NERD meets NIF: Lifting NLP Extraction Results to the Linked Data Cloud

Giuseppe Rizzo
EURECOM, France
Politecnico di Torino, Italy
giuseppe.rizzo@eurecom.fr

Raphaël Troncy
EURECOM, France
raphael.troncy@eurecom.fr

Sebastian Hellmann,
Martin Bruemmer
Universität Leipzig, Germany
hellmann@informatik.uni-leipzig.de

ABSTRACT

We have often heard that data is the new oil. In particular, extracting information from semi-structured textual documents on the Web is key to realize the Linked Data vision. Several attempts have been proposed to extract knowledge from textual documents, extracting named entities, classifying them according to pre-defined taxonomies and disambiguating them through URIs identifying real world entities. As a step towards interconnecting the Web of documents via those entities, different extractors have been proposed. Although they share the same main purpose (extracting named entity), they differ from numerous aspects such as their underlying dictionary or ability to disambiguate entities. We have developed NERD, an API and a front-end user interface powered by an ontology to unify various named entity extractors. The unified result output is serialized in RDF according to the NIF specification and published back on the Linked Data cloud. We evaluated NERD with a dataset composed of five TED talk transcripts, a dataset composed of 1000 New York Times articles and a dataset composed of the 217 abstracts of the papers published at WWW 2011.

Categories and Subject Descriptors

I.2.7 [**Artificial Intelligence**]: [Natural Language Processing - Language parsing and understanding]

General Terms

Measurement, Performance, Experimentation, Web

Keywords

Named Entity extractors, Information extraction, Linked Data, Evaluation

1. INTRODUCTION

The Web of Data is often illustrated as a fast growing cloud of interconnected dataset representing information about barely everything [6]. The Web hosts also millions of semi-structured texts such as scientific papers, news articles as well as forum and archived mailing list threads or (micro-)blog posts. This information has usually a rich semantic structure which is clear for the author but that remains mostly hidden to computing machinery. Named entity and information extractors aim to bring such a structure from those free texts. They provide algorithms for extracting semantic units identifying the name of people, organizations, locations, time references, quantities, etc. and classifying them according to predefined schema, increasing discoverability (e.g. through faceted search), reusability and the utility of information.

Since the 90's, an increasing emphasis has been given to the evaluation of NLP techniques. Hence, the Named Entity Recognition (NER) task has been developed as an essential component of the Information Extraction field. Initially, these techniques focused on identifying atomic information unit in a text, named entities, later on classified into predefined categories (also called context types) by classification techniques, and linked to real world objects using web identifiers. Such a task is called Named Entity Disambiguation. Knowledge bases affect the disambiguation task in several ways, because they provide the final disambiguation point where the information is linked. Recent methods leverage knowledge bases such as DBpedia [3], Freebase[1] or YAGO [21] since they contain many entries corresponding to real world entities and classified according to exhaustive classification schemes. A certain number of tools have been developed to extract structured information from text resources classifying them according to pre-defined taxonomies and disambiguating them using URIs. In this work, we aim to evaluate tools which provide such an online computation: AlchemyAPI[2], DBpedia Spotlight[3], Evri[4], Extractiv[5], Lupedia[6], OpenCalais[7], Saplo[8], Wikimeta[9], Yahoo! Content Analysis (YCA)[10] and Zemanta[11]. They represent a clear

LDOW2012, April 16, 2012, Lyon, France.

[1] http://www.freebase.com/
[2] http://www.alchemyapi.com
[3] http://dbpedia.org/spotlight
[4] http://www.evri.com/developer/index.html
[5] http://extractiv.com
[6] http://lupedia.ontotext.com
[7] http://www.opencalais.com
[8] http://saplo.com
[9] http://www.wikimeta.com
[10] http://developer.yahoo.com/search/content/V2/contentAnalysis.html
[11] http://www.zemanta.com

opportunity for the Linked Data community to increase the volume of interconnected data. Although these tools share the same purpose – extracting semantic units from text – they make use of different algorithms and training data. They generally provide a potential similar output composed of a set of extracted named entities, their type and potentially a URI disambiguating each named entity. The output vary in terms of data model used by the extractors. Hence, we propose the Named Entity Recognition and Disambiguation (NERD) framework which unifies the output results of these extractors, lifting them to the Linked Data Cloud using the new NIF specification.

These services have their own strengths and shortcomings but, to the best of our knowledge, few scientific evaluations have been conducted to understand the conditions under which a tool is the most appropriate one. This paper attempts to fill this gap. We have performed quantitative evaluations conducted on three different datasets covering different type of textual documents: a dataset composed of transcripts of five *TED*[12] talks, a dataset composed of 1000 news articles from *The New York Times*[13] and a dataset composed of the 217 abstracts of the papers published at *WWW 2011*[14]. We present statistics to underline the behavior of such extractors in different scenarios and group them according to the NERD ontology. We have developed the NERD framework, available at `http://nerd.eurecom.fr` to perform systematic evaluation of NE extractors.

The remainder of this paper is organized as follows. In section 2, we introduce a factual comparison of the named entity extractors investigated in this work. We describe the NERD framework in section 3 and we highlight the importance to have an output compliant with the Linked Data principles in section 4. Then, we describe the experimental results we obtained in section 5 and in section 6, we propose an overview on Named Entity recognition and disambiguation techniques. Finally, we give our conclusions and outline future work in section 7.

2. FACTUAL COMPARISON OF NAMED ENTITY EXTRACTORS

The NE recognition and disambiguation tools vary in terms of response granularity and technology used. As granularity, we define the way how the extraction algorithm works: One Entity per Name (OEN) where the algorithm tokenizes the document in a list of exclusive sentences, recognizing the dot as a terminator character, and for each sentence, detects named entities; and One Entity per Document (OED) where the algorithm considers the bag of words from the entire document and then detects named entities, removing duplicates for the same output record *(NE, type, URI)*. Therefore, the result set differs from the two approaches.

