Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 1772–1779
Marseille, 11–16 May 2020

Database Search vs. Information Retrieval: A Novel Method for Studying Natural Language Querying of Semi-Structured Data

Stefanie Nadig, Martin Braschler[1], Kurt Stockinger
Zurich University of Applied Sciences
Winterthur, Switzerland
steffinadig@gmail.com, {Martin.Braschler, Kurt.Stockinger}@zhaw.ch

Abstract

The traditional approach of querying a relational database is via a formal language, namely SQL. Recent developments in the design of natural language interfaces to databases show promising results for querying either with keywords or with full natural language queries and thus render relational databases more accessible to non-tech savvy users. Such enhanced relational databases basically use a search paradigm which is commonly used in the field of information retrieval. However, the way systems are evaluated in the database and the information retrieval communities often differs due to a lack of common benchmarks. In this paper, we provide an adapted benchmark data set that is based on a test collection originally used to evaluate information retrieval systems. The data set contains 45 information needs developed on the Internet Movie Database (IMDb), including corresponding relevance assessments. By mapping this benchmark data set to a relational database schema, we enable a novel way of directly comparing database search techniques with information retrieval. To demonstrate the feasibility of our approach, we present an experimental evaluation that compares SODA, a keyword-enabled relational database system, against the Terrier information retrieval system and thus lays the foundation for a future discussion of evaluating database systems that support natural language interfaces.

Keywords: database search, information retrieval, benchmark data set, query evaluation, relevance assessment

1. Introduction

The proliferation of World Wide Web use has democratized information search. Traditionally, technicians had crafted complicated search commands in languages such as SQL to retrieve results from relational databases. As more and more data and information became available to the public at large, new Web search services emerged that quickly became *the* main navigation tools for large segments of the online population. These services, with Google Web Search as the most successful among them, were built on technology from the academic field of Information Retrieval (IR) (Baeza-Yates & Ribeiro-Neto, 2011). An important characteristic of these services is that they allow the user to formulate queries in the form of either lists of keywords or even of complete, natural language sentences, thus dispensing with the technical barrier of using formal languages - allowing access to a much larger audience.

Web search services such as Google Web Search have indeed become so successful, that users today expect this kind of easy, non-formal access to data and information in many more searching scenarios. While the field of IR is concerned with the question of how to retrieve results from collections of unstructured (often textual) data, research over the last few years has also become more intensive on how to provide similar, natural language interfaces to databases operating on semi-structured or structured data. To this end, many different *natural language interfaces* (NLI) to databases have been proposed in the past several years (Affolter, Stockinger & Bernstein, 2019). The core idea is to map the natural language input of a (potentially non-technical) user to a formal language such as SQL. The yardstick that the NLI-enabled systems are held against is an ideal, manually crafted SQL statement for the same information need as expressed by the user's input.

In this paper, we claim that there is an alternative approach to measuring the effectiveness of such database systems: one which is built analogously to the methodologies employed in IR evaluation, and one that is more concerned with the question of whether the *relevant* data or information is contained in the result set, and less with rendering a (correct) statement in the formal target language.

The most commonly employed evaluation methodology in IR is the Cranfield methodology (Voorhees, 2001), which is test collection-driven: a set of sample information needs is used to query a given collection of retrievable items (documents), with known assessments of relevance for the query/document pairs. In the following, we will demonstrate how to adapt an IR test collection of this style from the INEX campaign 2011 for use with an NLI-equipped database system, and how to then compare the output of said system with an IR system that uses the data directly. The adapted INEX collection enables a whole range of new, result-focused evaluation experiments for NLI databases, some of which we report on as examples. As we discuss in the paper, the necessary scripts to process the adaptation are publicly available and should be interesting for a wide range of researchers in the field.

The paper makes the following contributions:

- We discuss two different architectures of implementing natural language interfaces to databases: Approach 1 is database-centric and treats the problem basically as a Steiner-tree graph search problem over a relational database. Approach 2 is information retrieval-centric where the problem is basically treated as a vector space problem.
- We provide a newly refactored data set for benchmarking NLI databases by adapting the INEX test collection. In particular we map the

[1] Corresponding author

data set to a relational database schema to enable querying it via SQL.

- We give a detailed experimental evaluation of querying NLI databases with a database search approach and compare it against an information retrieval approach. Our results lay the foundation for a future discussion on design and implementation of a hybrid approach that takes advantage of the strengths of database and information retrieval technologies to build even more powerful NLI databases than currently available.

