

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

Ghislain A. Atemezing¹, Ahmad Assaf¹, Raphaël Troncy¹ and Elena Cabrio¹²

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, scraping, 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

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 (GKB). To do so, we have to create a Node.js application that will

get all the DBpedia concepts that have *sameAs* links with Freebase. By doing so, we increase the probability that the search engine result page (SERP) will contain a GKB, 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.

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 GKB for all devices, so early scraping attempts failed as no GKB 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 GKB, 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 GKB. From our experiments, we found out that we do not always get a GKB 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 GKB was found again, we capture that for manual inspection later.

Algorithm 1 Google knowledge graph panel reverse engineering Algorithm

```

1: INITIALIZE equivalentClasses(DBpedia, Freebase) AS vectorClasses
2: Upload vectorClasses for querying processing
3: Set  $n$  AS number-of-instances-to-query
4: for each conceptType  $\in$  vectorClasses do
5:   SELECT  $n$  instances
6:   listInstances  $\leftarrow$  SELECT-SPARQL(conceptType,  $n$ )
7:   for each instance  $\in$  listInstances do
8:     CALL http://www.google.com/search?q=instance
9:     if knowledgePanel exists then
10:      SCRAP GOOGLE KNOWLEDGE PANEL
11:     else
12:      CALL http://www.google.com/search?q=instance+conceptType
13:      SCRAP GOOGLE KNOWLEDGE PANEL
14:     end if
15:     gkpProperties  $\leftarrow$  GetData(DOM, EXIST(GKP))
16:   end for
17:   COMPUTE occurrences for each prop  $\in$  gkpProperties
18: end for
19: return gkpProperties

```

3 Evaluation

3.1 User Survey

We set up a survey³ 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 results⁴ 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 occurrences 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 xx presents for the 9 classes surveyed the agreement percentage

4 Modeling the Preferred Properties with Fresnel

5 Conclusion and Future Work

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.
+Labex (Elena)

References

³ <http://eSurv.org?u=entityviz>

⁴ <https://github.com/ahmadassaf/KBE/blob/master/results/agreement-gkp-users.csv>