Table 1 provides an extensive comparison that take into account the technology used: algorithms used to extract NE, supported languages, ontology used to classify the NE, dataset for looking up the real world entities and all the technical issues related to the online computation such as the maximum content request size and the response format. We also report whether a tool provides the position where an NE is found in the text or not. We distinguish four cases: *char offset* considering the text as a sequence of characters, it reports the char index where the NE starts and the length (number of chars) of the NE; *range of chars* considering the text as a sequence of characters, it reports the start index and the end index where the NE appears; *word offset* the text is tokenized considering any punctuation, it reports the word number after the NE is located (this counting does not take into account the punctuation); *POS offset* the text is tokenized considering any punctuation, it reports the number of part-of-a-speech after the NE is located.

We performed an experimental evaluation to estimate the max content chunk supported by each API, creating a simple application that is able to send to each extractor a text of 1KB initially. In case that the answer was correct (HTTP status 20x), we performed one more test increasing of 1 KB the content chunk. We iterated this operation until we received the answer "text too long". Table 1 summarizes the factual comparison of the services involved in this study. The *** means the value has been estimated experimentally (as the content chunk), *+* means a list of other sources, generally identifiable as any source available within the Web, finally *N/A* means not available.

3. THE NERD FRAMEWORK

NERD is a web framework plugged on top of various NER extractors. Its architecture follows the REST principles [7] and includes an HTML front-end for humans and an API for computers to exchange content in JSON (another serialization of NERD output will be detailed in the section 4). Both interfaces are powered by the NERD REST engine.

3.1 The NERD Data Model

We propose the following data model that encapsulates the common properties for representing NERD extraction results. It is composed of a list of entities for which a label, a type and a URI is provided, together with the mapped type in the NERD taxonomy, the position of the named entity, the confidence and relevance scores as they are provided by the NER tools. The example below shows this data model (for the sake of brevity, we use the JSON syntax):

```
"entities": [{
     "entity":"Tim Berners-Lee",
     "type":"Person",
     "uri":"http://dbpedia.org/resource/
         Tim_berners_lee",
     "nerdType":"http://nerd.eurecom.fr/ontology#
         Person",
     "startChar":30,
     "endChar":45,
     "confidence":1,
     "relevance":0.5
    }]
```

3.2 The NERD REST API

The REST engine runs on Jersey[15] and Grizzly[16] technologies. Their extensible frameworks enable to develop several components and NERD is composed of 7 modules namely authentication, scraping, extraction, ontology mapping, store, statistics and web. The authentication takes as input a FOAF profile of a user and links the evaluations with the user who performs them (we are freezing an OpenID implementation and it will replace soon the simple authentica-

[12] `http://www.ted.com`

[13] `http://www.nytimes.com`

[14] `http://www.www2011india.com`

[15] `http://jersey.java.net`

[16] `http://grizzly.java.net`

	AlchemyAPI	DBpedia Spotlight	Evri	Extractiv	Lupedia	OpenCalais	Saplo	Wikimeta	YCA	Zemanta
Granularity	OED	OEN	OED	OEN	OEN	OED	OED	OEN	OEN	OED
Language support	English French German Italian Portuguese Russian Spanish Swedish	English German (partial) Portuguese (partial) Spanish (partial)	English Italian	English	English French Italian	English French Spanish	English Swedish	English French Spanish	English	English
Restriction on academic use (calls/day)	30000	unlimited	3000	1000	unlimited	50000	1333	unlimited	5000	10000
Sample clients	C/C++ C# Java Perl PHP-5 Python Ruby	Java Javascript PHP	Action Script Java PHP	Java	N/A	Java	Java Javascript PHP Python	Java Perl	Javascript PHP	C# Java Javascript Perl PHP Python Ruby
API interface	CLI JAX-RS SOAP	AJAX CLI JAX-RS SOAP	AJAX JAX-RS	AJAX CLI JAX-RS	CLI JAX-RS	AJAX CLI JAX-RS SOAP	AJAX CLI JAX-RS	CLI JAX-RS	JAX-RS CLI	AJAX CLI JAX-RS
Content chunk	150KB*	452KB*	8KB*	32KB*	20KB*	8KB*	26KB*	80KB*	7769KB*	970KB*
Response format	JSON Microformats XML RDF	HTML+uF(rel-tag) JSON RDF XHTML+RDFa XML	GPB HTML JSON RDF	HTML JSON RDF XML	HTML JSON RDFa XML	JSON Microformats N3 Simple Format	JSON	JSON XML	JSON XML	JSON WNJSON RDF XML
Entity type number	324	320	300*	34	319	95	5	7	13	81
Entity position	N/A	char offset	N/A	word offset	range of chars	char offset	N/A	POS offset	range of chars	N/A
Classification ontologies	Alchemy	DBpedia FreeBase Schema.org	Evri	DBpedia	DBpedia	OpenCalais	Saplo	ESTER (partial)	Yahoo	FreeBase
Deferencable vocabularies	DBpedia Freebase US Census GeoNames UMBEL OpenCyc YAGO MusicBrainz CIA Factbook CrunchBase	DBpedia	Evri	DBpedia	DBpedia LinkedMDB	OpenCalais	N/A	DBpedia Geonames CIAFactbook Wikicompanies others+	Wikipedia	Wikipedia IMDB MusicBrainz Amazon YouTube TechCrunch MusicBrainz Twitter MyBlogLog Facebook others+

Table 1: Factual information about 10 extractors under investigation

tion system working right now). The scraping module takes as input the URI of an article and extracts all its raw text. Extraction is the module designed to invoke the external service APIs and collect the results. Each service provides its own taxonomy of named entity types it can recognize. We therefore designed the NERD ontology which provides a set of mappings between these various classifications. The ontology mapping is the module in charge to map the classification type retrieved to our ontology. The store module saves all evaluations according to the schema model we defined in the NERD database. The statistic module enables to extract data patterns form the user interactions stored in the database and to compute statistical scores such as the Fleiss Kappa score and the precision measure. Finally, the web module manages the client requests, the web cache and generates HTML pages.

