ADEL: ADaptable Entity Linking : A Hybrid Approach to Link Entities with Linked Data for Information Extraction

Julien Plu, Giuseppe Rizzo, Raphaël Troncy

Semantic Web 1 (2017) 1–5
IOS Press

ADEL: ADaptable Entity Linking

A Hybrid Approach to Link Entities with Linked Data for Information Extraction

Editor(s): Name Surname, University, Country
Solicited review(s): Name Surname, University, Country
Open review(s): Name Surname, University, Country

Julien Plu [a], Giuseppe Rizzo [b] and Raphaël Troncy [a]
[a] *EURECOM, 450 Route des Chappes, 06410 Biot, France*
Email: {julien.plu,raphael.troncy}@eurecom.fr
[b] *ISMB, Via Pier Carlo Boggio, 61, 10138 Torino, Italie*
Email: giuseppe.rizzo@ismb.it

Abstract. Four main challenges can cause numerous difficulties when developing an entity linking system: i) the kind of textual documents to annotate (such as social media posts, video subtitles or news articles); ii) the number of types used to categorise an entity (such as Person, Location, Organization, Date or Role); iii) the knowledge base used to disambiguate the extracted mentions (such as DBpedia, Wikidata or Musicbrainz); iv) the language used in the documents. Among these four challenges, being agnostic to the knowledge base and in particular to its coverage, whether it is encyclopedic like DBpedia or domain-specific like Musicbrainz, is arguably the most challenging one. We propose to tackle those four challenges and in order to be knowledge base agnostic, we propose a method that enables to index the data independently of the schema and vocabulary being used. More precisely, we design our index such that each entity has at least two information: a label and a popularity score such as a prior probability or a Pagerank score. This results in a framework named ADEL, an entity recognition and linking system based on a hybrid linguistic, information retrieval, and semantics-based methods. ADEL is a modular framework that is independent to the kind of text to be processed and to the knowledge base used as referent for disambiguating entities. We thoroughly evaluate the framework on six benchmark datasets: OKE2015, OKE2016, NEEL2014, NEEL2015, NEEL2016 and AIDA. Our evaluation shows that ADEL outperforms state-of-the-art systems in terms of extraction and entity typing. It also shows that our indexing approach allows to generate an accurate set of candidates from any knowledge base that makes use of linked data, respecting the required information for each entity, in a minimum of time and with a minimal size.

Keywords: Entity Linking, Entity Recognition, Adaptability, Information Extraction, Linked Data

1. Introduction

Textual content represents the biggest part of content available on the Web but it comes in different forms (social media posts such as tweets, reviews, or Facebook status, video subtitles, news articles, etc.) and in different languages providing multiple challenges for researchers in natural language processing and understanding. Making use of textual content requires analysing and interpreting the information they contain. As a real example, entity linking and entity recognition are largely used in two projects, NextGenTV[1] and ASRAEL[2].

Within the NexGenTV project, we are developing authoring tools that enable to develop second screen applications and facilitate social TV. In particular, there is a need for near real-time automatic analysis to easily identify clips of interest, describe their content, and facilitate their enrichment and sharing [1]. In this context, we are analyzing the TV program subtitles

[1] http://nexgentv.fr/
[2] https://asrael.limsi.fr/

in French for extracting and disambiguating named entities and topics of interests. Within the ASRAEL project, we are analyzing large volume of English and French newswire content in order to induce fine grained schema that describe events being reported in the news. More precisely, we extract and disambiguate named entities that are head words to extract attribute values that best describe an event in a completely unsupervised manner [35].

1.1. Task Description

At the root of these two projects, there is a need of information extraction that aims to get structured information from unstructured text by attempting to interpret natural language for extracting information about entities, relations among entities and linking entities to external referents. More precisely, entity recognition aims to locate and classify entities in text into defined classes such as Person, Location or Organization. Entity linking (or entity disambiguation) aims to disambiguate entities in text to their corresponding counterpart, referred as resource, contained in a knowledge graph. Each resource represents a real world entity with a specific identifier.

In this paper, we denote a *mention* as the textual surface form extracted from a text. An *entity* as an annotation that varies depending of the task: *i)* when only doing the entity recognition task, an *entity* is the pair *(mention, class)*; *ii)* when only doing the entity linking task, an *entity* is the pair *(mention, link)*; *iii)* when doing both the entity recognition and linking task, an *entity* is the triplet *(mention, class, link)*. A *candidate entity* is one possible entity that we generate in order to disambiguate the extracted mention. *Novel entities* are entities that have not yet appeared in the knowledge base being used. This phenomenon happens mainly in tweets and sometimes in news: typically, people may just become popular but do not have yet an article in Wikipedia.

Many knowledge bases can be used for doing entity linking: DBpedia[3], Freebase[4], Wikidata[5] to name a few. Those knowledge bases are known for being broad in terms of coverage, while vertical knowledge bases also exist in specific domains, such as Geonames[6] for geography, Musicbrainz[7] for music, or LinkedMDB[8] for movies.

The two main problems when processing natural language text are ambiguity and synonymy. An entity may have more than one mention (synonymy) and a mention could denote more than one entity (ambiguity). For example, the mentions *HP* and *Hewlett-Packard* may refer to the same entity (synonymy), but the mention *Potter* can refer to many entities[9] (ambiguity) such as places, person, band, movie or even a boar. This problem can be extended to any language. Therefore, entity linking is also meant to solve the problems of synonymy and ambiguity intrinsic in natural language.

We illustrate the problems of ambiguity and synonymy in an example depicted in Figure 1: the mention *Noah* may correspond to at least two entities *Yannick Noah* and *Joakim Noah*. The need to have a knowledge base with Linked Data is crucial in order to properly disambiguate this example: *Yannick Noah* is a tennis player who has played for the Chicago ATP and US Open (in New York) tournaments, the Chicago tournament happening before the US Open one; *Joakim Noah* is a basketball player who has played for the Chicago Bulls before being enrolled by the New York Knicks team. Therefore, one key word in this example is the year 2007 since *Yannick Noah*'s tennis activity is well before 2007. Therefore, the proper entities for this example are *Joakim Noah*, *New York Nicks* and *Chicago Bulls*.

1.2. Challenges

Focusing on textual content, we can list four main challenges that the NLP community is addressing for performing such an intelligent processing and that entity recognition and entity linking systems are facing. These challenges primarily affect the strategy used to understand the text, for extracting meaningful information units and linking those to external referents.

1. the nature of the text, referring to the fact that there are two different categories of text: *i)* formal texts, usually well-written and coming from trusted sources such a newspaper, magazine, or encyclopedia; *ii)* informal texts that are texts

[3] http://wiki.dbpedia.org
[4] https://www.freebase.com
[5] https://www.wikidata.org
[6] http://www.geonames.org
[7] https://musicbrainz.org
[8] http://www.linkedmdb.org
[9] https://en.wikipedia.org/wiki/Potter

Fig. 1. Figure representing an entity linking task

coming from social media platforms or search queries. Each category of textual content has its own peculiarities. For example, tweets are often written without following any natural language rules (grammar-free, slangs, etc.) and the text is mixed with Web links and hashtags. A hashtag is a string preceded by the character # and used to give a topic or a context to a message. This is why one does not process a tweet like a Wikipedia article;

2. the language used: textual content on the Web is available in multiple languages and these languages have some particularities that make them more or less difficult to process (for instance, latin languages versus asian languages);
3. the entity types: they may exist multiple classes (types) in which an entity can be classified and where each type has a definition. The definition of a type may vary depending on the people. For example, in the text *Meet you at Starbucks on the 42nd street*, one may recognize *Starbucks* as an *Organization* while others may want to consider that *Starbucks* is a *Place* where the local branch of a coffee shop is making business. The two annotations may sound correct according to the setting but with two different definitions.
4. the knowledge base used: we can easily imagine that the results of an entity linking system highly depend on the knowledge base being used. First, the *coverage*: if a text is about a movie and one only uses a knowledge base containing descriptions of point of interests and places (such as Geonames), the number of disambiguated entities is likely to be small, contrarily if a general purpose or cinema specific knowledge base is being used. Second, the *data model*: knowledge bases may use different vocabularies and even models which prevent to query in a uniform way (e.g. Freebase vs DBpedia). They may also use different data modeling technology (e.g. relational database vs linked data). Third, *freshness*: if we use a release of DBpedia dated five years ago, it will not be possible to find the entity *Star Wars: The Force Awakens* and this will make the disambiguation of occurrences of this entity much harder.

1.3. Contributions

We propose a generic framework named ADEL which addresses, with some requirements, the four different challenges described in the Section 1.2:

1. We propose an entity recognition process that can be independent of the genre of the textual content (i.e. from Twitter or Wikipedia) and language. This process can also be adapted to the different definitions that may exist for extracting a mention and classifying an entity (Section 3.1).
2. We handle the different type of linked data models that may exist to design a knowledge base by providing a generic method to index its content and to improve the recall in terms of entity candidate generations (Section 3.2).

3. We propose a modular architecture that can be used to design an adaptable entity linking system (Figure 2).
4. ADEL is shown to be robust across different evaluation campaigns in terms of entity recognition, entity candidate generation, and entity linking (Section 5).

1.4. Paper Structure

The rest of the paper is structured as follows: Section 2 presents related work on entity recognition and entity linking. Sections 3 and 4 introduce our approach. Section 5 describes thoroughly numerous evaluations of our approach on standard benchmarks. Finally, conclusions and future work are proposed in Section 6.

2. Related Work

In Section 2.1, we list and detail the essential inputs needed for performing entity linking namely input text, knowledge base, and provenance of both input text and knowledge base. Next, in Section 2.2, we describe the different methods for each component used in the state-of-the-art approaches: mention extraction, entity linking, and joint recognition-linking.

2.1. External Entries Used for Entity Linking

We identify two external entries for an entity linking system: the text to process and the knowledge base to use for disambiguating the extracted mentions. We extend the definition of what is an external entry for an entity linking system defined in [43] where they define the text, the knowledge base and the entity as the three main external entries of an entity linking system. The authors classify the entity itself as a third component because there is currently no agreed upon definition of what is an entity. We identify two cases: *i)* named entities as defined in [19] during the MUC-6 evaluation campaign, is the most commonly used definition, and they represent instances of a defined set of categories with ENAMEX (entity name expressions e.g. Person, Location and Organization) and NUMEX (numerical expression). This definition is often extended by including other categories such as Event or Role. *ii)* named entities are a set of resources defined from a knowledge base. This definition allows to recognize and link only the entities contained in the knowledge base.

We have just seen two different definitions of what can be an entity. The current entity linking systems tends to adopt only one definition, making this as a requirement (an external entry) and not a feature to select. In ADEL, we have decided to integrate the two definitions in order to be able to extract, type and link entities belonging to each definition or the two at the same time.

