Enabling Entity-Based Aggregators for Web 2.0 data

Ekaterini Ioannou
L3S Research Center
ioannou@L3S.de

Claudia Niederée
L3S Research Center
niederee@L3S.de

Yannis Velegrakis
University of Trento
velgias@disi.unitn.eu

ABSTRACT

Selecting and presenting content culled from multiple heterogeneous and physically distributed sources is a challenging task. The exponential growth of the web data in modern times has brought new requirements to such integration systems. Data is not any more produced by content providers alone, but also from regular users through the highly popular Web 2.0 social and semantic web applications. The plethora of the available web content, increased its demand by regular users who could not any more wait the development of advanced integration tools. They wanted to be able to build in a short time their own specialized integration applications. *Aggregators* came to the risk of these users. They allowed them not only to combine distributed content, but also to process it in ways that generate new services available for further consumption.

To cope with the heterogeneous data, the Linked Data initiative aims at the creation and exploitation of correspondences across data values. In this work, although we share the Linked Data community vision, we advocate that for the modern web, linking at the data value level is not enough. Aggregators should base their integration tasks on the concept of an entity, i.e., identifying whether different pieces of information correspond to the same real world entity, such as an event or a person. We describe our theory, system, and experimental results that illustrate the approach's effectiveness.

Categories and Subject Descriptors: I.2.3: Resolution, D.2.2 [Software Engineering]: Design Tools and Techniques.

General Terms: Design, Algorithms.

Keywords: Entity Matching, Linked Data, Semantic Web.

1. INTRODUCTION

The paramount success of Wikipedia, Blogosphere, and other similar Social Web applications, are live evidences of the benefits that collaborative content creation can offer and of the fact that user-generated content can grow large both in terms of size, complexity, and importance. This increased the demand for integration tasks that the technical workforce could not easily cope with. An important breakthrough, was the development of mashups [7] technologies. Mashups [7] are designed for easy and fast integration tasks using open APIs and data sources. They typically combine data or functionality of many sources to create a new service.

This moved much of the developing burden for reusing and integrating existing web content for social and Web 2.0 applications from the technical experts, to the general public of web users [1]. It also allowed applications, such as DBPedia, Freebase, Spock, and DBLife, to easily incorporate knowledge already collected by other applications [4].

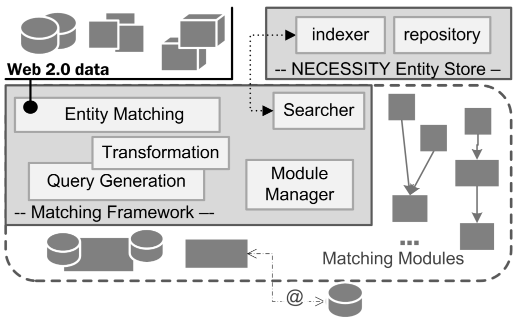

Figure 1: **Aggregator Matching Component Architecture**

The linked-data initiative [3], currently taking place in the web community, comes to the aid of integration. The aim is to build the infrastructure for re-using and interlinking existing data from different sources by relying on (Semantic) Web standards and principles for data publication on the Web. The challenge is to align multiple representations of the same real world object, or deal with *URI aliases*, typically represented as *owl:same* statements between resources. This will support the integration developer by facilitating the task of identifying different representations of the same objects across different sources, leaving him/her with only the task of combining these representations as suits best the application under development.

Unfortunately, the plethora of existing data matching algorithms [5] suggest the inability of a single matching algorithm to address the matching problem. We advocate an integration framework that is based on entities, instead of data structures like simple atomic values or tuples. We describe an entity matching framework that employees an extensible set of matching modules, each based on a different matching technique. Matching is performed as a series of steps that include selection of the most appropriate matching algorithm and combination of results from more than one modules.

2. ENTITY-BASED AGGREGATORS

Entities are the artifacts used to model real world objects and concepts. The characteristics of a real world object are modeled in an entity through a series of attributes. An attribute is a name-value pair, where the value can be an atomic value, or an identifier of another entity. This model is generic and flexible enough to represent relational, semi-structured, and RDF data [9],while at the same time, is closer to the human thinking. An entity has an identifier that has been assigned locally in the aggregator, and a set of *alternative* identifiers that correspond to identifiers given for the same real world object from other web sources. The goals of an aggregator is bifold: (i) process incoming queries describing an entity, and identify whether such an entity already exists in the system, and (ii) effectively merge the data of matched entities in order to maintain the repository.

Fig. 1 illustrates our aggregator's infrastructure. It incorporates an entity storage component, which in our current implementation uses the Necessity system [6]. This system provides a repository of entity profiles, i.e., entities alongside their attributes, and an index for efficient entity retrieval. The reason for including an entity store in our aggregators is to reduced the set of candidates in order to provide less data to the matching modules, since the matching algorithms are typically performing a lot of heavy operations.

When a query describing an entity is received, the matching framework invokes the entity matching component. This component first analyses this query to generates an initial query for the entity store, and identifies the matching module or modules that would more effectively evaluate the query. The initial query is then revised by the selected matching module(s) and send to the entity store. The store processes the query and returns a small set of entity candidates. These candidate entities are then given to the module(s) for performing matching and identifying the entity that corresponds to the given query. The following paragraphs provide the details for the main parts involved in this process.