Plugged on the top of this engine, there is an API interface[17]. It is developed following the REST principles and it has been implemented to enable programmatic access to the NERD framework. It follows the following URI scheme (the base URI is `http://nerd.eurecom.fr/api`):

`/document` : GET, POST, PUT methods enable to fetch, submit or modify a document parsed by the NERD framework;

`/user` : GET, POST methods enable to insert a new user to the NERD framework and to fetch account details;

`/annotation/{extractor}` : POST method drives the annotation of a document. The parametric URI allows to pilot the extractors supported by NERD;

`/extraction` : GET method allows to fetch the output described in section 3.1;

`/evaluation` : GET method allows to retrieve a statistic interpretation of the extractor behaviors.

3.3 The NERD Ontology

Although these tools share the same goal, they use different algorithms and different dictionaries which makes hard their comparison. We have developed the NERD ontology, a set of mappings established manually between the taxonomies of NE types. Concepts included in the NERD ontology are collected from different schema types: ontology (for DBpedia Spotlight, Lupedia, and Zemanta), lightweight taxonomy (for AlchemyAPI, Evri, and Yahoo!) or simple flat type lists (for Extractiv, OpenCalais, Saplo, and Wikimeta). The NERD ontology tries to merge the linguistic community needs and the logician community ones: we developed a core set of axioms based on the Quaero schema [8] and we mapped similar concepts described in the other scheme. The selection of these concepts has been done considering the greatest common denominator among them. The concepts that do not appear in the NERD namespace are sub-classes of parents that end-up in the NERD ontology. This ontology is available at `http://nerd.eurecom.fr/ontology`. To summarize, a concept is included in the NERD ontology as soon as there are at least two extractors that use it. The NERD ontology becomes a reference ontology for comparing the classification task of NE extractors. We show an example mapping among those extractors below: the `City` type is considered as being equivalent to `alchemy:City`, `dbpedia-owl:City`, `extractiv:CITY`, `opencalais:City`, `evri:City` while being more specific than `wikimeta:LOC` and `zemanta:location`.

```
nerd:City a rdfs:Class ;
  rdfs:subClassOf wikimeta:LOC ;
  rdfs:subClassOf zemanta:location ;
  owl:equivalentClass alchemy:City ;
  owl:equivalentClass dbpedia-owl:City ;
  owl:equivalentClass evri:City ;
  owl:equivalentClass extractiv:CITY ;
  owl:equivalentClass opencalais:City .
```

3.4 The NERD UI

The user interface[18] is developed in HTML/Javascript. Its goal is to provide a portal where researchers can find information about the NERD project, the NERD ontology, and common statistics of the supported extractors. Moreover, it provides a personalized space where a user can create a developer or a simple user account. For the former account type, a developer can navigate through a dashboard, see his profile details, browse some personal usage statistics and get a programmatic access to the NERD API via a NERD key. The simple user account enables to annotate any web documents via its URI. The raw text is first extracted from the web source and a user can select a particular extractor. After the extraction step, the user can judge the correctness of each field of the tuple *(NE, type, URI, relevant)*. This is an important process which gives to NERD human feedbacks with the main purpose of evaluating the quality of the extraction results collected by those tools [17]. At the end of the evaluation, the user sends the results, through asynchronous calls, to the REST API engine in order to store them. This set of evaluations is further used to compute statistics about precision measures for each tool, with the goal to highlight strengths and weaknesses and to compare them [18]. The comparison aggregates all the evaluations performed and, finally, the user is free to select one or more evaluations to see the metrics that are computed for each service in real time.

4. NIF: AN NLP INTERCHANGE FORMAT

The *NLP Interchange Format* (NIF) is an RDF/OWL-based format that aims to achieve interoperability between *Natural Language Processing* (NLP) tools, language resources and annotations. The NIF specification has been released in an initial version 1.0 in November 2011 and describes how interoperability between NLP tools, which are exposed as NIF web services can be achieved. Extensive feedback was given on several mailing lists and a community of interest[19] was created to improve the specification. Implementations for 8 different NLP tools (e.g. UIMA, Gate ANNIE and DBpedia Spotlight) exist and a public web demo[20] is available.

In the following, we will first introduce the core concepts of NIF, which are defined in a String Ontology[21] (STR). We will then explain how NIF is used in NERD. The resulting properties and axioms are included into a Structured Sentence Ontology[22] (SSO). While the String Ontology is used