2.1.1. Textual Content

In [43], the authors classify a textual content in two categories: short and long text. We propose a different orthogonal categorization where textual content is divided between formal text and informal text. Formal texts are well-written texts coming from trusted sources such a newspaper, magazine, or encyclopedia. These texts are often long texts and provide easier ways to detect the context in which the mentions are used. This context facilitates the way the algorithms used in entity linking are working. People who are writing these texts often use a proper and common vocabulary in order to be understood by the largest set of people and contain none (or a low amount) of misspellings. Nevertheless, formal texts can also be short texts, for example, the title of an article or the caption of a picture. It is then harder to detect the content with short texts, even if they have the same characteristics as long texts in terms of writing style. Generally, we argue that the longer is the text to process, the better the algorithms used in entity linking systems work [15].

On the contrary, informal texts are free-written texts mostly coming from social media posts (e.g. tweets) or search query logs. These texts are often short, but they can also be long (e.g. user reviews, forum posts), and generally contain many more misspellings than what formal texts can have. Tweets are the best example since they are often written without following any natural language rules (e.g. grammar-free and slangs) and the text is mixed with short Web links and hashtags. They can also be only composed of emojis. It is easy to imagine that the text *I <3 @justdemi* is more difficult to process by an entity linking system than *I love Demi Moore*.

This categorization is far from being exclusive and video subtitles is another kind of textual content that we aim to process. Subtitles are generally well-written and coming from trusted sources, but they can also come from an automatic speech recognition (ASR) system that will introduce errors and non-existing

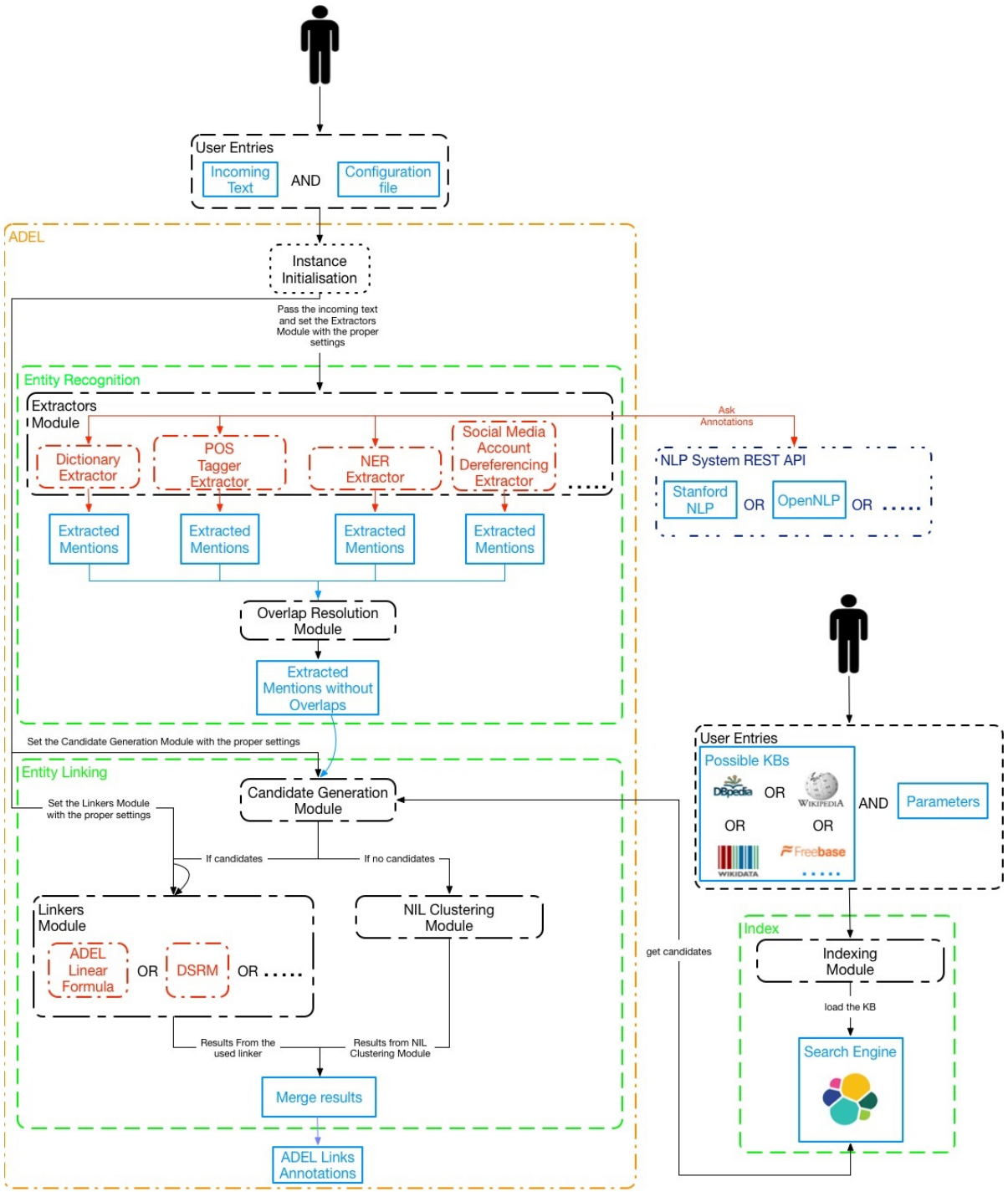

Fig. 2. ADEL architecture. There are two user entries, the text and the index (based on a knowledge base). A configuration file instantiates the launch of the framework. The text from the input goes to each extractor (relying on external NLP systems) and the output of each extractor goes to the overlap resolution. Next, we generate entity candidate, and link them to an entity from a knowledge base or to NIL.

words or generate awkward sentences that will make them informal. Similarly, if the video is a stream coming from Twitch[10], it is likely that the subtitles are informal texts.

2.1.2. Knowledge Bases

Knowledge bases are a fundamental resource for doing entity linking. They often use *linked data* to provide information about entities, their semantic categories and their mutual relationships. Nevertheless, knowledge bases can be stored in different models ranging from graph to relational databases such as Wikipedia. In [43], the authors define three characteristics of a knowledge base: *1)* domain-specific versus encyclopedic knowledge bases; *2)* relational database versus linked data; and *3)* updated versus outdated knowledge bases in terms of data freshness. We will complement this by *i)* introducing some existing knowledge bases that have been widely exploited in entity linking, and *ii)* add a fourth characteristic: the different ontologies (schemas) used to describe the data into a knowledge base. For example, Wikidata is not modeled in the same way than DBpedia [14]. We can reference multiple existing knowledge bases:

- Wikipedia[11] is a free online multilingual encyclopedia created through decentralized, collective efforts from a huge number of volunteers around the world. Nowadays, Wikipedia has become the largest and most popular encyclopedia in the world available on the Web that is also a very dynamic and quickly growing resource. Wikipedia is composed of pages (articles) that define and describe entities or a topic and each of these pages is referenced by a unique identifier. Currently, the English version of Wikipedia contains over 5.3 million pages. Wikipedia has a large coverage of entities and contains comprehensive knowledge about notable entities. Besides, the structure of Wikipedia provides a set of useful features for entity linking such as a unique label for entities, categories, redirect pages, disambiguation pages and links across Wikipedia pages.
- DBpedia [26] is a knowledge base built on top of Wikipedia. DBpedia is created by using the structured information (infobox, hierarchy of the categories, geo-coordinate and external links) contained in each Wikipedia page. Like Wikipedia, it exists also in multiple languages. The 2016-04 English version currently describes over 4.6 million entities and over 583 million relations. A large ontology is used to model the data and the number of entities grows similarly to Wikipedia at each release.
- Freebase [4] is a knowledge base owned by Google that aims to create a knowledge base of the world by merging a high scalability with a collaborative process. It means that anybody can update the knowledge base and anybody can access to it with a special language, MQL[12] (Metaweb Query Language) being a query language such as SPARQL but based on a JSON syntax. It contains 1.9 billion entities. Since March 2015, Google has decided to transfer the content of Freebase to Wikidata and has stopped to maintain Freebase.
- Wikidata [13] is a project from Wikimedia that aims to be a central hub for the content coming from the different Wikimedia projects. It has an evolving schema where new properties requested by the community are regularly added and it provides labels in many languages. More importantly, all entities across languages are linked and belong to the same big graph. The main goal of Wikidata is to become a central knowledge base and it contains so far over 25 million entities.
- YAGO [52] is a multilingual knowledge base that merges all multilingual Wikipedia versions with Wordnet. They use Wikidata as well to check in which language an entity is described. The aim is to provide a knowledge base for many languages that contains real world properties between entities and not only lexical properties. It contains over 4.5 million entities and over 8.9 million relations.
- Babelnet [33] is a multilingual knowledge base that merges Wikipedia, Wordnet, Open Multilingual Wordnet, OmegaWiki, Wiktionary and Wikidata. The goal is to provide a multilingual lexical and semantic knowledge base that is mainly based on semantic relations between concepts and named entities. It contains over 7.7 million entities.
- Musicbrainz[13] is a project that aims to create an open data music relational database. It captures information about artists, their recorded works,

[10] https://www.twitch.tv
[11] http://www.wikipedia.org
[12] https://discourse.cayley.io/t/query-languages-tour/191
[13] http://www.wikipedia.org

the relationships between them. Musicbrainz is maintained by volunteer editors and contains over 53 million entities. A linked data version of Musicbrainz nameed LinkedBrainz[14] is also regularly generated.
– 3cixty KB [45] is a collection of city-specific knowledge base that contains descriptions of events, places, transportation facilities and social activities, collected from numerous static, near- and real-time local and global data providers. The entities in the knowledge base are deduplicated, interlinked and enriched using semantic technologies.

Besides Wikipedia, all the other cited knowledge bases are available as linked data and are modelled using different ontologies. *DBpedia* uses the DBpedia Ontology[15]; Freebase uses its own data model[16] that has been mapped into RDF by keeping the same property names; YAGO uses its own data model [52]; Babelnet implements the lemon vocabulary[17]; Wikidata has developed its own ontology [13]. Knowing that, it is difficult to switch from one knowledge base to another due to the modelling problem as most of the disambiguation approaches uses specific values modelled with the schema of the referent knowledge base.

2.2. Common Entity Linking Components

Regardless of the different entity linking components that intervene in typical workflows [43], there are different ways to use these components. We have identified four different workflows:

1. systems composed of two independent stages: mention extraction and entity linking. For the mention extraction stage, this generally consists in mention detection and entity typing. For the entity linking stage, there is often entity candidate generation, entity candidate selection, and NIL clustering;
2. systems that give a type to the entity at the end of the worflow by using the types of the selected entity from the knowledge base when they exist;
3. systems that generate the entity candidates by using a dictionary during the extraction process, and, therefore, that will not be able to deal with NIL entities;
4. systems that is a merge of all these steps into a single one, called *joint recognition-linking*.

