From Web Data to Entities and Back

Zoltán Miklós[1], Nicolas Bonvin [1], Paolo Bouquet[2], Michele Catasta[1], Daniele Cordioli[3], Peter Fankhauser[4], Julien Gaugaz[4], Ekaterini Ioannou[4], Hristo Koshutanski[5], Antonio Maña[5], Claudia Niederée[4], Themis Palpanas[1], Heiko Stoermer[1]

[1] Ecole Polytechnique Fédérale de Lausanne (EPFL) {name.surname}@epfl.ch
[2] DISI, University of Trento {stoermer, bouquet, themis}@disi.unitn.it
[3] ExpertSystem s.p.a, Modena, Italy dcordioli@expertsystem.it
[4] L3S Research Center, Leibniz Universität Hannover {surname}@L3S.de
[5] Universidad de Málaga {hristo, amg}@lcc.uma.es

Abstract. We present the Entity Name System (ENS), an enabling infrastructure, which can host descriptions of named entities and provide unique identifiers, on large-scale. In this way, it opens new perspectives to realize entity-oriented, rather than keyword-oriented, Web information systems. We describe the architecture and the functionality of the ENS, along with tools, which all contribute to realize the Web of entities.
Keywords: entity, Web, unique identifier

1 Introduction

The information need of Web users is often related to their understanding and interpretation of real-world entities and their relationships. For example, someone might be interested in information about Paris, the picturesque capital of France. This mental interpretation of humans is very challenging to capture, thus information systems need a simplified representation.

The information on the Web is present mainly in the form of unstructured or semi-structured documents, thus it is very natural to model the Web data as a list of keywords, organized in documents, which might have links to each other. This simplified model was extremely successful, the Web search engines built around this model are used by millions of users every day.

This simple model however has a number of limitations, including the problem that the keywords are ambiguous. For example, if the keyword "Paris" appears on a page, it is not immediately clear, if this refers to the capital of France, to a city Paris in Texas, US or to the first name of Paris Hilton. This ambiguity is very problematic for human users, but it can cause more problems, if one intends to process the page automatically. There are numerous efforts to address this ambiguity problem, notably the Semantic Web community has proposed a number of solutions.

In this paper we present an information system, accompanied by a number of extendable tools, which enables to realize the "Web of entities". As opposed to the keyword-based model, in the Web of entities the keywords pointing to

named entities, such as persons, geographic locations, etc., are annotated with a reference to an entity description. These entity descriptions contain important information about the entities and also have a unique identifier (OKKAM ID). The basic idea of the Entity Name System (ENS), is to provide a free, open service to the users of the (Semantic) Web, such that they can annotate the Web content with references to entities.

We describe the design and architecture of the ENS. The main functionality of the ENS is to process entity search requests and return a unique identifier (OKKAM ID) of the entity. In this way one can annotate the page or the keyword with this unique identifier, so later this unique ID can be used to avoid ambiguities. We designed the ENS in a way such that it can work on web-scale and provide well within second response time even in the presence of a large number of users. The services of the ENS are openly available via Web Services, thus enabling Web application developers to adopt an entity-oriented, rather than keyword-oriented model. We already integrated this support into a number of popular tools, such as gmail, blogging, MS Word, etc. We populated the repository with an initial set of entities (with data related to LOD[1]), but the goal is to further continuously populate the entity descriptions, with the help of the tools, or through manual entity creation: similar to wikipedia, anyone can enter new entity descriptions to ENS. As entity descriptions might naturally evolve over time, ENS has support to accommodate these activities.

The rest of the paper is organized as follows. Section 2 discusses entity-oriented information systems, Section 3 explains the architecture and the functionality of the ENS. Section 4 presents a set of tools, which both use the services of the ENS and contribute to populate the entity repository. Section 5 evaluates how the ENS and the tools contribute towards realizing a Web of entities. Section 6 provides related work and finally Section 7 concludes the paper.

2 Entity-oriented information systems

The ENS was developed in the course of the OKKAM [2] project. The overall goal of the project is to enable the Web of entities, a global digital space for publishing and managing information about real-world entities, where each entity is linked to a unique identifier, named OKKAM id. The ENS plays a central role achieving this goal: the ENS can store a large number of entity descriptions, it assigns the unique OKKAM identifers to the descriptions (entity profiles). Furthermore, the ENS provides an entity search service, which returns the unique identifier of the requested entity.

This infrastructure intends to open the possibility to develop entity-oriented applications, by reusing the unique identifier one can avoid a number of ambiguity-related problems. We envision a continuous feedback between entity use and content creation. The OKKAM empowered tools (Section 4) shall recognize named entities in plain text documents and obtain the corresponding unique identifiers.

