Easy Access to the Freebase Dataset

Hannah Bast, Florian Bäurle, Björn Buchhold, Elmar Haußmann
Department of Computer Science
University of Freiburg
79110 Freiburg, Germany
{bast, baeurlef, buchhold, haussmann}@informatik.uni-freiburg.de

ABSTRACT

We demonstrate a system for fast and intuitive exploration of the Freebase dataset. This required solving several non-trivial problems, including: entity scores for proper ranking and name disambiguation, a unique meaningful name for every entity and every type, extraction of canonical binary relations from multi-way relations (which in Freebase are modeled via so-called mediator objects), computing the transitive hull of selected relations, and identifying and merging duplicates. Our contribution is two-fold. First, we provide for download an up-to-date version of the Freebase data, enriched and simplified as just sketched. Second, we offer a user interface for exploring and searching this data set. The data set, the user interface and a demo video are available from `http://freebase-easy.cs.uni-freiburg.de`.

Categories and Subject Descriptors

H.0 [**Information Systems**]: General

Keywords

Freebase; Knowledge Base; Ontology

1. INTRODUCTION

Freebase [2] is designed as an open, community-curated knowledge base. With more than 40 million topics and over 2 billion facts, it is today by far the most comprehensive publicly available source of general-knowledge facts.

The complete Freebase data is available for free use, sharing, and adaption (even commercially) under a creative commons license. The data format is N-Triples RDF, which is standard for triple data. In principle, the data can therefore be loaded into any state-of-the-art triple store and queried via standard semantic query languages such as SPARQL. Freebase also provides an own API. The query language used there is MQL.

However, when working with this raw data via SPARQL or with the Freebase API via MQL, several major usability issues arise, also for expert users. Consider the query for winners of the Palme d'Or[1], shown in Figure 1. This appears to be a simple query, which requires only a single relation. In SPARQL one would like to write something like this:

```
select ?x where { ?x Awards-Won "Palme d'Or" }
```

But the required SPARQL query on the provided RDF data dump looks like this:

```
select ?name where {
  ?x  ns:award/award_winner/awards_won  ?m  .
  ?m  ns:award/award_honor/award  ?a  .
  ?a  ns:type/object/name  "Palme d'Or"@en  .
  ?x  ns:type/object/name  ?name  .
}
```

Already this simple example hints at a number of usability issues. How to guess the right relation names? How to guess the right schema (the object of the *awards_won* relation is a so-called mediator object, which is linked, via another relation, to the actual award entity)? How to guess the right entity names (Palme d'Or in this case)? The results are opaque, too. Here is the link to the result for the equivalent MQL query (the complexity of which is similar to that of the SPARQL query above): `http://tinyurl.com/l2pdms5`. In particular, the ranking is merely lexicographic and there are ambiguous names like *Michael Moore*. For more complex queries, e.g. `http://tinyurl.com/qzrc77j`, these problems intensify.

In contrast, the query in Figure 1 is as one would expect. As we will see later, the user interface helps in finding the proper relation names. The results are properly ranked, with the most prominent hits (directors in this case) at the top. The names of the directors are as expected, and accompanied by pictures. What is not shown is that there are 16 persons in Freebase with the name *Michael Moore*. In our version of the Freebase data set, only the (in)famous director gets exactly that name. The others are disambiguated by meaningful suffixes, e.g. *Michael Moore (Soccer Forward)*. Finally, the user interface offers suggestions for sensible ways to augment the query, e.g. by the relation *Country of nationality*.

1.1 Our contribution

We address all of the problems from the example query above, as well as several other problems that occur with typical queries and impact usability.

Entity Scores. As in standard text search, long result lists demand for a proper ranking. For example, for our example query above, we would like to have the most prominent

WWW'14, April 7–11, 2014, Seoul, Korea.
ACM 978-1-4503-2744-2/14/04.
http://dx.doi.org/10.1145/2567948.2577016.

[1]This is the highest prize at the annual Cannes Film Festival.