The remainder of the paper is structured as follows. Section 2 surveys the related work on natural language interfaces to databases. Section 3 introduces the data set that serves as our benchmark and demonstrates how we map it to a relational database design. In addition, we describe the two systems that we use for evaluation, namely SODA and Terrier. In Section 4 we provide a systematic experimental evaluation of the both systems and, finally, draw conclusions in Section 5.

2. Related Work

Modern NLI databases are based on technologies at the intersection of three research communities in the areas of databases, information retrieval and machine learning. NLI databases can be classified into the following five different approaches according to a recent survey (Affolter, Stockinger et al. 2019): *keyword-based*, *pattern-based, parsing-based, grammar-based and neural machine translation-based*. We will briefly review the major ideas of these approaches.

Keyword-based systems are the earliest approaches. These systems do not support full natural language queries but only allow simple keyword queries. The basic idea is to use an inverted index on the base data and on the metadata, i.e. the database schema. The two inverted indexes are then applied for query understanding and generation of SQL statements by leveraging primary-foreign key relationships of the database schema. Examples of these systems are Precise (Simitsis et al., 2008), SODA (Blunschi et al., 2012) and Aqqu (Bast and Haussmann, 2015).

Pattern-based systems support answering more complex queries in natural language rather than only keyword-style. Typical examples are QuestIO (Damljanovic et al., 2008) and NLQ/A (Zheng et al., 2017).

Parsing-based systems use a natural language parser to analyze and reason about the grammatical structure of a query. This approach allows to better understand the dependencies between query tokens and to disambiguate query terms. Typical examples are NaLIR (Li and Jagadish, 2014) and ATHENA (Saha et al., 2016).

Grammar-based systems restrict the formulation of queries according to a specific set of rules, i.e. a grammar. This grammar defines how questions can be built, understood and answered by the system. Typical examples are TR Discover (Song et al., 2015), and SPARKLIS (Ferre, 2017).

Neural machine translation-based systems treat the translation of natural language to SQL as a Question/Answer problem. Given pairs of natural language questions and their respective SQL-statements, supervised machine learning techniques are used for translation. This approach is similar to translating between two natural languages such as from German to English. However, the major difference is that the translation process not only needs to learn language specific aspects but also the schema of the underlying databases. Typical examples are introduced by (Iyer et al., 2017) and (Basik et al., 2018).

All the above described approaches are usually studied in isolation. To the best of our knowledge, there is no systematic comparison of a database-centric vs. an information retrieval-centric approach. We assume that a lack of a suitable test collection is a major reason for this gap.

Available benchmark data sets typically applied in the database community only contain natural language/SQL pairs without sufficient information on relevance of the retrievable items - ie., relevance assessments commonly included in information retrieval test collections.

3. Data Set and System Architecture

In this section we will first describe the data set that underlies the test collection that we have adapted for use as a new benchmark for evaluating NLI databases. Afterwards we discuss two different implementations of NLI databases. In particular, we will introduce the database-centric approach based on SODA (Blunschi et al. 2012) as well as the information retrieval-centric approach based on the Terrier system (Macdonald and He, 2008). Next, we describe how to transform the XML-based INEX data set to a relational database design and finally, we show how to conduct experiments for both of these approaches on the newly adapted test collections.

3.1 Data Set

The test collection used for the INEX Data-Centric Track at the INEX 2011 conference (Wang et al., 2011) is used as the basis for our adaptations. The test collection includes a set of documents, a set of information needs ("topics") and the relevance assessments that link the two. This is paired with a scoring tool. The document set contains approximately 4.4 million XML files filled with information originally sourced from the Internet Movie Database (IMDb). Each file can be assigned to one of the two categories *movie* or *person* and provides further details related to each category, respectively.

3.1.1 Query Set

During the original INEX 2011 evaluation track, participating groups were asked to create ad-hoc search input representing user needs. The result is a set of 45 "topics" in XML syntax that serve as the basis for queries in the experiments.

A typical example of a topic is as follows:

```
<topic id="2011103" guid="5">
   <task>AdHoc</task>
   <title>Actors that played James Bond</title>
   <castitle> //person[about(.//act//movie//character,
                James Bond)]//name </castitle>
   <description>Name all actors that played the role of
                   James Bond</description>
   <narrative> List the names of all actors that played
                  the role of James Bond. </narrative>
</topic>
```

To generate the query set used for the experimental evaluation, the content from the title tag within the topic file was extracted in order to create natural language style user input. For illustration, the resulting queries 1 to 10 are shown in Table 1. This is the common way of treating topics in information retrieval experiments; the fields "description" and "narrative", while exploitable in cases where verbose query statements are desired, are mainly designed to assist relevance assessment by the test collection creators.