[1] Linked Open Data, `http://linkeddata.org/`

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

This process is often called OKKAMization. On the one hand, the ID can be reused, but on the other hand, the user can provide feedback, enter or update entity descriptions into the ENS. In this way, the entity population of the ENS will be continuously improved and enlarged, for the profit of the whole community.

3 The Entity Name System

In this section we give an overview of the functions of the ENS and its main components. In particular, Section 3.1 describes the main components of the system, Section 3.2 discusses the data population (currently available in the ENS) and its representation, Section 3.3 gives details about the storage infrastructure, Section 3.4 provides an overview of the entity search functionality, while the entity lifecycle management is discussed in Section 3.5. Finally, Section 3.6 discusses the security infrastructure of the ENS.

3.1 Overall system architecture

The *ENS* consists of the following four main components:

1. **Entity Store**. This component is responsible for the low-level tasks of storing and managing entity data, as well as relevant metadata which are necessary for finding existing identifiers. It follows the usual create-read-update-delete pattern (CRUD) [16] of serialized entity data accessed via a primary key.
2. **Index**. Search operations for an identifier via an entity profile in the *ENS* are accelerated using index structures.
3. ***ENS* Core**. This component is implementing the services that are exposed by the *ENS*, namely, entity creation and update, matching, as well as processes relevant to the entity lifecycle management.
4. **Offline Processing**. This component handles tasks which shall be performed offline rather than request processing time, including statistics computations, data quality related jobs, etc.

The *ENS* Core itself consists of several parts that we describe in the following. The **Storage API** abstracts from the complexities of accessing the Entity Store and Index components mentioned above. In particular, the underlying strategies for addressing the individual, load-balanced clusters and retrieving all relevant data are hidden from upper layers. The **Matching** component is responsible for ensuring a high top-k precision of entity identifier search results. It has two main tasks: (i) parsing, analyzing and – if necessary rewriting or expanding – the search request coming from a client, and (ii) applying specific ranking mechanisms with the aim to establish the best match between the search request and the candidate entities. The **Lifecycle Manager** takes care of the entity creation- and update-related tasks, and the important aspects of data quality and data lifecycle management. The **Access Manager** ensures that the requirements of access control and privacy are fulfilled. It performs, among other things, query

and result set filtering to avoid undesired use of the system. All services that the *ENS* offers as its public API are exposed through the **Web Services** component, which constitutes the uppermost layer of the *ENS* component stack. The ENS also equipped with an end-user frontend for identifier search and entity creation, which is available at `http://api.okkam.org/search/`.

3.2 Data

Entity Representation and Request language In the *ENS*, an entity is identified by an entity identifier, *oid*, assigned by the system. This ID is often referred as OKKAM ID. Along with this identifier, we store a set of alternative ids, *Aid*-s, that other systems have assigned to the same entity, which can also be used to access the entity or produce *same-as* statements for the LOD community (see Sect. 4.2). The preferred id, *prid*, is one of the above identifiers that we use when displaying the entity[3]. The descriptive part of the entity profile contains a set of attribute name value pairs that describe the entity. Some of these descriptive attribute-value pairs might contain external references, such as for example links to Web documents describing the entities. Finally, we also store a set of metadata for each entity, that include usage statistics, and provenance and access control metadata. All the above information is individually indexed to allow fast access to it, and then stored as a single XML file on disk.

Our goal in ENS is to keep the entity representation simple and at the same time as general as possible, without imposing a fixed schema of entity types or attributes [24]. In order to reach a compromise between the generality of the description and the precision in the functionality of the system, we use an internal classification of all entities in six broad, top-level categories, namely, *person*, *organization*, *location*, *event*, *artifact*, and *other*. For each one of these entity types, we have identified a set of default attributes that are most commonly used to describe them [2]. These default attributes are suggested to users during entity creation, but they are of course optional. Nevertheless, we expect that most of the created entities will use (at least some of) the default attributes (for example, give a name for a person), which in turn will have positive influence on the performance of the system.

We adopted a simple syntax for the entity ID request language: a request may contain a (list of) keywords and a (list of) attribute value pairs. For example, "Paris" or "first name=Paris". For a more precise description of the syntax we refer to `http://community.okkam.org/index.php/Documentation/APIs/entity-id-request-language.html`.

