Dexter 2.0 - an Open Source Tool for Semantically Enriching Data

Salvatore Trani[1,4], Diego Ceccarelli[1,2], Claudio Lucchese[1],
Salvatore Orlando[1,3], and Raffaele Perego[1]

[1]ISTI–CNR, Pisa, Italy, [2]IMT Lucca, Italy, [3]Ca' Foscari - University of Venice,
[4]University of Pisa
{name.surname}@isti.cnr.it

Abstract. Entity Linking (EL) enables to automatically link unstructured data with entities in a Knowledge Base. Linking unstructured data (like news, blog posts, tweets) has several important applications: for example it allows to enrich the text with external useful contents or to improve the categorization and the retrieval of documents. In the latest years many effective approaches for performing EL have been proposed but only a few authors published the code to perform the task. In this work we describe Dexter 2.0, a major revision of our open source framework to experiment with different EL approaches. We designed Dexter in order to make it easy to deploy and to use. The new version provides several important features: the possibility to adopt different EL strategies at run-time and to annotate semi-structured documents, as well as a well-documented REST-API. In this demo we present the current state of the system, the improvements made, its architecture and the APIs provided.

1 Introduction

In the latest years many researchers proposed new techniques for performing *Entity Linking* (or *Wikification*) that consists of enriching a document with the entities that are mentioned within it. For example, consider the document in Figure 1: an EL framework first detects the pieces of text that are referring to an entity e.g., **Maradona**, **Argentina**, or **Belgium**, (usually called *mentions* or *spots*); this step is known as *mention detection* or *spotting*. Then the system performs the *disambiguation* step: each spot is linked to an entity chosen from a list of candidates. The entity is represented by its URI or identifier in a knowledge base, in our case Wikipedia. As an example, in Figure 1 the correct entity for the spot **Argentina** is `http://en.wikipedia.org/wiki/Argentina_national_football_team`. Please note that linking the mention to the correct entity is not a trivial task since often a mention is *ambiguous*: indeed in the previous example **Argentina** is not referring to the most common sense (the country) but rather to the national football team.

In this demo we present the current status of Dexter, our open source framework for entity linking. We introduced Dexter one year ago [1] in order to provide

Maradona, [http://en.wikipedia.org/wiki/Diego_Maradona] played his first **World Cup tournament** [http://en.wikipedia.org/wiki/FIFA_World_Cup] in 1982 when **Argentina** [http://en.wikipedia.org/wiki/Argentina_national_football_team] played **Belgium** [http://en.wikipedia.org/wiki/Belgium_national_football_team] in the opening game of the **1982 Cup** [http://en.wikipedia.org/wiki/1982_FIFA_World_Cup] in **Barcelona** [http://en.wikipedia.org/wiki/Barcelona].

Fig. 1: Example of annotated document

a tool for implementing new EL methods, and for comparing or simply exploiting the existing EL methods on a common platform.

We designed the framework for researchers and students; Dexter is easy to deploy: it consists of a unique jar file without external dependencies, and some binary files representing the model. The user only has to run the program that will expose a web server providing both a Rest API and a web interface for performing EL. The framework is highly modular and it allows the developers to replace single parts of the EL process. It runs on commodity hardware and it requires only 3 gigabytes of memory.

2 Dexter Framework

2.1 Architecture

Dexter[1] is developed in Java, and is organized in several Maven[2] modules (as depicted in Figure 2):

Json-wikipedia[3] This module converts the Wikipedia XML Dump in a JSON Dump, where each line is a JSON record representing an article. The parser is based on the MediaWiki markup parser UKP[4]. While DBpedia only contains semistructured data extracted from the dump (mainly from the infoboxes) in RDF format, JSON-Wikipedia contains other fields, e.g., the section headers, the text (divided in paragraphs), the templates with their schema, text emphasized and so on. The module is designed to support different languages;

Dexter-Common Contains the domain objects, shared among all the modules of Dexter;

Dexter-Core The core implements the EL pipeline (illustrated on the right of Figure 2): the text is first processed by a *Spotter*, that produces a list of *spot matches*. Each spot match contains the offset of the match in the document, the list of entities that could be represented by the spot (produced by an *Entity Ranker*) and other features useful to perform the linking. The spot matches are then processed by a *Disambiguator* that for each spot tries to select the correct entity in the list of candidates (often relying on a *Relatedness* function, that estimates the semantic distance between two entities);

[1] The project page is http://dexter.isti.cnr.it, the website also presents a demo
[2] http://maven.apache.org/
[3] json-wikipedia is available at https://github.com/diegoceccarelli/json-wikipedia
[4] http://www.ukp.tu-darmstadt.de/software/jwpl/

Dexter-Webapp exposes a REST API for performing the annotations. It also implements a simple web interface for performing a demo. The current version of the REST API is briefly described in Table 1, and it is organized in 4 logical categories: the **Annotate API**, used for annotating a document, the **Spot API** that allows to retrieve the candidate spots in a document and to visualize their features, the **Graph API** and the **Category API** that allow to browse respectively the Wikipedia article's link graph and the category graph. The current API is available and testable. We provide a well written documentation for each method, in a web page that also allows the user to test the service;

Dexter-Client a simple client to perform EL from a client machine, implicitly calling the REST API.

