Query Answer Reformulation over Knowledge Bases

João Pedro V. Pinheiro[1], Marco A. Casanova[1], Elisa S. Menendez[2]

[1] Department of Informatics, PUC-Rio
[2] Federal Institute of Education, Science and Technology Baiano
{jpinheiro,casanova}@inf.puc-rio.br, elisa.menendez@ifbaiano.edu.br

Abstract. The answer of a query, submitted to a database or a knowledge base, is often long and may contain redundant data. The user is frequently forced to browse through a long answer or refine and repeat the query until the answer reaches a manageable size. Without proper treatment, consuming the answer may indeed become a tedious task. This article then proposes a process that modifies the presentation of a query answer to improve the quality of the user's experience in the context of an RDF knowledge base. The process reorganizes the original query answer by applying heuristics to summarize the results and to select template questions that create a user dialog that guides the presentation of the results. The article also includes experiments based on RDF versions of MusicBrainz, enriched with DBpedia data, and IMDb, each with over 200 million RDF triples. The experiments use sample queries from well-known benchmarks.

Categories and Subject Descriptors: H.2 [**Database Management**]: Heterogeneous Databases; H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

Keywords: Aggregation, Summarization, Natural Language Query (NLQ), Question Answering (QA), RDF, Semantic Web

1. INTRODUCTION

Question Answering (QA) systems combine techniques from multiple fields of computer science, including Natural Language Processing (NLP), Information Retrieval, Machine Learning (ML), and Semantic Web. Assuming that the user is interested in querying a database or a knowledge base, a QA system may be split into two parts: *question*, which receives a user's input in natural language, transforms it into a structured query and searches the data; and *answer*, which displays consistent results in a human-readable format to the user. The answer to a query is often long and may contain redundant data. The user is frequently forced to browse through a long answer or refine and repeat the query until the answer reaches a manageable size. Without proper treatment, consuming the answer may indeed become a tedious task.

This article addresses the problem of *query answer modification* to improve the quality of the user's experience, in the context of an RDF knowledge base. For example, imagine yourself as a user interacting with a virtual voice assistant, and you ask an open-ended question about a specific subject, e.g., *"Which artists were born on May 30th?"*. The query answer may have a long list of artists, partly shown in Table I. Instead of listing the results, the virtual assistant may formulate questions to the user based on the prior result set, such as: *"Do you want to list American or European artists?"*; *"Do you prefer Jazz, Pop, or Classical music?"*; and *"Do you want to filter by active artists?"*.

The article proposes a process that reorganizes the original query answer by applying heuristics to summarize the results and to select template questions that create a user dialog that guides the presentation of the results. The heuristics allow deciding which properties returned in the query

Table I. Example of question-answer for an open-ended question.

Artist	**Genre**	**Birth Date**	**Death Date**	**Gender**	**Nationality**
Goodman, Benny	Jazz	1909-05-30	1986-06-13	Male	American
Leonhardt, Gustav	Classical	1928-05-30	2012-01-16	Male	Dutch
Green, CeeLo	Pop	1974-05-30		Male	American
Biosphere	Eletronic	1962-05-30		Male	Norwegian
Fredriksson, Marie	Pop	1958-05-30	2019-12-09	Female	Swedish
Banhart, Devendra	Folk	1981-05-30		Male	American

answer are interesting to apply aggregations (**group by** operations) and which template questions best fit each case. The heuristics also help decide if the answer is ready to be displayed to the user, or if the answer must be improved. The decisions use global statistics about the RDF dataset, obtained a priori, and local statistics about the query answer, obtained dynamically. The statistics are related to the frequency of the class instances and the frequency of the predicates.

The article also includes experiments based on the RDF versions of MusicBrainz and IMDb described in [Menendez et al. 2019], and obtained by enriching a MusicBrainz dump with DBpedia data and transforming an IMDb relational database to RDF via R2RML, respectively. Each RDF dataset has over 200 million triples. Our experiments use sample queries from the QALD - Question Answering over Linked Data[1] challenge and from Coffman's benchmark [Coffman and Weaver 2010].

The rest of the paper is organized as follows. Section 2 summarizes related work. Section 3 discusses the query answer modification process. Section 4 describes the experiments. Finally, Section 5 presents the conclusions and directions for future research.

2. RELATED WORK

In a seminal paper, Webber proposed a theoretical framework that divided the communication between humans and machines into three parts [Webber 1986]. Instead of only considering questions and answers, Webber proposed a clear distinction between a question, an answer, and a response. A question is a request by a user that demands information or asks to perform an action. An answer is the information or performance directly requested. A response embraces multiple elements, such as a direct answer, information or actions related to the original request instead of an answer, and additional information or actions when no proper answer can be found (also called "did you mean?"). Comparing the proposed framework with our work, we consider only open-ended questions. Thus, we can assume that a question is strictly demanding information. From the point of view of the answer (called "response" in the framework), our process suggests information related to the original request instead of a single answer. This information is presented to the user as facets, which allows constructing a dialog and providing better guidance to the desired answer.

Knowledge Base systems are usually constructed from multiple sources, leading to the generation of duplicated data. By contrast, humans avoid redundancy in the act of writing or speaking. Indeed, reducing duplicated data in the communication between humans and machines is a challenging task. Aggregation and summarization are important techniques that help to solve this issue.

The problem of redundancy is addressed, for example, in [Dalianis and Hovy 1996]. The authors suggested aggregation strategies to remove redundancy from the text - usually retrieved answers from databases. An interesting example, used in the paper, to illustrate the problem goes as follows. Consider the question:

```
``Who is currently at ISI?''
```

[1]http://qald.aksw.org, (accessed 7 July 2020).