Entity Population We populated the repository of the ENS in two different ways. We imported openly available datasets (using their structured representations available online), such as Wikipedia (ca.700 000 entities) and Geonames[4]

[3] The only guaranteed, unique identifier is still the *oid*, which is always used internally in the ENS.

[4] `http://www.geonames.org/`

(ca. 6.5 million entities) to have an initial rich population[5]. We also added further entities manually, through the administrative interface of the ENS, so the current size of the entity population (as of February 2010) of the ENS is ca. 8 million.

3.3 Storage

The storage layer provides efficient means to store entity profiles and serves as an underlying infrastructure for the ENS. The storage layer also provides services for other components of the ENS, for example to realize entity search. The ENS adopts a two phase entity search approach. As a response to a (possibly reformulated) user query the storage layer first obtains a list of top-k candidates, which is then further processed in the search component (see Section 3.4). Thus, the primary goal of the storage layer to answer these queries with a very high recall, i.e. whenever a matching entity profile exists in the repository, it should be returned with a high probability.

The storage layer consists of two main building blocks: a key-value store and an inverted index [20]. The inverted index is used for query processing, to find the relevant documents (i.e. entity profiles), while the key-value store contains the entity profiles themselves. This organization of components is similar to the architecture of Web search engines [8]. The ENS shares many requirements with search engines, for example storing a large collection of files, having a high availability, fault tolerance, being able to process queries well below a second, even in the case of high query load, etc. To address these requirements we employed the same or similar techniques which are -to best of our knowledge- used to build Web search engines [8]. These techniques include the distribution of both the index and the data over several machines, replication of both the data and the inverted index and document partitioning.

There are however important differences to Search engines. The request language supported in the ENS allows to specify attributes to keywords (see Section 3.2). For example, the user might issue a query "city:Paris country:France", not just a simple keyword oriented query "Paris France". In order to support these queries, we apply a specific index structure. As opposed to the inverted indexes used by the Web search engines, our posting lists also contain the attribute information in addition to the document identifer, where the keyword occurs, for each element in the posting list. The other main difference is the ranking: we developed a ranking scheme specifically tailored for entity profiles.

The implementation of our storage relies on Hadoop [6], while we have chosen Apache Solr[7] to realize the large and distributed retrieval indexes.

[5] Overlaps between the datasets were eliminated by limiting the wikipedia import to entities of type person and organization.

[6] `http://hadoop.apache.org/`

[7] `http://lucene.apache.org/solr/`

3.4 Entity Search

A core functionality of the ENS is effective entity search: given a description of an entity, identify and return the corresponding entity already from the ENS's repository and then return the respective OKKAM identifier. Internally this translates into the matching of the requested entity with the entity descriptions available in the repository of the ENS.

At first sight, the entity search task has a strong similarity with entity linkage techniques [14], also known as data matching [4, 9], deduplication [27], resolution [3], merge-purge [13], entity identification [21], and reference reconciliation [10]. Entity Linkage is the process that decides whether two descriptions refer to the same real world entity (see [12] for an overview). Actually, state-of-the-art methods from this area have also been reused and adapted in implementing entity search.

However, there are some major additional challenges, since —in contrast to the ENS— entity linkage typically relies on a well-defined schemata. In the ENS, the need for such an agreement on a joint schema or ontology has been deliberately omitted, since it is generally accepted that such an agreement is not viable in a large, heterogeneous, and evolving environments that involve various user communities. Instead, our entity search assumes a flexible data model, also envisioned for dataspaces [11] (see also Section 3.2).

Entity search will accept entity requests from a wide variety of agents, including humans as well as applications. The employed very loose schema commitment and constraints within the repository, along with the variety of requesters, imposes a set of additional challenges on the described search task:

A. Over-Specification. The agent requesting an entity is not expected to have knowledge about the repository's data. Thus, the agent will use the information available to him, for example, extracted from text, for describing the searched entity. As a result, the information of the requested entity might have more or other knowledge about the entity than the ENS repository. Due to this, the interpretation of ENS requests deviate from the traditional query processing, which expects to only receive information that their systems contain. Entity search of ENS is forced to always consider that a given entity might contain information which the repository does not have.

B. Under-Specification. An opposite situation with over-specification is when the information of the requested entity is not sufficient to do a full disambiguation. This missing information can be collected in different ways, for example user interaction, analysis of repository entities (statistics), or analysis of previously requested entities.

C. Schema Heterogeneity. Omitting the commitment to an agreement on a common schema will leads to schema heterogeneity, which we have to deal with when processing the entity requests.

