Integrating NLP and SW with the KnowledgeStore

Marco Rospocher, Francesco Corcoglioniti, Roldano Cattoni,
Bernardo Magnini, and Luciano Serafini

Fondazione Bruno Kessler—IRST, Via Sommarive 18, Trento, I-38123, Italy
{rospocher,corcoglio,cattoni,magnini,serafini}@fbk.eu

Abstract. We showcase the KnowledgeStore (KS), a scalable, fault-tolerant, and Semantic Web grounded storage system for interlinking unstructured and structured contents. The KS contributes to bridge the unstructured (e.g., textual document, web pages) and structured (e.g., RDF, LOD) worlds, enabling to jointly store, manage, retrieve, and query, both typologies of contents.

1 Introduction: Motivations and Vision

Despite the widespread diffusion of structured data sources and the public acclaim of the Linked Open Data (LOD) initiative, a preponderant amount of information remains nowadays available only in unstructured form, both on the Web and within organizations. While different in form, structured and unstructured contents are often related in content, as they speak about the very same entities of the world (e.g., persons, organizations, locations, events), their properties, and relations among them. Despite the last decades achievements in Natural Language Processing (NLP), now supporting large scale extraction of knowledge about entities of the world from unstructured text, frameworks enabling the seamless integration and linking of knowledge coming both from structured and unstructured contents are still lacking.[1]

In this demo we showcase the KnowledgeStore (KS), a scalable, fault-tolerant, and Semantic Web grounded storage system to jointly store, manage, retrieve, and query, both structured and unstructured data. Fig. 1a shows schematically how the KS manages unstructured and structured contents in its three *representation layers*. On the one hand (and similarly to a file system) the *resource layer* stores unstructured content in the form of resources (e.g., news articles), each having a textual representation and some descriptive metadata. On the other hand, the *entity layer* is the home of structured content, that, based on Knowledge Representation and Semantic Web best practices, consists of *axioms* (a set of ⟨subject, predicate, object⟩ triples), which describe the *entities* of the world (e.g., persons, locations, events), and for which additional metadata are kept to track their provenance and to denote the formal *contexts* where they hold (e.g., point of view, attribution). Between the aforementioned two layers there is the *mention layer*, which indexes *mentions*, i.e., snippets of resources (e.g., some characters in a text document) that denote something of interest, such as an entity or an axiom of the entity layer. Mentions can be automatically extracted by NLP tools, that can enrich them with additional attributes about how they denote their referent (e.g., with which name, qualifiers, "sentiment"). Far from being simple pointers, mentions present both unstructured

[1] See [1] for an overview of works related to the contribution presented in this demo.

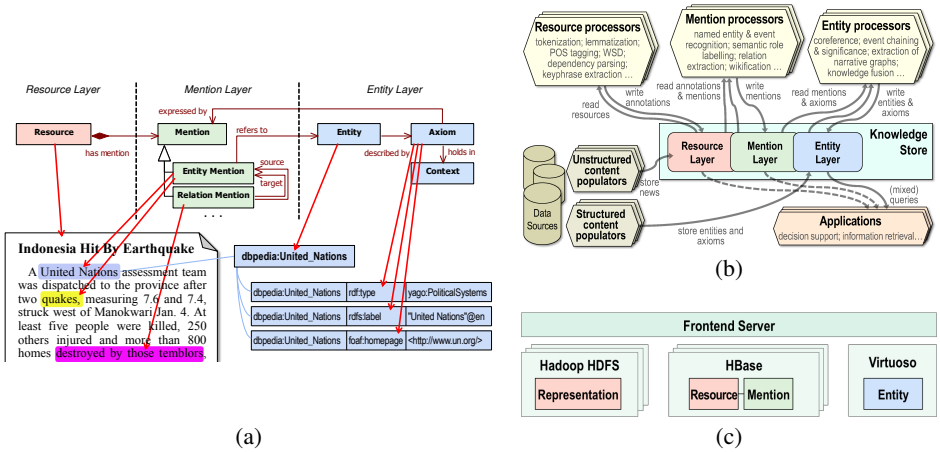

Fig. 1: (a) The three KS layers; (b) Interactions with external modules; (c) Components.

and structured facets (respectively snippet and attributes) not available in the resource and entity layers alone, and are thus a valuable source of information on their own.

