*, which describes the quality required for a graph to be considered relevant. In this evaluation framework, the relevance is therefore not an absolute, fixed concept, but can be varied with λ. This parameter models the preferences of a user. A user who does not want noise in its answers is modeled with a λ close to 1. A user that allows for imperfect answers is modeled with a parameter closer to 0. In what follows we study how different values of λ change the evaluation of the output of the considered systems.

As evaluation measures we use *recall* and *tb-DCG*.

Recall is the total number of relevant triples found in the ranking produced by a system – typically considering the first 1000 results – divided by $|GT|$. The recall enables to understand the extractive power of a system, the raw quantity of relevant triples extrapolated from the database. While useful, it does not convey any information about the disposition of the triples in the ranking and little information on the quantity of noise that surrounds them in the subgraphs where they are contained.

tb-DCG [5] is a more sophisticated metric, a derivation of the classic NDCG, which takes into consideration aspects such as the *relevance* of the graphs (the number of triples of the GT contained in them), their *position* (relevant graphs

put early weight more) and their *uniqueness* (repeated relevant triples are not counted).

3.3 Experimental Setup

We consider two databases: LinkedMDB and IMDB. LinkedMDB is a native RDF database of circa 7M triples. IMDB is a relational database that we converted into an RDF version of about 116M triples.

For both databases, we created 100 topics, divided into 5 classes of 20 queries each. 50 were used for training the parameters of TSA+VDP, the other 50 for testing. Every topic is represented as a pair made of a `CONSTRUCT` SPARQL query and a keyword counterpart, both representing the same information need. The SPARQL query is used to extract the subgraph which works as GT for the keyword query.

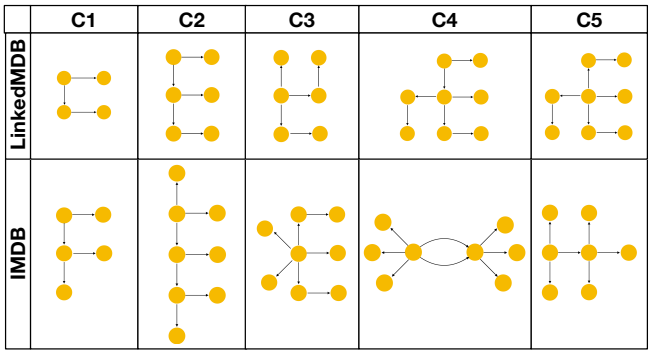

Fig. 1: The five query syntactical classes defined for LinkedMDB and IMDB.

Every class is syntactically different from the others, i.e. they differ in the shape of the pattern P, as can be seen in Figure 1. The figure presents the 10 patterns associated with each class ordered by increasing complexity. This complexity derives from the number of triples in the pattern and the shape of the connections created by these triples. Note that it is possible that two classes may differ only in the direction of one edge.

Each one of these classes is further divided into two sub-classes, differing by their semantic. For example, class `C1` of LinkedMDB is divided into `C1.1`, with queries asking for all the films of a certain director, and class `C1.2`, made of queries asking for all the films starring a certain actor.

The databases considered here, LinkedMDB 1M and IMDB 1M (of 1M triples each), are built by randomly extracting connected subsets of triples from the original datasets. The triples of the ground truths produced by the SPARQL are added to these datasets. This was necessary since only SUMM, TSA+BM25 and TSA+VDP are able to scale to the whole dimensions and we needed to be able to compare all systems.

Table 1: Performances of the algorithms on the 1M databases. † indicates the top performing systems returned by the Tukey HSD statistical test, with $\alpha < 0.01$. The best system is in bold. $\lambda = 0.1$

Dataset	Systems	tb-DCG	recall	time (sec)
LinkedMDB 1M	TSA+BM25	0.201±0.02	**0.733±0.02**†	**39.64±01.90**†
	TSA+VDP	**0.490±0.04**†	0.592±0.040†	318.78±21.60
	SLM	0.011±0.00	0.076±0.01	39.90±08.69†
	MRF-KS	0.400±0.03†	0.462±0.03	285.22±30.10
	SUMM	0.106±0.01	0.268±0.02	429.52±37.17
IMDB 1M	TSA+BM25	0.276±0.02	0.423±0.03	30.28±00.25†
	TSA+VDP	**0.583±0.04**†	**0.623±0.04**†	196.90±20.69
	SLM	0.011±0.00	0.057±0.00	**15.92±06.57**†
	MRF-KS	0.312±0.02	0.382±0.02	399.68±37.34
	SUMM	0.126±0.01	0.254±0.02	510.26±27.02

