Entity Search with NECESSITY

Ekaterini Ioannou[‡] Saket Sathe[§] Nicolas Bonvin[§]
Anshul Jain[§] Srikanth Bondalapati[§] Gleb Skobeltsyn[§]
Claudia Niederée[‡] Zoltán Miklós[§]

[§]Ecole Polytechnique Fédérale de Lausanne (EPFL)
{name.surname}@epfl.ch

[‡]L3S Research Center/Leibniz Universität Hannover
{surname}@L3S.de

ABSTRACT

Loosely structured heterogeneous information spaces are typically created by merging data from a variety of different applications and information sources. A common problem these information spaces need to address is that various data describe the same real-word entities (e.g., people, conferences, organizations). In this demo, we introduce NECESSITY, an efficient and scalable entity store. NECESSITY is able to handle a large number of entities and at the same time provide an efficient and highly accurate entity search functionality for heterogeneous and partially structured queries that follow the vision of dataspaces.

1. MOTIVATION AND OUTLINE

We are currently witnessing a rapid increase in the number of loosely structured *heterogeneous information spaces* - collections of data coming from a variety of different applications and information sources. One common problem these information spaces face, is managing their entities (e.g., organizations, events), since there will be given various representations for the same real world *entities*. The NECESSITY entity store[1] is able to address this challenge. Our system can handle a large number of entities and at the same time provides an efficient and highly accurate entity search functionality.

NECESSITY stores entity profiles composed by a set of attribute-value pairs. It allows efficient entity search over these loosely structured heterogeneous information spaces, with queries being conditions on the entity's attributes or values. Our approach contributes to the idea of realizing dataspaces as envisioned in [3]. This includes developing data models and designing search methods for a large collection of interrelated data, even if the entity queries are specific to our context.

NECESSITY was implemented and evaluated in the context of the Entity Name System (ENS), for the OKKAM project[2]. The aim of ENS is to foster the global re-use of entity identifiers and to mediate between existing identifiers for individual entities (details available in [1]). ENS receives queries and checks whether the entity described in each query exists in OKKAM. If the entity exists OKKAM returns the corresponding identifier. The core benefit of ENS is to ease integration of external applications. For example, repositories from personal information management systems can now rely on the service provided by ENS for creating their URIs for entities. As such, the integrating challenge of knowing which representations in different repositories refer to the same entity, would be resolved by the use of shared IDs as issued by OKKAM.

There are further application types that can profit from our proposed approach. One example is entity search in collaboratively authored information spaces, such as Wikipedia[3]. Each Wikipedia entry is composed by various contributors who are not enforced to follow a specific format or schemata in a consistent way. Moreover, processing a human query over Wikipedia data could benefit from matching the query with the heterogeneous data of the entities. Another example of targeted applications for NECESSITY is entity search engines. These applications are built upon information extracted from Web pages. Integrating this extracted data imposes a matching challenge of effectively identifying and merging the existing data that refer to the same real world entities. In addition, searching for a specific entity through this plethora of entities requires advanced matching functionalities.

The rest of this paper is organized as follows. Section 2 presents the NECESSITY entity store, with main focus on the entity search process. Section 3 describes the demo scenario, and Section 4 conclusions along with future work.

2. ENTITY SEARCH PROCESS

Entities in NECESSITY are modeled as a set of attribute-value pairs; a representation similar to dataspace proposal [3]. For example, a person entity will be represented by name, affiliation, email address, and whatever else is available. Entity search allows users or applications to retrieve the entities —ideally only one— already in NECESSITY that best match an entity description provided as a query to the system. An entity query is a set of predicates, where each predicate is a keyword or an attribute value pair, e.g., Q_1: name="John Smith" EPFL, and Q_2: name=Smith affiliation=EPFL.

[1]The name of the system was inspired by philosopher William of Ockham's famous principle: "Entities must not be multiplied beyond *necessity*".

[2]http://www.okkam.org

[3]http://www.wikipedia.org

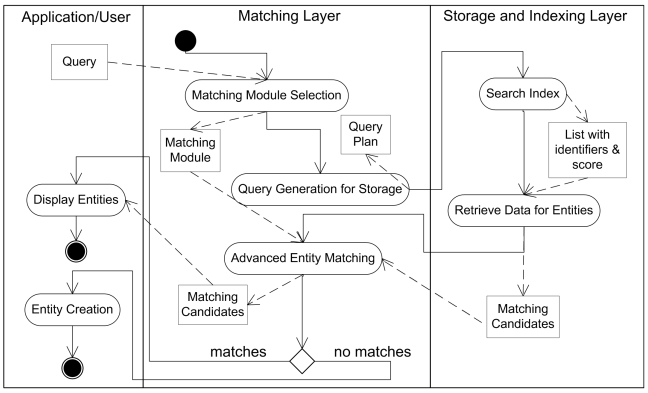

Figure 1: The search process of NECESSITY.