Suppose that the answer to this question is:

```
''Yigal is an employee at ISI. Hercules is a visitor at ISI.
Eduard is an employee at ISI. Kevin is an employee at ISI.
Vibhu is a student at ISI.''
```

Note that the answer is too long and repetitive. After applying the aggregation rules suggested in the paper, the modified answer is shortened and easier to understand:

```
''At ISI, Yigal, Eduard, and Kevin are all employees;
Vibhu is a student; and Hercules is a visitor.''
```

The authors developed a questionnaire and applied it to computer scientists. The questionnaire was composed of five example sets of input data, and each example contained between 11 and 18 propositions. After analyzing the answers, four classes or types of aggregation rules became obvious: grouping and collapsing rules, ordering rules, casting rules, and parsimony rules. Furthermore, the paper listed eight aggregation strategies, each related to one aggregation class. As future work, the authors suggested a follow-up study involving a larger group of people, not all of whom being computer scientists, for more general results. Also, in the last section of the paper, the authors presented some scenarios that lead to unsatisfactory results and listed future improvements.

Deutch et al. (2017) described an approach to present query results as sentences in Natural Language (NL) with provenance information. The authors argued that the answers in the query result lack justification and suggested the notion of provenance, which corresponds to including additional information to the query results. Also, provenance information helps validate answers. The paper used the MAS (Microsoft Academic Research) publication database to validate the results. The proposed solution had the following key contributions: provenance tracking based on the NL query structure, factorization, summarization, and implementation and experiments.

An important step was provenance tracking based on the query structure. Deutch et al. (2017) used two external tools in this process. The modified NaLIR (Natural Language Interface for Relational databases) tool was used to store exactly which parts of an NL query translate to which parts of the formal query. The evaluation of the formal query used the provenance-aware engine SelP - Selective tracking and presentation of data provenance. It stored which parts of the query "contributed" to which parts of the provenance. Thus, two mappings were available for the next steps: text-to-query-parts and query-parts-to-provenance.

This study was the first one to address provenance for the NL queries problem. After implementation and experiments, the authors listed two main limitations of their work. The sentence generation module was specifically designed for NaLIR and will need to be replaced if a different NL query engine is used. Second, the solution is limited to conjunctive queries, not supporting unions and aggregations.

In a similar direction, faceted browsing [Petzka et al. 2017] (also called "faceted search" [Wei et al. 2013]) is a complement to keyword search, which provides an iterative way to refine search results. Facets are usually displayed to the user as rectangles right next to the main list of results provided by keyword search. These facets contain relevant grouped information which guides users to the desired answer.

In [Moreno-Vega and Hogan 2018] and [Franz et al. 2009], faceted browsing was used to simplify the user's interaction with data. Moreno-Vega and Hogan (2018) allowed search by keyword or type. A type would be an IRI from the RDF graph. While there are facets available, the user can navigate interacting only with them. Facets with zero results are never offered. Franz et. al. (2009) used faceted browsing for user evaluation purposes. With fixed *subject* and *predicate*, the user receives a list of *objects* sorted by the so-called *TripleRank* and decides if each of the objects "is related", "does not know", or "is not related".

Finally, several studies addressed the problem of creating a Natural Language interface to databases, such as [Novello and Casanova 2020]. Usually, the proposed process has four steps: *Question Analysis*, *Phrase Mapping*, *Disambiguation*, and *Query Construction* - not necessarily in this order [Diefenbach et al. 2018]. In this article, we assume that the Natural Language interface is constructed over an RDF knowledge base and accessed through a SPARQL endpoint.

3. THE QUERY ANSWER MODIFICATION PROCESS

This section is organized as follows. Section 3.1 provides some basic definitions required for our discussion. Section 3.2 describes the process of transforming a single-column into a three-column result set. Section 3.3 addresses the use of frequency analysis based on RDF metadata. Section 3.4 briefly discusses how to construct the user dialog.

3.1 Basic Definitions

The Resource Description Framework - RDF [Cyganiak et al. 2014] is a standard model for data interchange on the Web. A key characteristic of RDF is the ability to model data and metadata without distinction, which facilitates the evolution of schemes over time without impairing the way data are consumed. An *RDF triple* is of the form (s, p, o), where s is the *subject*, p is the *predicate*, and o is the *object* of the triple. An *RDF triple set* (also called an *RDF dataset*) is a set of RDF triples, which induces a directed, labeled graph, whose nodes are the subjects and objects of the triples and there is an edge labeled with p iff there is a triple (s, p, o) in the triple set.

SPARQL is the standard query language for RDF [Prud'hommeaux and Seaborne 2008]. A simple example of a SPARQL query, which retrieves the URIs of all movies performed by *"Denzel Washington"*, is:

```
prefix imdb: <http://www.imdb.com/>

select distinct ?movie
where {
  ?movie a imdb:Movie .
  ?movie imdb:actor ?actor .
  ?actor imdb:name "Denzel Washington" .
}
```

The *query form* `select` followed by the *solution modifier* `distinct` guarantees that only unique URIs will be presented in the result set. The `where` *clause* restricts the result by applying a graph pattern matching over the RDF graph. Also the term `'a'` is the syntactical sugar form of the predicate `rdf:type`.

SPARQL also supports aggregation and subqueries, which are the main topics of this article, addressed in Section 3.2.

3.2 Transforming single-column into three-column result sets

The query answer modification process we propose starts after the query is executed. The expected inputs are the SPARQL query and the result set, as illustrated in Figure 1. There are two possible scenarios: the result set has a *single column*, or the result set has *multiple columns*. Our study focuses on the first case.

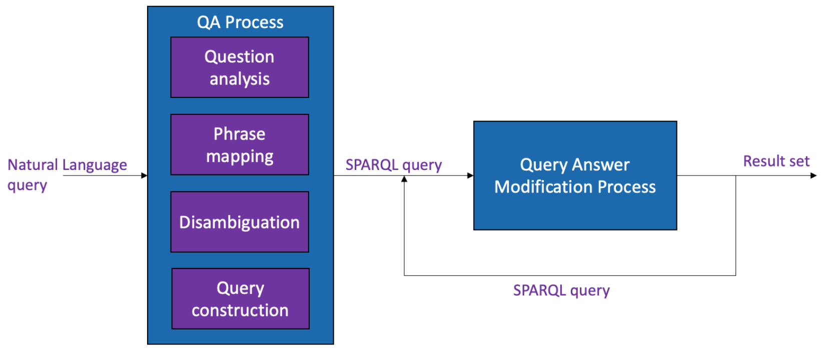


Fig. 1. Query answer modification process.