Since a few years, most of the current entity linking research endeavours are only focusing on linking process as they assume that the mention extraction is a solved problem. While the current state-of-the-art methods in mention extraction work very well for well-defined types on newswire content [47], it is far to be perfect for tweets and subtitles [18,46] or for fine-grained entity types. Current state-of-the-art, often, does not detail enough the way they generate the entity candidates or the way they index their knowledge base. Most of the time, they indicate the usage of a dictionary implemented as look up candidates over a Lucene index [39,15,30,48,5]. We believe that further investigating how this step is made, and how it can be optimized, improves the overall results of any entity linking system.

The tables 1, 2 and 3 provide a large overview of the methods and features used by the current state-of-the-art entity linking systems. The column *entity recognition* indicates if the entities are recognized during the mention extraction process or during the linking process; the column *entity candidate generation* indicates if the generation is applied during the mention extraction or during the linking process. In Table 1 and Table 2, we list the systems that provide a full entity linking workflow. We made the specific Table 3 for the joint recognition-linking systems as they cannot be split into the conventional workflow cited before.

In Table 1, we observe that most of the systems use what we call a semantic-based approach, because they make use of dictionaries that have been generated from semantic data (knowledge bases). When POS tagging is being used, it is essentially a secondary feature that aims to enforce or to discard what has been extracted with the dictionary. Contrarily to the others, AIDA [21] uses a pure NLP approach based on Stanford NER [16]. TagME [15] claims to make an overlap resolution between the extracted mentions at the end of this process. Overlap resolution is the process of resolving at least two mentions that overlap in order to make just one mention using a defined heuristic. Further explanation about overlap resolution is provided in the next sections.

[14] https://wiki.musicbrainz.org/LinkedBrainz
[15] http://wiki.dbpedia.org/services-resources/ontology
[16] https://developers.google.com/freebase/guide/basic_concepts
[17] http://lemon-model.net/lemon

Mention Extraction						
System	External Tool(s)	Main Features	Method	Language Resource(s)	Entity Recognition	Entity Candidate Generation
E2E [7]	-	N-Grams, stop words removal, punctuation as tokens	lexical similarity	Wikipedia and Freebase gazetteer	No	Yes
AIDA [21]	StanfordNER[ref]	-	-	NER Dictionary	yes	no
TagME [15] and DataTXT [48] and Dexter [5]	-	N-Grams	lexical similarity	Wikipedia gazetteer	no	yes
WAT [39]	OpenNLP	N-grams, allUpper, isCapitalized, numCapWords, ratioCapWords, mentionLength, numTokens, averageTokenLength, linkProb, boostedLinksRatio, ambiguityRatio, documentProb, mutualDependency, isPerson, isLocation, isOrganization, insidePersonSpans, insideLocationSpans, insideOrganizationSpans, hasSuperior, isWhitelisted	Collective agreement, Wikipedia statistics and SVM	Wikipedia gazetteer	no	yes
Babelfy [32]	-	N-Grams, POS	Lexical Similarity	Babelnet	no	yes
Spotlight Lucene [30]	LingPipePOS	N-Grams, POS	lexical similarity	DBpedia gazetteer	no	yes
Spotlight Statistical [9]	OpenNLP[ref]	POS, capitalized words	finite-state automaton, lexical similarity	DBpedia gazetteer and NER dictionary	yes	yes
VINCULUM [27]	StanfordNER	-	-	NER dictionary	yes	yes
FOX [50]	StanfordNER, OpenNLP, Illinois NE Tagger, Ottawa Baseline IE	-	Ensemble Learning	NER Dictionary	yes	no

Table 1
Analysis of Named Entity Extraction systems

EL (Entity Linking)						
System	Main Features	Method	Knowledge Base(s)	NIL Clustering	Entity Recognition	Entity Candidate Generation
E2E [7]	ngrams, lower case, entity graph features (entity semantic cohesiveness), popularity-based statistical features (clicks and visiting information from the Web)	DCD-SSVM and SMART gradient boosting	Wikipedia, Freebase	no	yes	no
AIDA [21]	popularity based on Wikipdia, similarity, coherence, densest subgraph	graph-based	YAGO2	no	yes	no
TagME [15] and DataTXT [48] and Dexter [5]	mention-entity commonness, Wikipedia statistics	collective agreement, link probability, C4.5	Wikipedia	no	yes	no
WAT [39]	string similarity, commonness, context similarity, PageRank, Personalized PageRank, HITS, SALSA	graph-based	Wikipedia	no	yes	no
Babelfy [32]	densest subgraph	graph-based	Babelnet	no	yes	no
Spotlight Lucene [30]	TF*ICF	VSM, cosine similarity	DBpedia	no	yes	no
Spotlight Statistical [9]	popularity based on Wikipedia, string similarity, context similarity	multiplication among the three features, best result is taken	DBpedia	no	yes	no
VINCULUM [27]	types, co-reference, coherence	sum maximisation among the coherence scores	Wikipedia	no	yes	no
FOX [53]	HITS	graph-based	DBpedia	no	yes	no

Table 2

Analysis of Named Entity Linking systems

In Table 2, we observe three approaches: graph-based (use the graph structure of the data), arithmetic formula (combining different scores with mathematical operations) and pure machine learning using different features. At the end of the linking process, the TagME [15] and WAT [39] systems also do a pruning. None of these systems claim to be able to handle novel entities, that is, disambiguate some entities to *NIL* mainly due to their extraction approach. We also observe how linked data versus relational data is used: most of the listed methods that use a linked data knowledge base tend to have a graph-based linking approach meaning that the structure of the data has a key role into this process. We conclude that linked data promotes graph-based methods and that such knowledge base eases the making of a collective disambiguation. This kind of disambiguation means that we use the relations that the extracted entities have among them in a knowledge base in order to disambiguate them all at the same time.

In Table 3, all methods are CRF-based but with some differences among the features being used. The structure of the knowledge base does not really matter since these methods aims primarily to extract or to compute specific features from it.

3. Approach

The goal of an entity linking approach is to recognize and to link all mentions occurring in a text to existing linked data knowledge base entries and to identify new entities not yet included in the knowledge base. ADEL comes with a new architecture (Figure 2) compared to the state-of-the art ones. Those architectures are typically static and show little flexibility for extracting and linking entities. They generally cannot be extended without making important changes that would require to spend a lot of time in terms of integration. For example, for the extraction, it is not possible to add a dictionary extraction engine to AIDA [21] or a NER extraction to TagME [15]. Next, the linking process is also fixed as, for example, we cannot add a method based on a linear formula to Babelfy [32] which uses a graph-based approach. Finally, the knowledge base being used is often fixed as well: it is difficult to change as we cannot ask to Babelfy [32] to switch from Babelnet [33] to another knowledge base that belongs to the Linked Open Data cloud.

ADEL has been designed to enable all those changes. The ADEL architecture is modular where modules fall within three main categories. The first part, (*Entity Recognition*), contains the modules *Extractors* and *Overlap Resolution*. The second part, (*Index*), contains the module *Indexing*. Finally, the third part, (*Entity Linking*), contains the modules *Candidate Generation*, *NIL Clustering* and *Linkers*. The architecture works with what we call *modules* defined as a piece of the architecture configurable through a configuration file and where each component of a module (in red color on the schema) can be activated or deactivated depending on the pipeline one wants to use. Each module is further detailed in Section 3.1, 3.2 and 3.3. A general pipeline can also be automatically configured for some modules.

3.1. Entity Recognition

In this section, we describe how we recognize mentions from texts that are likely to be selected as entities with the *Extractor Module*. After having identified candidate mentions, we resolve their potential overlaps using the *Overlap Resolution Module*.

Extractors Module. Currently, we make use of six different extractors: *1)* Gazetteer Tagger, *2)* POS Tagger, *3)* NER Tagger, *4)* Date Tagger, *5)* Number Tagger and *6)* Co-reference Tagger. If two or more of these extractors are activated, they run in parallel. The recognition process is based on external NLP systems such as Stanford CoreNLP [29], GATE, NLTK or OpenNLP. To be compliant with any external NLP system, we have based our recognition process on a Web API interface that uses NIF as data exchange format [17]. Therefore, by using this module, it is possible to switch from one NLP system to another one without changing anything in the code or to combine different systems. An example is available with Stanford CoreNLP[18].

1. The Gazetteer Tagger relies on the integrated handling proposed in NLP systems such as *RegexNER*[19] of Stanford CoreNLP, *DictionaryNameFinder*[20] for OpenNLP or the *Dictionary Setup*[21] for GATE. We also propose an automated way to generate a dictionary by issuing

[18] https://github.com/jplu/stanfordNLPRESTAPI

[19] http://stanfordnlp.github.io/CoreNLP/regexner.html

[20] http://opennlp.apache.org/documentation/apidocs/opennlp-tools/opennlp/tools/namefind/DictionaryNameFinder.html

[21] https://gate.ac.uk/sale/tao/splitch13.html#x18-34700013.9.2

JRL (Joint Recognition-Linking)				
System	Main Features	Method	Knowledge Base(s)	NIL Clustering
J-NERD [34]	linguistic features (POS, dependency, tokens, etc.), coherence among entities, context similarity and entity type	CRF probabilistic graphical model and inference	Wikipedia	No
JERL [28]	NER features (tokens, POS, etc.), linking features (probabilities word sequence, context similarity) and mutual dependency (categories information, popularity, relationship, etc.)	semi-CRF	Wikipedia	yes
Durett et al. [11]	co-reference features (first word, headword, last word, context, length, mention type), NER features (word identity, POS, word class, word shape, Brown clusters, common bigrams, capitalization)	CRF and inference	Wikipedia	yes
NEREL [49]	token-level, capitalization, entity co-occurrence, entity type, binary relations, exact match and fuzzy match	maximum entropy	Wikipedia and Freebase	yes

Table 3
Analysis of joint recognition-linking systems

SPARQL queries to a linked data knowledge base that is inspired from how GATE generates its dictionaries. While using a dictionary as extractor, it gives the possibility to be very flexible in terms of entities to extract and their corresponding type, and allows to handle multiple languages.

2. The POS Tagger extractor is configured to extract singular and plural proper nouns and to attach the generic type *THING*. In order to handle tweets, we use the model proposed in [10].
3. The NER Tagger extractor aims to extract named entities that are classified through the taxonomies used by Stanford CoreNLP, OpenNLP, GATE or others NLP systems. In order to handle tweets, we train a model using the data from the NEEL Challenge [43].
4. The Date Tagger aims to recognize all surface forms that represents temporal expression such as *Today*, *December 18, 1997* or *1997/12/18* and relies on current temporal systems such as *SUTime*[22], *ManTIME*[23] or *HeidelTime*[24].
5. The Number Tagger aims to recognize the digit numbers (e.g. 15, 1, 35) or their textual representation (e.g. one, thirty), and can be done by either a NER Tagger (with Stanford NER), a POS Tagger (with the CD[25] tag) or regular expressions.
6. The Co-reference Tagger aims to extract co-references inside a same document but not across documents. The annotators provided by Stanford CoreNLP, OpenNLP, GATE or others NLP systems can be used.