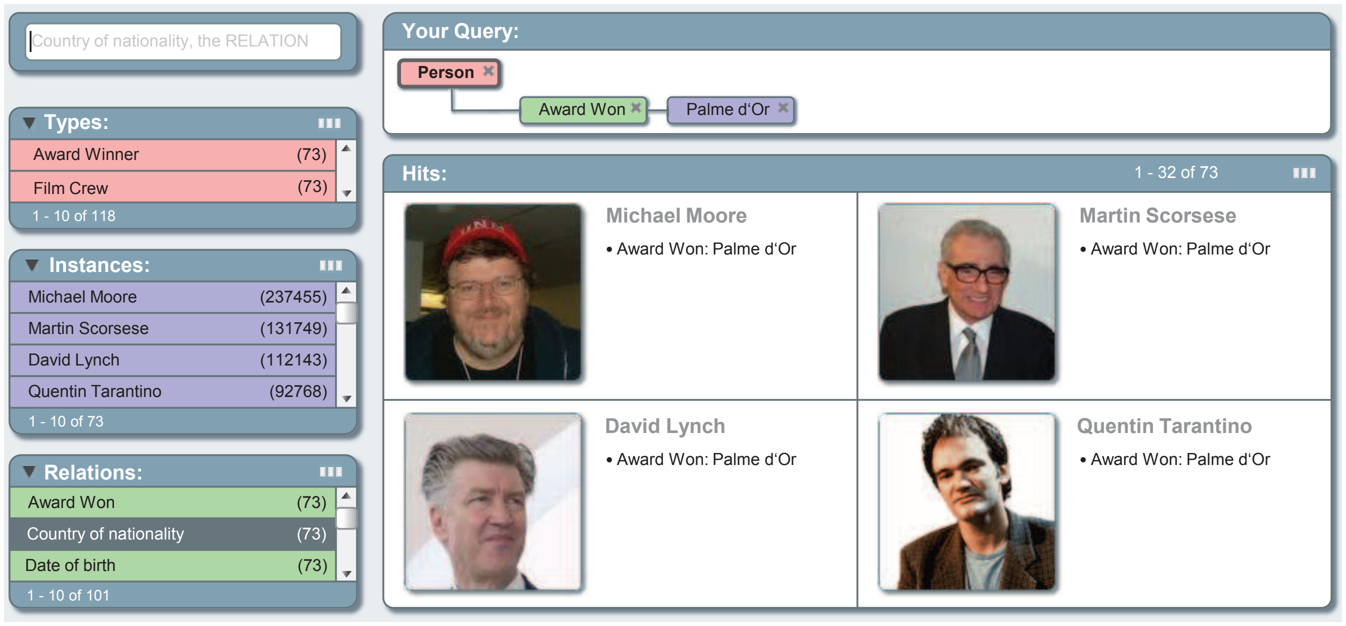

Figure 1: A screenshot of our demo system showing results for a query for winners of the Palme d'Or. For explanations of the various components and features, see the paragraph before Section 1.1 and Section 3.

people at the top. We provide a prominence score for each entity in Freebase; see Section 2.2.

Entity Names. In Freebase, each entity has a unique alpha-numerical so-called machine id or mid, e.g. */m/0jw67*. In most applications, it is desirable to also have unique names that are meaningful for humans. This is also the approach Wikipedia takes. There, entities are distinguished with suffixes. For example, *Europe* denotes the continent as expected, while the Swedish rock band with the same name is called *Europe (band)*. For the sake of consistency, these suffixes follow several rules, but ultimately they are chosen by humans. We automatically compute such names for each entity in Freebase. This is described in Section 2.3.

Mediators. In our introductory example, we have encountered the complex *awards_won* relation. It involves a mediator object that itself is related to several entities, including not only the person who won the award and the award won, but also supplementary information like the date of the award and the winning work. Still, for many queries the "main" binary relation (between the person and the award in this case) is all that is needed, and would be much easier to use. We automatically extract this binary relation from each mediator; see in Section 2.4.

Transitivity. Many relations are practically unusable when they are not closed under transitivity. The relation *Contained by* between locations is a prominent example. We compute the transitive hull for several large (manually selected) relations from Freebase; see Section 2.5.