Figure 1 shows an illustration of the search process incorporated in NECESSITY for answering entity queries. When NECESSITY receives an entity query, it selects the appropriate matching module to process it. We consider the possibility of different methods for selecting the module, for example explicit selection by user/application (e.g., based on previous experience), or selection using query information such as restrictions in execution time. Once the matching module is selected, the query is reformulated to realize the semantics of the entity storage and to cope with incomplete or imprecise information. More specifically, this action can generate a disjunction of predicates over the entity profiles in the storage layer, or include missing schema information.

The generated query is then sent to the storage and indexing layer for evaluation. The storage will use the distributed index to retrieve a *constant* number of the most relevant entities, named "matching candidates", and then their corresponding entity profiles. The matching module will receive these candidates and perform advanced entity matching. This will select the most relevant entities for the given query by computing the matching probability between each candidate and the entity described in the given query.

There are two possible outcomes of the entity search process. The first is an empty list, which indicates that the entity described by the query is not in NECESSITY, and thus the application/user may choose to create it. The second possible result is a ranked list of entities which were found to match the query.

We evaluated NECESSITY with 1M entities, and 500 queries[4] extracted from various real data and web pages, extracted using Cogito extractor[5]. Each query was manually processed to identify the corresponding entity. The average time taken for executing queries was bellow 0.9 second. The following table shows the fraction of queries for which the requested entity was returned among Top-5 and Top-10 results for three different sizes of NECESSITY:

	Entities in Store		
	1.005.004	980.980	940.940
Entity found in Top-5	0.89121	0.89032	0.88864
Entity found in Top-10	0.93724	0.93763	0.93987

3. DEMONSTRATION

In the demo, users will be able to perform their own entity search queries on the NECESSITY entity store. NECESSITY will contain at least 1M entities, including people and organizations from *Wikipedia*, locations from *GeoNames*, and proteins from *UniProt*. In addition, we will present the details and discuss the various aspects of NECESSITY, particularly in relation to the topics shortly described in the following paragraphs.

Entity Queries. NECESSITY can handle various types of queries made from different sources. The various types include; queries which are over-specified or under-specified, contain only keywords, or incomplete information.

Queries Generated for the Storage. User queries are internally reformulated in several ways. This includes query extension, and incorporation of higher weights to attributes which are considered more important than others (e.g., the attribute *name* of person entities).

Identification of the Matching Candidates. NECESSITY uses a key-value store for storing the entity profiles along with a document partitioned inverted index specially designed for processing entity queries. The primary reason for selecting a document partitioned scheme for indexing is to ensure scalability [2]. These are used to efficiently drill down top-k matching candidates from the storage that best match the given query.

Advance Matching. Our current implementation of NECESSITY contains two matching modules. The first is group linkage, which returns entities whose attribute value pairs have high similarity with query predicates [4]. The second module is generic matching, an extension of group linkage that in addition considers domain specific similarity functions and the selectivity of predicates included in the query.

4. CONCLUSIONS & FUTURE WORK

This paper introduced NECESSITY – an entity storage for managing entities. NECESSITY is able to handle a large number of entities while providing an efficient and highly accurate search functionality, with a long-term goal of handling entities on web-scale. Our future plans include performing a new set of experiments that aim at testing the system in a distributed setup. In addition, we plan better translation of the requests into queries for the storage with special focus on reducing the time needed for answering these requests and increasing the result quality.

5. ACKNOWLEDGMENTS

The authors thank Oleksandr Druzhynin for helping in implementation; Dr. Peter Fankhauser, and Juri Luca De Coi for allowing the use of their generic matcher; and the participants of the OKKAM project for valuable discussions. This work is partially supported by FP7 EU Project OKKAM (contract no. ICT-215032).

6. REFERENCES

[1] P. Bouquet, H. Stoermer, C. Niederee, and A. Mana. Entity name system: The backbone of an open and scalable web of data. In *ICSC*, 2008.

[2] J. Dean. Challenges in building large-scale information retrieval systems: invited talk. In *WSDM*, 2009.

[3] A. Y. Halevy, M. J. Franklin, and D. Maier. Principles of dataspace systems. In *PODS*, 2006.

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

[4]http://www.okkam.org/resources/query-set.txt
[5]http://www.expertsystem.net/page.asp?id=1515/