Number	Query
1	social network
2	best movie award "James Cameron"
3	Actors that played James Bond
4	movie Ellen Page thriller
5	king kong jack black
6	movie "Terry Gilliam" "Benicio del Toro" "Dr gonzo"
7	Tom Hanks biography
8	director of artificial intelligent "Haley Joel Osment"
9	Trivia "Don Quixote"
10	plot seven

Table 1: First 10 queries of the INEX test collection.

3.1.2 Relevance Assessment

INEX participants used this topic set for experiments on their own, different systems. The output of those original experiments was then used as the basis to create relevance assessments. The specific strategy to keep relevance assessment cost within practical limits lies outside the scope of this paper, and the interested reader is referred to (Voorhees, 2002). For those documents that have been assessed with respect to their relevance for a specific topic, a binary value (relevant/non-relevant) is recorded. The full set of relevance assessments was used for scoring systems in the original INEX experiments, and also serves the same purpose in our adapted test collection.

3.2 Search Over Data Warehouse (SODA)

The SODA system was developed to simplify search processes for business analysts who are not familiar with SQL syntax (Blunschi et al. 2012). Using SODA enables searching within relational databases in a Google-like manner where executable SQL statements are generated automatically from ad-hoc keyword queries.

SODA is based on a graph model represented by base data and metadata stored in a relational database. SODA also enables enriching the database schema with ontologies as well as with external sources such as DBpedia. In terms of implementation, SODA leverages two inverted indexes. One index is created on the base data of the underlying database. The other index is created on the metadata of the database as well as additional ontologies.

A system such as SODA sits on top of a relational database and cannot directly use the INEX test collection, which is in XML form. Hence, we need to transform the original INEX collection to conform to a relational database schema. The goal is to design a suitable relational model to store all the data from the XML files, making experiments thus comparable. After loading into the database, the data from the INEX collection is in semi-structured form: the records are fitted to the schema, but contain natural language text in most fields.

The relational model is relatively simple: The two main categories "movie" and "person" become the main tables. Both tables are linked to two auxiliary tables using the entity-attribute-value pattern (Dinu, Nadkarni, 2007). The two main tables are linked through roles. Finally, two extra tables for "links" and "biographies" were needed since they contain unique attributes. For a simplified entity-relationship diagram of this model, containing a total of ten tables, see Figure 1.

Ultimately, after running experiments on this new relational version of the test collection, we have to link back to the original XML files for scoring. To this end, all records in the tables hold information about their origin.

Note that there are several different database schema designs already available if one uses the internet movie database directly as a data source. A particular example is currently commonly used for query optimization benchmarks such as the Join Order Benchmark (Leis et al. 2015). The core reason why we cannot use these existing schemas and why we use our own transformation lies in the leveraging of the relevance assessments: we need a direct transformation path between the INEX version of the documents and the records in the relational database. This is only possible by starting with the exact XML version that was used in that campaign in 2011, and not with any data dumps sourced from IMDb directly. We thus use our own Python package to create SQL statements for all tables. We have made this package publicly available[2]. The package allows creating and populating the relational database, which is suitable for all experiments that use the INEX relevance assessments, as described in the remainder of the paper.

[2] https://github.zhaw.ch/SODA-IR-Ranking/nli-db-benchmark

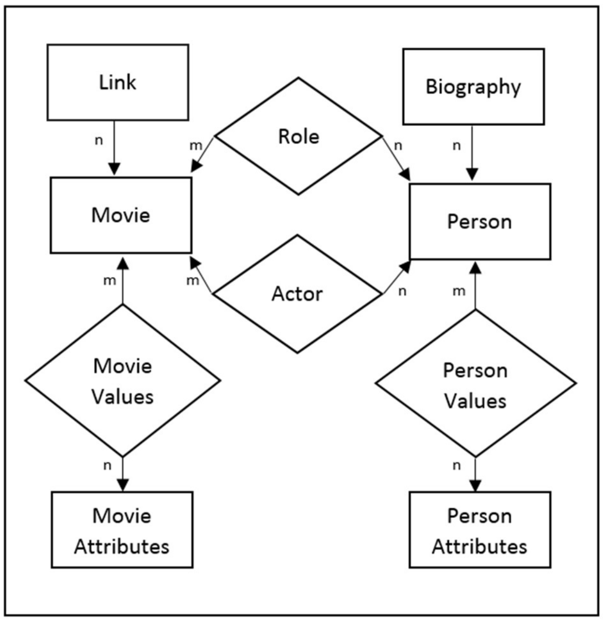

Figure 1: Entity-relationship diagram of data set.

