What Are the Important Properties of an Entity? Studying the Knowledge Graph View

Ghislain A. Atemezing[1], Ahmad Assaf[1], Raphaël Troncy[1] and Elena Cabrio[1 2]

[1] EURECOM, Sophia Antipolis, France,
<atemezin@eurecom.fr>
[2] INRIA, France, <elena.cabrio@inria.fr>

Abstract. In knowledge bases and more precisely in the Web of Data, entities have a lot of properties. A quick view to the different versions of DBpedia can give an idea of the phenomenon. However, it is still difficult to decide which ones are important than others depending on how we want to use these entities, such as for a visualization of some basic facts about the given entity. In this paper, we perform reverse engineering on the Google Knowledge graph panel to find out what are the properties shown according to the type of the entity. We compare the results obtained with users surveyed on some Entities. The preliminary results are promising as they shape the path towards a recommendation tool for detecting the core properties important to Entities.

Keywords: Crowdsourcing, Google Knowledge panel, visualization, scrapping, knowledge elicitation, intrinsic properties

1 Introduction

{TODO: rewrite this**}** - 1) Motivation: in knowledge bases, entities have a lot of properties. Deciding which ones are more important than others depending on how we want to use these entities. Two use cases: . a) visualization of some basic facts about entities, for a multimedia QA system (QakisMedia) or for a second screen application (LinkedTV) . b) data integration (ontology matching), those properties having a bigger weights when computing alignments - 2) Approach 1: Google knowledge graph panel reverse engineering ... algorithms + first results - 3) Approach 2: User survey ... setup + results analysis - 4) Vocab for representing those "important" properties and dataset publication
{TODO: @summarize the work of Thomas, Michiel Hildebrand, etc**}**

2 Reverse Engineering the Google Knowledge Graph Panel

{TODO: @Ahmad to Run this will full dbpedia**}**

Web scraping is a technique to extract data from Web pages. For our purpose, we need to capture the properties information contained in the Google

Knowledge Panel (GKP) [**?**]. To do so, we have a create a Node.js application that that will get all the DBpedia concepts that have `owl:sameAs` links with Freebase. By doing so, we increase the probability that the search engine result page (SERP) will contain a GKP, since the underlying knowledge graph relies on Freebase as one of the data sources. Moreover, we filter out generic concepts by excluding those who are direct subclasses of $owl : Thing$. The SPARQL query gets back a total of 352 concepts[3].

Now, for each of these concepts we need to retrieve back n number of instances. For our experiment, n was equal to 100 random instances. For each of these instances, we need to issue a search query to Google containing the instance label. Google doesn't serve the GKP for all devices, so early scraping attempts failed as no GKP was present in the results. To overcome that, we had to mimic a browser behavior by setting the $user - Agent$ to that of compatible one. To check the existence of and extract data from a GKP, we use CSS selectors. An exemplary query selector is $._om$ (all elements with class name $\_om$), which returns the property DOM element(s) for the concept described in the GKP. From our experiments, we found out that we do not always get a GKP in a SERP. If this happens, we try to disambiguate our instance by issuing a new query with the concept type attached. However, if no GKP was found again, we capture that for manual inspection later. Listing 1 gives the high level algorithm for extracting the GKP.

Algorithm 1 Google Knowledge Panel reverse engineering Algorithm

```
INITIALIZE equivalentClasses(DBpedia, Freebase) AS vectorClasses
Upload vectorClasses for querying processing
Set n AS number-of-instances-to-query
for each conceptType ∈ vectorClasses do
    SELECT n instances
    listInstances ← SELECT-SPARQL(conceptType, n)
    for each instance ∈ listInstances do
        CALL http://www.google.com/search?q=instance
        if knowledgePanel exists then
            SCRAP GOOGLE KNOWLEDGE PANEL
        else
            CALL http://www.google.com/search?q=instance + conceptType
            SCRAP GOOGLE KNOWLEDGE PANEL
        end if
        gkpProperties ← GetData(DOM, EXIST(GKP))
    end for
    COMPUTE ocurrences for each prop ∈ gkpProperties
end for
return gkpProperties
```

[3] `http://goo.gl/EYuGm1`

3 Evaluation

We present our prelimimary evalutions by setting a user survey and the results of the extraction of the GKP. We then compare the properties shared by both settings and provide the agreement percentage for the concepts types used for the evaluation.

