FreeQ: An Interactive Query Interface for Freebase *

Elena Demidova
L3S Research Center
Leibniz University of Hanover
Appelstr. 9a
30165 Hanover, Germany
demidova@L3S.de

Xuan Zhou
DEKE Lab, MOE
Renmin University of China
Zhongguancun Street 59
10087 Beijing, China
xuan.zhou.mail@gmail.com

Wolfgang Nejdl
L3S Research Center
Leibniz University of Hanover
Appelstr. 9a
30165 Hanover, Germany
nejdl@L3S.de

ABSTRACT

Freebase is a large-scale open-world database where users collaboratively create and structure content over an open platform. Keyword queries over Freebase are notoriously ambiguous due to the size and the complexity of the dataset. To this end, novel techniques are required to enable naive users to express their informational needs and retrieve the desired data. FreeQ offers users an interactive interface for incremental query construction over a large-scale dataset, so that the users can find desired information quickly and accurately.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval - Query formulation

Keywords

Freebase, query construction, database keyword search

1. INTRODUCTION

Freebase [5] is a large-scale open-world database where users collaboratively create, structure and maintain content over an open platform. Freebase data comes from a large number of high-quality open data sources, such as Wikipedia, MusicBrainz [2], and others. The data size of Freebase is increased continuously using algorithmic data imports, data extraction, and other techniques. At the time of writing, Freebase contained about 22 millions entities and more than 350 millions facts in about 100 domains[1]. Freebase data is organized as a relational database with more than 7500 tables, mostly containing textual data.

Keyword queries over Freebase are notoriously ambiguous due to the size and the complexity of the dataset. Entity-oriented keyword queries exposed by the current search interface of Freebase can retrieve heterogeneous results from different domains. For example, the query *"Tom Hanks"* can retrieve entities coming from *person*, *film*, *book*, *tv* and *music* domains. The results would include the actor Tom Hanks, the person Thomas C. Hanks, the book entitled "Tom Hanks", and others. As there could be a large number of such results, the user may have to scan through a long list to find the desired entities, without any possibility to refine the result set. Moreover, this interface can only be used to retrieve individual entities. It is unable to retrieve complex results, including collective results, e.g. *"All films acted by Tom Hanks"*, or results involving more than one entity, e.g. *"The role of Tom Hanks in the film The Terminal"*. These complex results have to be retrieved using structured queries.

As an alternative, Freebase offers a structured query language named MQL, through which skilled users can manually create structured queries on the dataset. A MQL query can span over multiple tables and contain several predicates. The following example illustrates a MQL query, which aims to retrieve the role of Tom Hanks in the film *"The Terminal"*:

```
[{"!pd:/film/actor/film":  [{
                "name":  "Tom Hanks"
                "type":  "/film/actor"}],
        "film":  [{
                "name":  "The Terminal"
                "type":  "/film/film"}],
        "character":  {
                "name":  null,}
        "type":  "/film/performance"}]
```

The creation of an MQL query requires users to study the database schema to identify the relevant tables as well as the required attributes and join paths. This is not a trivial exercise even for a database expert, especially when the scale of the database schema is large. Furthermore, the composition of a MQL query is a time consuming task.

In this demonstration, we introduce FreeQ - an interactive query interface that enables a naive user to start with simple keywords and incrementally refine them into a structured query that can accurately express the user's informational need. Fig. 1 presents the graphical user interface of FreeQ. The interface is composed of four parts: (1) an input field for keyword queries, (2) a query construction panel, (3) a query window for presenting structured queries, and (4) a result window for query results.

When a user Alice issues a keyword query, e.g. *"Tom Hanks Terminal"*, FreeQ tries to guess the user's intent and generates the top-k most likely structured queries in the query window. If a query in the top-k list satisfies the Alice's informational need, she can click the query to obtain the query results in the result window. If no query in the query window makes sense to Alice, she can choose some

*A demonstration of the FreeQ system is available online at: `http://iqp.l3s.uni-hannover.de`

[1]http://activity.freebaseapps.com/freebase

Copyright is held by the International World Wide Web Conference Committee (IW3C2). Distribution of these papers is limited to classroom use, and personal use by others.
WWW 2012 Companion, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1230-1/12/04.