As explained in Section 3.3, the search process in the ENS is divided into two main phases. First a set of candidates is selected, then they are ranked with the help of additional domain knowledge an heuristics. Below, we explain this second phase, which relies on a Generic Matching Module (explained in the next

paragraphs). This module is extensible, one can implement additional modules for specific types of entity profiles.

The input to the **Generic Matching Module** is a request in our simple request language see Section 3.2. A request consists of segments, which may either be attribute value pairs or unqualified values. The requests intend to find an entity profile, that we interpret such that that each segment of a request *should* match the sought entity, much like usually done in Information Retrieval. It takes into account fuzzy matches between attribute names and values, and also partial mismatches, which may arise due to over-specification.

Given a request and a candidate entity, the Generic Matching Module computes a final *matching score* resulting from the aggregation of several features of two kinds: attribute level features and entity level features. The attribute level features are attribute *label similarity*, attribute *value similarity* and an attribute *boosting* factor. These three features are aggregated into *attribute similarity*. For each request segment, the entity attribute with the maximum attribute similarity is chosen, and the individual *segment similarities* are aggregated to obtain the first entity level feature: the *entity similarity*. The second kind of entity level feature is the *entity popularity*, which further differentiates matching entities especially for under-specified queries. Finally, in order to deal with geographical entities, two special entity level features are dedicated to them: the *location type* and *location population*. This gives four entity level scores which are aggregated to the final score for each candidate matching entity.

At all levels, feature scores are aggregated by a weighted log-based tempered geometric mean. Given the scores to aggregate $S = \{s_1, \ldots, s_t\}$ and their respective weights $W = \{w_1, \ldots, w_t\}$, the aggregated score is

$$\epsilon GM(S, W) = \exp \frac{\sum_{i=1}^{t} w_i \log(s_i + \epsilon)}{\sum_{j=1}^{t} w_i} - \epsilon$$

This is a weighted version of the ϵ-*adjusted geometric mean* proposed by Robertson et al. in [26]. According to [25] the geometric mean is less affected by outlying values than the arithmetic mean, and since it is based on multiplication, there is no requirement that the scores are on the same scale. The ϵ-adjustment avoids a null score forcing the aggregation to zero, and computing in the log space helps avoiding underflows.

The label and value similarities are standard string similarity metrics[8] from the SimMetrics library[9]. The attribute boosting factor is a combination of two values: attribute selectivity and attribute popularity. We determined these factors through statistics on the data and query logs and through extensive experimentation.

[8] Respectively Needleman-Wunch and a combination of Levensthein and Monge-Elkan
[9] `http://www.dcs.shef.ac.uk/~sam/stringmetrics.html`

3.5 Entity lifecycle management

Lifecycle management of entities in the ENS includes aspects of entity representation, data quality, repository evolution, and online monitoring of the use of the repository, which we are going to illustrate in the following.

Data Quality: The aim of data quality at entity creation time is to ensure that the new entities satisfy a minimal set of quality requirements. This assessment also takes place when a new entity profile is created or when an existing entity is being modified.

The data quality assessment process at creation time that is currently in place within ENS, consists of the following three types of checks.

1. Attribute value quality, where we want to ensure that the values entered for the various attributes in the entity description are correct and valid (when possible). For example, we check that the provided attribute value is not empty or overly long, and that date and time attributes adhere to some known format.
2. Intra-entity description quality, where we check the quality of the entity description as a whole. For example we check whether attributes appear with the same name and different value (that would lead to a warning), etc.
3. Inter-entity description quality, where we check that changes in the repository will not degrade the overall quality performance of the system. In particular, we make sure that modifications do not introduce duplicate entity profiles, which would degrade the quality of search results. The duplicate detection relies on the matching techniques, discussed in Section 3.4.

Evolution of Identifiers: Even though the identifier of an entity should never change, in some special circumstances this may happen. For example, if we realize that two different entity profiles refer to the same real world entity, or when the same entity profile is already being used to refer to two distinct real world entities. In these cases, we would like to take corrective action, by performing a *merge* and a *split*, respectively.

The ENS supports the merge and split operations as follows. In the case of a merge operation, we select one of the existing identifiers (i.e., the one that according to the system statistics belongs to the most popular entity of the two) to be the identifier of the merged entity. The other identifier will still exist, so that users can refer to it, but only the selected identifier will be returned as a result. When splitting an entity representation into two new ones, we have no option but creating two new identifiers, since the old identifier has been used to refer to a non-existing (wrong) entity. In both cases, the system keeps the history of changes in order to be able to undo these operations. We also provide a service that makes the information about these changes available for users (and/or machines) to read, however in the current version the merge/split operations can be performed only manually.