[17] http://nerd.eurecom.fr/api/application.wadl

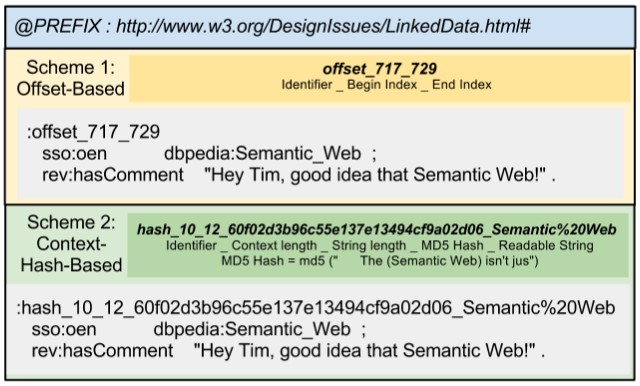
[18] http://nerd.eurecom.fr

[19] http://nlp2rdf.org/get-involved

[20] http://nlp2rdf.lod2.eu/demo.php

[21] http://nlp2rdf.lod2.eu/schema/string

[22] http://nlp2rdf.lod2.eu/schema/sso


@PREFIX : http://www.w3.org/DesignIssues/LinkedData.html#
Scheme 1: Offset-Based
offset_717_729
Identifier _ Begin Index _ End Index
Scheme 2: Context-Hash-Based
hash_10_12_60f02d3b96c55e137e13494cf9a02d06_Semantic%20Web
Identifier _ Context length _ String length _ MD5 Hash _ Readable String
MD5 Hash = md5 ("   The (Semantic Web) isn't jus")

```
:offset_717_729
  sso:oen          dbpedia:Semantic_Web  ;
  rev:hasComment   "Hey Tim, good idea that Semantic Web!" .
```

```
:hash_10_12_60f02d3b96c55e137e13494cf9a02d06_Semantic%20Web
  sso:oen          dbpedia:Semantic_Web  ;
  rev:hasComment   "Hey Tim, good idea that Semantic Web!" .
```

Figure 1: NIF URI schemes: Offset (top) and context-hashes (bottom) are used to create identifiers for strings

to describe the relations between strings (i.e. Unicode characters), the SSO collects properties and classes to connect strings to NLP annotations and NER entities as produced by NERD.

4.1 Core Concepts of NIF

The motivation behind NIF is to allow NLP tools to exchange annotations about documents in RDF. Hence, the main prerequisite is that parts of the documents (i.e. strings) are referenceable by URIs, so that they can be used as subjects in RDF statements. We call an algorithm to create such identifiers *URI Scheme*: For a given text t (a sequence of characters) of length $|t|$ (number of characters), we are looking for a *URI Scheme* to create a URI, that can serve as a *unique* identifier for a substring s of t (i.e. $|s| \leq |t|$). Such a substring can (1) consist of adjacent characters only and it is therefore a unique character sequence within the text, if we account for parameters such as context and position or (2) derived by a function which points to several substrings as defined in (1).

NIF provides two URI schemes, which can be used to represent strings as RDF resources. We focus here on the first scheme using offsets. In the top part of Figure 1, two triples are given that use the following URI as subject:

```
http://www.w3.org/DesignIssues/LinkedData.html#
                offset_717_729
```

According to the above definition, the URI points to a substring of a given text t, which starts at character index 717 until the index 729 (counting all characters). NIF currently mandates that the whole string of the document has to be included in the RDF output as an `rdf:Literal` to serve as the reference point, which we will call *inside context* formalized using an OWL class called `str:Context`. The term *document* would be inappropriate to capture the real intention of this concept as we would like to refer to an arbitrary grouping of characters forming a unit, which could also be applied to a *paragraph* or a *section* and is highly dependent upon the *wider context* in which the string is actually used such as a Web document reachable via HTTP.

To appropriately capture the intention of such a class, we will distinguish between the notion of outside and inside context of a piece of text. The inside context is easy to explain and formalise, as it is the text itself and therefore it provides a reference context for each substring contained in the text (i.e. the characters before or after the substring). The outside context is more vague and is given by an outside observer, who might arbitrarily interpret the text as a "book chapter" or a "book section".

The class `str:Context` now provides a clear reference point for all other relative URIs used in this context and blocks the addition of information from a larger (outside) context by definition. For example, `str:Context` is disjoint with `foaf:Document` since labeling a context resource as a document is an information which is not contained within the context (i.e. the text) itself. It is legal, however, to say that the string of the context occurs in (`str:occursIn`) a `foaf:Document`. Additionally, `str:Context` is a subclass of `str:String` and therefore its instances denote Unicode text as well. The main benefit to limit the context is that an OWL reasoner can now infer that two contexts are the same, if they consist of the same string, because an inverse-functional data type property (`str:isString`) is used to attach the actual text to the context resource.

```
:offset_0_26546   a   str:Context  ;
#the exact retrieval method is left underspecified
  str:occursIn <http://www.w3.org/DesignIssues/
       LinkedData.html> ;
# [...] are all 26547 characters as rdf:Literal
    str:isString    "[...]" .
:offset_717_729   a   str:String  ;
  str:referenceContext  :offset_0_26546  .
```

A complete formalisation is still work in progress, but the idea is explained here. The NIF URIs will be grounded on Unicode Characters (especially Unicode Normalization Form C[23]. For all resources of type *str:String*, the universe of discourse will then be the words over the alphabet of Unicode characters sometimes called $\sigma*$. Perspectively, we hope that this will allow for an unambiguous interpretation of NIF by machines.

Within the framework of RDF and the current usage of NIF for the interchange of output between NLP tools, the definition of the semantics is sufficient to produce a working system. However, problems arise if additional interoperability with Linked Data or fragment identifiers and ad-hoc retrieval of content from the Web is demanded. The actual retrieval method (such as content negotiation) to retrieve and validate the content for `#offset_717_729_Semantic%20Web` or its reference context is left underspecified as is the relation of NIF URIs to fragment identifiers for MIME types such as text/plain (see RFC 5147[24]). As long as such issues remain open, the complete text has to be included as RDF Literal.

4.2 Connecting String to Entities

For NERD, three relevant concepts have to be expressed in RDF and were included into the Structured Sentence Ontology (SSO): OEN, OED and NERD ontology types.

One Entity per Name (OEN) can be modeled in a straightforward way, by introducing a property `sso:oen`, which connects the string with an arbitrary entity.

```
:offset_717_729   sso:oen dbpedia:Semantic_Web  .
```

One Entity per Document (OED). As *document* is an outside interpretation of a string, the notion of context in NIF

[23] `http://www.unicode.org/reports/tr15/#Norm_Forms` counted in Code Units `http://unicode.org/faq/char_combmark.html#7`

[24] `http://tools.ietf.org/html/rfc5147`

has to be used. The property `sso:oec` is used to attach entities to a given context. We furthermore add the following DL-Axiom:

$$sso{:}oec \supseteq str{:}referenceContext^{-1} \circ sso{:}oen$$

As the property `oen` contains more specific information, `oec` can be inferred by the above role chain inclusion. In case the context is enlarged, any materialized information attached via the `oec` property needs to be migrated to the larger context resource.