We have the possibility to combine all these extractors, but also to combine the various NER models into one NER Tagger extractor. More precisely, we use a model combination method that aims to jointly make use of different CRF models in Stanford NER as described in the Algorithm 1. This algorithm shows that the order in which the models are applied is important. In Stanford NER, it is called *NER Classifier Combiner*. This logic can be extended to any other NER tagger. We explain the logic of this NER model combination using the following example: *William Bradley Pitt (born December 18, 1963) is an American actor and producer.*. The details for the models being used are

[22] https://nlp.stanford.edu/software/sutime.shtml
[23] https://github.com/filannim/ManTIME/
[24] https://github.com/HeidelTime/heideltime/releases
[25] https://sites.google.com/site/partofspeechhelp/#TOC-CD-

Algorithm 1: Algorithm used in ADEL to combine multiple CRF models

Result: Annotated tokens
Input : (Txt, M) with Txt the text to be annotated and M a list of CRF models
Output: $A = List(\{token, label\})$ a list of tuples $\{token, label\}$

```
begin
    finalTuples ← EmptyList();
    foreach model in M do
        /* tmpTuples contains the
           tuples {token, label} got from
           model                          */
        tmpTuples ←apply model over Txt;
        foreach {token, label} in tmpTuples do
            if token from {token, label} not in
            finalTuples then
                add {token, label} in finalTuples;
            end
        end
    end
end
```

available in the Stanford NER documention[26]. If we only apply the 4 classes model, we get the following result: *William Bradley Pitt* as *PERSON*, and *American* as *MISC*. If we only apply the 7 classes model, we get the following result: *William Bradley Pitt* as *PERSON* and *December 18, 1963* as *DATE*. If we apply both models at the same time using the model combination logic, wet get the following result: *William Bradley Pitt* as *PERSON*, *December 18, 1963* as *DATE* and *American* as *MISC* corresponding here to the sets union.

This combination of different models can, however, lead to a labelling problem. Let's imagine two models trained on two different datasets, where in one dataset a location is labelled as *LOC* but in the other dataset, it is labelled as *Place*. Therefore, if we apply a combination of these two models, the results will contain labelled entities that represents a location but some of them with the label *LOC* and others with the label *Place* and some mentions could have one label or the other depending on the order in which the models have been applied. In this case, the classes are not anymore harmonized because we are mixing models that have been trained with different labels for representing the same type of entities. In order to solve this labelling problem, we propose a two-step solution: *i)* do not mix models that have been trained with different labels to represent the same entity type but, instead, create two instances of a NER extractor where each one has a combination of compatible models; and *ii)* use an overlap resolution module that resolves the overlaps among the extracted mentions from each extractor and harmonize the labels coming from models of different instances of a NER extractor into a same labelling definition.

Overlap Resolution Module. This module aims to resolve the overlaps among the outputs of the extractors and to give one output without overlaps. The logic of this module is as follows: given two overlapping mentions, *e.g.* `States of America` from the NER Tagger and `United States` from the POS Tagger, we only take the union of the two phrases. We obtain the mention `United States of America` and the type provided by the NER Tagger is selected. The overlaps in terms of text are easy to resolve, but it becomes much harder for the types when we have to decide which type to keep when two types come from two different extractors.

A first case is when two labels represent the same category, for example *LOCATION* from the Stanford 3-class model and *dul:Place* from a model trained with the OKE2015 dataset[27]. In order to solve this ambiguity, we have developed a mapping represented in SKOS between the types from multiple sources where the sources are: the labels given by the three default models of Stanford NER, the DUL ontology[28], the Schema.org ontology[29], the DBpedia ontology[30], the Music ontology [41], the NERD ontology [44] and the NEEL taxonomy [43]. A sample of this mapping for the type *Person* is provided at `https://gist.github.com/jplu/74843d4c09e72845487ae8f9f201c797` and the same logic is applied for the other types. With this mapping, it is then possible to jump from one source to another with a SPARQL query. We are also using the notion of *broad* and *narrow* matches from SKOS

[26] `https://nlp.stanford.edu/software/CRF-NER.shtml#Models`

[27] `https://ckan.project-hobbit.eu/fr/dataset/oke2015_task1`

[28] `http://www.ontologydesignpatterns.org/ont/dul/DUL.owl`

[29] `http://schema.org`

[30] `http://mappings.dbpedia.org/server/ontology/classes/`

in order to introduce a hierarchy among the types allowing the possibility to get a parent or sub-category if an equivalent one does not exist.

This recognition process allows us to handle a large set of languages and document types by *i)* cleverly combining different annotators from multiple external systems, and *ii)* merging their results by resolving their overlaps and aligning their types. Once we succeed to recognize the entities, we generate entity candidates retrieved from the knowledge base. In the next section, we describe in detail the process of indexing a knowledge base as an essential task for the retrieval.

3.2. Indexing Linked Data

In this section, we describe how we index a knowledge base and how we optimize the search over it with the *Indexing Module*. The module is composed of two steps: *i)* indexing and *ii)* search optimization. As detailed in Section 2.1.2, there are multiple differences across the existing knowledge bases that make the indexing process very complex. The following process can be applied to any knowledge base that uses linked data. We will detail what are the minimum linked data requirements that a knowledge base should comply with, but also the extra other linked data that they might contain.

Indexing. The first step consists in extracting all entities that will be indexed using a SPARQL query. This query defines as many constraints as necessary. The minimum requirements for an entity to be indexed is to have an ID, a label, and a score. This score can correspond to the PageRank of the entity, or to any other way to score the entities in a linked data knowledge base. For example, with DBpedia, the corresponding required dumps[31] are: Labels, Page Ids and Page Links. The Page Links dump is only used to compute the PageRank of the DBpedia entities and will not be loaded. We use a dedicated graph library[32] in order to compute the PageRank and generate an RDF file that contains the PageRank score for all entities. In general, one needs to generate a file that contains only the links across the entities from the same source in order to compute their PageRank. For DBpedia, we are also using other dumps: anchor texts, instance types, instance type transitive, disambiguation links, long abstracts, mapping-based literals, and redirects. Once done, we load all the dumps into a triple store and use a SPARQL query (Query 1 for DBpedia or Query 3 for Musicbrainz) that retrieves the wanted entities. In the case of DBpedia, we add an additional constraint such as not be a redirect or a disambiguation page. Next, for each entity we got via this first query, we run a second SPARQL query that has for role to retrieve all the data we want to index. The Query 2 and the Query 4 are respectively used for DBpedia and Musicbrainz.

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT DISTINCT ?s
FROM <http://dbpedia.org> WHERE {
    ?s rdfs:label ?label .
    ?s dbo:wikiPageRank ?pr .
    ?s dbo:wikiPageID ?id .
    filter not exists{?s dbo:wikiPageRedirects ?x} .
    filter not exists{?s dbo:wikiPageDisambiguates ?y} .
}
```

Listing 1: SPARQL query that filters the entities we would like to index

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT DISTINCT ?p
(GROUP_CONCAT(DISTINCT ?o;separator="------") AS ?vals)
FROM <http://dbpedia.org> WHERE {
    {
        <http://dbpedia.org/resource/Barack_Obama> ?p ?o .
        FILTER(DATATYPE(?o) = xsd:string ||
            LANG(?o) = "en") .
    } UNION {
        VALUES ?p {dbo:wikiPageRedirects
            dbo:wikiPageDisambiguates} .
        ?x ?p <http://dbpedia.org/resource/Barack_Obama> .
        ?x rdfs:label ?o .
    } UNION {
        VALUES ?p {rdf:type} .
        <http://dbpedia.org/resource/Barack_Obama> ?p ?o .
        FILTER(CONTAINS(str(?o),
            "http://dbpedia.org/ontology/")) .
    } UNION {
        VALUES ?p {dbo:wikiPageRank dbo:wikiPageID} .
        <http://dbpedia.org/resource/Barack_Obama> ?p ?o .
    }
}
```

Listing 2: SPARQL query to retrieve interesting content for the entity *http://dbpedia.org/resource/Barack_Obama*. This query is extended to each entity retrieved from the first DBpedia query

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX mo: <http://purl.org/ontology/mo/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT DISTINCT ?s
FROM <http://musicbrainz.org> WHERE {
    ?s mo:musicbrainz_guid ?id .
    ?s dbo:wikiPageRank ?pr .
    {
        ?s rdfs:label ?label .
    } UNION {
```

[31] http://wiki.dbpedia.org/downloads-2016-04
[32] http://jung.sourceforge.net/

```
        ?s foaf:name ?label .
    } UNION {
        ?s dc:title ?label .
    }
}
```

Listing 3: SPARQL query 1 for Muscbrainz. In Musicbrainz, the labels for an entity might be represented with three different properties *rdfs:label*, *foaf:name*, or *dc:title*

```
PREFIX mo: <http://purl.org/ontology/mo/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT DISTINCT ?p
(GROUP_CONCAT(DISTINCT ?o;separator="------") AS ?vals)
FROM <http://musicbrainz.org> WHERE {
    {
        <http://musicbrainz.org/artist/0002cb05-044d
        -46b8-98e2-8115ba9d24cb#_> ?p ?o .
        FILTER(DATATYPE(?o) = xsd:string ||
            LANG(?o) = "en") .
    } UNION {
        VALUES ?p {dbo:wikiPageRank mo:musicbrainz_guid} .
        <http://musicbrainz.org/artist/0002cb05-044d
        -46b8-98e2-8115ba9d24cb#_> ?p ?o .
    }
}
```

Listing 4: SPARQL query 2 for Musicbrainz to retrieve interesting content for the entity *http://musicbrainz.org/artist/0002cb05-044d-46b8-98e2-8115ba9d24cb#_*. This query is extended to each entity retrieved from the first Musicbrainz query

The result of this second query is then used to obtain an index of the knowledge base.

Optimizing. Once we have this index, we can search for a mention and retrieve entity candidates. Searching over all columns negatively impacts the performance of the index in terms of computing time. In order to optimize the index, we have developed a method that maximizes the coverage of the index while querying a minimum number of columns (or entity properties). Therefore, we need to know in advance over which columns to search. We experimented with an optimization logic for the following benchmark datasets: AIDA and NEEL2015. These datasets have to be annotated with the proper targeted knowledge base. For this reason, we take as example how to optimize a DBpedia index but the proposed logic can be extended to any other knowledge base.

The DBpedia index has 4726950 rows (entities) and 281 columns (datatype properties). Given some benchmark datasets such as OKE2015, OKE2016, NEEL2014, NEEL2015 and NEEL2016, we parse their content in order to extract a list of distinct pairs (mention, link). Next, for every pair, we query the index against every single columns (in the case of dbpedia, this represents 281 queries for each pair), and for each query, we check whether the proper link of the pair is among the results or not. If yes, we put the property in a white list, and if not, the property is ignored as not being helpful to retrieve the good candidate link. At the end, we end up with a file that looks like the excerpt depicted in the Listing 5.