3.3 Terrier IR System

The information retrieval-centric approach is built around the Terrier IR system[3]. Terrier allows direct indexing of XML documents if properly configured. In practice, all documents from the INEX Data-Centric Track were collected and enriched with some necessary additional tags.

For the indexing pipeline proper, we used a standard setup, employing both Terrier's own stopword list, as well as the Porter stemming algorithm (Porter 1980). The query set can be executed in batch mode (Macdonald and He, 2008).

3.4 Ranking Algorithms

While the two systems used in our experiments, SODA and Terrier, can both handle the same keyword-style queries, their approaches to retrieval and presentation of results is fundamentally different. SODA operates on top of a relational database system, and thus returns results as sets - all those items (documents) that "match" the query. There is no inherent ordering to this set, and in fact, the SQL standard makes no provision for how to order records by default. Terrier, being an information retrieval system, uses a ranked list paradigm: the system estimates the probability of relevance (score) for each item with respect to the information need that underlies the query. This score is then used to sort the result list. In the following, we introduce both retrieval mechanisms in more detail.

3.4.1 SODA Retrieval Mechanism

SODA operates by transforming a keyword query to one or more SQL statements. The following five steps are executed:

- *Lookup*: the input keywords are looked up in the inverted index to retrieve the entry points to the graph model. Multiple matches result in multiple entry points.
- *Rank and Top N*: Corresponding to the location where keywords were found in the graph, a score is provided. Multiple solutions are ranked and SODA returns the top N results.
- *Tables and Joins*: The tables holding keywords are identified as well as the relationships between them.
- *Filters*: names from columns where keywords were identified are applied as filters (WHERE-clause) for the required table.
- *SQL Generation*: All identified tables, relationships and applied filters are used to finally generate the executable SQL statement.

Finally, the output SQL statement(s) are executed on the respective database. The corresponding result lists are then returned.

3.4.2 Terrier Retrieval Mechanism

Terrier allows the configuration of different weighting schemes for retrieval, such as the commonly used tf-idf-Cosine method. In this basic scheme, tf describes the term frequency (number of occurrences of a term t in a document), a statistic local to a specific document, while the inverse document frequency idf (a weight inversely tied to the number of documents a term occurs in) is a statistic that considers the whole document collection N(number of documents in collection):

$$idf(t) = log\left(\frac{1+N}{1+d\ (t)}\right)$$

(where *df* is the document frequency, i.e. in how many documents a term appears). The relevance weight for a keyword t in a document D is calculated as follows:

$$tf.idf(t,D) = tf(t,D) * idf(t)$$

This is followed by a Cosine-length normalization of overall retrieval scores. Terrier also provides more advanced weighting schemes such as the widely used BM25 scheme (Walker et al. 1998). BM25 is derived from the probability ranking principle (Robertson, 1977). The weight for a term t in the document D is calculated as follows:

$$BM25(t,D) = \left(\frac{3 * tf(t,D)}{2(\frac{1}{4}+\frac{3}{4}\frac{l}{L}) + tf(t,D)}\right)$$

with l being the document length and L denoting the average document length.

4. Experiments

In this section we demonstrate how to use the adapted test collection to perform a direct comparison of a database-centric approach based on SODA with an information retrieval-centric approach based on Terrier. It is not the scope of the paper to exhaustively optimize these experiments. Rather, the description serves to illustrate some of the potential for analysis that is unlocked by having

[3] http://terrier.org/

a common benchmark for systems from these two different domains.

The main research questions that can be addressed by such experiments include:

- Given a set of natural language questions, is it better to use a database-centric or information retrieval-centric approach to answer the underlying information needs?
- Can we quantify which approach is better for which kind of query?
- How would we need to design a hybrid-approach that leverages the advantages of both approaches and minimizes the disadvantages of the individual approaches?

Many other research questions, e.g. pertaining to the optimization of the individual systems, could additionally be pursued.