The connection between NERD types and strings is done via a linked data URI, which disambiguates the entity. Overall three cases can be distinguished: In case, the NER extractor has provided a linked data URI to disambiguate the entity, we simply re-use it as in the following example:

```
# this URI points to the string "W3C"
:offset_23107_23110
  rdf:type              str:String ;
  str:referenceContext  :offset_0_26546 ;
  sso:oen               dbpedia:W3C ;
  str:beginIndex        "23107"  ;
  str:endIndex          "23110"  .
dbpedia:W3C rdf:type    nerd:Organization .
```

If, however, the NER extractor provides no disambiguation link at all or just a non-linked data URI for the entity (typically, the `foaf:homepage` of an organization such as *http://www.w3.org/*), we plan to mint a new linked data URI for the respective entity that could then be further sameAs with other identifiers in a data reconciliation process.

5. EVALUATIONS

We performed a quantitative experiment using three different datasets: a dataset composed of transcripts of five *TED* talks (different category of talks), a dataset composed of 1000 news articles from *The New York Times* (collected from 09/10/2011 to 12/10/2011), and a dataset composed of the 217 abstracts of the papers published at *WWW 2011* conference. The aim of these evaluations is to assess how these extractors perform in different scenarios, such as news articles, user generated content and scientific papers. The total number of document is 1222, with an average word number per document equal to 549. Each document was evaluated using 6 extractors supported by the NERD framework[25]. The final number of entities detected is 177, 823 and the average of unique entity number per document is 20.03. Table 2 shows statistics about grouped view according to the source documents.

We define the following variables: the number n_d of evaluated documents, the number n_w of words, the total number n_e of entities, the total number n_c of categories and n_u URIs. Moreover, we compute the following measures: word detection rate $r(w, d)$, i.e. the number of words per document, entity detection rate $r(e, d)$, i.e. the number of entities per document, the number of entities per word $r(e, w)$, the number of categories per entity $r(c, e)$ (this measure has been computed removing not relevant labels such as “null” or “LINKED_OTHER”) and the number of URIs per entity $r(u, e)$.

	WWW2011	TED	NYTimes
n_d	217	5	1,000
n_w	38,062	13,381	62,0567
r_w	175.4	2,676.2	620.567
n_e	12,266	1,441	164,116
r_e	56.53	288.2	164.1

Table 2: Statistics about the three dataset used in the quantitative experiment, grouped according to the source where documents were collected.

5.1 User Generated Content

In this experiment, we focus on the extractions performed by all tools for 5 *TED* talk transcripts. The goal is to find out NE extraction ratio for user generated content, such as speech transcripts of videos. First, we propose general statistics about the extraction task and then, we focus on the classification, showing statistics grouped according to the NERD ontology. DBpedia Spotlight classifies each resource according three different schema (see Table 1). For this experiment, we consider only the results which belong to the DBpedia ontology. The total number of documents is 5, with an overall number of total words equal to 13, 381. The word detection rate per document $r(w, d)$ is equal to 2, 676.2 with an overall number of entities equal to 1, 441, and the $r(e, d)$ is 288.2. Table 3 shows the statistics about the computation results for all extractors. DBpedia Spotlight is the extractor which provides the highest number of NE and disambiguated URIs. These values show the ability from this extractor to locate NE and to exploit the large cloud of LOD resources. In parallel, it is crucial noting that it is not able to classify these resources, although it uses a deep classification schema. All the extractors show high ability for the classification task, except Zemanta as shown by the $r(c, e)$. Contrarily, Zemanta shows strong ability to disambiguate NE via URI identification, as shown by $r(u, e)$. It is worth noting OpenCalais and Evri have almoust the same performances of Zemanta. The last part of this experiment consists in aligning all the classification types provided by these extractors, while performing the analysis of *TED* talk transcripts, using the NERD ontology. For the sake of brevity, we report all the grouping results according to 6 main concepts: Person, Organization, Country, City, Time and Number. Table 4 shows the comparison results. AlchemyAPI classifies a higher number of Person, Country and City than all the others. In addition, OpenCalais obtains good performances to classify all the concepts except Time and Number. It is worth noting that Extractiv is the only extractor able to locate and classify Number and Time. In this grouped view, we consider all the results classified with the 6 main classes and we do not take into account all potentially inferred relationships. This is why the Evri results contrast with what is showed in the Table 3. Indeed, Evri provides a precise classification about Person such as Journalist, Physicist, Technologist but it does not describe the same resource as a sub-classes of the Person axiom.

5.2 Scientific Documents

In this experiment, we focus on the extraction performed by all tools for the 217 abstract papers published at the *WWW 2011* conference, with the aim to seek NE extraction patterns for scientific contributions. The total number

[25] At the time this evaluation has been conducted Lupedia, Saplo, Wikimeta and YCA were not part of the NERD framework.

	n_e	n_c	n_u	$r(e,d)$	$r(e,w)$	$r(c,e)$	$r(u,e)$
AlchemyAPI	141	**141**	71	28.2	0.01	**1**	0.504
DBpedia Spotlight	**810**	0	**624**	**162**	**0.06**	0	0.77
Evri	120	120	113	24	0.009	**1**	0.942
Extractiv	60	53	22	12	0.004	0.883	0.367
OpenCalais	163	136	157	32.6	0.012	0.834	0.963
Zemanta	50	17	49	10	0.003	0.34	**0.98**

Table 3: Statistics about computation results for the sources coming from *TED* talks of all extractors used in the comparison.