We base our discussion on a series of question-answering challenges over Linked Data, referred to as QALD - Question Answering over Linked Data[2]. Several papers use QALD to measure quality metrics of the system's answers [Diefenbach et al. 2017]. We noticed that most queries listed in the QALD challenges had single-column answers, which calls for enriching the answers for the purposes of this paper. A simple approach is to add to the instances returned their property values. Indeed, frequently, the answers represent sets of instances of the same `rdf:type`. So, it is straightforward to modify the original SPARQL query to also retrieve the desired property values.

As an example, consider again the question *"Which artists were born on May 30th?"*. The result set of the corresponding SPARQL query has instances of type `mo:MusicArtist`, as in Figure 2(c). Then, modifying the original SPARQL query also enables to retrieve property values, as shown in Figure 2(d). Note that, in Figure 2(d), the column *artist* has repeated values. However, instead of normalizing the returned table, we decided to keep this three-column format to simplify data manipulation.

3.3 Frequency analysis based on computed metadata

In the process we propose, a set of SPARQL queries is used to generate graph statistics, which help decide what to do next. These statistics are related to the frequency of the instances by class and the frequency of the predicates.

There are two types of frequencies used. A *global frequency* is defined over the full graph and is computed only once before any query is executed. On the other hand, a *local frequency* is defined over the sub-graph generated as in Section 3.2 and is computed at run time. Both *global* and *local frequencies* are computed over predicates pointing to literals only.

Entity ranking is based on *InfoRank*, a family of importance measures proposed in [Menendez et al. 2019]. The proposed importance measures are combinations of three intuitions: (I) "important things have lots of information about them"; (II) "important things are surrounded by other important things"; (III) "few important relations (e.g., friends) are better than many unimportant relations (e.g. acquaintances)". Hence, the strategy is based on the level of informativeness of an entity, represented as literals in RDF graphs. They use a PageRank-inspired approach to propagate the importance scores from entity to entity. The *InfoRank* metric helps our process prioritize the most relevant triples of the result set.

As an example, Figure 3 shows an instance **A1**. The initial state (Figure 3(a)) has predicates pointing to literals and other instances. Notice that the final state (Figure 3(b)) only has predicates pointing to literals and an extra predicate called *InfoRank*.

[2]http://qald.aksw.org, (accessed 7 July 2020).

```
prefix mo: <http://purl.org/ontology/mo/>
prefix dbo: <http://dbpedia.org/ontology/>

select distinct ?artist
where {
  ?artist a mo:MusicArtist .
  ?artist dbo:birthDate ?date .
  filter(regex(?date, "5-30$", "i")) .
}
```

(a) Single-column query

```
select distinct ?artist ?predicate ?object
where {
  {
    select ?artist
    where {
      ?artist a mo:MusicArtist .
      ?artist dbo:birthDate ?date .
      filter(regex(?date, "5-30$", "i"))
    }
  }  # graph pattern 1 - prior query
  .  # conjunction (inner join)
  {
    select ?artist ?predicate ?object
    where {
      ?artist ?predicate ?object .
      filter(isLiteral(?object))
    }
  }  # graph pattern 2
}
```

(b) Three-column query

artist
mo:MusicArtist/1
mo:MusicArtist/2
mo:MusicArtist/3
mo:MusicArtist/4
mo:MusicArtist/5
mo:MusicArtist/6
mo:MusicArtist/7
mo:MusicArtist/8
mo:MusicArtist/9
mo:MusicArtist/10

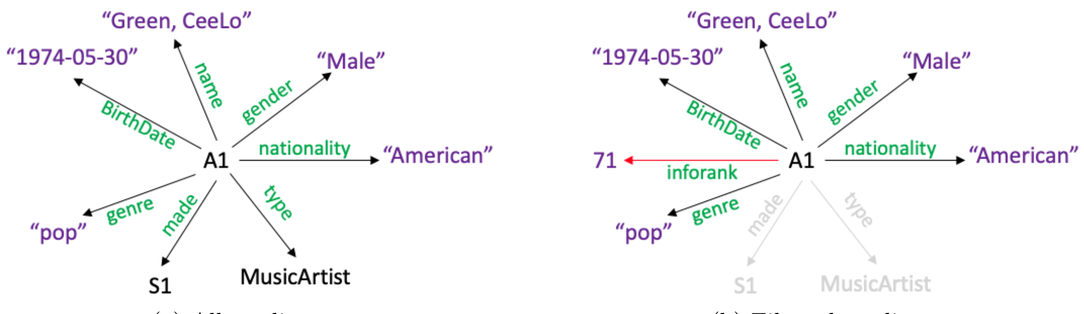
(c) Single-column result set

artist	predicate	object
mo:MusicArtist/1	`foaf:name`	`"Green, CeeLo"`
mo:MusicArtist/1	`mo:genre`	`"pop"`
mo:MusicArtist/1	`dbo:BirthDate`	`"1974-05-30"`
mo:MusicArtist/1	`dbo:DeathDate`	`""`
mo:MusicArtist/1	`foaf:gender`	`"Male"`
mo:MusicArtist/1	`dbp:nationality`	`"American"`
mo:MusicArtist/2	`foaf:name`	`"Leonhardt, Gustav"`
mo:MusicArtist/2	`mo:genre`	`"Classical"`
mo:MusicArtist/2	`dbo:BirthDate`	`"1928-05-30"`
mo:MusicArtist/2	`dbo:DeathDate`	`"2012-01-16"`
mo:MusicArtist/2	`foaf:gender`	`"Male"`
mo:MusicArtist/2	`dbp:nationality`	`"Dutch"`
mo:MusicArtist/3	`foaf:name`	`"Goodman, Benny"`
mo:MusicArtist/3	`mo:genre`	`"Jazz"`
mo:MusicArtist/3	`dbo:BirthDate`	`"1909-05-30"`
mo:MusicArtist/3	`dbo:DeathDate`	`"1986-06-13"`
mo:MusicArtist/3	`foaf:gender`	`"Male"`
mo:MusicArtist/3	`dbp:nationality`	`"American"`

(d) Three-column result set

Fig. 2. Transformation with SPARQL queries.