4.1 Evaluation Metric

In order to gain comparable data points from both systems, the widely used evaluation measures precision and recall can be computed:

$$precision = \frac{\#relevant\ documents\ in\ result}{\#documents\ in\ result}$$

$$recall = \frac{\#relevant\ documents\ in\ result}{\#relevant\ documents\ in\ collection}$$

These are set-based measures, so computation for the output of SODA is straight-forward. Since SODA delivers multiple SQL statements as a possible transformation of a keyword query, we need a strategy to pick one set of values for each query. For further presentation of the results in this paper, the maximum values in each case were chosen. We thus present an upper bound on the performance that SODA could attain on the query set.

The respective values for the Terrier output were computed with the official INEX scoring tool, which is a version of the tool "trec_eval" used by the TREC campaign[4]. The output of Terrier is fundamentally different from SODA: in our configuration, Terrier always attempts to return up to 1,000 scored documents, with shorter lists only returned in cases where no positive scores for that many documents exist. In contrast to SODA, however, the documents in the result lists are ranked, i.e., users have full freedom to decide on their preferred result set size. For evaluation, we have decided to use *R-precision*. The value of this measure is obtained by truncating the result set at the point where precision and recall values are equal.

For future work, we are still actively pursuing the question of which measure(s) will maximize comparability between the set-based and rank list-based results. One additional direction we intend to explore is the truncation of Terrier's lists according to the number of results returned by SODA.

4.2 Experimental Setup

We executed all queries both on SODA and Terrier. The SODA output is a collection of SQL statements ranked by the score calculated by the default algorithm. The generated SQL statements were executed on the database to retrieve results including the file number.

Terrier returns a list of retrieved documents containing the ranked file numbers according to the computed score. For our experiments we used the BM25 score. Since Terrier is an IR system, we did not need to execute any SQL statements.

Finally, we compared the results of both systems against the relevance assessment to calculate precision and recall values. These calculations were performed by using the trec_eval tool for Terrier results and a Python implementation of the precision/recall formula for the SODA results.

4.3 Results

For our experiments we executed all 45 queries of the INEX test collection. However, it turns out that only 37 queries out of 45 contain relevant documents or records according to the relevance assessment. Hence, the analysis is restricted to those 37 queries.

Figure 2 shows the precision of all queries sorted by the precision obtained by SODA, while Figure 3 shows the precision sorted by the R-precision obtained by Terrier. By displaying the results from two different perspectives, we can more easily analyze the behavior of both systems.

Let us now analyze the results in more detail. For the 37 executed queries, Terrier scored precision values higher than zero in 21 cases (see Figure 3). The maximum precision score that Terrier reached is 0.92 indicating that the optimum of 1.0 was never achieved. On the other hand, SODA reached the maximum score of 1.0 in five cases. For 29 queries the score is above 0; however, for 14 of these it is below 0.01, and thus barely or not visible in the figure.

[4] https://trec.nist.gov/

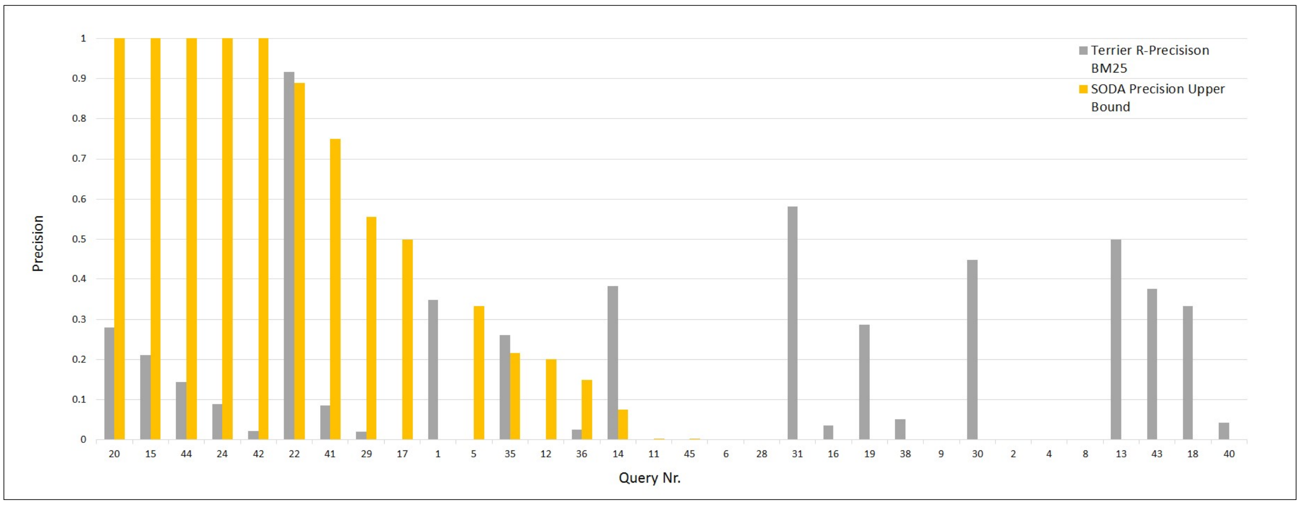

Figure 2: Precision, sorted by precision obtained by SODA.

