Retrieving attributes using Web tables

Arlind Kopliku
IRIT
Paul Sabatier University
Toulouse, France
Arlind.Kopliku@irit.fr

Karen Pinel-Sauvagnat
IRIT
Paul Sabatier University
Toulouse, France
Karen.Sauvagnat@irit.fr

Mohand Boughanem
IRIT
Paul Sabatier University
Toulouse, France
Mohand.Boughanem@irit.fr

ABSTRACT

In this paper we propose an attribute retrieval approach which extracts and ranks attributes from Web tables. We use simple heuristics to filter out improbable attributes and we rank attributes based on frequencies and a table match score. Ranking is reinforced with external evidence from Web search, DBPedia and Wikipedia. Our approach can be applied to whatever instance (e.g. Canada) to retrieve its attributes (capital, GDP). It is shown it has a much higher recall than DBPedia and Wikipedia and that it works better than lexico-syntactic rules for the same purpose.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval

General Terms

Experimentation, Performance

Keywords

information retrieval, attribute retrieval, entity-oriented search

1. INTRODUCTION

Traditional Web search provides a list of documents to a user query, but there are many information needs that should be answered with information within documents. For instance, the query "features of Mac Book" is not asking for all pages speaking of Mac Books rather than for their features. Inspired by the work in [2], we distinguish three types of information needs that can be answered differently:
- *query by attribute* (GDP of Canada, major of Boston)
- *query by instance* (Samsung Galaxy, Canada, Oscar Wilde)
- *query by class* (Macintosh notebooks, British writers)

When the query is specifically asking for an attribute, we can return its value right away. When the query is an instance, we can propose a summary of salient attributes. When the query is a class, the result can be a comparative table of the class instances with their attributes. Figure 1 shows what these results might look like.

In this paper we adress attribute retrieval. **Given a set of instances $I = i_1, i_2, ..i_n$ where $n \geq 1$, we want to extract and rank attributes with respect to their relevance to I.** Different approaches exist util now to extract attributes, but they are mostly precision oriented and they do not have enough recall to answer all attribute-oriented information needs. We can mention wrapper induction used to learn extraction rules [5] for attributes. It is also common to use for the same purpose lexico-syntactic rules [1] (e.g. "A of I" and "I's A").

JCDL'11, June 13–17, 2011, Ottawa, Ontario, Canada.
ACM 978-1-4503-0744-4/11/06.

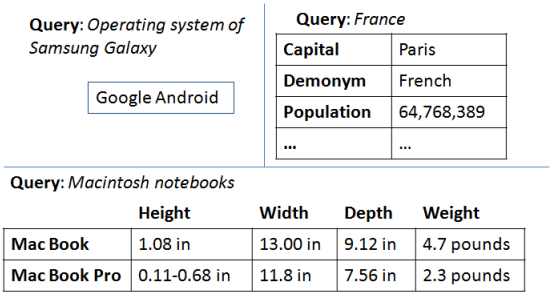

Figure 1: Examples of alternative search results

In this paper we propose a recall oriented and domain-independent technique that uses Web HTML tables to extract and rank attributes. Our method can be applied to whatever instance or set of instances.

2. ATTRIBUTE RETRIEVAL

We consider retrieving attributes for one or more instances of the same class. For every instance we retrieve a seed of documents using the top search results of a search engine. The seed is used to extract attributes from HTML tables. Finally, we rank attributes using a set of features. Each of these steps is further detailed below.

An important hypothesis in this paper is that having multiple instances can improve attribute retrieval. In fact, relevant attributes are likely to repeat across instances of the same class. Experimental setup will help us investigate this hypothesis.

Retrieval: Relevant attributes have to be within relevant documents for the instance (by definition of relevance). Using a search engine, we can obtain a seed of potentially relevant documents to extract attributes from. This is different from most attribute acquisition techniques which extract only from documents where the instance has an exact match. Exact matches can decrease recall significantly. "Prince William Arthur Philip Louis" will not also match with "Prince William", "Prince William of Wales", etc. Relying on a search engine is a permissive oriented choice, which might not necessarily return only rel-

evant documents, but will in most cases return at least some documents.

Extraction: We use Web tables to extract attributes from. Tables are probably the largest source of attributes in the Web. Cafarella et al. [4, 3] show that only about 1.1% of the tables are relational. Still, it represents the hugest source of relational data. They propose a relation search with these tables. They show that it is better to rank first tables that match the query. In our work, we filter out many irrelational tables as they do, but we are more permissive and we extract attributes also from tables that do not match the query.