Thanks to the explicit representation and alignment of information at different levels, from unstructured to structured knowledge, the KS supports a number of usage scenarios. It enables the development of enhanced applications, such as effective *decision support systems* that exploit the possibility to semantically query the content of the KS with requests combining structured and unstructured content, such as "retrieve all the documents mentioning that person Barack Obama participated to a sport event". Then, it favours the design and empirical investigation of information processing tasks otherwise difficult to experiment with, such as cross-document *coreference resolution* (i.e., identifying that two mentions refer to the same entity of the world) exploiting the availability of interlinked structured knowledge. Finally, the joint storage of (i) extracted knowledge, (ii) the resources it derives from, and (iii) extracted metadata provides an ideal scenario for developing, training, and evaluating ontology population techniques.

2 An overview of the KnowledgeStore

In this section we briefly outline the main characteristics of the KS. For a more exhaustive presentation of the KS design, we point the reader to [1]. More documentation, as well as binaries and source code,[2] are all available on the KS web site [2].

Data Model The data model defines what information can be stored in the KS. It is organized in three layers (resource, mention and entity), with properties that relate objects across them. To favour the exposure of the KS content according to LOD principles, the data model is defined as an OWL 2 ontology (available on [2]). It contains the TBox definitions and restrictions for each model element and can be extended on a per-deployment basis, e.g., with domain-specific resource and linguistic metadata.

API The KS presents a number of interfaces through which external clients may access and manipulate stored data. Several aspects have been considered in defining them

[2] Released under the terms of the Apache License, Version 2.0.

(e.g, operation granularity, data validation). These interfaces are offered through two HTTP ReST endpoints. The *CRUD endpoint* provides the basic operations to access and manipulate (CRUD: create, retrieve, update, and delete) any object stored in any of the layers of the KS. Operations of the CRUD endpoint are all defined in terms of sets of objects, in order to enable bulk operations as well as operations on single objects. The *SPARQL endpoint* allows to query axioms in the entity layer using SPARQL. This endpoint provides a flexible and Semantic Web-compliant way to query for entity data, and leverages the grounding of the KS data model in Knowledge Representation and Semantic Web best practices. A Java client is also offered to ease the development of (Java) client applications.

Architecture At its core, the KS is a storage server whose services are utilized by external clients to store and retrieve the contents they process. From a functional point of view, we identify three main typologies of clients (see Fig. 1b): (i) *populators*, whose purpose is to feed the KS with basic contents needed by other applications (e.g., documents, background knowledge from LOD sources); (ii) *linguistic processors*, that read input data from the KS and write back the results of their computation; and, (iii) *applications*, that mainly read data from the KS (e.g., decision support systems). Internally, the KS consists of a number of software components (see Fig. 1c) distributed on a cluster of machines: (i) the *Hadoop HDFS* filesystem provides a reliable and scalable storage for the physical files holding the representations of resources (e.g., texts and linguistic annotations of news articles); (ii) the *HBase* column-oriented store builds on Hadoop to provide database services for storing and retrieving semi-structured information about resources and mentions; (iii) the *Virtuoso* triple-store stores axioms to provide services supporting reasoning and online SPARQL query answering; and, (iv) the *Frontend Server* has been specifically developed to implement the operations of the CRUD and SPARQL endpoints on top of the components listed above, handling global issues such as access control, data validation and operation transactionality.

User Interface (UI) The KS UI (see Fig. 2) enables human users to access and inspect the content of the KS via two core operations: (i) the *SPARQL query* operation, with which arbitrary SPARQL queries can be run against the KS SPARQL endpoint, obtaining the results directly in the browser or as a downloadable file (in various file formats, including the recently standardized JSON-LD); and, (ii) the *lookup* operation, which given the URI of an object (i.e., resource, mention, entity), retrieves all the KS content about that object. These two operations are seamlessly integrated in the UI, to offer a smooth browsing experience to the users.