```
{
  "Abrams----http://dbpedia.org/resource/J._J._Abrams": [
              "dbo_abstract",
              "dbo_birthName",
              "dbo_wikiPageWikiLinkText",
              "dbo_wikiPageRedirects",
              "rdfs_label",
              "foaf_name"
          ],
  "AlArabiya_Eng----http://dbpedia.org/resource/Al_Arabiya": [],
  "America----http://dbpedia.org/resource/United_States": [
              "dbo_wikiPageDisambiguates",
              "dbo_wikiPageWikiLinkText",
              "dbo_wikiPageRedirects",
              "dbo_longName"
          ],
  "AnonyOps----http://dbpedia.org/resource/Anonymous_(group)": [
              "dbo_wikiPageWikiLinkText"
          ],
  "AnotherYou----http://dbpedia.org/resource/Another_You": [],
  "CNN----http://dbpedia.org/resource/CNN": [
              "dbo_abstract",
              "dbo_wikiPageDisambiguates",
              "dbo_wikiPageWikiLinkText",
              "dbo_wikiPageRedirects",
              "rdfs_label",
              "foaf_name",
              "dbo_slogan"
  ]
}
```

Listing 5: Excerpt of the result file for the optimization process

This file indicates the columns that must be queried to get the proper link for each pair. We notice that most of the pairs share similar columns. Therefore, we make a union of all these columns to obtain a list of unique columns to use to query the index. For the excerpt depicted in Listing 5, the distinct union yields the following list of 9 properties:

1. `dbo_abstract`
2. `dbo_birthName`
3. `dbo_wikiPageWikiLinkText`
4. `dbo_wikiPageRedirects`
5. `rdfs_label`
6. `foaf_name`
7. `dbo_wikiPageDisambiguates`
8. `dbo_longName`
9. `dbo_slogan`

In the case of dbpedia, this reduces the number from 281 to 72 columns to query but this list is still too large. If we check closely this excerpt, we notice that the column *dbo_wikiPageWikiLinkText* belongs to each list

which means that with 1 single column (instead of 9) we can retrieve all pairs except the pair *AnotherYou—-http://dbpedia.org/resource/Another_You*. The logic behind is that we have to maximize the number of pairs we retrieve for each column, and the goal is then to minimize the number of columns. At the end, we finish with a minimum list of columns that maximize the coverage of the pairs. This optimization can be done with the Algorithm 2. The source code is also available[33].

Algorithm 2: Algorithm used in ADEL to optimize a search query for a specific index.

Result: Optimized set of columns
Input : two-dimentional array I where a row is an instance of a couple and a column is a proper queried column in the index
Output: A a set of columns

```
1  begin
2      current ← EmptySet();
3      tmp ← EmptySet();
4      A ← EmptySet();
5      foreach row E in I do
6          foreach column P in I do
7              add I[P][E] in current;
8          end
9          if size(current) == 1 and
           size(A ∩ current) == 0 then
10             A ← A ∪ current;
11         else if size(A ∩ current) == 0 and
           size(tmp ∩ current) > 0 then
12             tmp ← tmp∪
               firstElement(current ∩ tmp);
13             A ← A ∪ tmp;
14         else
15             tmp ← current;
16         end
17         current ← EmptySet();
18     end
19     if size(tmp) > 0 then
20         A ← A∪ firstElement(tmp);
21     end
22 end
```

At the end of this optimization, we produce a reduced list of 4 properties that are necessary to maximize the coverage of the pairs in the benchmark dataset:

1. `dbo_wikiPageRedirects`
2. `dbo_wikiPageWikiLinkText`
3. `dbo_demonym`
4. `rdfs_label`

This optimization reduces the time of the query to generate the entity candidates from around 4 seconds to less than one second. This ratio is an average time computed across all the queries response. The indexing process allows us to index a large set of knowledge bases that uses linked data and optimize the search against them. The latter is possible at the condition to have at least one benchmark dataset using the targeted knowledge base.

3.3. Entity Linking

The entity linking component starts with the *Candidate Generation Module* that queries the index and generates a list of entity candidates for each extracted entity. If the index returns a list of entity candidates, then the *Linkers Module* is invoked. Alternatively, if an empty list of entity candidates is returned, then the *NIL Clustering Module* is invoked.

NIL Clustering Module. We propose to group the *NIL* entities that may identify the same real-world thing. The role of this module is to attach the same *NIL* value within and across documents. For example, if we take two different documents that share the same emerging entity, this entity will be linked to the same *NIL* value. We can then imagine different *NIL* values, such as *NIL_1*, *NIL_2*, etc. We perform a string strict matching over each possible NIL entities (or between each token if it is a multiple token mention). For example, two mentions: “Sully” and “Marine Jake Sully” will be linked to the same NIL entity.

Linkers Module. Similarly to the *Extractors Module*, this module can handle more than one linking method. The one detailed in this paper is an empirically assessed function represented by Equation 1 that ranks all possible candidates given by the *Candidate Generation Module*.

$$r(l) = (a \cdot L(m, title) + b \cdot max(L(m, R)) + c \cdot max(L(m, D))) \cdot PR(l) \quad (1)$$

[33] `https://gist.github.com/jplu/a16103f655115728cc9dcff1a3a57682`

The function $r(l)$ is using the Levenshtein distance L between the mention m and the title, the maximum distance between the mention m and every element (title) in the set of Wikipedia redirect pages R and the maximum distance between the mention m and every element (title) in the set of Wikipedia disambiguation pages D, weighted by the PageRank PR, for every entity candidate l. The weights a, b and c are a convex combination that must satisfy: $a + b + c = 1$ and $a > b > c > 0$. We take the assumption that the string distance measure between a mention and a title is more important than the distance measure with a redirect page which is itself more important than the distance measure with a disambiguation page.

4. Implementation

The ADEL framework is implemented in Java and is publicly accessible via a REST API[34]. ADEL addresses the aforementioned four challenges being adaptable to the language and the kind of text to process, the types of entity to extract and the knowledge base to use for providing identifiers to entities.

ADEL needs a configuration file expressed in YAML that we call *profile* (Listing 6) in order to adapt its workflow. In the reminder of this section, we will detail how each component works.

```
extract:
  mapping: mappings/types.skos
  reference: stanford
  ner:
    - address:http://localhost/v4/ner
      name:stanfordner
      profile:none
      className:package.ExtractionNER
  pos:
    - address:http://localhost/v4/pos
      name:stanfordpos
      tags:NNP
      profile:none
      className:package.ExtractionPOS
index:
  type:elasticsearch
  address:http://localhost:9200
  query:query.txt
  strict:true
  name:dbpedia201604
link:
  method: package.AdelFormula
```

Listing 6: An example of an ADEL profile

Extract. In Listing 6, the object *extract* configures the entity recognition component. It is composed of one object for each extractor used (NER, POS, COREF, dic, date and number.), the value of these objects being a list of instances. For example, in Listing 6, there are two extractors: *ner* and *pos*, where each extractor generates one instance. An instance is composed of four mandatory properties: *address*, *name*, *profile*, *className*, and an optional one: *tags*. The property *address* is the Web API HTTP address used to query the extractor. The property *name* is a unique name given to the instance of the extractor. The property *profile* is the profile that the extractor has to adopt[35]. The property *className* is the full name of the Java class (package + class) that has to be used internally to run the extractor. This property allows anyone to manage the extractor behavior via the reflection of Java[36]. The single optional property, *tags*, represents the list of tags that have to be extracted (all if empty or not present). It is also composed of two other mandatory properties that are *mapping* and *reference*. The former is the location of the SKOS mapping file for the types, and the latter is the source that will be used for typing the entities.

Index. In Listing 6, the object *index* configures the index that is composed of four mandatory properties: *type*, *address*, *strict* and *name*. The property *address* is the Web API HTTP or the folder address used to locate the index. The property *type* defines the index type to be used. Currently, we only handle Elasticsearch and Lucene but our indexing process can be extended to any other indexing system. In case of an Elasticsearch index, the properties *query* and *name* are mandatory, the former is the file where to find the Elasticsearch query template and the latter is the name of the index. In case of Lucene, these properties are replaced by two other mandatory properties that are *fields* and *size*, the former being the list of fields that will be queried and the latter being the maximum number of candidate to retrieve 7. The property *strict* can have two values: *true* if we want a strict search, or *false* if we want a fuzzy search.