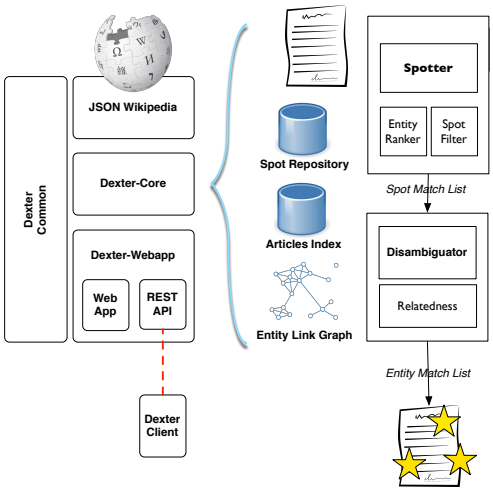

Fig. 2: Dexter Architecture

2.2 Novel Features

Since the first version we added the possibility to replace and combine different versions of the components of the system (the spotter, the disambiguator, the relatedness function *etc.*). An EL annotation can then be performed providing to the linker the symbolic names of the components that the developer wants to use (the spotter x, the disambiguator y ...). More in detail, in the `annotate` REST API the spotter and the disambiguator components are parameters, allowing to make use of different EL techniques at run-time. Another interesting feature is the possibility to annotate *semi-structured documents*; the previous version, as well as other EL frameworks, annotates only flat text, i.e., a plain string. In the new version we added the possibility to annotate documents composed by several fields (*e.g.*, title, headlines, paragraphs); when developing a new spotter/disambiguator, a researcher can exploit this information about the structure

of a document. It is worth to observe that in the new version each candidate spot contains also the field where the match was performed. The system also offers a Category API (extracted from the DBpedia categories).

Annotate API	
`api/rest/annotate`	Performs the EL on a given text
`api/rest/get-desc`	Given the Wikipedia Id of an entity, returns an object describing the entity (title, short description, ...)
Spot API	
`api/rest/spot`	Performs only the spotting step on the document, returning a list of mentions detected in the document, and for each mention some useful features and the list of possible candidate entities
`api/rest/get-spots`	Given an entity returns all the spots used in the Wikipedia dump for referring to the entity, for example given the entity **Mona_lisa**, it returns `mona lisa`,`gioconda`, `la joconde` ...
Graph API	
`api/rest/get-target-entities`	Returns the entities linked by the given entity
`api/rest/get-source-entities`	Returns the entities that link to the given entity
`api/rest/relatedness`	Returns the semantic relatedness between two entities (by default using the Milne and Witten formula [2])
Category API	
`api/rest/get-parent-categories`	Given an category, returns its parent categories
`api/rest/get-child-categories`	Given an category, returns its child categories
`api/rest/get-entity-categories`	Given an entity, returns its categories
`api/rest/get-belonging-entities`	Given a category, returns the entities belonging to the category

Table 1: The current version of the Dexter's REST-API

Finally, we released a framework for evaluating the quality of the annotations and comparing our framework with the others[5]. We are also planning to integrate our tool with the NERD framework [3].

The demonstration will present the main functionalities provided by our system. We will illustrate how to use the API, how to deploy the system on a server, how to extend the components, and we will show some applications built on top of Dexter.

Future work. We are planning to add several disambiguators and spotters proposed in literature, and produce a performance comparison on different types of datasets.

Acknowledgements This work was partially supported by the EU project E-CLOUD (no. 325091), the Regional (Tuscany) project SECURE! (POR CReO FESR 2007/2011), and the Regional (Tuscany) project MAPaC (POR CReO FESR 2007/2013).

References

1. D. Ceccarelli, C. Lucchese, S. Orlando, R. Perego, and S. Trani. Dexter: an open source framework for entity linking. In *ESAIR*, 2013.
2. D. Milne and I. H. Witten. Learning to link with wikipedia. In *Proceedings of CIKM*, 2008.
3. G. Rizzo and R. Troncy. Nerd: a framework for unifying named entity recognition and disambiguation extraction tools. In *Proceedings of EACL*, 2012.

[5] `https://github.com/diegoceccarelli/dexter-eval`