Parameterized thresholds are used to filter predicates that are candidates to be used in a **group_by** operation. By default, these threshold values (δ) are set between 0.4 and 2.0, which means the predicate must appear in at least 40% and not more than 200% of the unique subjects. For clarity, consider again the question *"Which artists were born on May 30th?"*. Analyzing the three-column result set, the predicate `rdfs:comment` appears 497 times and there are 123 unique artists. This means the predicate appears 4.04/artist on average. Thus, the predicate `rdfs:comment` is removed by the process because it exceeds the maximum threshold.



(a) All predicates (b) Filtered predicates

Fig. 3. Filtered predicates by literal and highlighted *InfoRank*.

3.4 Computing the user dialog

As mentioned in the previous section, an aggregation process is applied over the filtered predicates. These predicates are evaluated and sorted by their *local frequency*. Another threshold is related to the number of aggregated values (α), which has 10 as the default value. After this filter, the process chooses which template question must be used and returns a single new question to the user.

For example, consider once again the question *"Which artists were born on May 30th?"*. Analyzing the previous filtered candidates, the predicate `dbo:activeYearsStartYear` has 35 unique values. Although it might be a promising candidate, the predicate `dbo:activeYearsStartYear` is removed by the process because it exceeds the α threshold.

Template questions are choice questions, formulated in natural language, that offer aggregated predicates as alternatives to the user. Using the same example, the filtered predicates are: `foaf:gender`, with two aggregated values - *female* and *male*; and `dbo:background`, with three aggregated values - *non_performing_personnel*, *non_vocal_instrumentalist*, and *solo_singer*. Thus, the process may generate the following questions:

(1) Between *{non_performing_personnel}*, *{non_vocal_instrumentalist}*, and *{solo_singer}*, which artists do you prefer?
(2) Do you prefer *{female}*, or *{male}* artists?

Since `dbo:backgroud` precedes `foaf:gender` in the computed frequencies, only question (1) is returned to the user. If the user keeps interacting with the system, the whole process restarts. Also, there is a final threshold (β) responsible for limiting the number of elements returned to the user, which has 15 as the default value.

4. EXPERIMENTS

4.1 Setup

We performed initial experiments using the RDF versions of MusicBrainz and IMDb[3] described in [Menendez et al. 2019], obtained by enriching a MusicBrainz dump with DBpedia data and transforming an IMDb relational database to RDF via R2RML, respectively. These datasets also contain the *InfoRank* scores of instances, properties and classes. Figure 4 shows the resulting MusicBrainz and IMDb schemas, respectively. Each RDF dataset has over 200 million triples. We used sample queries from the QALD challenge and Coffman's benchmark [Coffman and Weaver 2010].

[3]https://sites.google.com/view/quira/, (accessed 7 July 2020).

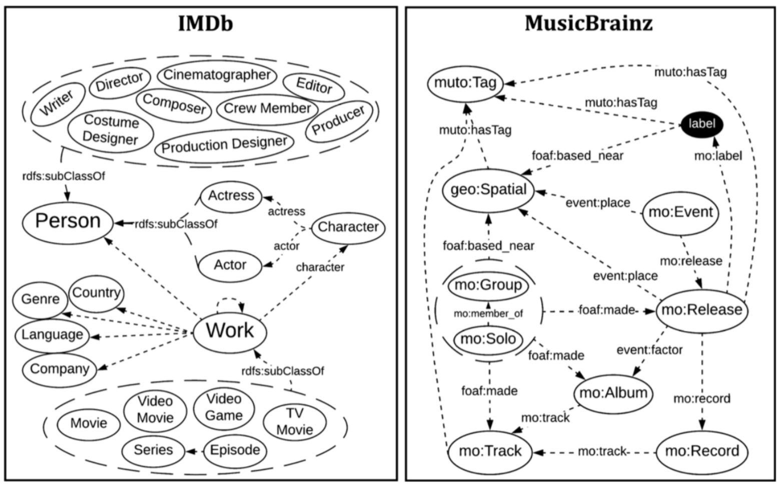


Fig. 4. IMDb and MusicBrainz schemas.

To store and manage the RDF datasets, we used the component TDB2 of Apache Jena for RDF[4]. Apache Jena Fuseki (a SPARQL server) ran on a server machine with OS GNU/Linux Ubuntu 16.04.6 LTS, a quad-core processor Intel(R) Core(TM) i7-5820K CPU @ 3.30GHz, 64 GB of RAM and SSD 1TB.

In the following sections, we discuss the effect of the proposed query reformulation process - referred to as the *process*, for brevity - over both datasets on the query result. The thresholds used for these experiments were: $\alpha = 10$, $\beta = 15$ and $\delta = (\delta_{min}, \delta_{max}) = (0.4, 2.0)$.

4.2 MusicBrainz Results

In this section, we detail the experiments with the QALD query for MusicBrainz presented earlier: *"Which artists were born on May 30th?"*. The initial process generated the following SPARQL query, with results ranked by the *InfoRank* score.

```
prefix mo: <http://purl.org/ontology/mo/>
prefix dbo: <http://dbpedia.org/ontology/>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix quira: <http://www.quira.org/>

select distinct ?artist ?label ?inforank
where {
  ?artist a mo:MusicArtist .
  ?artist dbo:birthDate ?date .
  ?artist rdfs:label ?label .
  ?artist quira:inforank ?inforank .
  filter (regex(?date, "5-30\$", "i")) .
}
order by desc(?inforank)
```

Table II shows a preview of the original result. Note that the SPARQL query returned 122 artists that were born on May 30th, which the user might consider to be a long list to interact with. We defined a threshold to indicate when the (reformulation) process should be applied. For these experiments, we chose a maximum of $\beta = 15$ lines, that is, a list with only β artists in this example. Hence, the

[4]https://jena.apache.org, (accessed 12 August 2020).

process reformulated the SPARQL query to capture the predicates that are candidates to be used for aggregation.

```
prefix mo: <http://purl.org/ontology/mo/>
prefix dbo: <http://dbpedia.org/ontology/>

select distinct ?artist ?predicate ?object
where {
  ?artist a mo:MusicArtist .
  ?artist dbo:birthDate ?date .
  filter (regex(?date, "5-30\$", "i")) .
  ?artist ?predicate ?object .
  filter(isLiteral(?objetct)) .
}
```

The result was stored in memory to facilitate manipulation and to avoid further access to the database. Then, the process grouped the results by predicate and counted the distinct object values. Table III presents these results.