```
index:
  type: lucene
  address: /path/to/the/index
  fields: field1,field2,field3
  size: 1000
```

Listing 7: Lucene example for an index object

[34] http://adel.eurecom.fr/api

[35] The available list of existing profile for the NER extractor starting with the prefix *ner_* is described at https://github.com/jplu/stanfordNLPRESTAPI/tree/develop/properties

[36] Reflection allows to examine, introspect, and modify the code structure and behaviour at runtime.

Link. In Listing 6, the object *link* configures the linkers module. This property contains the full name of the Java class (package + class) that has to be used internally to run the corresponding linking method.

5. Evaluation

In this section, we present a thorough evaluation of ADEL over different benchmark datasets namely OKE2015 [36], OKE2016 [37], NEEL2014 [2], NEEL2015 [42], NEEL2016 [46] and AIDA [21]. Each of these datasets have its own characteristics detailed in Table 4. The scores are computed with the official scorers of each challenge: GERBIL [54] for OKE2015 and OKE2016, neleval [20] for NEEL2015 and NEEL2016, and neeleval[37] for NEEL2014. As there is no official scorer for AIDA we used GERBIL. The neeleval scorer does not allow to compute a score at extraction and linking level, and the neleval scorer does not allow to compute a score at linking level. For these two reasons, we used GERBIL for scoring NEEL2014 at extraction and linking level, and for scoring NEEL2015 and NEEL2016 at linking level. Following the terminology proposed by GERBIL, extraction is defined as *Entity Recognition*, typing as *Entity Typing*, recognition as *RT2KB*, and linking as *D2KB*. Following the terminology proposed by neleval, extraction is referred as *strong_mention_match*, recognition as *strong_types_mention_match*, and extraction + linking as *strong_link_match*.

The results in Table 14 shows ADEL compared to the best participant at OKE2015 and OKE2016, while the Tables 18 and 19 show ADEL compared to the best participant at NEEL2014, NEEL2015 and NEEL2016 for each level evaluated in the respective guidelines. Tables 9, 11, 10 and 12 do the same but with the best systems in GERBIL.

5.1. Experimental Setup

We evaluate our approach at different level: extraction (Tables 6, 5, 7 and 8), recognition (Tables 15 and 16), linking (Table 13) and indexing (Table 17).

Besides, for the two first level we will evaluate different possible configurations:

1. *Conf1*: Use one NER tagger with a model combination setting where the models are the 3 default CRF models provided by Stanford CoreNLP.

[37] https://github.com/giusepperizzo/neeleval

	NEEL2014		
	Precision	Recall	F1
conf1	74.61	29.38	42.16
conf4	**67.79**	**52.47**	**59.15**
conf7	66.67	49.04	56.51
conf15	51.02	35.96	42.19
conf16	54.40	59.32	56.75
conf17	53.90	57.26	55.53

Table 5

Results over the NEEL2014 dataset at extraction level for different ADEL Entity Recognition module configurations. Scores in bold represent the best ADEL configuration.

	OKE2015			OKE2016		
	Precision	Recall	F1	Precision	Recall	F1
conf1	90.69	55.72	69.03	89.35	44.41	59.33
conf2	77.98	39.46	52.4	88.08	39.12	54.18
conf3	95.17	62.35	75.34	87.18	50	63.55
conf4	79.13	57.68	66.72	78.22	51.76	62.3
conf5	74.8	54.97	63.37	78.22	51.76	62.3
conf6	75.7	64.76	69.81	79.34	56.47	65.98
conf7	65.58	51.66	57.79	57.48	42.94	49.16
conf8	**89.54**	**70.93**	**79.16**	90.76	66.47	76.74
conf9	80.45	53.31	64.13	89.3	56.47	69.19
conf10	83.49	67.77	74.81	88.42	67.35	76.46
conf11	80.67	72.89	76.58	**82.3**	**73.82**	**77.83**
conf12	77.2	68.83	72.77	82.03	73.82	77.71
conf13	77.68	78.61	78.14	82.03	73.82	77.71
conf14	69.22	66.72	67.94	66.17	65	65.58

Table 6

Results over the OKE2015 and OKE2016 datasets at extraction level for different ADEL Entity Recognition module configurations. Scores in bold represent the best ADEL configuration.

	NEEL2015			NEEL2016		
	Precision	Recall	F1	Precision	Recall	F1
conf1	83.3	29.5	43.6	77.7	9.9	17.6
conf2	86.3	63.3	73.3	91.6	69.7	79.2
conf3	**85.2**	**72.4**	**78.3**	90.6	70.7	79.4
conf4	67.8	77.4	72.3	75.1	84.8	79.7
conf5	67.9	80.7	73.7	74.2	86	79.7
conf6	67.8	81.6	74.1	74.2	85.9	79.6
conf7	67.6	76.4	71.7	**75.4**	**85.3**	**80.1**

Table 7

Results over the NEEL2015 and NEEL2016 datasets at extraction level for different ADEL Entity Recognition module configurations. Scores in bold represent the best ADEL configuration.

Datasets	Co-references	Classification	Novel Entities	Dates	Numbers	Tweets	Newswire
OKE2015	✓	✓	✓	✗	✗	✗	✓
OKE2016	✓	✓	✓	✗	✗	✗	✓
NEEL2014	✗	✗	✗	✓	✓	✓	✗
NEEL2015	✗	✓	✓	✗	✗	✓	✗
NEEL2016	✗	✓	✓	✗	✗	✓	✗
AIDA	✗	✗	✓	✗	✗	✗	✓

Table 4

Characteristics for each benchmark dataset

	AIDA		
	Precision	Recall	F1
conf1	95.82	91.45	93.58
conf2	**96.59**	**94.24**	**95.4**
conf3	95.82	91.45	93.58
conf4	81	88.21	84.45
conf5	81.94	89.83	85.7
conf6	81	88.21	84.45
conf7	76.76	75.66	76.21

Table 8

Results over the AIDA dataset at extraction level for different ADEL Entity Recognition module configurations. Scores in bold represent the best ADEL configuration.

	OKE2015			OKE2016		
	Precision	Recall	F1	Precision	Recall	F1
extraction ADEL	**89.54**	**70.93**	**79.16**	**82.3**	**73.82**	**77.83**
extraction BG	89.54	55.42	68.47	90.24	43.53	58.73
linking ADEL	78.98	44.13	56.62	50.2	37.06	42.64
linking BG	**83.93**	**49.55**	**62.31**	**65.14**	**62.65**	**63.87**
extraction + linking ADEL	60.46	47.89	53.45	41.31	37.06	39.07
extraction + linking BG	**76.63**	**42.47**	**54.65**	**85.82**	**35.59**	**50.31**

Table 9

Compared results between ADEL best configuration and the best system according to GERBIL (BG) over the OKE 2015 and OKE 2016 datasets. Scores in bold represent the best system.

2. *Conf2*: Use one NER tagger with a model trained with the respective training data of each benchmark dataset.
3. *Conf3*: Use two NER taggers: one with a model combination setting where the models are the 3 default CRF models provided by Stanford

	NEEL2015			NEEL2016		
	Precision	Recall	F1	Precision	Recall	F1
extraction ADEL	**85.2**	**72.4**	**78.3**	**75.4**	**85.3**	**80.1**
extraction BG	39.16	59.22	47.15	4.07	56.37	7.59
linking ADEL	61.45	60.38	60.91	**56.32**	**57.09**	**56.70**
linking BG	**63.15**	**63.05**	**63.1**	45.09	45	45.04
extraction + linking ADEL	**52.9**	**45**	**48.7**	**49.9**	**58.3**	**53.8**
extraction + linking BG	45.58	29.3	35.67	3.28	13.24	5.26

Table 10

Compared results between ADEL best configuration and the best system according to GERBIL (BG) over the NEEL2015 and NEEL2016 datasets. GERBIL does not propose to do entity recognition for the NEEL2015, NEEL2016. Scores in bold represent the best system.

	NEEL2014		
	Precision	Recall	F1
extraction ADEL	**67.79**	**52.47**	**59.15**
extraction BG	36.13	45.62	40.32
linking ADEL	46.89	46.89	46.89
linking BG	**78.74**	**72.85**	**75.68**
extraction + linking ADEL	37.26	28.84	32.51
extraction + linking BG	**34.76**	**34.95**	**34.86**

Table 11

Compared results between ADEL best configuration and the best system according to GERBIL (BG) over the NEEL2014 dataset. Scores in bold represent the best system.

	AIDA		
	Precision	Recall	F1
extraction ADEL	**96.59**	**94.24**	**95.4**
extraction BG	98.75	83.33	90.39
linking ADEL	55.95	55.81	55.88
linking BG	**77.76**	**65.87**	**71.32**
extraction + linking ADEL	55.25	53.81	54.52
extraction + linking BG	**73.64**	**61.89**	**64.27**

Table 12

Compared results between ADEL best configuration and the best system according to GERBIL (BG) over the AIDA dataset. GERBIL does not propose to do entity recognition for the AIDA dataset. Scores in bold represent the best system.

	Precision	Recall	F1
OKE2015	78.98	44.13	56.62
OKE2016	50.2	37.06	42.64
NEEL2014	46.89	46.89	46.89
NEEL2015	61.45	60.38	60.91
NEEL2016	56.32	57.09	56.70
AIDA	55.95	55.81	55.88

Table 13

Results at linking level for ADEL

CoreNLP and a second one with a CRF model trained with the respective training data of each benchmark dataset.

4. *Conf4*: Use one NER tagger with a model combination setting where the models are the 3 default CRF models provided by Stanford CoreNLP. Use one POS tagger with the proper model, for tweets if the benchmark dataset is based on tweets or for newswire if the benchmark dataset is based on newswire text.
5. *Conf5*: Use one NER tagger with a model trained with the respective training data of each benchmark dataset. Use one POS tagger with the proper model, for tweets if the benchmark dataset is based on tweets or for newswire if the benchmark dataset is based on newswire text.
6. *Conf6*: Use two NER taggers: one with a model combination setting where the models are the 3 default CRF models provided by Stanford CoreNLP and a second one with a CRF model trained with the respective training data of each benchmark dataset. Use one POS tagger with the proper model, for tweets if the benchmark dataset is based on tweets or for newswire if the benchmark dataset is based on newswire text.
7. *Conf7*: Use one POS tagger with the proper model, for tweets if the benchmark dataset is based on tweets or for newswire if the benchmark dataset is based on newswire text.

	OKE2015			OKE2016		
	Precision	Recall	F1	Precision	Recall	F1
extraction ADEL	**89.54**	**70.93**	**79.16**	**82.3**	**73.82**	**77.83**
extraction BP	-	-	-	74.03	81.05	77.38
typing ADEL	**79.24**	**66.39**	**72.24**	**82.04**	**69.57**	**75.29**
typing BP	-	-	-	63.07	62.58	62.83
linking ADEL	**78.98**	**44.13**	**56.62**	50.2	37.06	42.64
linking BP	-	-	-	**71.82**	**51.63**	**60.08**

Table 14

Compared results between ADEL best configuration and the best participant (BP) of the OKE challenges. Scores in bold represent the best system.

Seven other configurations are evaluated by adding a dictionary and coreference extractors but only over the OKE2015 and OKE2016 datasets because we do not have proper dictionaries for the other benchmarks and only those benchmarks recognize co-references:

$$conf_i = conf_{i-7} + COT + DIC \quad (2)$$

with $i \in [8, 14]$, COT stands for the coreference tagger, and DIC stands for the dictionary containing the name of jobs coming from Wikipedia.

Three other configurations are evaluated by adding a date and number extractors but only over the NEEL2014 dataset because this benchmark recognizes this kind of entities:

$$conf_i = conf_j + NT + DT \quad (3)$$

with the couple $(i, j) \in [(15, 1); (16, 4); (17, 7)]$, NT stands for the number tagger, and DR stands for the date tagger.

The NEEL2014 and AIDA dataset are not evaluated at recognition level because the guidelines do not require such evaluation. We also remove the ADEL con-

figurations that use the POS Tagger because the POS Tagger cannot type an entity. The Table 13 has no specific configuration because, for now, we do have only one linking method to evaluate.

	NEEL2015			NEEL2016		
	Precision	Recall	F1	Precision	Recall	F1
conf1	72.3	25.6	37.8	61.5	7.9	13.9
conf2	66.1	48.5	56	**75.6**	**57.5**	**65.3**
conf3	**66.7**	**56.7**	**61.3**	74	57.8	64.9

Table 15

Results over the NEEL2015 and NEEL2016 datasets at recognition level for different ADEL Entity Recognition module configurations. Scores in bold represent the best ADEL configuration.

	OKE2015			OKE2016		
	Precision	Recall	F1	Precision	Recall	F1
conf1	76.47	48.21	59.14	82.67	39.01	53
conf2	64.19	31.93	42.65	84.9	32.87	47.39
conf3	87.62	53.27	66.26	81.56	43.41	56.66
conf8	**81.34**	**62.59**	**70.74**	**86.43**	**61.98**	**72.19**
conf9	73.57	45.72	56.39	84.09	49.25	62.12
conf10	78.04	62.65	69.5	85.23	59.17	69.85

Table 16

Results over the OKE2015 and OKE2016 datasets at recognition level for different ADEL Entity Recognition module configurations. Scores in bold represent the best ADEL configuration.