Duplicates. Duplicate entities or types with the same or a similar name are frequent in Freebase. For example, there are four classes called *Person* or *person*. Usually, additional types with the same name have few instances and are added as a user's mistake. The problem is aggravated by our own addition of types to the taxonomy; see the next item. We identify duplicates, merge them and give them a proper canonical name; see Section 2.6.

Taxonomy. Freebase by itself has a comparably shallow taxonomy (3,557 different types at the time of this writing) expressed via its *type/object/type* relation. However, many intuitive semantic classes like *plant* or *politician* are not types. Instead this information is available only via relations, e.g. *Profession*. We apply a set of configurable relations with objects that are to be included in the taxonomy. Our resulting taxonomy has a total of 21,042 different types. See `http://freebase-easy.cs.uni-freiburg.de` for more details.

User Interface. We provide a fully-functional user interface that allows for an interactive exploration and search using all of the features above. See Figure 1 and our description in Section 3. The demo is available under the link above; we encourage the reader to try it out.

Download. Along with the demo, we also provide our version of the Freebase data set, with all of the mentioned features, for download. A zip file (2.4 GB at the time of this writing) is available under the link above. Our data curation pipeline (see Section 2) is fully automized. This allows us to easily update the data set on a regular basis, and thus keep pace with the continuously growing Freebase data.

We remark that some of the items above represent major research challenges. For this demo paper, we apply comparably simple and straightforward solutions. However, as can be seen from the demo, these already go a long way towards an easier access to and better usability of the Freebase data.

1.2 Related Work

There is an abundance of work on providing more convenient front ends for semantic search. See [4] for a small survey, and the many papers citing that work. None of these achieve context-sensitive query suggestions at interactive speed as in our user interface (for a data set as large as Freebase); see also [1, Section 2].

Concerning our data curation pipeline, we do not claim particular novelty for any of the components. Our contri-

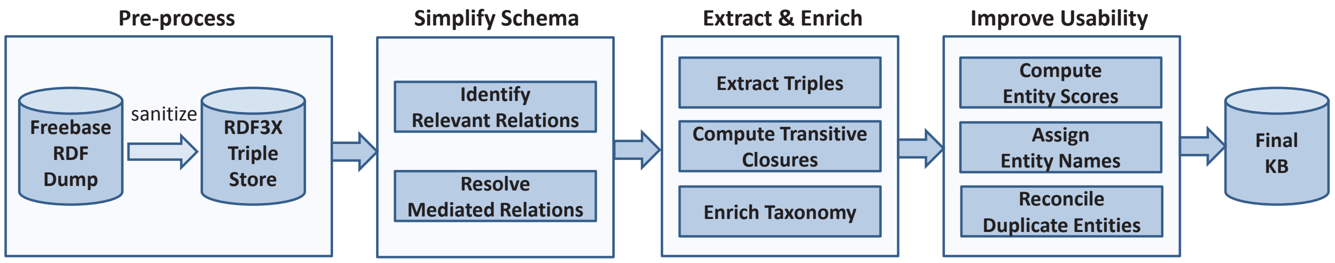

Figure 2: Architectural overview of our pipeline for a more easy-to-use version of the Freebase data set.

bution is that we have identified the major issues for the (widely used) Freebase data set, and provide a version that is much more easily accessible, and a ready-to-use demo application. We know of no comparable effort to date.

2. DESCRIPTION OF OUR PIPELINE

Freebase provides raw data dumps in the form of RDF-triples. As explained above, working with this raw data is complex for a variety of reasons. We therefore seek to simplify and enrich it in several ways. Figure 2 illustrates the general pipeline of our architecture. The various steps of the pipeline are described in the following subsections.

2.1 Data Sanitization

The raw RDF data contains redundant information as well as information which is undesired or even annoying in most use cases. We therefore first load the raw RDF data into RDF-3X [5], a fast triple store, and then extract the relations we are interested in using appropriate SPARQL queries. Namely, we omit relations with few facts (< 5) and relations that are not part of the core data in Freebase (e.g., facts in the domains *user* and *base*). Further, Freebase contains many (but not all) relations in two directions, e.g., *place-of-birth* and *people-born-here*. For all those, we only extract one direction (the one with more subjects than objects)[2].