4 Experimental Performances

4.1 Average Analysis

Table 1 reports the average performances obtained by the five systems on the two databases, already presented in [6]. Considering tb-DCG, TSA+VDP is the best performing system, immediately followed by MRF-KS. The former also obtains good values of recall, only exceeded by TSA+BM25 on LinkedMDB 1M.

The two fastest algorithms are SLM and TSA+BM25. The former is very quick to produce its answers but has low effectiveness. The latter is fast thanks to its reliance on state-of-the-art implementations of BM25. On this note, TSA+VDP presents higher execution time due to its heuristics which prune and re-rank the graphs, obtaining more relevant graphs that are better ranked.

Analysis of the quality of the answers through the λ factor Note that in Table 1 we fixed λ to 0.1. This parameter was empirically chosen to show the differences in the effectiveness of the different systems. A higher λ could provoke all the values to be flattened to 0.

One may ask now what happens if we change λ. A higher λ describes a user who tolerates less noise in her answer graphs. It asks for answers that are more "tailored" around the GT. Figure 2 presents the values of recall and tb-DCG of the five systems on the two datasets as λ varies. On the x-axis we have different values for λ, from 0 to 1, with step 0.1. Every point in the plots is obtained computing the averages on the 50 queries issued on each dataset.

Let us start on LinkedMDB 1M. TSA+BM25 appears to be consistently the top-performing system in terms of recall at the varying of λ. Moreover, when $\lambda = 0$, that is when we do not count the noise in the answers, the values of average recall is around 0.9, illustrating how, on average, many relevant triples are retrieved by the system in the top 1000 positions. Immediately below TSA+BM25,

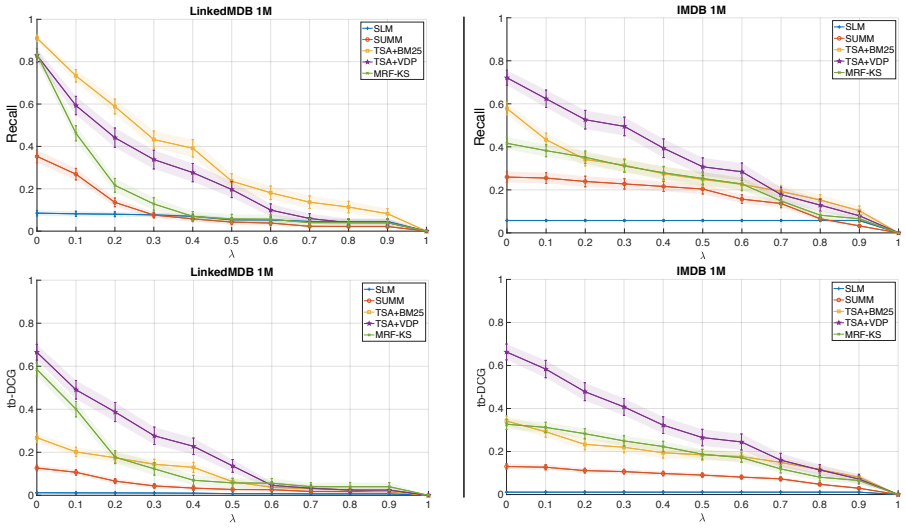

Fig. 2: Plot to confront average performances of recall and tb-DCG on the real databases LinkedMDB 1M and IMDB 1M for the different systems at the varying of λ.

we find TSA+VDP. The heuristics used by TSA+VDP eliminate some relevant triples from the ranking, thus reducing the overall recall.

For tb-DCG, it is evident how TSA+VDP sacrifices recall to obtain a better tb-DCG. In particular, TSA+VDP is consistently the top-performing system until we reach $\lambda = 0.5$. After this point, all the systems collapse on low values, due to the presence of few high-quality answers.