Query Generation. The entity matching component needs to generate a query for the storage, which for the Necessity store used corresponds to a Lucene query. Since the store offers very efficient but restricted search functionality, this step might also require the generation of more that one queries, with the final entity candidate list being the merging of the entity candidates returned by the store for all generated queries. In addition, the query can be enhanced and refined by the matching modules according to their needs, e.g., transformations on the schema level, or query relaxation.

Matching Modules. Individual matching modules implement their own method for matching queries with entity profiles, with each module focusing on a specific matching task (e.g., matching in the absence of attribute names, or using associations). Modules may also use a local database for storing their internal data, or even communicate with external sources for retrieving information useful for their matching algorithm.

In addition to individual modules, the matching framework can also contain modules that do not compute matches directly, but by combining the results of other modules. One methodology is the sequential processing, where a module invokes other modules in a sequence. So, each module receives the results of previously invoked module, and the resulted entity matches are the ones returned by the last module.

The other possible methodology is parallel processing, where a module invokes a set of modules at the same time. Once all modules return their matches, the module needs to combine their results to a final list. This process has recently attracted research attention and especially for probabilistic data. For example, the approach presented in [2] aims at identifying the most possible entity merge from the ones generated by various algorithms.

Module Section. To know the abilities of each module, the matching framework maintains the profiles of the modules. These profiles do not only contain module description and classification, but also information about their matching capabilities. For example, the average time required for processing queries, and the query formats that they can handle.

The module profile along with the information of the query are used for selecting the module that is more appropriate for evaluating the specific query. For example, this may include requirements with respect to performance, existence or not of attribute names.

Module Selection & Result Combination. The current aggregator's implementation aims at effectively handling entity queries coming from free text (i.e., keywords) and from information extractors such as OpenCalais[1] or Cogito [2]. We employ two modules, the 'Group Linkage' invoked when the query contains attribute names and the 'Eureka' module invoked in the absence of attribute names. The first is an adaptation of the algorithm suggested in [8], and computes matches by detecting the similarity fraction between the attributes from the query and the attributes of the entity profile in the store. The second module deals with the lack of attribute names in the queries by using importance weights on the attribute names in the entity profiles, e.g., matching with attribute names *full_name* will have a higher score than with *residence*.

3. EXPERIMENTAL EVALUATION

Our evaluation aimed at investigating efficiency and effectiveness of matching via entity-based aggregation of Web 2.0 data. We imported in our entity store 6.865.392 entities coming from people and organizations from Wikipedia, and geographical entities (e.g., describing countries, cities, mountains) from GeoNames. We then evaluated entity matching with the following query sets:

(A) Queries created by data generated using the OpenCalais extractor from news events, e.g., " name='Carla Boni' position='singer' country='italian' medicalCondition='long illness' ". Our approach was able to correctly match and return the requested entity as the first answer for 81% of the queries with exeuction time 0.02 seconds (average over 2.000 queries).

(B) Queries created by collecting text describing entities as available on Web pages, e.g., "Jacques-Yves Cousteau French explorer". Entities were returned as the first answer for 74% with execution time 0.03 seconds (average over 2.000 queries).

Please contact the authors for entity dataset and queries.

4. CONCLUSIONS

In this work we presented an approach for entity-based aggregators that lead to more efficient and effective integration of Web 2.0 data. Ongoing work includes the investigation of matching in the presence of associations, which should be also considered to improve aggregation's performance and quality. We also aim at improving aggregation through external knowledge, e.g., WordNet.

Acknowledgments. This work is partially supported by FP7 EU Project OKKAM (contract no. ICT-215032).

5. REFERENCES

[1] S. Amer-Yahia, V. Markl, A. Y. Halevy, A. Doan, G. Alonso, D. Kossmann, and G. Weikum. Databases and web 2.0 panel at vldb 2007. *SIGMOD Record*, 2008.

[2] P. Andritsos, A. Fuxman, and R. J. Miller. Clean answers over dirty databases: A probabilistic approach. In *ICDE*, 2006.

[3] C. Bizer, T. Heath, and T. Berners-Lee. Linked Data - The story so far. *IJSWIS*, 2009.

[4] N. N. Dalvi, R. Kumar, B. Pang, R. Ramakrishnan, A. Tomkins, P. Bohannon, S. Keerthi, and S. Merugu. A web of concepts. In *PODS*, 2009.

[5] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Trans. Knowl. Data Eng.*, 2007.

[6] E. Ioannou, S. Sathe, N. Bonvin, A. Jain, S. Bondalapati, G. Skobeltsyn, C. Niederée, and Z. Miklos. Entity Search with Necessity. In *WebDB*, 2009.

[7] G. D. Lorenzo, H. Hacid, H. young Paik, and B. Benatallah. Data integration in mashups. *SIGMOD Rec.*, 2009.

[8] B.-W. On, N. Koudas, D. Lee, and D. Srivastava. Group linkage. In *ICDE*, 2007.

[9] M. Zhong, M. Liu, and Q. Chen. Modeling heterogeneous data in dataspace. In *IEEE IRI*, 2008.

[1]http://www.opencalais.com/

[2]http://www.expertsystem.net/page.asp?id=1515/