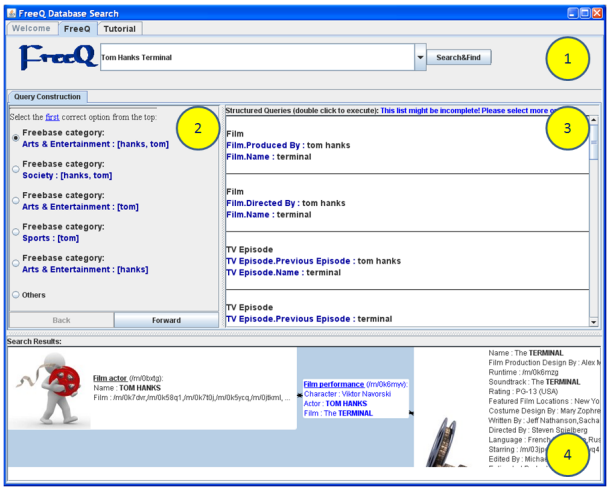

Figure 1: FreeQ GUI. The components of the FreeQ GUI include: (1) an input field for keyword query, (2) interaction options, (3) top-k structured queries, and (4) query results[1].

interaction options in the query construction panel (2) to further clarify her intent. The interaction options are simple questions, such as *"Is Tom Hanks a person?"* and *"Is Terminal an entity in the film domain?"*. Whenever Alice clicks on an interaction option that she regards as true, the interface is refreshed to present only the structured queries, query results and further interactive items that comply with the Alice's selection. Through the interaction process, the system becomes more certain about the Alice's intent. The interaction process continues until Alice identifies the intended structured query and obtains the desired information. As Freebase is very heterogeneous and complex, it is usually difficult for FreeQ to identify the right structured query immediately. Therefore, the interactive query construction process plays the most crucial role in the interface.

The interface of incremental query construction was first introduced in [8, 7]. The systems described in [8, 7] worked well for small and medium databases, such as Internet Movie Database [1] and Lyrics [14], which contain around 20 tables. However, they do not scale on Freebase, which is composed of several thousands of tables. On the one hand, the interpretation space of a keyword query over Freebase is usually too big to materialize. On the other hand, the interactive items generated by the previous approaches are not informative enough to enable efficient query construction on Freebase. In contrast, FreeQ is a highly scalable system. It applies a set of new techniques to handle the heterogeneity and complexity of the Freebase dataset. In this demonstration we briefly describe the underlying technology of FreeQ and demonstrate how FreeQ can support users in information seeking on Freebase.

[1]Fig.1 includes the following images: http://de.fotolia.com/id/10056489 ©ioannis kounadeas - Fotolia.com; http://de.fotolia.com/id/9974098 ©XYZproject - Fotolia.com

2. QUERY CONSTRUCTION

The main function of FreeQ is to translate a user's keyword query into the intended structured query. We call such translation *query interpretation*. In a query interpretation, each keyword in the initial keyword query is interpreted either as a constant in a predicate or as a name of the schema element, such as a table name or an attribute name. For instance, a user looking for all the films played by Tom Hanks can issue a keyword query *"Tom Hanks film"*, which can be interpreted as:

structure: $actor_1 \bowtie acts \bowtie film_1$,
predicates: *"Tom Hanks"* $\in actor_1.name$

In this interpretation, *"Tom Hanks"* is mapped to a constant of a predicate, and *"film"* is mapped to a table name. The structured queries generated by FreeQ support the operators of selection, projection and join. The results of a structured query can be obtained using a traditional SPJ execution plan.

The aim of Freebase is to create a global structured resource for human's common knowledge. It is of high heterogeneity and complexity. Usually, a simple keyword, such as *"film"*, has hundreds of occurrences on the Freebase schema and data, and thus can be interpreted in hundreds of ways. Therefore, the interpretation space of a keyword query on Freebase can be extremely large and simple ranking may often fail to find the intended query interpretation immediately. The query construction panel of the FreeQ interface empowers users to incrementally construct the desired query interpretation in these cases.