Monitoring of Repository Usage: The way that users access the system and interact with it may provide useful insight on what actions to take in order to

improve the performance of the ENS. Therefore, we monitor the data streams relevant to the usage patterns of the system. We have extended and adapted algorithms that can operate in an online fashion, and are flexible enough to allow effective and efficient data analysis of the incoming data streams [28, 19]. By monitoring and analyzing the way users interact with the ENS we can further improve the ENS services.

3.6 Security and trust

To provide services for a large number of users, the ENS has a specific set of security requirements. These include access control requirements (fine grained policies, easy policy specification, automated access control enforcement), legal and privacy requirements (to be able to control in a confidential way, who can modify entity profiles) and usability requirements.

Our security architecture is based on advanced use of certificate technologies. Figure 1 shows a high-level view of the main elements of the security infrastructure and their interactions. Trust in ENS community is based on certificate authorities that qualify the ENS, third party service providers and users by means of identity and attribute certificates, compliant with X.509 [29] standard. The infrastructure adopts several widely-used security technologies such as https[10], Web Services security standards[11], and secure e-mail[12].

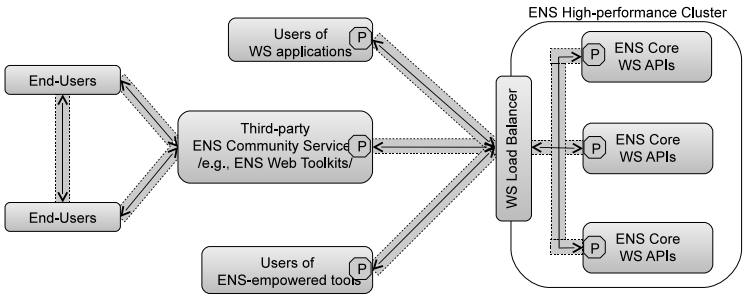

Fig. 1. ENS Security Architecture

All communications between a user and the ENS are realized through security proxies located at both at the ENS and the user sides. We have chosen a proxy-based solution to decouple all security management and technological aspects from application-level development. The proxy component allows transparent security management, so that also thin ENS-empowered tools (e.g., MS Word plug-in, see Section 4) can achieve a high level of secure communications. The

[10] Secure HTTP communications based on SSL/TLS standards.
[11] WS-Security, WS-SecureConversation etc. at `http://www.oasis-open.org/specs`
[12] OpenPGP-compliant e-mail security `http://tools.ietf.org/html/rfc4880`

proxy implements latest WS security standards[13] and advanced authorization process based on certificate-based automated trust negotiation model [17].

4 Tools

This section discusses tools, which benefit from the services of the ENS. OKKAM Empowered Tools are extensions of existing tools that both use the services of ENS, to enrich the processed plain text with entities and at the same time enable content creation: if an entity description does not exist in the ENS, the user might decide to create it. In this way, while they benefit from ENS, the ENS and the whole community benefits from the new content. This mutual feedback mechanism is intentional and shall foster the adoption of the ENS.

In the course of the OKKAM EU project, a whole suite of such tools is developed. These tools include an extension to ontology editors Protege and NeON [18], plugins for Microsoft office products (MS word, outlook), plugins for popular browsers (Firefox). In the following we demonstrate the use of OKKAM-plugins for popular Web tools.

4.1 For Web Users

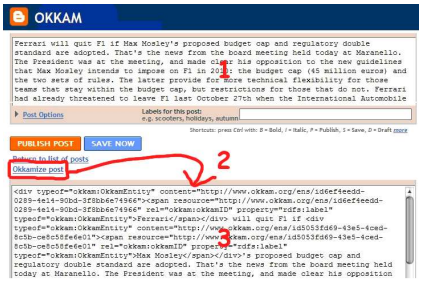

Fig. 2. Semantic annotation of entities via the Okkam4Blogger extension.

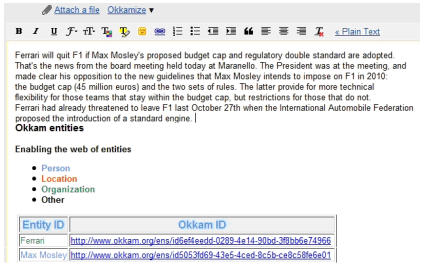

Fig. 3. The Okkam4Gmail extension for annotating named entities in emails.