For IMDB 1M and the recall value, the top-performing system is now TSA+VDP, with TSA+BM25 immediately behind. This means that in the case of IMDB, BM25 is less effective in extracting relevant triples from the whole graph. The pruning heuristics of TSA+VDP improve the quality of some answers, making them relevant and therefore counted in the computation of the average recall. We also note that the trend of the SUMM method is more stable than the other systems. In fact, the system manages to get high-quality answers that satisfy many λ values. Only after a value of $\lambda = 0.5$ there is a performance decline.

Regarding tb-DCG, TSA+VDP remains the top system, with a sizeable difference from the second-placed TSA+BM25 and MRF-KS, which appear to be quite similar in performances while λ increases. SUMM maintains its stability but is still below the other systems.

This analysis clearly shows how bigger values of λ make the task harder. In comparison, however, the TSA-based systems are able to obtain consistently better results with respect to the state of the art systems.

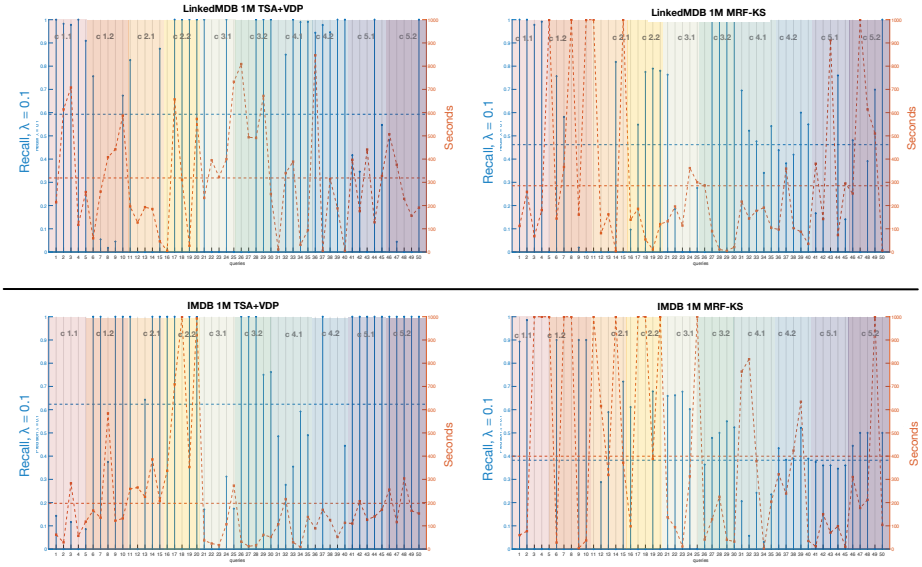

Fig. 3: Plots to compare performances of time and recall on LinkedMDB 1M and IMDB 1M for the two systems TSA+VDP and MRF-KS. The horizontal lines represent the averages for recall (blue) and time (red). A time limit of 1000s was considered for the execution of a query.

4.2 Query-by-query analysis

Recall Figure 3 shows the behavior query-by-query in terms of recall and time for the two datasets and for the two top-performing systems TSA+VDP and MRF-KS with $\lambda = 0.1$. We grouped the queries by their classes and subclasses and highlighted them with different colors.

Starting from LinkedMDB 1M, the queries of certain classes such as `C1.1` obtain good results in both systems. Others, like class `C5`, appear hard for both systems.

What is evident is that a simpler P does not necessarily imply high performances. In fact, a class such as `C1.2` presents low results in both systems, while `C4`, which has a more complex P, has higher recall with TSA+VDP. It seems that the semantic of the queries plays a critical role in determining their difficulty. Subclasses such as `C1.1` and `C1.2`, for example, show different behaviors, even if they share the same pattern structure.

For IMDB 1M, we can make the same observations. It is even more evident in this instance how a simpler syntax does not correspond to higher recall. For TSA+VDP class `C5` almost always presents a recall of 1, while many queries of class `C1` present values close to 0. This may be explained considering the fact that a simpler pattern corresponds to a ground truth with fewer relevant triples, harder to extract from the database in an effective way.