The conceptual process of incremental query construction can be modeled as follows:

1. Given a database whose schema is $G = (V, E)$, a user issues a keyword query $K = \{k_1, k_2, ..., k_n\}$.
2. The system generates all possible query interpretations of K based on G. We call this set of query interpretations the *interpretation space* of K, denoted by ζ.
3. The system generates an interaction option IO and presents it to the user.
4. If the user accepts IO, then the system removes all the query interpretations that cannot imply IO from ζ. Otherwise, the system removes all the interpretations that imply IO from ζ.
5. The system retrieves the top-k query interpretations from ζ, and presents them to the user.
6. If the user finds the intended query interpretation from the top-k, the query construction process terminates. Otherwise, the process continues with Step 3.

As we can see, as the interaction goes on, the interpretation space ζ of the keyword query shrinks gradually. When the interpretation space becomes sufficiently small, it is easy for the user to find the desired query in the query window.

2.1 Generation of Interaction Options

In each cycle of query construction, FreeQ needs to select a few interaction options for the user to interactively construct the intended structured queries. To ensure efficiency, it is essential to select the interaction options that

can reveal as much information about the user's intent as possible. In other words, the interaction options with the highest information gain need to be identified.

The previous approaches of incremental query construction [7] use the sub-structures of the query interpretations as interaction options. As these sub-interpretations are usually not informative enough in the context of a large-scale database such as Freebase, they make the query construction process very slow. To achieve efficient query construction, FreeQ utilizes the hierarchical conceptual models on top of the database schema to generate interaction options that yield high information gain.

For example, suppose a user issues a keyword query "Hanks Character". The possible interaction options that can be presented to the user include:

- "$Hanks$" $\in$ *actor.name* (indicating that "Hanks" is an actor's name)
- "$Hanks$" $\in$ *author.name* (indicating that "Hanks" is a book author's name)
- "$Hanks$" $\in$ *domain* : $film$ (indicating that "Hanks" appears in the domain of film)
- etc.

It is obvious that the first two options, "$Hanks$" $\in$ *actor.name* and "$Hanks$" $\in$ *author.name*, can be too specific to reveal much information about the user's intent. Once the user rejects these two options, there are still a large number of possible interpretations remained. In contrast, "$Hanks$" $\in$ *domain* : $film$ appear to be more general and a better interaction option. If the user accepts this option, we can zoom the interpretation space into the domain of film. If the user rejects the option, we can remove the domain of film from the interpretation space completely. In both ways, we can shrink the interpretation space significantly. In essence, "$Hanks$" $\in$ *domain* : $film$ offers much higher information gain than the other two options do.

To obtain such informative interaction options as "$Hanks$" $\in$ *domain* : $film$, we need to summarize the database schema using a set of abstract concepts such as "film", "book", etc. To achieve this, FreeQ extends the Freebase schema with the domain hierarchy of Freebase. Besides domain specific interaction options, FreeQ also provides some other types of general interaction options based on hierarchical ontologies, e.g. "$Hanks$" $\in$ *person.name*, which will be enumerated in our demonstration. To this end, FreeQ makes use of some external ontologies, such as WordNet [9] and YAGO [17].

In the conceptual query construction process introduced previously, we need first to materialize the complete interpretation space, before we can obtain the top-k query interpretations and estimate the information gain of the interaction options. However, this materialization is infeasible for a large-scale database like Freebase. The interpretation space of a simple two-keyword query on Freebase can already contain millions of interpretations, which can consume all the main memory. To achieve scalability, FreeQ makes use of a query hierarchy, which was first introduced in [7]. The hierarchy grows incrementally during the interaction process with the user and materializes only the most probable query construction options that comply with the user's selection in each step. The query construction options and complete queries can be generated through a bread-first-search and a depth-first-search algorithms on the top of the query hierarchy respectively.