Okkam4Blogger is a web plugin for the free blog application Blogger[14] from Google. Before posting a new document, the user can press the "Okkamize post" button (see Fig.2). The extension identifies precisely and automatically entities by means of a thorough processing and linguistic disambiguation, which includes morphological, grammar, syntactic and semantic analysis of the text. All fundamental information for the disambiguation process, i.e. the whole system knowledge, is represented as a concept-based semantic network. Expert System's semantic network, called Sensigrafo[15], is a rich conceptual representation of the

[13] Using Metro high-performance Web Services stack at `https://metro.dev.java.net`

[14] `http://www.blogger.com/`

[15] Sensigrafo is a trademark of Expert System S.p.A., Italy. `http://www.expertsystem.it/`

language containing more than 400,000 concepts and millions of links between these concepts. Once an entity is identified by linguistic and semantic analysis, the plugin returns its unique identifier from the ENS and injects it in an RDFa annotation inside the document(see bottom part of Fig. 2). If the ENS does not contain a record about the entity, a new identifier can be generated. The use of *Okkam4Gmail*, the Okkam Web plugin for Google's webmail is depicted in Figure 3.

4.2 For the Linked Data community

The goal of the Linked Open Data [16] community is to connect several openly available datasets on the Web. The initiative is closely related to the Semantic Web community, indeed many of the datasets are published in the form of RDF/OWL. A link among two URLs shall be interpreted as a `owl:sameAs` statement. Identifying such links is a very work-intensive task and typically requires data-source specific heuristics[17] .

The ENS can help the Linked Data community in particular to store and represent the links, as the correspond to equivalent entities. The ENS implements a functionality exposed through the method `getAlternativeIDs()` (accessible via Web services), which allows someone working with Linked Data to retrieve all the alternative identifiers known to the ENS for the same entity [18].

5 Evaluation

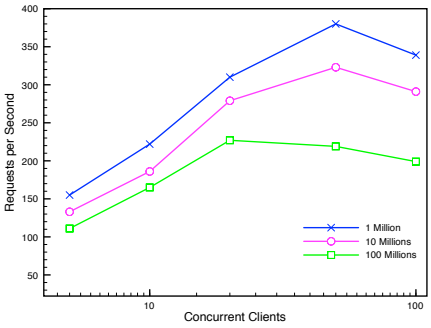

Fig. 4. Average query throughput

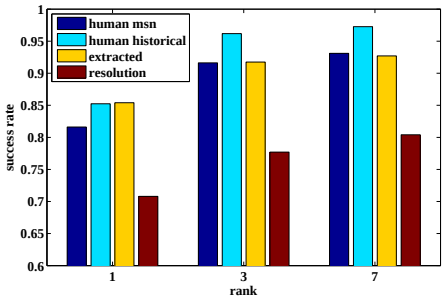

Fig. 5. Okkam entity search success rates for various query sets

In this section we report about some of our evaluation experiments, which support that the ENS can cope with large entity profile collections, show scalable behavior and high search quality.

[16] `http://linkeddata.org/`

[17] `http://esw.w3.org/topic/TaskForces/CommunityProjects/LinkingOpenData/EquivalenceMining`

[18] This function is inspired by what has been called "coreference service" by Glaser et al. in [15].

5.1 Scalability and performance

For our scalability experiments we used a large dataset, which we syntactically generated based on the US Census statistics[19]. The dataset consists of 100 million entities and has a size of ca. 300GB. We queried the system using concurrent clients. For the performance and scalability evaluation we used a synthetic query set. Figure 4 depicts the average query throughput figures we measured with a population of 1, 10 and 100 million entities, with 1-100 clients. For this measurements we used the following configuration: 1 Index server with 8 cores (2.66GHz), 16GB RAM, 1TB disk, Solr 1.4, Tomcat 6, and 16 HBase nodes with 8 cores (2.66GHz), 8GB RAM, 2x1TB disks, HBase 0.20.2.

5.2 Search quality

We evaluated the quality and performance of entity resolution on different kinds of queries over a storage containing 6.5 million entities, include people, organizations, and locations. We aimed at using queries covering a wide user and applications range. For this reason, we collected query sets from four different sources. The first query set, named 'human msn', contains 610 queries from the MSN Search query Log (RFP 2006 dataset)[20]. These queries are quite short in length that user provided as input in web search engines, e.g., "whitney houston", and "Bloody Mary of England". The next set, named 'human historical', are queries created from text that people included on web pages, for example to describe members of organizations, e.g., "Patrick de Gayardon French skydiver skysurfing pioneer". Our third set, named 'extracted' contains 315 queries created from data extracted from web blogs and news articles. We used two extraction tools, the OpenCalais, and the Expert System's[21] Cogito. Examples of the resulted queries are name="Hillary R. Clinton", and name="Barack Obama". The last set, named "resolution", has 1000 queries created from structured DBPedia data. In contrast to the previous sets, these queries provide much information for matching while also providing the attribute values. For all queries, we manually or automatically retrieved the Okkam identifier. Queries are available upon request.