3.1 User Survey

We set up a survey[4] on the February 25th, 2014 for three weeks gathering the preferences of users in term of the properties they would like to be shown. We pick up nine entities per classes, namely TennisPlayer, Museum, Politician, Company, Country, City, Film, SoccerClub and Book. We received quite a good number of participation (152 in total), with almost 72% of users from academia, and 20% coming from the industry. Generally, 94% have heard about Semantic Web, and 35% of the surveyed were not familiar with visualization tools. The detailed results[5] presents for each question in the file, the properties ranked by percentage received. We decide the more important properties to be the ones receiving more than for 10% of the surveyed users. For example, users surveyed don't care much about showing the INSEE code of a city, while they will love to see mostly population, points of interest properties.

3.2 Comparison with the Knowledge Graph

We limit the properties with moore than 10% of answers from users surveyed. And on the Google Knowledge Panel (GKP) scrapping results, we just pick the first top N ocurrences[6] properties coming just after `label`, `type` and `properties`. These latter are called *by-default-properties* as they are always presented in more than 98% of the entities in the GKP. Table 1 presents for the 9 classes surveyed the agreement percentage with the GKP. The highest agreement with Museum(66.97%) while the lowest one for TennisPlayer (20%) concept.

Classes	TennisPlayer	Museum	Politician	Company	Country	City	Film	SoccerClub	Book
Agreement	20%	66.97%	50%	40%	60%	60%	60%	50%	60%

Table 1. Agreement on properties for 9 concept types, between users surveyed and Google Knowledge Panel.

With this set of 9 Concepts, We are covering 301, 189 entities of DBpedia that have keys in Freebase. And for each of them, we are empirically giving the most

[4] `http://eSurv.org?u=entityviz`

[5] `https://github.com/ahmadassaf/KBE/blob/master/results/agreement-gkp-users.csv`

[6] The results show that N can be 4, 5 or 6

important properties are likely to be found in the ones where there is agreement between one of the biggest knowledge base (Google) and users preferences.

4 Modeling the Preferred Properties with Fresnel

Fresnel[7] is a presentation vocabulary for displaying RDF data. It specifies *what* what information contained in an RDF graph should be presented, with the core concept `fresnel:Lens` [**?**]. We use Fresnel and PROV-O ontology[8] with the property prov:wasDerivedFrom to specify the source of the data.

```
:tennisPlayerGKPDefaultLens rdf:type fresnel:Lens ;
fresnel:purpose fresnel:defaultLens ;
fresnel:classLensDomain dbpedia-owl:TennisPlayer ;
fresnel:group :tennisPlayerGroup ;
fresnel:showProperties  (dbpedia-owl:abstract dbpedia-owl:birthDate
 dbpedia-owl:birthPlace dbpprop:height dbpprop:weight
 dbpprop:turnedpro dbpprop:siblings) ;
prov:wasDerivedFrom <http://www.google.com/insidesearch/features/search/knowledge.html>
```

5 Conclusion and Future Work

We have shown that it is possible to reveal important properties pf entities in a large knowlege base by starting comparing the properties obtained in the Knowledge Google Panel and the users preferences. We have provided an algorithm that for a given entity both in DBpedia and Freebase, compute the N ocurrences of the properties shown in the Google "infobox". We are sure these prelimary results can be benefit and helpful to decide which properties of an entity is worth for visualizing.

For future work, we would improve the vocabulary used to describe the results obtained and make them available on a SPARQL endpoint. We are also investigating the use of Mechanical Truck to perform the survey for the rest of the classes and provide a complete comprehense dataset with the results obtained with the classes of DBpedia.

Acknowledgments

This work has been partially supported by the French National Research Agency (ANR) within the Datalift Project, under grant number ANR-10-CORD-009. Elena is supported by the Labex project.

References

[7] `http://www.w3.org/2005/04/fresnel-info/`
[8] `http://www.w3.org/TR/prov-o/`