3. IMPLEMENTATION

FreeQ is implemented as a client-server Java application. The user interface of FreeQ runs as a Java applet in a browser, whereas the database search is performed on the server side.

In order to deploy the FreeQ system on Freebase, we made use of the data dumps provided by Freebase in June 2011 [10]. This dataset includes approximately 7500 tables, which contain more than 20 million entities in about 100 domains. To facilitate our query construction algorithms, we imported the data into a MySQL database system, and performed additional pre-processing, such as foreign key definition and data indexing.

3.1 Foreign Key Definition

As the Freebase data dumps do not directly expose any foreign key definitions, we query Freebase API to retrieve additional information. Each entity table in Freebase possesses a name column, which can be regarded as the identifier for the entities in the table. If an attribute of a Freebase table is associated with an expected type that points to another table, it indicates that this attribute is a foreign key. For example, attribute $film.written\_by$ has an expected type $writer$. In this case we establish a foreign key relationship between the $film.written\_by$ and $writer.name$ attributes.

3.2 Data Indexing

To identify occurrences of keywords in the database attributes, database search systems create inverted indexes. To this end, FreeQ makes use of a Lucene index [15].

The indexes in various systems differ in granularity of the index items. Some systems view the whole content of a database attribute as a single document [18], others perform indexing at the value and tuple levels. This primary design decision affects availability of statistics that can be retrieved from an index as well as the size and efficiency of the index.

FreeQ chooses to index each column (the whole contents of each attribute) as a single document. This is the least costly alternative in terms of the index size and efficiency, as the maximal length of the posting list in an inverted index is restricted by the total number of attributes in the database. From this index we can retrieve term frequency of the keyword in an attribute as well as approximate statistics of phrase occurrences.

4. DEMONSTRATION OVERVIEW

The aim of FreeQ is to assist users in defining structured queries over Freebase by starting from simple keywords. In our demonstration we will primarily show how FreeQ works and how users can use it to obtain desired information from the Freebase dataset, without a-priori knowledge of the database schema and the dedicated structured query language. First, we will demonstrate the complete query process. This process starts from submitting a keyword query to the system, followed by a presentation of the top-k structured queries that give different interpretations to the keywords, followed by an execution of the queries to retrieve search results. Second, we will show how the interaction scheme provided by FreeQ can guide users to quickly construct desired structured queries.

Our audience can take a chance to try the FreeQ interface. To highlight the advantages of our approach, we will ask our audience to perform query construction using the FreeQ interface as well as the original search interfaces of Freebase. The original interfaces of Freebase include the entity-oriented keyword search interface and the MQL (Metaweb Query Language[2])interface - a structured query interface. Through the comparison, the audience can get some hand-on experience about information seeking problems on structured data. We will provide different information seeking tasks for the audience to try.

5. RELATED WORK

Recently the problem of structured query generation from user's keywords have received a lot of attention [7, 13, 18, 19]. Initial approaches to incremental query construction included a ranking approach called SUITS [8], and the probabilistic model of IQP [6, 7]. These interfaces can efficiently assist users to construct queries over small and medium database schemas, such as the Internet Movie Database [1] and the Lyrics Database [14], which contain less than 20 tables. However, the previous approaches do not scale well on a large and heterogeneous database such as Freebase. This is because the previously developed methods relied only on the database internal statistics and internal schemas to generate query construction options. FreeQ is the first system that enables incremental query construction over large databases with similar scale as Freebase. In FreeQ, new techniques are used to scale up the query construction algorithms. FreeQ alleviates the scalability problem by integration more general interaction options that offer high information gain.

Database usability is a long-term research issue [11]. In this context, Natural Language Query Interfaces, e.g. [4] are designed to enable users to specify structured queries in a human language. Other techniques such as query auto-completion [16] assist users to form structured queries by suggesting possible structures or terms based on the already entered user's sub-query. Forms are the most typical interface used to query information in a database through a pre-defined template, which can be optionally modified [12]. FreeQ offers similar expressivity as the interfaces mentioned above, but with much more flexibility. It enables users to start with arbitrary keywords and identifies the intended structured query efficiently using a small number of meaningful interaction items.