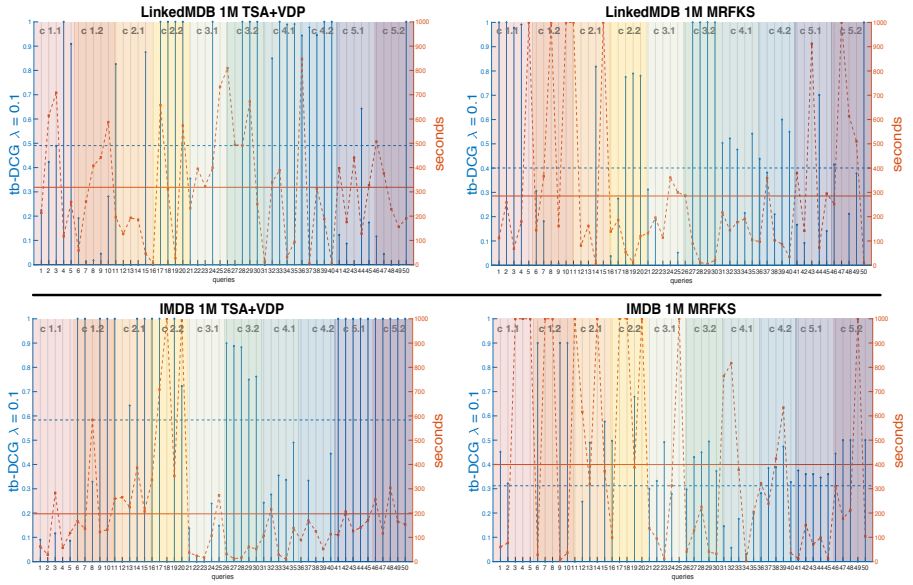

Fig. 4: Plots to compare performances of time and rb-DCG on LinkedMDB 1M and IMDB 1M for the two systems TSA+VDP and MRF-KS. The horizontal blue line represents the average precision, the red one the time.

tb-DCG Figure 4 presents similar graphics but for the tb-DCG metric.

The high variability in the behavior of the queries is immediately evident from the different performances obtained by the systems. Let us start with LinkedMDB 1M. Certain classes, such as `C4`, appear to be relatively easy for TSA+VDP. On the other hand, they are harder for MRF-KS. If we consider that `C1.2`, instead, is hard for both systems, we see how it is not necessarily true that a simpler syntax (i.e. a simpler pattern P) corresponds to an easier query to answer.

On the other hand, a class such as `C2` presents subclass `C2.1` which appears hard for both systems, while `C2.2` is easier. This suggests again how the same syntactic class can become harder or easier depending on its semantic nature, independently from the system.

For IMDB 1M, we see how similar observations can be made. Class `C5` appears simpler for TSA+VDP and harder for MRF-KS. More in general, it appears that MRF-KS struggles more in IMDB to answer the queries: its average tb-DCG is much lower than the one of TSA+VDP. This shows how it may be possible that the nature of a database (the number and structure of connections among nodes and the quantity and disposition of keywords among them) may influence the effectiveness of a system.

Time The execution times are presented in both Figure 3 and 4. Note how we used a time limit of 1000s.

Once again, queries in the same class may present different behaviors in terms of execution time. For example, MRF-KS presents relatively low values of time for subclass `C5.1` in IMDB, while subclass `C5.2` presents much higher times on average. This new evidence of the fact that the syntactical factor is not the only one that plays a role in the determination of the execution time. The choice of keywords (more or less frequent inside the database) appears to also play a role. Queries with more frequent keywords will, in fact, demand much more execution time.

5 Conclusions and Future Works

In this paper, we presented an in-depth analysis of the behavior of state-of-the-art systems with a particular focus on the quality of required answers and the behavior presented with different kinds of queries.

We used a sub-set of LinkedMDB and IMDB as databases. We defined five different classes of queries for each dataset, which differ from one another on their syntax and semantic, for a total of 100 test queries. We used recall and tb-DCG as evaluation metrics. The first one describes the simple raw power of extraction of relevant triples of a system. The latter is a more sophisticated measure that takes into consideration the relevance of the answers, their position in the ranking and the quantity of noise in them. We considered five different state-of-the-art systems: SLM, MRF-KS, SUMM, TSA+BM25, and TSA+VDP.