We defined a threshold of α as the maximum number of distinct values for the predicates, so the questions formulated to the user are not too long. A heuristic is to choose the predicate with the highest number of distinct values from the set of predicates with less distinct values than the maximum. Following this heuristic, the process chose the predicate `dbo:background`, which refers to the type of music artists. Hence, the process formulated the query *"Which of the above artist background do you prefer?"* and presented a few options for the user to choose from, as shown in Table IV.

Note that Table IV provides five options to the user: the first three options correspond to the three distinct values of the predicate `dbo:background`; the fourth option to artists that have some background defined (not all artists have the background defined in the database); and the last option to all artists (artists, without any filter). Suppose the user chose option "1. Solo singer", which has one of the highest artist counts. The final result decreased to 27 artists, as shown in Table V.

Since this result was still higher than our threshold of $\beta = 15$ lines, the process was reapplied. Again, the process grouped the results by predicate and counted the distinct object values. Table VI presents this result.

Continuing with the proposed heuristic, the process chose the predicate `dbp:occupation` and formulated a new question to the user *"Which of the above artist occupation do you prefer?"* and presented a few options for the user to choose from, as shown in Table VII.

Suppose the user chose option "1. Singer-songwriter, musician" as the artist occupation. The process finally stopped since it achieved our threshold of β lines. Hence, the final result was presented to the user in a decreased order of *InfoRank* score, as shown in Table VIII.

Table II. Preview of the original SPARQL result.

#	**Artist partial URI**	**Artist name**
1	/artist/b09ae88f-4156-4caa-b129-1cacb5e1632e	Benny Goodman
2	/artist/27b0750a-7318-4075-9470-43b82d454ea0	Gustav Leonhardt
3	/artist/2c69465c-0f76-45ce-90a2-1ed0fdacc997	CeeLo Green
...	...	...
120	/artist/e00871b0-f6b5-41cf-b758-f2f1ea467818	Frank St. Leger
121	/artist/09ffe9f4-d54e-4943-8297-4456963f0def	Josephine Preston Peabody
122	/artist/22b95e86-0749-4df1-ae29-cd5acfe5a285	Jim Murray

Table III. Available predicates in the first reformulation process.

#	Predicate	Distinct Values
1	http://xmlns.com/foaf/0.1/name	144
2	http://dbpedia.org/property/birthDate	131
3	http://xmlns.com/foaf/0.1/givenName	129
4	http://www.w3.org/2000/01/rdf-schema#label	123
5	http://dbpedia.org/ontology/wikiPageID	120
6	http://dbpedia.org/ontology/wikiPageRevisionID	120
7	http://purl.org/dc/terms/description	91
8	http://xmlns.com/foaf/0.1/surname	87
9	http://dbpedia.org/ontology/deathDate	53
10	http://dbpedia.org/ontology/activeYearsStartYear	35
11	http://dbpedia.org/ontology/background	3
12	http://xmlns.com/foaf/0.1/gender	2

Table IV. Options for "Which of the above artist background do you prefer?".

#	Options	Counts
1	Solo singer	27
2	Non vocal instrumentalist	25
3	Non-performing personnel	5
4	Show me all artists with some background	57
5	Show me all artists	123

Table V. Preview of the SPARQL result filtered by "Solo singer" background.

#	Artist partial URI	Artist name
1	/artist/2c69465c-0f76-45ce-90a2-1ed0fdacc997	CeeLo Green
2	/artist/0110e63e-0a9b-4818-af8e-41e180c20b9a	Devendra Banhart
3	/artist/3a0373c0-f9c1-4eb3-9c10-53cc18193b07	Marie Fredriksson
...	...	...
25	/artist/0de740a2-a651-4d76-9cd5-54912a64070f	Gladys Horton
26	/artist/be0c5489-92e2-4149-b094-48293606f34b	Brian Fair
27	/artist/ae148627-23cc-48d3-a1a7-804f2af6b7dc	Rick DePiro (Ricky Dee)

4.3 IMDb Results

In this section, we use a query adapted from Coffman's benchmark over IMDb: "*Which movies did Denzel Washington starred?*". The initial process generated the SPARQL query below, ranking the results by the *InfoRank* score.

```
prefix imdb: <http://www.imdb.com/>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix quira: <http://www.quira.org/>

select distinct ?movie ?label ?inforank
where {
  ?movie a imdb:Movie .
  ?movie imdb:actor ?actor .
  ?movie rdfs:label ?label .
  ?movie quira:inforank ?inforank .
  ?actor imdb:name "Denzel Washington" .
}
order by desc(?inforank)
```

Table IX shows a preview of the original result. Note that this SPARQL query returned 49 movies starred by Denzel Washington. Again, we used the maximum of $\beta = 15$ lines as the threshold to

Table VI. Available predicates in the second reformulation process.

#	**Predicate**	**Distinct Values**
1	http://dbpedia.org/ontology/birthDate	40
2	http://xmlns.com/foaf/0.1/givenName	35
3	http://xmlns.com/foaf/0.1/name	31
4	http://www.w3.org/2000/01/rdf-schema#label	27
5	http://dbpedia.org/ontology/wikiPageID	26
6	http://dbpedia.org/ontology/wikiPageRevisionID	26
7	http://purl.org/dc/terms/description	20
8	http://dbpedia.org/ontology/activeYearsStartYear	18
9	http://xmlns.com/foaf/0.1/surname	16
10	http://dbpedia.org/property/caption	13
11	http://dbpedia.org/property/occupation	10
12	http://xmlns.com/foaf/0.1/gender	2

Table VII. Options for "Which of the above artist occupation do you prefer?".