5.2. Results Analysis

OKE2015 and OKE2016. Regarding the OKE datasets, it is interesting to notice that the models trained with the corresponding training sets is less performing in comparison to a general purpose model learned on news, probably due to the amount of data, the datasets being too small, while having a dictionary can significantly improve the results (+13% in average). By analysing the results, we have seen that the coreference Tagger is not that useful for extracting entities if we use the respective OKE models. Basically, these models are able to extract the coreference mentions (e.g. he, she, him, etc.) because these mentions are well represented into the training datasets. While this fact is interesting, the coreference Tagger is important as it links these mentions to their proper reference, what the NER Tagger cannot do because it is not possible for such tagger to make a relation between the extracted entities. For example, in the sentence *Barack Obama was the President of the United States. He was born in Hawaii.*, a NER Tagger might extract *Barack Obama* and *He* and type them as a Person, but will never make the relation that *He* refers to *Barack Obama* and then that *Barack Obama* must be used to disambiguate *He*. This is why we need a Coreference Tagger that provides this relation.

NEEL2014. This dataset is difficult because it requires to extract (but not type) and link only the entities that belong to DBpedia and not the novel entities. As there is no typing, it is not possible for us to train a NER model with the training set, which makes the POS Tagger becoming an important extractor.

NEEL2015 and NEEL2016. The first configuration mainly fails to identify the hashtags and user mentions while the second configuration works relatively well. We also notice that adding a POS Tagger increases the recall but decreases the precision. The best configuration for doing entity recognition is the same than for the extraction. Contrarily to the NEEL2015 dataset, for NEEL2016, the test set has a lower amount of annotated tweets (1663 against 296). Inside this small amount, most of the entities are hashtags or Twitter user mentions, explaining why the *conf1* performs poorly. For NEEL2016, it is interesting to notice that, to only extract entities but not typing them, the *conf7* performs the best. For entity recognition, for both datasets, the best configurations are different from the extraction, which shows that it is not necessarily the best extraction process that will have the best recognition. Furthermore, for these two datasets, we can see that the best configuration is not the same, due to a more important training set for NEEL2016, the resulting model is more accurate. For analysing tweets in general, a simple POS tagger can achieve good results in terms of extraction, which is something useful as one can do entity linking on tweets without a NER model. While NER models trained over newswire content seem not to be appropriate for a proper entity recognition on tweets, we can still achieve fair results as long as there are not too many hashtags and Twitter user mentions.

AIDA. We observe that using a specific NER model yields better results than a combination of models. Using the POS Tagger as the only extractor can provide fair results. Unfortunately, the GERBIL scorer does not give the possibility to score a system at recognition level for the AIDA dataset.

As an overall overview of these per level evaluation, we can see that rarely the best configuration implies only one extractor, showing that our extractor combination approach is playing a key role. It is

	OKE2015	OKE2016	NEEL2014	NEEL2015	NEEL2016	AIDA
Recall	98.38	97.34	93.35 (61.91)	93 (61.84)	93.55 (60.68)	99.62

Table 17

Indexing optimization evaluation: measure if the correct entity is among the list of entity candidates retrieved by the index

	NEEL2015			NEEL2016		
	Precision	Recall	F1	Precision	Recall	F1
recognition ADEL	66.7	56.7	61.3	**75.6**	**57.5**	**65.3**
recognition BP	**85.7**	**76.1**	**80.7**	45.3	49.4	47.3
extraction + linking ADEL	52.9	45	48.7	**49.9**	**58.3**	**53.8**
extraction + linking BP	**81**	**71.9**	**76.2**	45.4	56	50.1

Table 18

Compared results between ADEL best configuration and the best participant (BP) of the NEEL2015 and NEEL2016 challenges. Scores in bold represent the best system.

	NEEL2014		
	Precision	Recall	F1
extraction + linking ADEL	37.26	28.84	32.51
extraction + linking BP	**77.10**	**64.20**	**70.06**

Table 19

Compared results between ADEL best configuration and the best participant (BP) of the NEEL2014 challenge. Scores in bold represent the best system.

also interesting to notice that the best configuration for the NEEL2015 dataset is not the same than for the NEEL2016 dataset despite the fact that both datasets are made of tweets.

Index Optimization. Our index optimization process allows us to get a high score in terms of recall for the entity linking process. The results have been computed with a list of at most 8177 candidates. Providing more candidates does not further increase the recall. We originally observe, though, a significant drop in terms of recall for the NEEL datasets which is mainly due to the presence of hashtags and Twitter user mentions (see the numbers in parenthesis for the 3 NEEL datasets in the Table 17). For example, it is hard to retrieve the proper candidate link *db:Donald_Trump_presidential_campaign,_2016* for the mention corresponding to the hashtag *#TRUMP2016*. We tackle this problem by developing a novel hashtag segmentation method inspired by [51,24]. For the previous example, this will result in *trump 2016*, those two tokens being then enough to retrieve the good disambiguation link in the candidate set. The 3 NEEL datasets, when using the hashtag segmentation method, and the 3 other datasets (OKEs and AIDA) have then a near-perfect recall if one retrieves sufficient candidate links. The few errors encountered correspond to situations where there is no match between the mention and any property values describing the entity in the index.

Comparison with Other Systems. Tables 14, 18, 19, 9, 11, 10 and 12 show that ADEL outperforms all other state-of-the art systems in terms of extraction and recognition, except for the NEEL2015 dataset. The reason is because the system that achieves the best score makes use of a full machine learning approach for each subtask: entity linking (mention extraction + disambiguation), type prediction for entities, NIL mention extraction and type prediction for NIL entities. It works very well but needs a large amount of data for being trained, and, therefore, it will not perform efficiently over the OKE datasets (3498 tweets for NEEL2015 and 95 sentences in OKE2015). In Table 14, we did not put another system for OKE2015 because the winner of the challenge was ADEL. The best system at linking level for OKE2016, is the challenge winner [6]. In Table 19, the winner [7] has the best score. In Ta-

ble 18, for NEEL2015, the winner has the best scores as well [55]. In Tables 9, 11, 10 and 12, ADEL is not the best system for linking, except for NEEL2016. At the linking level, xLisa-NGRAM [38] is the best for OKE2015, DoSeR [56] is the best for OKE2016 and NEE2014, AGDISTIS [53] is the best for NEEL2015, and WAT [39] is the best for AIDA. At extraction and linking level: AIDA [21] is the best for OKE2015, xLisa-NER [38] is the best for OKE2016, DBpedia Spotlight [9] is the best for NEEL2014, and AIDA [21] is the best for AIDA.

Although the linking results are encouraging, they are still a bit low compared to the other state-of-the-art methods. This can be explained for two reasons:

1. It is sensitive to the noise brought at the extraction step since this formula does not take into account the entity context but instead relies on a combination of string distances and the PageRank global score. For example, the string distance score over the title, the redirect and the disambiguation pages between the mention *Trump* and the entity candidate `db:Trumpet` is higher than with the correct entity candidate `db:Donald_Trump`, as *Trump* is closer from *Trumpet* than from *Donald Trump*.
2. It is sensitive to the PageRank as if an entity got a very low score in terms of string comparison, if its PageRank is high enough, this entity can become the one with the best final score.

6. Conclusion and Future Work

The results are encouraging since we demonstrate that our approach enables to be adaptable for at least three challenges:

- text: different kind of text (newswire, tweets, blog posts, etc.) can be processed;
- knowledge base: different knowledge bases (in terms of language, content and model) can be indexed;
- entity: although focusing on common types (PERSON, LOCATION and ORGANIZATION), dates, numbers and more fine grained types can also be independently extracted and linked.

The fourth challenge is the language: another language than English can be used by changing the language of the knowledge base, the models used by the NLP system and the surface forms that the dictionary may contain. We have a functional pipeline for French but it has not been evaluated yet. Evaluating ADEL over multiple languages is also part of our future work.

Linking. The linking step is currently the main bottleneck in our approach. The performance drops significantly at this stage mainly due to a fully unsupervised method. Two new methods will be investigated in order to improve this step. The first one consists in using the new fastText[3] method which is an efficient learning of word representations and sentence classification. In comparison to Word2Vec [31], fastText is robust against out of vocabulary words allowing to create and compute similarities between words that do not belong to its model. The second method is to use the Deep Structured Semantic Models [23] as a relatedness score. This method can be customized to compute a relatedness score of entities in a knowledge base. Next, with this score, we can build a graph regularization as detailed in [22] in order to properly disambiguate the entities. We are also investigating how to use the French lexical network Rezo [25] in order to link entities in French texts. Finally, other general knowledge bases such as Freebase and Wikidata will be tested, but also specific ones like Geonames and 3cixty for different kind of text in order to broaden the evaluation domain of our approach.

Recognition. We are currently working on a coreference approach based on [8] to improve the accuracy of their approach by adding a semantic layer detailed in [40] to the deep neural network. During the overlap resolution, when we merge the results from multiple extractor, if at least two of them extract the same entity but assign a different type (e.g. one with PERSON and the other one with LOCATION), then it is difficult to select the proper type. Therefore, it can be improved by using an ensemble learning approach over each extractor such as the method proposed in [12].

Architecture. Although ADEL has a parallel architecture, we are not yet capable of handling live streams of text as the current system is not designed to be distributed. However, multiple instances of ADEL can run at the same time, and a solution could be to plug on top of multiple instances (workers) a load balancing implementation such as the one proposed in Apache Spark[38].

[38] `http://spark.apache.org`

7. Acknowledgments