We performed two different classes of experiments. In the first one, we varied the minimal quantity of SNR required from an answer to be considered as relevant. We showed that the higher the required threshold, the lower the performances of the systems. TSA+BM25 and TSA+VDP consistently appeared to obtain the higher results, although we registered a progressive detriment of the performances.

The second class of experiments showed that certain classes of queries seem to be simpler than others for both systems. It resulted that not necessarily a simpler pattern P implies better results. The complexity of the query is therefore not present only in its syntax, but also in its semantic and in the keywords of choice. To corroborate this intuition, it appeared that queries belonging to the same syntactic class with different semantics presented completely different performances.

For our future works, we will keep researching, also on different databases, how different kinds of queries correspond to different behaviors, and how poor results may be mitigated with techniques as query rewriting.

References

1. Balmin, A., Hristidis, V., Koudas, N., Papakonstantinou, Y., Srivastava, D., Wang, T.: A system for keyword proximity search on XML databases. In: Proceedings of

29th International Conference on Very Large Data Bases, VLDB. pp. 1069–1072. Morgan Kaufmann (2003)
2. Bast, H., Buchhold, B., Haussmann, H.: Semantic search on text and knowledge bases. Foundations and Trends in Information Retrieval **10**(2-3), 119–271 (2016)
3. Bhalotia, G., Hulgeri, A., Nakhe, C., Chakrabarti, S., Sudarshan, S.: Keyword searching and browsing in databases using BANKS. In: Proceedings of the 18th International Conference on Data Engineering. pp. 431–440. IEEE Computer Society (2002)
4. Coffman, J., Weaver, A.C.: A framework for evaluating database keyword search strategies. In: Proc. of the 19th ACM International Conference on Information and knowledge management. pp. 729–738. ACM Press (2010)
5. Dosso, D., Silvello, G.: 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 (2019)
6. Dosso, D., Silvello, G.: Search text to retrieve graphs: A scalable RDF keyword-based search system. IEEE Access **8**, 14089–14111 (2020)
7. Elbassuoni, S., Blanco, R.: Keyword Search over RDF Graphs. In: Proc. of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011,. pp. 237–242. ACM Press, New York, USA (2011)
8. He, H., Wang, H., Yang, J., Yu, P.S.: BLINKS: ranked keyword searches on graphs. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. pp. 305–316. ACM (2007)
9. Le, W., Li, F., Kementsietsidis, A., Duan, S.: Scalable keyword search on large RDF data. IEEE Trans. Knowl. Data Eng. (11), 2774–2788. https://doi.org/10.1109/TKDE.2014.2302294
10. Lopez-Veyna, J.I., Sosa-Sosa, V.J., López-Arévalo, I.: A virtual document approach for keyword search in databases. In: DATA. pp. 39–48. SciTePress (2012)
11. Mass, Y., Sagiv, Y.: 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 (2016)
12. Pound, J., Mika, P., Zaragoza, H.: Ad-hoc object retrieval in the web of data. In: Proc. of the 19th International Conference on World Wide Web, WWW 2010. pp. 771–780. ACM Press, New York, USA (2010)
13. Prud, E., Seaborne, A., et al.: Sparql query language for rdf (2006)
14. Sahu, S., Mhedhbi, A., Salihoglu, S., Lin, J., Özsu, M.T.: The ubiquity of large graphs and surprising challenges of graph processing. PVLDB **11**(4), 420–431 (2017)
15. Su, Q., Widom, J.: Indexing Relational Database Content Offline for Efficient Keyword-Based Search. In: Proc. of the 9th International Database Engineering and Applications Symposium (IDEAS 2005). pp. 297–306. IEEE Computer Society (2005)
16. Wang, H., Aggarwal, C.C.: A survey of algorithms for keyword search on graph data. In: Managing and Mining Graph Data, pp. 249–273. Springer (2010)
17. Wu, W.: Proactive Natural Language Search Engine: Tapping into Structured Data on the Web. In: Proc. of the Joint 2013 EDBT/ICDT Conferences. pp. 143–148. ACM Press, New York, USA (2013)
18. Yu, J.X., Qin, L., Chang, L.: Keyword Search in Relational Databases: A Survey. IEEE Data Eng. Bull. **33**(1), 67–78 (2010)