2.2 Entity Scores

Scores indicating the prominence of entities are essential when ranking result entities (rank prominent entities first) and when resolving naming conflicts (assign the most prominent entity the canonical name, see Section 2.3). Intuitively, the more people talk (or write) about an entity the more prominent it is. We utilize the mentions of Freebase entities in the ClueWeb'12 Corpus[3] (733M web pages) from [3] to count the number of mentions of each entity and use it as a score. Given a set of mentions M_{CW_e} of an entity e we use s_{CW} as the resulting score:

$$s_{CW}(e) = |M_{CW_e}|$$

About 4.5 million distinct entities were recognized in ClueWeb, but our knowledge base contains a total of 39.6 million distinct entities. Therefore, we additionally compute a score based on the knowledge base and its relations in the following way:

[2]Most applications, including our own here, can handle queries for the reverse direction without requiring a copy of it.

[3]http://lemurproject.org/clueweb12/

$$\begin{aligned} s_{KB}(e) = &\sum_{r \in R} \log(\max(1, |\{x \mid (e, r, x) \in KB\}|)) \\ &+ \sum_{r \in R} \log(\max(1, |\{x \mid (x, r, e) \in KB\}|)) \end{aligned}$$

R is the set of all relation types in the knowledge base and KB denotes its set of relational triples (x, r, y). The above is the sum of the log of a per-relation out-degree and in-degree with the intuition that an entity with many incoming and outgoing relations is more prominent. The main effect of this score is as a tie-braker, when two entities have the same number of occurrences in the ClueWeb collections or were not mentioned or recognized at all. As final score for an entity we use the sum of the ClueWeb and knowledge based score:

$$s(e) = s_{CW}(e) + s_{KB}(e)$$

2.3 Entity names

As discussed in the introduction, a unique meaningful name for each entity is highly desirable in many applications. However, the raw Freebase data only provides alphanumerical ids and highly ambiguous names. In Wikipedia, this problem is solved manually as follows. For an ambiguous name, the most prominent entity gets the name without further additions. For example, the director from our example query in Figure 1 is called *Michael Moore*. Other contestants for the same name are distinguished by a meaningful suffix, e.g. *Michael Moore (Australian politician)*.

For the Freebase data, we automatically assign unique names as follows.[4] If there is no name at all, use the alphanumerical id from Freebase. Otherwise, there will be a set of candidates that compete for a name. Note that these candidates can be types (e.g. *Director*) as well as entities (e.g. *Michael Moore*) Also note that a type and an entity can have the same name in Freebase (e.g., there is a type *Person* and several entities with that name). The score for an entity is simply the score from Section 2.2. The score for a type is simply the maximum score of an entity plus the number of instances of that type. The literal name (without suffixes) then goes to the candidate with the highest score.

The remaining candidates are disambiguated as follows. If they are located in a country, they compete for the name *<name> (<country>)*. Again, the entity with the highest score gets that name. The others get an additional numerical suffix, e.g. *Berlin (United States) #2*. Entities without locational information are disambiguated using their *notable-for* relation, e.g. *Michael Moore (Soccer Forward)*. If that is not enough to achieve unique names, again a numerical suffix is

[4]Note that for most Freebase entries, there is no associated Wikipedia entry.

added. Entities that have neither locational nor *notable-for* information are disambiguated using their Freebase ids, e.g. *Maria (m/0760g8)*.

2.4 Mediators

As explained in the introduction, Freebase realizes multi-way relations using so-called mediator objects. For example, for a fact from Freebase's *Awards won* relation, the object is such a mediator object of type *award_honor*. This object is then related to the actual award, but also to supplementary information such as the winning work or the date of the award.

For each mediator type m (e.g. *award_honor*), we do the following. Intuitively, there are two types of mediators, which require a different approach. Namely, m either mediates between two entities in different roles (e.g. a musician and a group) or in the same role (e.g. siblings). We found the following strategy to differentiate very well between these two cases.