	AlchemyAPI	DBpedia Spotlight	Evri	Extractiv	OpenCalais	Zemanta
Person	42	-	10	6	27	4
Organization	15	-	-	-	20	1
Country	16	-	11	1	16	3
City	14	-	3	3	7	-
Time	-	-	-	1	-	-
Number	-	-	-	5	-	-

Table 4: Number of axioms aligned for all the extractors involved in the comparison according to the NERD ontology for the sources coming from *TED* talks.

	n_e	n_c	n_u	$r(e,d)$	$r(e,w)$	$r(c,e)$	$r(u,e)$
AlchemyAPI	323	171	39	1.488	0.008	0.529	0.121
DBpedia Spotlight	**3,699**	0	**1,062**	**17.04**	**0.097**	0	0.287
Evri	282	282	167	1.299	0.007	**1**	0.592
Extractiv	1,345	725	415	6.198	0.035	0.539	0.309
OpenCalais	1,158	**1,158**	713	5.337	0.03	**1**	**0.616**
Zemanta	1,748	97	757	8.055	0.046	0.055	0.433

Table 5: Statistics about extraction results for the 217 abstract papers published at the *WWW 2011* conference of all extractors used in the comparison.

	AlchemyAPI	DBpedia Spotlight	Evri	Extractiv	OpenCalais	Zemanta
Person	17	-	12	6	6	1
Organization	20	-	24	-	5	-
Country	9	-	8	14	7	6
City	4	-	3	8	9	-
Time	-	-	-	-	-	-
Number	-	-	-	184	-	-

Table 6: Number of axioms aligned for all the extractors involved in the comparison according to the NERD ontology for the sources coming from the *WWW 2011* conference.

of words is $13,381$, while the word detection rate per document $r(w,d)$ is equal to 175.40 and the total number of recognized entities is $12,266$ with the $r(e,d)$ equal to 56.53. Table 5 shows the statistics of the computation results for all extractors. DBpedia Spotlight keeps a high rate of NEs extracted but shows some weaknesses to disambiguate NEs with LOD resources. $r(u,e)$ is equal to 0.2871, lesser than is performance in the previous experiment (see section 5.1). OpenCalais, instead, has the best $r(u,e)$ and it has a considerable ability to classify NEs. Evri performed in a similar way as shown by the $r(c,e)$. The last part of this experiment consists in aligning all the classification types retrieved by these extractors using the NERD ontology, aligning 6 main concepts: Person, Organization, Country, City, Time and Number. Table 6 shows the comparison results. AlchemyAPI still preserves the best result to classify named entities as Person. Instead, differently to what happened in the previous experiment, Evri outperforms AlchemyAPI while classifying named entities as Organization. It is important to note that Evri shows an high number of NEs classified using the class Person in this scenario, but does not explore deeply the Person inference (as shown in the user generated content experiment). OpenCalais has the best performance to classify NEs according to the City class, while Extractiv shows reliability to recognize Country and, especially, to classify Number.

5.3 News Articles

For this experiment, we collected 1000 news articles of *The New York Times* from 09/10/2011 to 12/10/2011 and we performed the extraction for the tools involved in this comparison. The goal is to explore the NE extraction ratio with this dataset and to assess commonalities and differences with the previous experiments. The total number of words is $620,567$, while the word detection rate per document $r(w,d)$ is equal to 620.57 and the total number of recognized entities is $164,12$ with the $r(e,d)$ equal to 164.17. Table 7 shows the statistics of the computation results for all extractors.

Extractiv is the tool which provides the highest number of NEs. This score is considerably greater than what does the same extractor in the other test scenarios (see section 5.1 and section 5.2), and it does not depend from the number of words per document, as reported by $r(e,w)$. In contrast, DBpedia Spotlight shows a $r(e,w)$ which is strongly affected by the number of words: indeed, the $r(e,w)$ is 0.048 lower than the same score in the previous experiment. Although the highest number of URIs detailed is provided by OpenCalais, the URI detection rate per entity is greater for Zemanta, with a score equal to 0.577. Alchemy, Evri, and OpenCalais confirm their reliability to classify NEs and its detection score value $r(c,e)$ is sensibly greater than all the others. Finally, we propose the alignment of the 6 main types recognized by all extractors using the NERD ontology. Table 8 shows the comparison results. Differently to what has been detailed previously, DBpedia Spotlight recognizes few classes, although this number is not comparable with what performed by the other extractors. Zemanta and DBpedia Spotlight increase classification performances with respect to the experiments detailed in the two previous test cases, obtaining a number of recognized Person which is lower than one magnitude order. AlchemyAPI preserves strong ability to recognize Person, but still shows great performance to recognize City and significant scores for Organization and Country. OpenCalais shows meaningful results to recognize the class Person and especially a strong ability to classify NEs with the label Organization. Extractiv holds the best score for classifying Country and it is the only extractor able to seek the classes Time and Number.

