Dexter: an Open Source Framework for Entity Linking

Diego Ceccarelli
IMT Lucca
ISTI CNR, Pisa
Università di Pisa
diego.ceccarelli@isti.cnr.it

Claudio Lucchese
ISTI CNR, Pisa
claudio.lucchese@isti.cnr.it

Raffaele Perego
ISTI CNR, Pisa
raffaele.perego@isti.cnr.it

Salvatore Orlando
Università Ca' Foscari Venezia
ISTI CNR, Pisa
orlando@unive.it

Salvatore Trani
ISTI CNR, Pisa
Università di Pisa
salvatore.trani@isti.cnr.it

ABSTRACT

We introduce Dexter, an open source framework for entity linking. The entity linking task aims at identifying all the small text fragments in a document referring to an entity contained in a given knowledge base, e.g., Wikipedia. The annotation is usually organized in three tasks. Given an input document the first task consists in discovering the fragments that could refer to an entity. Since a mention could refer to multiple entities, it is necessary to perform a disambiguation step, where the correct entity is selected among the candidates. Finally, discovered entities are ranked by some measure of relevance. Many entity linking algorithms have been proposed, but unfortunately only a few authors have released the source code or some APIs. As a result, evaluating today the performance of a method on a single subtask, or comparing different techniques is difficult. In this work we present a new open framework, called Dexter, which implements some popular algorithms and provides all the tools needed to develop any entity linking technique. We believe that a shared framework is fundamental to perform fair comparisons and improve the state of the art.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information Filtering, Search process*

Keywords

Entity linking; Annotations; Evaluation.

1. INTRODUCTION

Most Web documents currently do not contain semantic annotations, and they are commonly modeled as simple bags of words. One of the main approaches for enhancing search effectiveness on these documents consists in automatically enriching them with the most relevant related entities [?]. Given a plain text, the Entity Linking task consists in identifying small fragments of text (in the following interchangeably called spots or mentions) referring to any entity (represented by a URI) that is listed in a given knowledge base. Usually the task is performed in three steps: i) **Spotting**: a set of candidate mentions is detected in the input document, and for each mention a list of candidate entities is retrieved; ii) **Disambiguation**: for each spot associated with more than one candidate, a single entity is selected to be linked to the spot; iii) **Ranking**: the list of entities detected is ranked according to some policy, e.g., annotation confidence.

Wikipedia is a perfect resource for performing this task in a open domain context. Each article can be considered as an entity, and possible mentions for an entity can be retrieved considering all the anchor texts of the links that point to that entity. Many approaches use Wikipedia for performing the task and also the two main entity linking tracks: the Knowledge Base Population[1] organized by the U.S. National Institute of Standards and Technology (NIST), and the TREC Knowledge Base Acceleration[2].

The spotter detects the entities by looking in the text for any fragment matching any of the Wikipedia mentions, and therefore potentially referring to a entity. Once we retrieve the spots, the main challenge to cope with is the ambiguity of natural language mentions. In fact a fragment could refer to more than one entity. For example consider the sentence:

*On July 20, 1969, the **Apollo 11** astronauts - **Neil Armstrong**, **Michael Collins**, and **Edwin "Buzz" Aldrin Jr.** - realized **President Kennedy**'s dream.*

It is quite easy to map the spot `Michael Collins` to the entity *Michael Collins*[3], since in Wikipedia there are 98 anchors linking to that page. On the other hand, it could be not so easy for a software to decide if `Michael Collins` is an astronaut, an Irish leader or the president of the Irish provisional government in 1922.

ESAIR'13, October 28, 2013, San Francisco, CA, USA.
Copyright 2013 ACM 978-1-4503-2413-7/13/10
http://dx.doi.org/10.1145/2513204.2513212 ...$15.00.