#	**Options**	**Counts**
1	Singer-songwriter, musician	5
2	Singer	2
3	Musician	1
4	Singer, actor	1
5	Musician, songwriter	1
6	Singer-songwriter	1
7	Singer, author, philanthropist, actress	1
8	Musician, singer-songwriter, record label owner	1
9	Singer-songwriter, musician, visual artist	1
10	Singer, rapper, songwriter, record producer, actor, businessman	1
11	Show me all artists with some occupation	15
12	Show me all artists	27

Table VIII. The final result presented to the user.

#	**Artist partial URI**	**Artist name**
1	/artist/a0580131-73f3-49c8-aac5-2c478f64a363	Stephen Duffy
2	/artist/19e07fd0-5642-47a0-a2b9-b8176e6b06e5	Brooke Waggoner
3	/artist/23c738ed-5dc4-4ff7-8c00-3c1c54e8eb89	Kevin Barnes
4	/artist/1d566a14-4094-4f96-abb7-969b4f439728	Geva Alon
5	/artist/4e0e884d-099b-4ca9-bf4d-bcb31e739540	Duffy

indicate when the process should be applied. In this example, it would be a list with a maximum of β movies. Hence, the process reformulated the SPARQL query to capture the predicates used for aggregation.

```
prefix imdb: <http://www.imdb.com/>

select distinct ?movie ?predicate ?object
where {
  ?movie a imdb:Movie .
  ?movie imdb:actor ?actor .
  ?actor imdb:name "Denzel Washington" .
  ?movie ?predicate ?object .
   filter(isLiteral(?object)) .
}
```

Once again, the process grouped the results by predicate and counted the distinct object values.

Table IX. Preview of the original SPARQL result.

#	**Movie URI**	**Movie title**
1	http://www.imdb.com/work/1996688	Malcolm X
2	http://www.imdb.com/work/1592464	American Gangster
3	http://www.imdb.com/work/2354723	Unstoppable
...	...	...
47	http://www.imdb.com/work/1675254	Champs
48	http://www.imdb.com/work/2255730	The Equalizer
49	http://www.imdb.com/work/2356601	Uptown Saturday Night

Table X presents part of the available predicates. Following the proposed heuristic, the process chose the predicate `imdb:label`, which refers to the production company of the film. Hence, the process formulated the query *"Which of the above movie label do you prefer?"* and presented a few options for the user to choose from, as shown in Table XI.

Note that Table XI provides 12 options to the user: the first ten options refer to the movie label; the 11^{th} option to all movies with some label defined (not all movies have the label information defined in the database); and the last option to all movies (movies, without any filter). Suppose the user chose option "1. Columbia/Tristar" film label, which has one of the highest movie counts. The final result had only five movies, and the process stopped at this point since it achieved a reasonably compact result to present to the user. The final result was presented to the user in decreasing order of *InfoRank* score, as shown in Table XII.

Table X. Available predicates.

#	**Predicate**	**Distinct Values**
1	http://www.imdb.com/tag	2408
2	http://www.imdb.com/release_dates	1211
3	http://www.imdb.com/quotes	936
...	...	...
36	http://www.imdb.com/novel	14
37	http://www.imdb.com/label	10
38	http://www.imdb.com/number_of_chapter_stops	8
...	...	...
66	http://www.imdb.com/interviews	1
67	http://www.imdb.com/master_format	1
68	http://www.imdb.com/quality_program	1

Table XI. Options for "Which of the above movie label do you prefer?".

#	**Options**	**Counts**
1	Columbia/Tristar	5
2	Encore	5
3	Warner Home Video	3
4	MCA/Universal Home Video	2
5	Paramount	2
6	20th Century Fox Home Entertainment	1
7	Hollywood Pictures	1
8	Philips	1
9	Pioneer	1
10	RCA/Columbia	1
11	Show me all movies with some label	23
12	Show me all movies	47

Table XII. Final result presented to the user.

#	**Movie URI**	**Movie title**
1	http://www.imdb.com/work/2095826	Philadelphia
2	http://www.imdb.com/work/1824613	Glory
3	http://www.imdb.com/work/2031102	Much Ado About Nothing
4	http://www.imdb.com/work/2020598	Mississippi Masala
5	http://www.imdb.com/work/1731607	Devil in a Blue Dress

4.4 Qualitative Discussion

In this section, we discuss the results obtained for MusicBrainz and IMDb, from a qualitative point of view. We adopted a metric, called *compression rate*, to help compare the results obtained. The metric is denoted γ and defined as follows:

$$\gamma = 1 \text{ - } \kappa \ / \ \eta$$

where η is the number of lines of the initial result set and κ is the number of lines of the final result set.

Note that, when $\kappa \sim \eta$, the compression is very low. On the other hand, when $\kappa << \eta$, the compression is very high. It is important to recall that β guarantees that the final result set will not exceed a maximum number of lines. The analysis that follows will reflect this fact when the selected facet does not significantly reduce the result set.

Positioned at the end of this section, Tables XIV and XV have the same headers. Column `IRS` means *Initial Result Set* and is related to the number of lines from the original result set. Column `FRS` means *Final Result Set* and is related to the number of lines from the compressed result set. Column `Facets` has the selected predicate and facet separated by the pipe symbol `''|''`. Note that the number between parenthesis is the number of unique subjects or, in other words, the number of lines of the result set filtered by facet. Also, this column may have multiple facets displayed in multi-lines. The number of lines reflects the number of `Steps` the process took.