Ranking: Extracted attributes are ranked with domain-independent features. Attributes are ranked on the following features: document frequency (how many documents it occurs in), instance frequency (how many instances it is found for), number of search hits on "A of I" (where A is the attribute and I is the instance), the presence of the attribute-instance relation in Wikipedia or DBpedia, and a table match score. The table match score for a table T and an instance i is computed as the maximimal cosine similarity between each table cell $T_{x,y}$ and the instance i. $match(i,T) = \max_{T_{x,y} \in T} cos(i, T_{x,y})$.

For an instance i and a candidate attribute a, the final ranking score $\phi(a,i)$ is a simple linear combination of the above features. When a set of instance I is considered we use the score $\phi(a,I) = \sum_{i \in I} \phi(a,i)$.

3. EXPERIMENTAL SETUP

To test our approach we use 190 instances from 19 classes (10 instances each), namely *"rock bands, laptops, universities, hotels, software, British army generals, chancellors, films, IR papers, SLR cameras, 2000 novels, Nirvana songs, Nissan vehicles, programmable calculators, countries, drugs, companies, cities, painters"*. Given a set of instances $I = \{i_0, i_1, ..\}$, we extract all tables from top search results on every instance using Yahoo! BOSS API. Similarly to [4], we filter out tables with too few cells. Attributes are taken from table headers (first col. and first row). 5 assessors evaluated top attributes for relevance. Cohen Kappa' inter-assessor agreement coefficient was 0.63 which falls in the substantial agreement range.

4. RESULTS

We tested our approach varying the number of instances per class, the number of search results per query and the linear combination of features. In figure 2, we can see precision at rank when we use one instance per class and 25 search results per query. Results are promising. If we apply this configuration for query suggestion, we will have that about 4 correct suggestions within top 5 suggestions.

In the same figure, we can also see results when we use 10 instances per class with the same number of search results per query. We noticed that **increasing the number of instances improves significantly results**.

Our experiments confirm also that increasing the number of search results per query helps. For instance, using 10 instances per class and respectively 10, 25 and 50 search results per query we obtain the following precision at rank 30 (P@30): 0.65, 0.73 and 0.74. Improvement is less significant after 30 search results per query.

Our dataset is too big to assess all attributes. However, to estimate the recall of our approach, we assessed all attributes for 4 classes "cameras", "countries", "companies", "vehicles". We found an average of 918 distinct attributes retrieved per class and about 256 relevant attributes per class. We compared these statistics with instances from the same classes in Wikipedia and DBpedia. Summing over both these sources, we could not find more than an average of 25 relevant attributes per class. We can confirm that **recall of Web tables is high.**

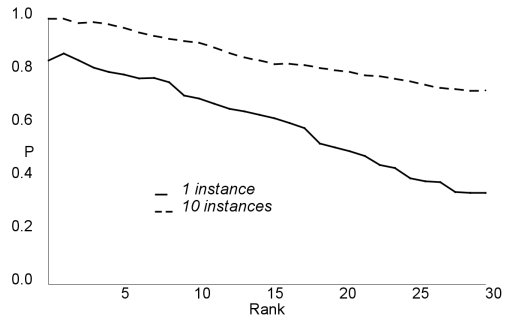

Figure 2: Impact of the number of instances

We also compared our approach with the use of lexico-syntactic rules for the same purpose. We use a configuration of 10 instances per class.The top 25 search results for all instances are used as extraction seed for both techniques. Results are shown in table 1.

Approach	p@10	p@20	p@30
Our approach	0.91	0.80	0.72
Lexico-syntactic rules	0.48	0.43	0.33

Table 1: Comparision to other techniques

5. CONCLUSIONS

In this paper, we propose a domain-independent attribute retrieval approach which can retrieve attributes at Web scale using Web tables. Our approach combines search, extraction and ranking to enable retrieval for whatever instance. Results are promising. Our approach has a much higher recall than quality sources such as DBPedia and Wikipedia. Furthermore, we show that our approach works better than lexico-syntactic rules for the same purpose.

6. REFERENCES

[1] E. Alfonseca, M. Pasca, and E. Robledo-Arnuncio. Acquisition of instance attributes via labeled and related instances. In *SIGIR '10*.

[2] M. J. Cafarella, M. Banko, and O. Etzioni. Relational Web Search. Technical report, U. Washington, 2006.

[3] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. Webtables: exploring the power of tables on the web. *Proc. VLDB Endow.*, 1(1):538–549, 2008.

[4] M. J. Cafarella, A. Y. Halevy, Y. Zhang, D. Z. Wang, and E. Wu. Uncovering the Rel. Web. In *WebDB*, 2008.

[5] C.-H. Chang, M. Kayed, M. R. Girgis, and K. F. Shaalan. A survey of web information extraction systems. *IEEE Trans. on Knowl. and Data Eng.*, 18:1411–1428, October 2006.