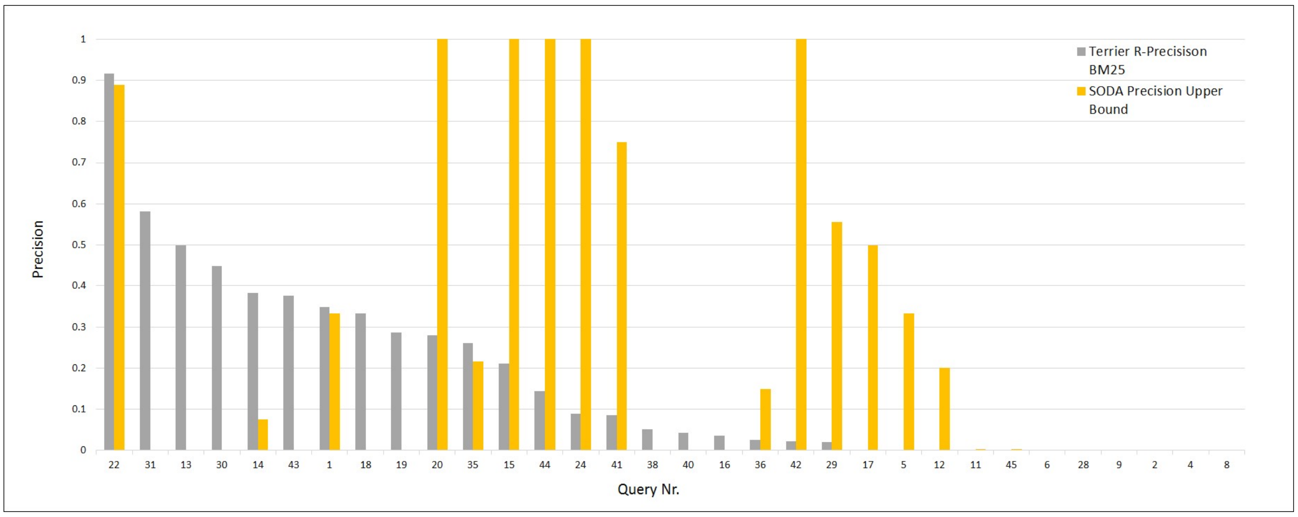

Figure 3: Precision, sorted by precision obtained by Terrier.