Consider the k relations that have m as subject.[5] Let $n_1 \geq \ldots \geq n_k$ be the number of facts in each of these k relations, sorted in decreasing order. Let r be the relation pertaining to n_1. If $k \geq 2$ and $n_2 \geq n_1/2$, let r' be the relation pertaining to n_2, otherwise let $r' = r$. Intuitively, r and r' are hence the most "frequent" relations, with $r = r'$ for a relation like "sibling". It remains to "merge" r and r' to the desired binary relation and give it a proper name.

Let n be the name of the reverse direction of the relation r according to Freebase. If no such name can be obtained, try the reverse relation of r'. In the very rare event that this fails too, fall back to $n = r$. We then extract a binary relation r_m in the following way.

$$r_m = \{(s, n, o) \mid (x, r, s) \in KB \land (x, r', o) \in KB \land s \neq o\}$$

This gives us exactly one binary relation for each mediator type m.

2.5 Transitivity

Freebase does not provide the transitive closure of transitive relations. Given R_1 and R_2, the tuples of two relations r_1 and r_2, we compute the transitive closure of tuples to be added during extraction as follows:

$$R_t = R_1 \circ R_2^+$$

Where R_2^+ is the transitive of relation r_2 and $\circ$ is relation composition. This allows computing the transitive closure over two relations, e.g., *profession* and *is-specialization-of* to ensure that a person with the profession *physicist* also has the profession *scientist* (because the profession *physicist* is a specialization of *scientist*). The classic transitive closure is a special case where r_1 equals r_2. We currently provide a manually compiled list of relations for which the transitive closure should be computed.

2.6 Duplicate Classes

A common problem in knowledge bases is that of duplicate entities or classes, often with identical or slightly different names. We follow a simple approach and merge two classes if they have the same name, ignoring case, and if the instances of one class are included in the other by a threshold. Let I_A and I_B be the instances/entities of some class A and B, respectively. We only merge class A into class B if the instances of class A are contained to at least 70% in class B, that is when:

$$\frac{|I_A \cap I_B|}{|I_A|} \geq 0.7$$

and vice versa.

[5]We always have $k \geq 1$ and for few relations, like "sibling", we indeed have $k = 1$.

3. USER INTERFACE

We provide a convenient user interface for performing complex searches on our version of the Freebase dataset, as described in the previous section. The main features are as follows. We encourage the reader to try our demo under `http://freebase-easy.cs.uni-freiburg.de`.

(1) A single input field, as in standard text search.

(2) Incremental query construction with suggestions (for matching types, instance and relations) after each keystroke.

(3) Example tooltips for each relation (shown on mouse over), to help understand what the relation is about (relation names in Freebase are sometimes opaque).

(4) Visual editing of the current query graph (e.g., removing a part or double-clicking a node to make it the new root).

(5) Meaningful names (following Section 2.3) and images (loaded from Freebase, if available).

(6) Proper ranking of results, using the scores from Section 2.2 where appropriate.

(7) Sort by an arbitrary query element, i.p. dates and values.

(8) Interactive query times, using the index from [1].

4. CONCLUSION

We provide a curated version of the Freebase data set that fixes several major usability issues with the original data set. We also provide a convenient user interface for interactive search and exploration, making good use of the various features we added. Several of the problems we addressed are major research problems in their own right. The solutions we provided here are simple and effective, yet by no means perfect. For example, our entity scores (derived from counts in the ClueWeb'12 corpus) work very well to bring the prominent entities to the top, but in some cases show an undesirable topic drift (e.g., Celine Dion is the fourth most prominent person). Our canonical entity names work like a charm for the more frequent entities, while names like *Berlin (United States) #2* could be improved.

5. REFERENCES

[1] H. Bast and B. Buchhold. An index for efficient semantic full-text search. In *CIKM*, pages 369–378, 2013.

[2] K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pages 1247–1250, 2008.

[3] E. Gabrilovich, M. Ringgaard, and A. Subramanya. FACC1: Freebase annotation of ClueWeb corpora, Version 1. (Release date 2013-06-26, Format version 1, Correction level 0).

[4] E. Kaufmann and A. Bernstein. How useful are natural language interfaces to the semantic web for casual end-users? In *ISWC*, pages 281–294, 2007.

[5] T. Neumann and G. Weikum. Scalable join processing on very large RDF graphs. In *SIGMOD*, pages 627–640, 2009.