Faceted search engines, e.g. Yippy [3], organize search results into meaningful groups, called facets to enable users shrinking the scope of the search by focusing on a small number of facets. The interface of FreeQ is similar to a faceted interface, whereas each facet corresponds to a query construction option.

6. ACKNOWLEDGMENT

The authors would like to thank Irina Oelze for supporting implementation of the system. This work is partly funded by the National Basic Research Program of China (973 Program) Project No. 2012CB316205 and by the European Commission under the grant agreements 270239 (ARCOMEM) and 231126 (LivingKnowledge).

[2]http://wiki.freebase.com/wiki/MQL

7. REFERENCES

[1] The internet movie database. www.imdb.com.
[2] Musicbrainz - the open music encyclopedia. http://musicbrainz.org.
[3] Yippy search engine. http://search.yippy.com.
[4] M. J. Al-Muhammed and D. W. Embley. Ontology-based constraint recognition for free-form service requests. In *Proc. of the ICDE 2007*, Los Alamitos, CA, USA, 2007. IEEE Computer Society.
[5] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proc. of the SIGMOD 2008*, pages 1247–1250, New York, NY, USA, 2008. ACM.
[6] E. Demidova, X. Zhou, and W. Nejdl. Iq^P: Incremental query construction, a probabilistic approach. In *Proc. of the ICDE 2010, Long Beach, California, USA, March 1-6, 2010*. IEEE Computer Society, 2010.
[7] E. Demidova, X. Zhou, and W. Nejdl. A probabilistic scheme for keyword-based incremental query construction. *IEEE Trans. Knowl. Data Eng.*, 24(3):426–439, 2012.
[8] E. Demidova, X. Zhou, G. Zenz, and W. Nejdl. Suits: Faceted user interface for constructing structured queries from keywords. In *Proc. of the DASFAA 2009*, Berlin, Heidelberg, 2009. Springer-Verlag.
[9] e. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, 1998.
[10] Google. Freebase data dumps. http://download.freebase.com/datadumps/, 2011.
[11] H. V. Jagadish, A. Chapman, A. Elkiss, M. Jayapandian, Y. Li, A. Nandi, and C. Yu. Making database systems usable. In *Proc. of the SIGMOD 2007*, pages 13–24, New York, NY, USA, 2007. ACM.
[12] M. Jayapandian and H. V. Jagadish. Expressive query specification through form customization. In *Proc. of the EDBT 2008*, New York, NY, USA, 2008. ACM.
[13] E. Kandogan, R. Krishnamurthy, S. Raghavan, S. Vaithyanathan, and H. Zhu. Avatar semantic search: a database approach to information retrieval. In *Proc. of the SIGMOD 2006*, pages 790–792, New York, NY, USA, 2006. ACM.
[14] F. Liu, C. Yu, W. Meng, and A. Chowdhury. Effective keyword search in relational databases. In *Proc. of the SIGMOD 2006*, New York, NY, USA, 2006. ACM.
[15] M. McCandless, E. Hatcher, and O. Gospodnetić. *Lucene in Action, Second Edition*. Manning Publications Co., 2009.
[16] A. Nandi and H. V. Jagadish. Assisted querying using instant-response interfaces. In *Proc. of the SIGMOD 2007*, New York, NY, USA, 2007. ACM.
[17] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A Core of Semantic Knowledge. In *Proc. of the WWW 2007*, New York, NY, USA, 2007. ACM Press.
[18] E. Tata and G. M. Lohman. Sqak: doing more with keywords. In *Proc. of the SIGMOD 2008*, pages 889–902, New York, NY, USA, 2008. ACM.
[19] Q. Zhou, C. Wang, M. Xiong, H. Wang, and Y. Yu. Spark: adapting keyword query to semantic search. In *Proc. of the ISWC 2007*. Springer-Verlag, 2007.