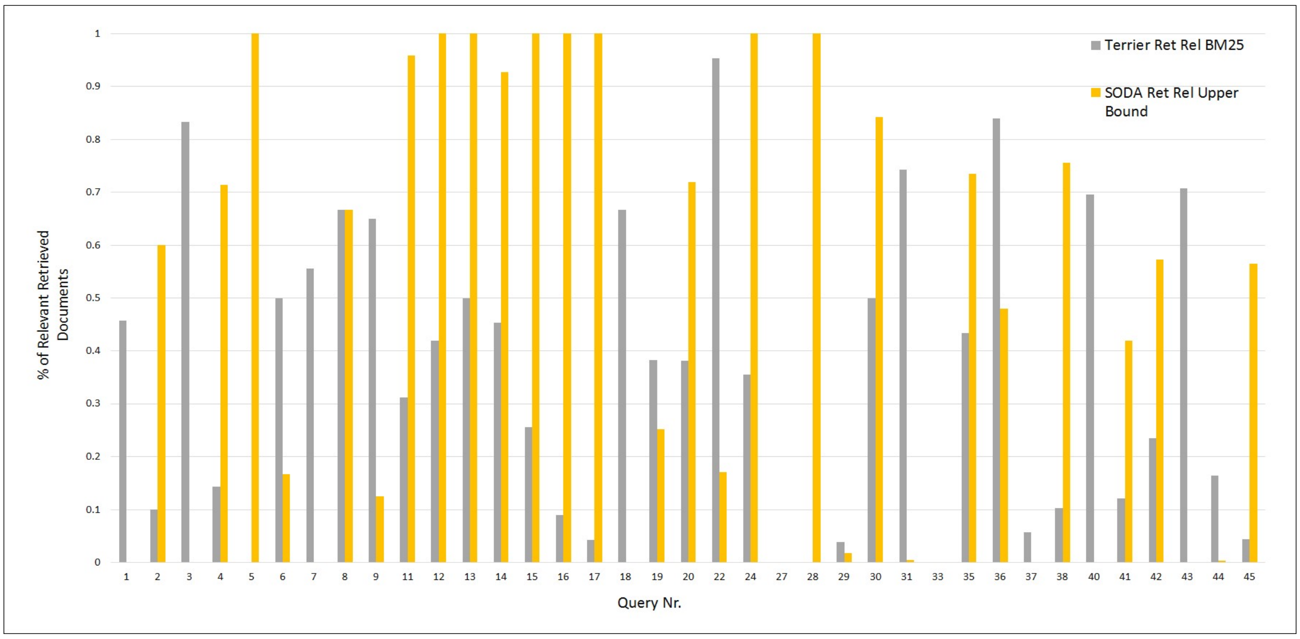

Figure 4: Recall of SODA and Terrier.

Note that SODA was not able to generate a SQL statement for query number 37 ("romance movies by Leonardo DiCaprio or Tom Cruise"). Therefore, we were not able to calculate the corresponding precision. For the same query, Terrier's R-precision is 0 as well while it finds only two out of 35 relevant documents (further down in the ranked list). For this query both systems show poor performance.

As shown in Figure 2, SODA reaches the optimum recall of 1.0 for 8 queries. For all of these queries, SODA generates simple *SELECT * FROM* statements which return the whole data set from the respective tables. For 16 queries SODA does not find any relevant documents at all. The maximum value scored by Terrier is 0.9532 and it does not find any relevant documents in 4 cases (see Figure 3).

For query 15 ("aliens usa") and 24 ("survive desert island") SODA scores the maximum for both values precision and recall. Terriers' best performing query is number 22 scoring 0.9532 for recall and 0.9164 for precision.

Let us now analyze the recall of both systems. As shown in Figure 4, retrieval results differ the most for query 5 ("king kong jack black") and 28 ("lovers arctic Spanish") when SODA finds all relevant documents while Terrier retrieves none of them. 7 queries perform better with Terrier as it is able to find documents whereas SODA retrieves zero relevant documents.

Overall, these measurements are consistent with our expectations considering the different result formats of the two systems: the set-based results of SODA, plus the fact that SODA generates multiple SQL commands and thus result sets, of which we chose the upper bound, favor precision. The ranked lists returned by Terrier conversely favor recall.

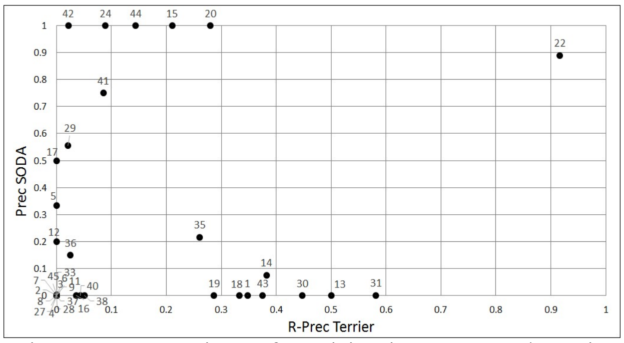

Figure 5: Comparison of precision in SODA and Terrier.

A different view on the results is obtained when contrasting the values in a scatterplot. The goal of this evaluation is to obtain a rough categorization of query performance: which queries perform equally on both systems vs. which queries have substantially different performance (see Figure 5). It will be part of future work to try to find a link between these performance categories and query characteristics, which may then either form the basis for optimizations of either system or the basis for a hybrid system that chooses the right approach (SQL on relational database or IR) query by query.

5. Conclusion

In this paper we show how to adapt an IR test collection to obtain a novel benchmark for evaluating systems that enable natural language querying over databases. The benchmark is targeted for comparing database-centric approaches that rely on a relational database schema to information retrieval approaches that operate on unstructured, textual data. As such, we are confident that it should inspire many additional research lines.

In order to demonstrate the feasibility of our approach, we evaluated SODA - using a database-centric approach - against Terrier - an information retrieval-centric approach. Preliminary analysis confirms our expectation that the two approaches inherently favor different types of information needs: SODA shows good results on queries that benefit from a precision-oriented, set-based result, whereas Terrier shows its main strengths when considering recall.