Figure 5 shows the success rate at different ranks, which measures the performance of Okkam for returning the requested entity in the rank position, i.e., as the first answer, third, and seventh. Except for resolution queries, we observe that the success rate stabilizes relatively quickly around rank 3 at values above 90%. Resolution queries show lower success rates, even though 80% at rank 7 is acceptable. The difference between resolution and extracted queries is that resolution queries tend to be over-specified, and a wider range of attribute label are used than in extracted queries. It is interesting to note that for those structured queries over-specification performs worse than under-specification, whereas for human keyword queries the inverse is true. This might indicate a problem with

[19] `http://www.census.gov/genealogy/names/`
[20] `http://research.microsoft.com/en-us/um/people/nickcr/WSCD09/`
[21] `http://www.expertsystem.net`

the attribute label similarity. Schema matching techniques will be investigated to correct this. The average time for processing queries is bellow 0.5 seconds, which means that Okkam can process up to $\tilde{1}20$ queries per minute.

6 Related work

This paper presents the overall ENS and demonstrates through experiments, that the system can cope with large entity profile collections and high number of users. Thus, together with the OKKAM-empowered tools it can contribute to realize entity oriented information systems. The conceptual model for reusing OKKAM identifiers on the Web is discussed in [5]. The paper [7] presents and early prototype of the ENS, the system and the corresponding ecosystem we are presenting in this paper has significantly improved since these early versions. The paper [6] focuses business-oriented use cases, scenarios and application prototypes that might benefit from the use of the ENS.

OpenCalais [22] is a popular plugin for Web browsers. Using the plugin, the identified named entities are highlighted on the Web page. The entities have an identifier, however this might be not globally unique. The entity descriptions (if they exist) are kept private and the Web users are not allowed to enter or edit their own entities. On the contrary, the ENS provides unique identifiers and also one can edit the entity profiles, under some access restrictions.

The zemanta [30] plugin, which is available for popular email and blog software, empowers their users to automatically associate the text they are writing with resources on the web, such as pictures, wikipedia links, etc. As far as we know, they do not use globally unique identifiers.

Bautin et al. [1] propose an entity search engine. They construct concordance documents for each entity consisting all sentences, which contain references to the entity from their large text corpus. They do not provide tools or the possibility of editing entity profiles (concordance documents) and also they do not use globally unique identifiers. On the other hand they try to discover and depict the relations among entities, which the ENS does not support in this version.

OpenID [23] uses unique identifiers, but for a very different goal: they try to realize that users can identify themselves at different services with the same id and password.

7 Conclusion and future work

We implemented an enabling infrastructure, the Entity Name System, which opens the possibility to realize the web of entities. Our scalability and search quality experiments suggest that the ENS is able to serve a larger user community. The ENS is accompanied by a number of tools, which both exploit the services of the ENS, thus improving application experiences for users through disambiguation, and offer the possibility of content creation. In this way the whole user community can benefit from the individual contributions, and shall result a continuous entity population growth.

The ENS is currently used in several pilot projects including the semantic information mashup Sig.ma [22] and at the Italian new agency ANSA[23]. The unique identifiers provided by ENS are also utilized in a private setting to manage publication records at Elsevier[24] and to manage a large software product portfolio at SAP[25]. The ENS is also used to link physical entities to electronic ones, through QR codes, in the city of Manor, Texas, US.

Acknowledgements

This work is partially supported by the by the FP7 EU Large-scale Integrating Project **OKKAM – Enabling a Web of Entities** (contract no. ICT-215032). More details are available here: http://www.okkam.org. The authors would like to express their gratitude to all members of the OKKAM consortium for their contributions.