6. RELATED WORK

The Named Entity (NE) recognition and disambiguation problem has been addressed in different research fields such as NLP, Web mining and Semantic Web communities. All of them agree on the definition of a Named Entity, which was coined by Grishman *et al.* as an information unit described by the name of a person or an organization, a location, a brand, a product, a numeric expression including time, date, money and percent found in a sentence [9]. One of the first research papers in the NLP field, aiming at automatically identifying named entities in texts, was proposed by Rau [16]. This work relies on heuristics and definition of patterns to recognize company names in texts. The training set is defined by the set of heuristics chosen. This work evolved and was improved later on by Sekine *et al.* [20]. A different approach was introduced when Supervised Learning (SL) techniques were used. The big disruptive change was the use of a large dataset manually labeled. In the SL field, a human being usually trains positive and negative examples so that the algorithm computes classification patterns. SL techniques exploit Hidden Markov Models (HMM) [4], Decision Trees [19], Maximum Entropy Models [5], Support Vector Machines (SVM) [2] and Conditional Random Fields (CRF) [13]. The common goal of these approaches is to recognize relevant key-phrases and to classify them in a fixed taxonomy. The challenges with SL approaches is the unavailability of such labeled resources and the prohibitive cost of creating examples. Semi-Supervised Learning (SSL) and Unsupervised Learning (UL) approaches attempt to solve this problem by either providing a small initial set of labeled data to train and seed the system [11], or by resolving the extraction problem as a clustering one. For instance, a user can try to gather named entities from clustered groups based on the similarity of context. Other unsupervised methods may rely on lexical resources (e.g. WordNet), lexical patterns and statistics computed on large annotated corpus [1].

The NER task is strongly dependent on the knowledge base used to train the NE extraction algorithm. Leveraging on the use of DBpedia, Freebase and YAGO, recent methods, coming from Semantic Web community, have been introduced to map entities to relational facts exploiting these fine-grained ontologies. In addition to detect a NE and its type, efforts have been spent to develop methods for disambiguating information unit with a URI. Disambiguation is one of the key challenges in this scenario and its foundation stands on the fact that terms taken in isolation are naturally ambiguous. Hence, a text containing the term `London` may refer to the city `London in UK` or to the city `London in Minnesota, USA`, depending on the surrounding context. Similarly, people, organizations and companies can have multiple names and nicknames. These methods generally try to find in the surrounding text some clues for contextualizing the ambiguous term and refine its intended meaning. Therefore, a NE extraction workflow consists in analyzing some input content for detecting named entities, assigning them a type weighted by a confidence score and by providing a list of URIs for disambiguation. Initially, the Web mining com-

	n_e	n_c	n_u	$r(e, d)$	$r(e, w)$	$r(c, e)$	$r(u, e)$
AlchemyAPI	17,433	17,443	3,833	17.44	0.028	**1**	0.22
DBpedia Spotlight	30,333	20	8,702	30.33	0.048	0.001	0.287
Evri	16,944	16,944	8,594	16.94	0.027	**1**	0.507
Extractiv	**47,455**	**41,393**	8,177	**47.45**	**0.076**	0.872	0.172
OpenCalais	23,625	23,625	**12,525**	23.62	0.038	**1**	0.53
Zemanta	9,474	4621	5,467	9.474	0.015	0.488	**0.577**

Table 7: Statistics about extraction results for the 1000 news articles published by *The New York Times* from 09/10/2011 to 12/10/2011 of all extractors used in the comparison.

	AlchemyAPI	DBpedia Spotlight	Evri	Extractiv	OpenCalais	Zemanta
Person	6,246	14	2,698	5,648	5,615	1,069
Organization	2,479	-	900	81	2,538	180
Country	1,727	2	1,382	2,676	1,707	720
City	2,133	-	845	2,046	1,863	-
Time	-	-	-	123	1	-
Number	-	-	-	3,940	-	-

Table 8: Number of axioms aligned for all the extractors involved in the comparison according to the NERD ontology for the sources collected from the *The New York Times* from 09/10/2011 to 12/10/2011.

munity has harnessed Wikipedia as the linking hub where entities were mapped [12, 10]. A natural evolution of this approach, mainly driven by the Semantic Web community, consists in disambiguating named entities with data from the LOD cloud. In [14], the authors proposed an approach to avoid named entity ambiguity using the DBpedia dataset.

Interlinking text resources with the Linked Open Data cloud becomes an important research question and it has been addressed by numerous services which have opened their knowledge to online computation. Although these services expose a comparable output, they have their own strengths and weaknesses but, to the best of our knowledge, few research comparisons have been spent to evaluate them. The creators of the DBpedia Spotlight service have compared their service with a number of other NER extractors (OpenCalais, Zemanta, Ontos Semantic API[26], The Wiki Machine[27], AlchemyAPI and M&W's wikifier [15]) according to an annotation task scenario. The experiment consisted in evaluating 35 paragraphs from 10 news articles in 8 categories selected from the *The New York Times* and has been performed by 4 human raters. The final goal was to create wiki links and to provide a disambiguation benchmark (partially, re-used in this work). The experiment showed how DBpedia Spotlight overcomes the performance of other services under evaluation, but its performances are strongly affected by the configuration parameters. Authors underlined the importance to perform several set-up experiments and to figure out the best configuration set for the specific disambiguation task. Moreover, they did not take into account the precision of the NE and type.

We have ourselves proposed a first qualitative comparison attempt, highlighting the precision score for each extracted field from 10 news articles coming from 2 different sources, *The New York Times* and *BBC*[28] and 5 different categories: business, health, science, sport, world [18]. Due to the news articles length, we face a very low Fleiss's kappa agreement score: many output records to evaluate affected the human rater ability to select the correct answer. In this paper, we advance these initial experiments by providing a full generic framework powered by an ontology and we present a large scale quantitative experiment focusing on the extraction performances with different type of text: user-generated content, scientific text, and news articles.

7. CONCLUSION AND FUTURE WORK

In this paper, we presented NERD a web framework which unifies 10 named entity extractors and lift the output result to the Linked Data Cloud following the NIF specification. To motivate NERD, we presented a quantitative comparison of 6 extractors in particular task, scenario and settings. Our goal was to assess the performance variations according to different kind of texts (news articles, scientific papers, user generated content) and different text length. Results showed that some extractors are affected by the word cardinality and the type of text, especially for scientific papers. DBpedia Spotlight and OpenCalais are not affected by the word cardinality and Extractiv is the best solution to classify NEs according to "scientific" concepts such as Time and Number.