[1]`http://www.nist.gov/tac/2013/KBP/index.html`
[2]`http://trec-kba.org/`
[3]`http://en.wikipedia.org/wiki/Michael_Collins_(astronaut)`

Several techniques were recently proposed for annotating documents with entities [?, ?, ?, ?]. These techniques usually rely on features extracted from Wikipedia for performing the disambiguation. Ratinov *et al.* [?] distinguish the methods between *local* and *global* approaches. Local approaches exploit clues such as the textual similarity between the document and each candidate disambiguation's Wikipedia page [?], or the probability (*commonness*, estimated from the Wikipedia dump) for a mention to refer to a particular entity [?, ?]. Global approaches try to optimize the coherence among the entities in the documents. Coherence is a subtle concept often modeled as the notion of *relatedness* between two entities (or by their semantic distance). Many approaches were proposed for estimating the relatedness of two entities. Some approaches are based on the similarity between the textual description of the articles [?], or on comparing the categories to which they belong. Many methods use measures defined on the Wikipedia entity link graph. A popular measure was defined by Milne and Witten [?] that model the relatedness between two entities e_1 and e_2 as a variation of the Jaccard distance between the set of the entities that link to e_1 and e_2. In [?] several techniques for computing relatedness with the Wikipedia graph are experimented.

Performing a fair comparison among these techniques is very hard. To the best of our knowledge, only a few authors released the source code of their methods (WikiMiner by Milne and Witten[4] and Aida[5] by Hoffart *et al.*), or provide a REST API for annotating documents using their method (TAGME by Ferragina and Scaiella). Hachey *et al.* [?] propose a framework (not published) for entity linking in which they implemented and compared three methods in the state of the art. In their experiments they found that spotting is more important than disambiguation and that mixing the spotting and the disambiguation strategies of different methods can lead to interesting results.

A good performance obtained in spotting may heavily impact on the performance of the whole system, as well as using a different dump of Wikipedia (i.e., old dumps contain less entities, but also have less ambiguity for each spot), or removing all the candidate entities of a spot which have commonness below a certain threshold, or spots with a low *link probability* (probability to be a link, estimated as the occurrences of the text as anchor divided the occurrences as pure text). Moreover, *efficiency* of the proposed methods is in many cases ignored even if, depending on the use case, it could be of paramount importance (for example for annotating a huge repository of webpages).

For these reasons, we strongly believe that for this kind of research it is important to share a unique framework where spotting, disambiguation and ranking are well separated and easy to isolate in order to study their performance. Our proposal goes in the direction of developing an open and flexible framework for implementing entity linking strategies. The framework, called Dexter, provides several facilities for implementing annotation strategies. Differently from other approaches which require high-performance hardware or to install additional software (e.g., databases), Dexter is a standalone program designed to easily work with a small effort from the user.

[4]http://wikipedia-miner.cms.waikato.ac.nz/
[5]http://www.mpi-inf.mpg.de/yago-naga/aida/

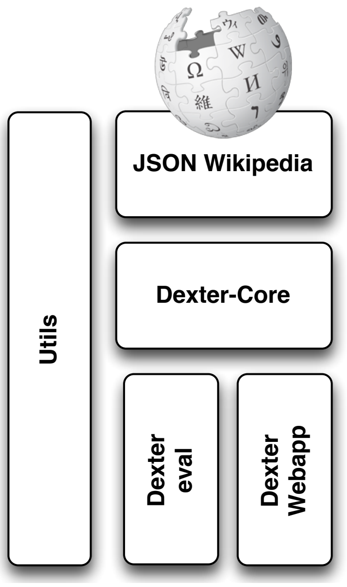

Figure 1: Dexter architecture

2. DEXTER

Dexter is developed in Java, and is organized in several Maven modules as shown in Figure 1. The **Utils** module contains utility code for performing general operations: logging, compressed file I/O, management of properties, writing command line programs, etc. **Dexter-Eval** implements utilities for performing benchmarks of the methods. The module relies on the trec-eval framework[6]. **Dexter-Webapp** exposes REST API for performing the annotations. It also implements a simple web interface for deploying demo and user studies. In the following we will briefly describe the two main modules: the Json-Wikipedia and the Core.