3 Showcasing the KnowledgeStore and concluding remarks

During the Posters and Demos session, we will demonstrate live how to access the KS content via the UI (similarly to the detailed demo preview available at [3]), highlighting the possibilities offered by the KS to navigate back and forth from unstructured to structured content. For instance, we will show how to run arbitrary SPARQL queries, retrieving the mentions of entities and triples in the query result set, and the documents where they occur. Similarly, starting from a document URI, we will show how to access the mentions identified in the document, up to the entities and triples they refer to.

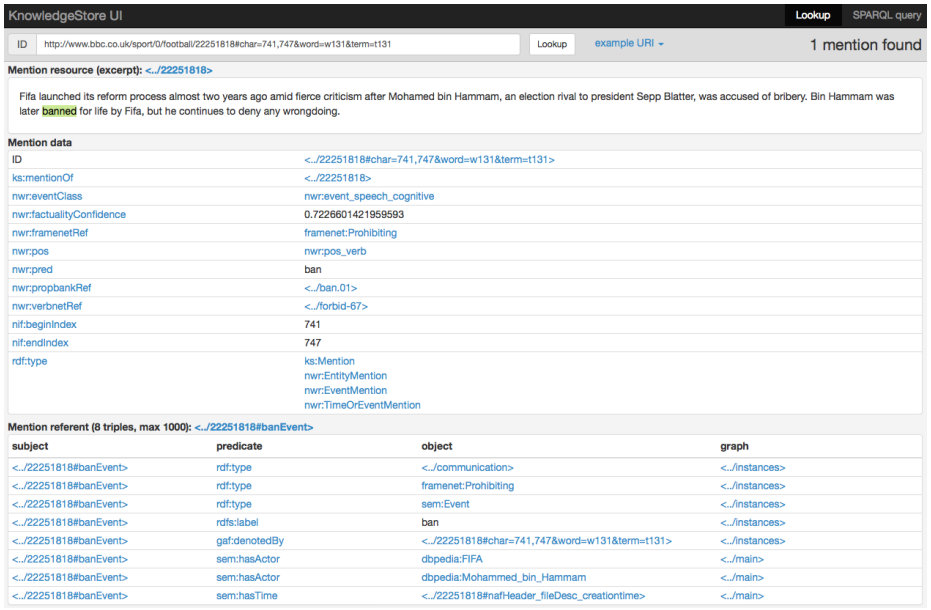

Fig. 2: KS UI. Lookup of a mention. Note the three boxes (Mention resource, Mention Data, Mention Referent) corresponding to the three representation layers of the KS.

In the last few months, several running instances of the KS were set-up (on a cluster of 5 average specs servers) and populated using the NewsReader Processing Pipeline [4] with contents coming from various domains: to name a few, one on the global automotive industry [5] (64K resources, 9M mentions, 316M entity triples), and one related to the FIFA World Cup (212K resources, 75M mentions, 240M entity triples). The latter, which will be used for the demo, was exploited during a Hackathon event [6], where 38 web developers accessed the KS to build their applications (over 30K SPARQL queries were submitted – on average 1 query/s, with peaks of 25 queries/s).

Acknowledgements The research leading to this paper was supported by the European Union's 7th Framework Programme via the NewsReader Project (ICT-316404).

References

1. Corcoglioniti, F., Rospocher, M., Cattoni, R., Magnini, B., Serafini, L.: Interlinking unstructured and structured knowledge in an integrated framework. In: 7th IEEE International Conference on Semantic Computing (ICSC), Irvine, CA, USA. (2013)
2. http://knowledgestore.fbk.eu
3. http://youtu.be/if1PRwSll5c
4. https://github.com/newsreader/
5. http://datahub.io/dataset/global-automotive-industry-news
6. http://www.newsreader-project.eu/come-hack-with-newsreader/