Table XIV presents the results for MusicBrainz and lists seven questions. Recall that ten questions were originally tested over MusicBrainz, but only 3 of them had no predicate selected. This happened because the process could not find a predicate that respects the thresholds $\alpha = 10$ and $\delta = (0.4, 2.0)$ defined for the experiments. For instance, the question *"What are the songs performed by Aretha Franklin?"* has the following predicates: `mo:track_number` (54), `rdfs:label` (1,120), and `mo:duration` (1,881). In these cases, all three numbers in parenthesis are higher than $\alpha = 10$. The original result set had 2,945 lines, and the final result set had $\beta = 15$ lines. Finally, we recall that, even when the process does not perform any steps, it ranks the results using the *InfoRank* metric and returns the first β lines to the user.

In the music context, it might be interesting to group songs by minutes. Based on the Music Ontology Specification[5], the predicate `mo:duration` represents the duration of a track or a signal in ms. Analyzing the songs by Aretha Franklin, her longest song is "Amazing Grace" with 10min and 48s (∼11min). So, we can guarantee that the heuristic would apply one of the ten possible length-facets (2min - 11min).

Continuing the analysis, the third question *"Which artists played on the same groups that David Bowie was a member of?"* had the original result set with length 17, which is very close to β = 15. Thus, the compression rate obtained by applying the heuristic was surprisingly large. But the predicate `dbo:wikiPageID` does not seem very interesting, considering a non-technical user. Hence,

[5]http://musicontology.com/specification/#term-duration, (accessed 12 August 2020).

the compression rate metric by itself does not capture all aspects. We still need to analyze the path explored by the process.

Table XV presents the results for IMDb and also lists seven questions. In this case, the most interesting predicates chosen and the facets selected were for the first question *"Which movies did Denzel Washington starred?"*. Although the compression rates were good, when compared to the other questions, the compression rates were the worst (together with the sixth question). This happened because all other questions had no restrict facet and the original result set was very long. Thus, $\kappa << \eta$ since $\kappa = \beta = 15$.

In this specific scenario, the use of a ranked list of results is important, as otherwise the process would randomly choose β results and return them to the user. Our process uses *InfoRank* to guarantee meaningful results. In Table XIII, the intersection between the results randomly sorted and the results sorted by *InfoRank* represents only $\sim 27\%$ of $\beta = 15$ movies. For instance, the multi-award-winning movie *"Gladiator"* would be missing in the result set.

Table XIII. Results comparison for "Which movies were released in 2000?".

	Results randomly sorted		**Results sorted by InfoRank**	
#	**Movie partial URI**	**Movie title**	**Movie partial URI**	**Movie title**
1	/work/1637423	Big Momma's House	/work/2007705	Me, Myself & Irene
2	**/work/2393883**	**X-Men**	/work/1823621	Gladiator
3	**/work/1625822**	**Battle Royale**	/work/2289533	The Patriot
4	**/work/1592761**	**American Psycho**	**/work/2393883**	**X-Men**
5	/work/1551278	2001: A Space Travesty	**/work/1625822**	**Battle Royale**
6	/work/1790475	Faust: Love of the Damned	**/work/2333414**	**Traffic**
7	/work/2363075	Versus	**/work/1592761**	**American Psycho**
8	/work/1812463	Före stormen	/work/1676655	Charlie's Angels
9	/work/2234180	The Adventures of Rocky & Bullwinkle	/work/2389882	Crouching Tiger, Hidden Dragon
10	/work/2016980	Militia	/work/2133745	Requiem for a Dream
11	/work/2350711	Unbreakable	/work/1685528	Citizen Toxie: The Toxic Avenger IV
12	/work/1976757	Little Nicky	/work/2061093	O Brother, Where Art Thou?
13	**/work/2333414**	**Traffic**	/work/2158237	Scary Movie
14	/work/1807398	Frequency	/work/2019684	Miss Congeniality
15	/work/2008819	Meet the Parents	/work/2139796	Road Trip

Finally, we investigated the lack of predicates with restrictive facets. Since the questions were related to `imdb:Actor` and `imdb:Actress`, we combined the sets of resources from `imdb:Actor` and `imdb:Actress` and called then *Artists*.

Analyzing the unique predicates related to instances of `imdb:Movie`, from 68 predicates, only 15 satisfy the α threshold. Also, considering the δ threshold, the set of selectable predicates is small, and their expressiveness is also low. Except for `imdb:category`, all selectable predicates were related to technical information about the movies. A similar scenario was observed for predicates related to instances of *Artists*. From the 26 available predicates, only one was selectable, regarding the defined thresholds.

5. CONCLUSIONS

The main contribution of this study was the definition of a process, called the *Query Answer Modification Process*, based on simple heuristics and parameterized thresholds, that analyses the query answer and improves the quality of the user's experience. This study addressed open-ended questions since specific questions do not generate long result sets, and would therefore not benefit from the summarization process.

To validate the proposed process, sample queries from the QALD challenge and Coffman's benchmark were used. These queries were applied over RDF versions of MusicBrainz and IMDb, respectively.

Table XIV. Compression Rate comparison over Music Brainz.

		MusicBrainz		
Question	**1**	**Which artists were born on May 30th?**		
Steps	Facets	IRS	FRS	Compression Rate

2	dbo:background | solo_singer (27);
	dbp:occupation | Singer-songwriter (5);
		123	5	95,93%

Question	**2**	**Which songs by Miles Davis are longer than 20 minutes?**		
Steps	Facets	IRS	FRS	Compression Rate
1	mo:track_number | 1 (48)	89	15	83,15%
Question	**3**	**Which artists played on the same groups that David Bowie was member of?**		
Steps	Facets	IRS	FRS	Compression Rate
1	dbo:wikiPageID | 1515176 (1);	17	1	94,12%
Question	**4**	**What are the albums from Michael Jackson?**		
Steps	Facets	IRS	FRS	Compression Rate
1	dbp:years | –05-24 (1)	23	1	95,65%
Question	**5**	**What are the albums from Kraftwerk?**		
Steps	Facets	IRS	FRS	Compression Rate
1	dbp:writer | Hütter (2)	13	2	84,62%
Question	**6**	**Which artists were born on September, 1964?**		
Steps	Facets	IRS	FRS	Compression Rate

2	dbo:background | solo_singer (18);
	foaf:surname | Anastasio (1);
		66	1	98,48%

Question	**7**	**Which bands broke up in 2010?**		
Steps	Facets	IRS	FRS	Compression Rate

2	dbo:background | group_or_band (236);
	dbo:activeYearsEndYear | 2010 (236);
		238	15	93,70%

Table XV. Compression Rate comparison over IMDb.