2.1 Json Wikipedia

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[7]. DBPedia only contains semistructured data extracted from the dump (mainly from the infoboxes) in RDF format, while JSON-Wikipedia contains other fields, e.g., the section headers, the text (divided in paragraphs), the templates with their schema, emphasized and so on. These fields are not properly attributes of an entity but they can be useful for performing the annotation or a posteriori, for presenting an annotated entity. Another main difference is that in our dump all the articles are converted to JSON records, so we have also Disambiguation records, Redirect records, Category records etc. (the type of the record is encoded in a *type* field). The module is designed to manage different languages. Given a locale file describing how disambiguations, categories, redirects, etc are denoted in a given language, Dexter parses the XML dump in the specified language and produces the JSON dump. All the languages will share the same JSON schema. This should simplify the development of techniques combining different languages. Moreover, JSON is easy to parse and to be used

[6]http://trec.nist.gov/trec_eval/
[7]http://www.ukp.tu-darmstadt.de/software/jwpl/

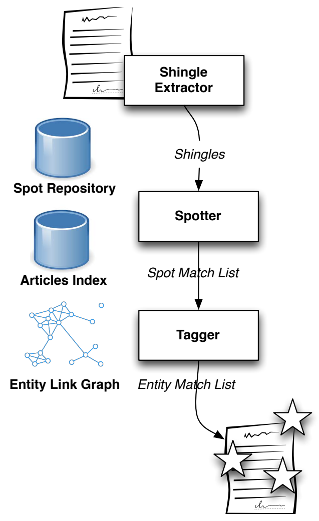

Figure 2: The Dexter-Core module

in scalable Map Reduce frameworks like Hadoop, Pig or Cascading. In the future, we plan to improve the creation of entity linking models to be used in map reduce jobs.

2.2 Dexter Core

The core contains all the code to manipulate the JSON dump in order to generate: the *spot repository*, the *article index*, and the *entity graph*. The main components are depicted in Figure 2. The spot repository contains all the anchors used in Wikipedia for intra-linking the articles. For each spot, the spot index contains the link probability and the list of entities that could be represented by the spot (i.e, all the articles targeted by that particular spot). The article index is a multi-field index of the article, built by using Lucene. The entity graph stores the connections among the entities. The Shingle Extractor produces a list of possible spots from a given document. At the time of writing, the extractor produces all the possible n-grams of terms, where n ranges from one to six. The Spotter then associates with each fragment a list of entity candidates (if any) using the spot repository. Finally the Tagger takes in input the list of spot matches produced by the spotter and selects the best entity for each spot, performing the disambiguation if the spot has more than one candidate. Disambiguation can be performed using the features provided by the spot repository, the article index, and the entity graph. The Tagger outputs an entity match list, with the position in the original text of each annotated entity and a confidence score.

3. FUTURE ROADMAP

We implemented three methods in Dexter: TAGME [?], the collective linking approach [?] and WikiMiner [?]. We plan to implement other methods in the state of the art in order to show their performance and efficiency on all the datasets that will be able to obtain. Of course, we will give our best to implement the proposed approaches as described in the papers, and we will be happy to publish corrections or better implementations. Currently we are working on refactoring the code, and on writing documentation on the project page [8]. We would like to provide a demo of our system at the workshop.

We are also investigating on evaluation datasets. We tried several datasets, and we experienced that switching from a dataset to another can change the ranking of the entity linking methods. The best method on a certain dataset may not be the first on another dataset. This seems reasonable if the datasets contain different types of documents (e.g., TAGME works better on short texts, such as tweets), but it seems to happen also within the same type. We would like to investigate if there is a method able to annotate documents of different type and on different domains with a good performance, or if it is better to switch the model depending on either the type or the semantic domain. For these reasons, we are considering to perform a deep study on the available datasets in order to evaluate their quality, as we observed that some datasets miss relevant annotations or contain non-relevant annotations. It would be also interesting to build a new benchmark collection, using different types of documents not covered by copyright in order to facilitate the sharing.

Acknowledgements.

This work was partially supported by the EU projects InGeoCLOUDS (no. 297300), MIDAS (no. 318786), E-CLOUD (no. 325091) and the Regional (Tuscany) project SECURE! (FESR PorCreo 2007-2011).

[8]`http://dexter.isti.cnr.it`