This work has evidenced the need to follow up with such systematic comparisons between NE extractor tools, especially using a large golden dataset. We believe that the NERD framework we have proposed is a suitable tool to perform such evaluations. In this work, the human evaluation has been conducted asking all participants to rate the output results of these extractors. An important step forward would be to investigate about the creation of an already labeled dataset of triples *(NE, type, URI)* and then assessing how these extractors adhere to this dataset. Future work will include a thorough comparison with the ESTER2 and CONLL-2003 datasets (datasets well-known in the NLP community) studying how it may fit the need of comparing those extractor tools and more importantly, how to combine them. In terms of manual evaluation, Boolean decision is not enough for judging all tools. For example, a named entity

[26] `http://www.ontos.com`

[27] `http://thewikimachine.fbk.eu/`

[28] `http://www.bbc.com`

type might not be wrong, but not precise enough (Obama is not only a person, he is also known as the American President). Another improvement of the system is to allow the input of additional items or correct miss-understanding or ambiguous items. Finally, we plan to implement a “smart” extractor service, which takes into account extraction evaluations coming from all raters to assess new evaluation tasks. The idea is to study the role of the relevance field in order to create a set of not-discovered NE from one tool, but which may be find out by other tools.

Acknowledgments

This work was supported by the French National Agency under contracts 09.2.93.0966, “Collaborative Annotation for Video Accessibility” (ACAV), ANR.11.EITS.006.01, “Open Innovation Platform for Semantic Media” (OpenSEM) and the European Union’s 7th Framework Programme via the projects LOD2 (GA 257943) and LinkedTV (GA 287911). The authors would like to thank Pablo Mendes for his fruitful support and suggestions and Ruben Verborgh for the NERD OpenID implementation.

8. REFERENCES

[1] E. Alfonseca and S. Manandhar. An Unsupervised Method for General Named Entity Recognition And Automated Concept Discovery. In *1st International Conference on General WordNet*, 2002.

[2] M. Asahara and Y. Matsumoto. Japanese Named Entity extraction with redundant morphological analysis. In *International Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL’03)*, pages 8–15, Edmonton, Canada, 2003.

[3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. DBpedia: A Nucleus for a Web of Open Data. In *6th International Semantic Web Conference (ISWC’07)*, pages 722–735, Busan, South Korea, 2007.

[4] D. Bikel, S. Miller, R. Schwartz, and R. Weischedel. Nymble: a high-performance learning name-finder. In *5th International Conference on Applied Natural Language Processing*, pages 194–201, Washington, USA, 1997.

[5] A. Borthwick, J. Sterling, E. Agichtein, and R. Grishman. NYU: Description of the MENE Named Entity System as Used in MUC-7. In *7th Message Understanding Conference (MUC-7)*, 1998.

[6] R. Cyganiak and A. Jentzsch. Linking Open Data cloud diagram. LOD Community (`http://lod-cloud.net/`), 2011.

[7] R. T. Fielding and R. N. Taylor. Principled design of the modern web architecture. *ACM Transaction Interneternet Technology*, 2:115–150, May 2002.

[8] O. Galibert, S. Rosset, C. Grouin, P. Zweigenbaum, and L. Quintard. Structured and extended named entity evaluation in automatic speech transcriptions. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 518–526, Chiang Mai, Thailand, November 2011.

[9] R. Grishman and B. Sundheim. Message Understanding Conference-6: a brief history. In *16th International Conference on Computational linguistics (COLING’96)*, pages 466–471, Copenhagen, Denmark, 1996.

[10] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum. Robust Disambiguation of Named Entities in Text. In *Conference on Empirical Methods in Natural Language Processing*, pages 782–792, 2011.

[11] H. Ji and R. Grishman. Data selection in semi-supervised learning for name tagging. In *Workshop on Information Extraction Beyond The Document*, pages 48–55, Sydney, Australia, 2006.

[12] S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti. Collective annotation of Wikipedia entities in Web text. In *15th ACM International Conference on Knowledge Discovery and Data Mining (KDD’09)*, pages 457–466, Paris, France, 2009.

[13] A. M. W. Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *7th International Conference on Natural Language Learning at HLT-NAACL (CONLL’03)*, pages 188–191, Edmonton, Canada, 2003.

[14] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. DBpedia Spotlight: Shedding Light on the Web of Documents. In *7th International Conference on Semantic Systems (I-Semantics)*, 2011.

[15] D. Milne and I. H. Witten. Learning to link with Wikipedia. In *17th ACM International Conference on Information and Knowledge Management (CIKM’08)*, pages 509–518, Napa Valley, California, USA, 2008.

[16] L. Rau. Extracting company names from text. In *7th IEEE Conference on Artificial Intelligence Applications*, volume i, pages 29–32, 1991.

[17] G. Rizzo and R. Troncy. NERD: A Framework for Evaluating Named Entity Recognition Tools in the Web of Data. In *10th International Semantic Web Conference (ISWC’11), Demo Session*, pages 1–4, Bonn, Germany, 2011.

[18] G. Rizzo and R. Troncy. NERD: Evaluating Named Entity Recognition Tools in the Web of Data. In *Workshop on Web Scale Knowledge Extraction (WEKEX’11)*, pages 1–16, Bonn, Germany, 2011.

[19] S. Sekine. NYU: Description of the Japanese NE system used for MET-2. In *7th Message Understanding Conference (MUC-7)*, 1998.

[20] S. Sekine and C. Nobata. Definition, Dictionaries and Tagger for Extended Named Entity Hierarchy. In *4th International Conference on Language Resources and Evaluation (LREC’04)*, Lisbon, Portugal, 2004.

[21] F. Suchanek, G. Kasneci, and G. Weikum. Yago: a Core of Semantic Knowledge. In *16th International Conference on World Wide Web (WWW’07)*, pages 697–706, Banff, Alberta, Canada, 2007.