An exciting future research question is how to predict the user preference for either precision or recall from the query, and thus build a hybrid-approach that leverages the advantages of both worlds - databases and IR. The key to this end will be the identification of query characteristics, which will allow the selection of the appropriate retrieval strategy.

6. References

Affolter, K., Stockinger, K., & Bernstein, A., A Comparative Survey of Recent Natural Language Interfaces for Databases, VLDB Journal, 2019.

Baeza-Yates, R., & Ribeiro-Neto, B. (2011). Modern information retrieval (2nd Edition). New York: ACM press.

Basik, F., Hättasch, B., Ilkhechi, A., Usta, A., Ramaswamy, S., Utama, P., ... & Cetintemel, U. (2018, May). DBPal: A Learned NL-Interface for Databases. In Proceedings of the 2018 International Conference on Management of Data (pp. 1765-1768). ACM.

Bast, H., & Haussmann, E. (2015, October). More accurate question answering on freebase. In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (pp. 1431-1440). ACM.

Blunschi, L., Jossen, C., Kossmann, D., Mori, M., & Stockinger, K. (2012). Soda: Generating sql for business users. Proceedings of the VLDB Endowment, 5(10), 932-943.

Damljanovic, D., Agatonovic, M., & Cunningham, H. (2010, May). Natural language interfaces to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction. In Extended Semantic Web Conference (pp. 106-120). Springer, Berlin, Heidelberg.

Dinu, V., & Nadkarni, P. (2007). Guidelines for the effective use of entity–attribute–value modeling for biomedical databases. International journal of medical informatics, 76(11-12), 769-779.

Ferré, S. (2017). Sparklis: an expressive query builder for SPARQL endpoints with guidance in natural language. Semantic Web, 8(3), 405-418.

Iyer, S., Konstas, I., Cheung, A., Krishnamurthy, J., & Zettlemoyer, L. (2017, April). Learning a Neural Semantic

Parser from User Feedback. In 55th Annual Meeting of the Association for Computational Linguistics.

Leis, V., Gubichev, A., Mirchev, A., Boncz, P., Kemper, A., & Neumann, T. (2015). How good are query optimizers, really?. Proceedings of the VLDB Endowment, 9(3), 204-215.

Li, F., & Jagadish, H. V. (2014). Constructing an interactive natural language interface for relational databases. Proceedings of the VLDB Endowment, 8(1), 73-84.

Macdonald, C., & He, B. (2008). Researching and Building IR applications using Terrier. In European Conference on Information Retrieval.

Porter, M. F. (1980). An algorithm for suffix stripping. Program, 14(3), 130-137.

Saha, D., Floratou, A., Sankaranarayanan, K., Minhas, U. F., Mittal, A. R., & Özcan, F. (2016). ATHENA: an ontology-driven system for natural language querying over relational data stores. Proceedings of the VLDB Endowment, 9(12), 1209-1220.

Simitsis, A., Koutrika, G., & Ioannidis, Y. (2008). Précis: from unstructured keywords as queries to structured databases as answers. The VLDB Journal—The International Journal on Very Large Data Bases, 17(1), 117-149.

Song, D., Schilder, F., Smiley, C., Brew, C., Zielund, T., Bretz, H., ... & Harrison, J. (2015, October). TR discover: A natural language interface for querying and analyzing interlinked datasets. In International Semantic Web Conference (pp. 21-37). Springer, Cham.

Voorhees, E. M. (2001, September). The philosophy of information retrieval evaluation. In Workshop of the cross-language evaluation forum for european languages (pp. 355-370). Springer, Berlin, Heidelberg.

Walker, S., Robertson, S. E., Boughanem, M., Jones, G. J., & Jones, K. S. (1997, November). Okapi at TREC-6 Automatic ad hoc, VLC, routing, filtering and QSDR. In TREC (pp. 125-136).

Wang, Q., Ramírez, G., Marx, M., Theobald, M., & Kamps, J. (2011, December). Overview of the inex 2011 data-centric track. In International Workshop of the Initiative for the Evaluation of XML Retrieval (pp. 118-137). Springer, Berlin, Heidelberg.

Zheng, W., Cheng, H., Zou, L., Yu, J. X., & Zhao, K. (2017, November). Natural language question/answering: let users talk with the knowledge graph. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (pp. 217-226). ACM.