References

1. M. Bautin and S. Skiena. Concordance-Based Entity-Oriented Search. *Web Intelligence and Agent Systems*, 7(4):303–320, December 2007.
2. B. Bazzanella, J. A. Chaudhry, T. Palpanas, and H. Stoermer. Towards a general entity representation model. In *SWAP*, 2008.
3. O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, and J. Widom. Swoosh: a generic approach to entity resolution. *The VLDB Journal*, 18(1):255–276, January 2009.
4. M. Bilenko, R. J. Mooney, W. W. Cohen, P. Ravikumar, and S. E. Fienberg. Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18(5), September 2003.
5. P. Bouquet, T. Palpanas, H. Stoermer, and M. Vignolo. A Conceptual Model for a Web-Scale Entity Name System. In *The Semantic Web, Fourth Asian Conference, (ASWC'09)*, volume 5926 of *LNCS*, pages 46–60. Springer, 2009.
6. P. Bouquet, H. Stoermer, W. Barczynski, and S. Bocconi. *Cases on Semantic Interoperability for Information Systems Integration : Practices and Applications*, chapter Entity-centric Semantic Interoperability, pages 1–21. IGI Global, 2009.
7. P. Bouquet, H. Stoermer, and B. Bazzanella. An Entity Name System (ENS) for the Semantic Web. In *The Semantic Web: Research and Applications, 5th European Semantic Web Conference (ESWC)*, volume 5021 of *LNCS*, pages 258–272. Springer, 2008.
8. J. Dean. Challenges in building large-scale information retrieval systems (invited talk). In *Proceedings of the Second ACM International Conference on Web Search and Data Mining (WSDM)*, 2009.
9. A. Doan, Y. Lu, Y. Lee, and J. Han. Object matching for information integration: A profiler-based approach. In *Proceedings of IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03)*, pages 53–58, 2003.
10. X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *SIGMOD*, pages 85–96, 2005.

[22] `http://sig.ma`
[23] `http://ansa.it`
[24] `http://www.elsevier.com`
[25] `http://www.sap.com`

11. X. Dong and A. Y. Halevy. Indexing dataspaces. In *SIGMOD*, pages 43–54, 2007.
12. A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16, 2007.
13. M. A. Hernández and S. J. Stolfo. Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem. *Data Min. Knowl. Discov.*, 2(1):9–37, 1998.
14. E. Ioannou, C. Niedere, and W. Nejdl. Probabilistic entity linkage for heterogeneous information spaces. In *Proceedings of the 20th international conference on Advanced Information Systems Engineering (CAiSE)*, volume 5074 of *LNCS*, pages 556–570. Springer, 2008.
15. A. Jaffri, H. Glaser, and I. Millard. URI Identity Management for Semantic Web Data Integration and Linkage. In *IFIP WG 2.12 and WG 12.4 International Workshop on Semantic Web and Web Semantics (SWWS)*, volume 4806 of *LNCS*, pages 1125–1134. Springer, 2007.
16. H. Kilov. *Business Specifications: The Key to Successful Software Engineering*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998.
17. H. Koshutanski and F. Massacci. A negotiation scheme for access rights establishment in autonomic communication. *Journal of Network and System Management*, 15(1), March 2007.
18. X. Liu, H. Stoermer, P. Bouquet, and S. Wang. Supporting the Reuse of Global Unique Identifiers for Individuals in OWL/RDF Knowledge Bases (demo paper). In *Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications (ESWC)*, volume 5554 of *LNCS*, pages 868–872. Springer, 2009.
19. N. Manerikar and T. Palpanas. Frequent Items in Streaming Data An Experimental Evaluation of the State-of-the-Art. *Data Knowl. Eng.*, 68(4):415–430, April 2009.
20. C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.
21. A. Morris, Y. Velegrakis, and P. Bouquet. Entity Identification on the Semantic Web. In *Proceedings of the 5th Workshop on Semantic Web Applications and Perspectives (SWAP2008)*, 2008.
22. OpenCalais. http://www.opencalais.com/.
23. OpenID. http://openid.net/.
24. T. Palpanas, J. A. Chaudhry, P. Andritsos, and Y. Velegrakis. Entity Data Management in OKKAM. In *Proceedings of the 2008 19th International Conference on Database and Expert Systems Application (DEXA)*, pages 729–733. IEEE, 2008.
25. S. D. Ravana and A. Moffat. Score aggregation techniques in retrieval experimentation. In *ADC*, pages 59–67, 2009.
26. S. Robertson. On gmap: and other transformations. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 78–83, New York, NY, USA, 2006. ACM.
27. S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, pages 269–278, 2002.
28. F. I. Tantono, N. Manerikar, and T. Palpanas. Efficiently Discovering Recent Frequent Items in Data Streams. In *Proceedings of the 20th international conference on Scientific and Statistical Database Management (SSDBM)*, volume 5069 of *LNCS*, pages 222–239. Springer, 2008.
29. X.509. The directory: Public-key and attribute certificate frameworks, 2005. ITU-T Recommendation X.509:2005 | ISO/IEC 9594-8:2005.
30. Zemanta. http://www.zemanta.com/.