		IMDb		
Question	**1**	**Which movies did Denzel Washington starred?**		
Steps	Facets	IRS	FRS	Compression Rate
1	imdb:label | Columbia/Tristar (5);	49	5	89,80%
Question	**2**	**Which movies are available in spanish language?**		
Steps	Facets	IRS	FRS	Compression Rate
1	imdb:color_info | Color (67933);	77670	15	99,98%
Question	**3**	**Which actors or actresses were born on May 30th?**		
Steps	Facets	IRS	FRS	Compression Rate
1	imdb:gender | Male (409);	595	15	97,48%
Question	**4**	**Which movies were released in 2000?**		
Steps	Facets	IRS	FRS	Compression Rate

2	imdb:color_info | Color (5815);
	imdb:year | 2000 (5815);
		7206	15	99,79%

Question	**5**	**Which movies were produced in Brazil?**		
Steps	Facets	IRS	FRS	Compression Rate
1	imdb:color_info | Color (4432);	7016	15	99,79%
Question	**6**	**Which movies were produced in Brazil in 2010?**		
Steps	Facets	IRS	FRS	Compression Rate

2	imdb:color_info | Color (74);
	imdb:year | 2000 (74);
		91	15	83,52%

Question	**7**	**Which Brazilian artists starred foreign movies?**		
Steps	Facets	IRS	FRS	Compression Rate
1	imdb:gender | Male (733);	1261	15	98,81%

Also, a *compression rate* metric was defined, enabling a comparison and discussion over the compiled results.

The first suggestion for future work would be to allow users to provide a list of predicates to ignore or to enable them to dynamically exclude undesired predicates from the available list in each step of the process. Thus, it would be possible to exclude predicates without a clear contribution, from the user's point of view, and the results would possibly improve. Also, in this context, registering users' feedback after each query answer modification would provide a strategy to construct this list automatically.

A second suggestion would be to develop and apply a questionnaire to users interested in using the system. As a result, we would be able to segment users by preferences and create user profiles. These profiles would enrich the qualitative discussion in Section 4 and could be used to adapt the heuristics to the users' preferences.

Another interesting suggestion would be to allow users to indicate that specific predicates would actually define a taxonomy so that the system would be able to aggregate over each level of the taxonomy. In a second version, together with the initial exploration of the RDF Knowledge Base to calculate frequencies, the system would automatically infer the related taxonomy.

Finally, it would be profitable to develop a user interface similar to GraFa [Moreno-Vega and Hogan 2018] which would allow navigation through predicates and facets. An extra feature that would impact the user's navigation is the possibility to select more than one option at each interaction. Hence, disjunctive filtering would be possible, together with conjunctive filtering, as defined in this article.

ACKNOWLEDGMENT

This work was partly funded by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, and grants CAPES 88881.134081/2016-01, CNPq 302303/2017-0 and FAPERJ E-26-202.818/2017.

REFERENCES

Coffman, J. and Weaver, A. C. A framework for evaluating database keyword search strategies. In *Proceedings of the 19th ACM international conference on Information and knowledge management.* pp. 729–738, 2010.

Cyganiak, R., Wood, D., and Lanthaler, M. RDF 1.1 Concepts and Abstract Syntax, 2014. W3C Recommendation 25 February 2014.

Dalianis, H. and Hovy, E. Aggregation in natural language generation. In *Trends in Natural Language Generation An Artificial Intelligence Perspective*, J. G. Carbonell, J. Siekmann, G. Goos, J. Hartmanis, J. Leeuwen, G. Adorni, and M. Zock (Eds.). Vol. 1036. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 88–105, 1996.

Diefenbach, D., Lopez, V., Singh, K., and Maret, P. Core techniques of question answering systems over knowledge bases: a survey. *Knowledge and Information Systems* 55 (3): 529–569, June, 2018.

Diefenbach, D., Tanon, T. P., Singh, K., and Maret, P. Question Answering Benchmarks for Wikidata. In *ISWC 2017.* Vienne, Austria, 2017.

Franz, T., Schultz, A., Sizov, S., and Staab, S. TripleRank: Ranking semantic web data by tensor decomposition. In *Lecture Notes in Computer Science.* Springer Berlin Heidelberg, Berlin, pp. 213–228, 2009.

Menendez, E. S., Casanova, M. A., Leme, L. A. P., and Boughanem, M. Novel node importance measures to improve keyword search over rdf graphs. In *International Conference on Database and Expert Systems Applications.* Springer, pp. 143–158, 2019.

Moreno-Vega, J. and Hogan, A. Grafa: Faceted search & browsing for the wikidata knowledge graph. In *International Semantic Web Conference.* Springer, Cham, 2018.

Novello, A. and Casanova, M. A. A novel solution for the aggregation problem in natural language interface to databases (nlidb). In *Anais do XXXV Simpósio Brasileiro de Bancos de Dados.* SBC, Porto Alegre, RS, Brasil, pp. 217–222, 2020.

Petzka, H., Stadler, C., Katsimpras, G., Haarmann, B., and Lehmann, J. Benchmarking faceted browsing capabilities of triplestores. In *Proceedings of the 13th International Conference on Semantic Systems.* Semantics2017. Association for Computing Machinery, New York, NY, USA, pp. 128–135, 2017.

Prud'hommeaux, E. and Seaborne, A. SPARQL Query Language for RDF, 2008. W3C Recommendation 15 January 2008.

Webber, B. L. Questions, answers and responses: Interacting with knowledge-base systems. In *Topics in Information Systems.* Springer New York, pp. 365–402, 1986.

Wei, B., Liu, J., Zheng, Q., Zhang, W., Fu, X., and Feng, B. A survey of faceted search. *Journal of Web Engineering* vol. 12, pp. 41–64, 02, 2013.