This work has been partially supported by the French National Research Agency (ANR) within the ASRAEL project (ANR-15-CE23-0018), the French Fonds Unique Interministériel (FUI) within the NexGen-TV project and the innovation activities 3cixty (14523) and PasTime (17164) of EIT Digital (`https://www.eitdigital.eu`).

References

[1] Olfa Ben Ahmed, Gabriel Sargent, Florian Garnier, Benoit Huet, Vincent Claveau, Laurence Couturier, Raphaël Troncy, Guillaume Gravier, Philémon Bouzy, and Fabrice Leménorel. NexGen-TV: Providing Real-Time Insights During Political Debates in a Second Screen Application. In *25th ACM International Conference on Multimedia (ACMMM), Demo Track*, 2017.

[2] Amparo Elizabeth Cano Basave, Giuseppe Rizzo, Andrea Varga, Matthew Rowe, Milan Stankovic, and Aba-Sah Dadzie. Making Sense of Microposts (#Microposts2014) Named Entity Extraction & Linking Challenge. In *4th Workshop on Making Sense of Microposts*, Seoul, Korea, 2014.

[3] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606*, 2016.

[4] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *ACM SIGMOD International Conference on Management of Data*, 2008.

[5] Diego Ceccarelli, Claudio Lucchese, Salvatore Orlando, Raffaele Perego, and Salvatore Trani. Dexter: an open source framework for entity linking. In *6th International Workshop on Exploiting Semantic Annotations in Information Retrieval*, 2013.

[6] Mohamed Chabchoub, Michel Gagnon, and Amal Zouaq. Collective disambiguation and Semantic Annotation for Entity Linking and Typing. In *Semantic Web Challenges: 2nd OKE Challenge at ESWC 2016*, 2016.

[7] Ming-Wei Chang, Bo-June Paul Hsu, Hao Ma, Ricky Loynd, and Kuansan Wang. E2E: An End-to-End Entity Linking System for Short and Noisy Text. In *4th Workshop on Making Sense of Microposts*, Seoul, Korea, 2014.

[8] Kevin Clark and Christopher D. Manning. Improving Coreference Resolution by Learning Entity-Level Distributed Representations. In *54th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016.

[9] Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. Improving efficiency and accuracy in multilingual entity extraction. In *9th International Conference on Semantic Systems (I-SEMANTICS)*, Graz, Austria, 2013.

[10] Leon Derczynski, Alan Ritter, Sam Clark, and Kalina Bontcheva. Twitter Part-of-Speech Tagging for All: Overcoming Sparse and Noisy Data. In *International Conference on Recent Advances in Natural Language Processing (RANLP)*, 2013.

[11] Greg Durrett and Dan Klein. A Joint Model for Entity Analysis: Coreference, Typing, and Linking. *TACL*, 2014.

[12] Marieke Van Erp, Giuseppe Rizzo, and Raphaël Troncy. Learning with the Web: Spotting Named Entities on the Intersection of NERD and Machine Learning. In *Making Sense of Microposts (#MSM2013) Concept Extraction Challenge*, 2013.

[13] Fredo Erxleben, Michael Günther, Markus Krötzsch, Julian Mendez, and Denny Vrandečić. Introducing Wikidata to the Linked Data Web. In *Proceedings of the 13th International Semantic Web Conference (ISWC)*, 2014.

[14] Michael Färber, Frederic Bartscherer, Carsten Menne, and Achim Rettinger. Linked Data Quality of DBpedia, Freebase, OpenCyc, Wikidata and YAGO. *Semantic Web Journal*, 2016.

[15] Paolo Ferragina and Ugo Scaiella. TAGME: on-the-fly annotation of short text fragments (by wikipedia entities). In *19th ACM Conference on Information and Knowledge Management (CIKM)*, 2010.

[16] Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2005.

[17] Jorge Gracia, Daniel Vila-Suero, John P. McCrae, Tiziano Flati, Ciro Baron, and Milan Dojchinovski. Language Resources and Linked Data: A Practical Perspective. In *Knowledge Engineering and Knowledge Management (EKAW) Satellite Events, VISUAL, EKM1, and ARCOE-Logic*, 2014.

[18] Guillaume Gravier, Gilles Adda, Niklas Paulsson, Matthieu Carré, Aude Giraudel, and Olivier Galibert. The ETAPE corpus for the evaluation of speech-based TV content processing in the French language. In *8th International Conference on Language Resources and Evaluation (LREC)*, 2012.

[19] Ralph Grishman and Beth Sundheim. Design of the muc-6 evaluation. In *6th Conference on Message Understanding (MUC)*, 1995.

[20] Ben Hachey, Joel Nothman, and Will Radford. Cheap and easy entity evaluation. In *52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2014.

[21] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. Robust Disambiguation of Named Entities in Text. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Edinburgh, UK, 2011.

[22] Hongzhao Huang, Yunbo Cao, Xiaojiang Huang, Heng Ji, and Chin-Yew Lin. Collective Tweet Wikification based on Semi-supervised Graph Regularization. In *52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2014.

[23] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning Deep Structured Semantic Models for Web Search Using Clickthrough Data. In *22nd ACM International Conference on Information & Knowledge Management (CIKM)*, 2013.

[24] Jhonata Pereira-Martins Jack Reuter and Jugal Kalita. Segmenting twitter hashtags. *International Journal on Natural Language Computing*, 2016.

[25] Mathieu Lafourcade and Alain Joubert. Increasing Long Tail in Weighted Lexical Networks. In *Cognitive Aspects of the Lexicon (CogAlex-III), COLING*, 2012.

[26] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mo-

hamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal*, 2015.

[27] Xiao Ling, Sameer Singh, and Daniel S. Weld. Design Challenges for Entity Linking. *Transactions of the Association for Computational Linguistics TACL*, 2015.

[28] Gang Luo, Xiaojiang Huang, Chin yew Lin, and Zaiqing Nie. Joint Named Entity Recognition and Disambiguation. In *International Conference on Empirical Methods on Natural Language Processing (EMNLP)*, 2015.

[29] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP Natural Language Processing Toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, 2014.

[30] Pablo N. Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. DBpedia Spotlight: shedding light on the web of documents. In *7th International Conference on Semantic Systems (I-SEMANTICS)*, Graz, Austria, 2011.

[31] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. *CoRR*, 2013.

[32] Andrea Moro, Alessandro Raganato, and Roberto Navigli. Entity Linking meets Word Sense Disambiguation: a Unified Approach. *TACL*, 2014.

[33] Roberto Navigli and Simone Paolo Ponzetto. BabelNet: The Automatic Construction, Evaluation and Application of a Wide-Coverage Multilingual Semantic Network. *Artificial Intelligence*, 2012.

[34] Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. J-NERD: Joint Named Entity Recognition and Disambiguation with Rich Linguistic Features. *TACL*, 2016.

[35] Kiem-Hieu Nguyen, Xavier Tannier, Olivier Ferret, and Romaric Besançon. Generative Event Schema Induction with Entity Disambiguation. In *53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2015.

[36] Andrea Giovanni Nuzzolese, Anna Lisa Gentile, Valentina Presutti, Aldo Gangemi, Darìo Garigliotti, and Roberto Navigli. The First Open Knowledge Extraction Challenge. In *12th European Semantic Web Conference (ESWC)*, 2015.

[37] Andrea Giovanni Nuzzolese, Anna Lisa Gentile, Valentina Presutti, Aldo Gangemi, Robert Meusel, and Heiko Paulheim. The Second Open Knowledge Extraction Challenge. In *13th European Semantic Web Conference (ESWC)*, 2016.

[38] Christian Paul, Achim Rettinger, Aditya Mogadala, Craig A. Knoblock, and Pedro Szekely. Efficient Graph-based Document Similarity. In *13th Extended Semantic Web Conference (ESWC)*, 2016.

[39] Francesco Piccinno and Paolo Ferragina. From TagME to WAT: a new entity annotator. In *1st International Workshop on Entity Recognition & Disambiguation (ERD)*, Gold Coast, Queensland, Australia, 2014.

[40] Roman Prokofyev, Alberto Tonon, Michael Luggen, Loic Vouilloz, Djellel Eddine Difallah, and Philippe Cudré-Mauroux. SANAPHOR: Ontology-Based Coreference Resolution. In *14th International Semantic Web Conference (ISWC)*, 2015.

[41] Yves Raimond, Samer Abdallah, Mark Sandler, and Frederick Giasson. The Music Ontology. In *8th International Conference on Music Information Retrieval (ISMIR)*, 2007.

[42] Giuseppe Rizzo, Amparo Elizabeth Cano Basave, Bianca Pereira, and Andrea Varga. Making Sense of Microposts (#Microposts2015) Named Entity rEcognition and Linking (NEEL) Challenge. In *5th Workshop on Making Sense of Microposts*, Florence, Italy, 2015.

[43] Giuseppe Rizzo, Bianca Pereira, Andrea Varga, Marieke van Erp, and Amparo Elizabeth Cano Basave. Lessons Learnt from the Named Entity rEcognition and Linking (NEEL) Challenge Series. *Semantic Web Journal*, 2017.

[44] Giuseppe Rizzo and Raphaël Troncy. NERD: A Framework for Unifying Named Entity Recognition and Disambiguation Extraction Tools. In *13th Conference of the European Chapter of the Association for Computational Linguistics (EACL), Demo Track*, 2012.

[45] Giuseppe Rizzo, Raphäel Troncy, Oscar Corcho, Anthony Jameson, Julien Plu, Juan Carlos Ballesteros Hermida, Ahmad Assaf, Catalin Barbu, Adrian Spirescu, Kai-Dominik Kuhn, Irene Celino, Rachit Agarwal, Cong Kinh Nguyen, Animesh Pathak, Christian Scanu, Massimo Valla, Timber Haaker, Emiliano Sergio Verga, Matteo Rossi, and José Luis Redondo García. 3cixty@Expo Milano 2015: Enabling Visitors to Explore a Smart City. In *14th International Semantic Web Conference, Semantic Web Challenge (ISWC)*, 2015.

[46] Giuseppe Rizzo, Marieke van Erp, Julien Plu, and Raphaël Troncy. NEEL 2016: Named Entity rEcognition & Linking challenge report. In *6th International Workshop on Making Sense of Microposts*, 2016.

[47] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 Shared Task: Language-independent Named Entity Recognition. In *7th Conference on Natural Language Learning HLT-NAACL*, 2003.

[48] Ugo Scaiella, Michele Barbera, Stefano Parmesan, Gaetano Prestia, Emilio Del Tessandoro, and Mario Verí. DataTXT at #Microposts2014 Challenge. In *4th Workshop on Making Sense of Microposts*, Seoul, Korea, 2014.

[49] Avirup Sil and Alexander Yates. Re-ranking for Joint Named-entity Recognition and Linking. In *22nd ACM International Conference on Information & Knowledge Management (CIKM)*, 2013.

[50] René Speck and Axel-Cyrille Ngonga Ngomo. Ensemble Learning for Named Entity Recognition. In *13th International Semantic Web Conference (ISWC)*, Riva del Garda, Italy, 2014.

[51] S.P.Sharmila and P.Kola Sujatha. Segmentation based representation for tweet hashtag. In *7th International Conference on Advanced Computing,*, 2015.

[52] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO: A Large Ontology from Wikipedia and WordNet. *Journal of Web Semantics*, 6:203–217, 2008.

[53] Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Michael Röder, Daniel Gerber, SandroAthaide Coelho, Sören Auer, and Andreas Both. AGDISTIS - Graph-Based Disambiguation of Named Entities Using Linked Data. In *13th International Semantic Web Conference (ISWC)*, 2014.

[54] Ricardo Usbeck, Michael Röder, Axel-Cyrille Ngonga Ngomo, Ciro Baron, Andreas Both, Martin Brümmer, Diego Ceccarelli, Marco Cornolti, Didier Cherix, Bernd Eickmann, Paolo Ferragina, Christiane Lemke, Andrea Moro, Roberto Navigli, Francesco Piccinno, Giuseppe Rizzo, Harald Sack, René Speck, Raphaël Troncy, Jörg Waitelonis, and Lars Wesemann. GERBIL – General Entity Annotation Benchmark Framework. In *24th World Wide Web Conference (WWW)*, 2015.

[55] Ikuya Yamada, Hideaki Takeda, and Yoshiyasu Takefuji. An end-to-end entity linking approach for tweets. In *5th Workshop on Making Sense of Microposts*, 2015.

[56] Stefan Zwicklbauer, Christin Seifert, and Michael Granitzer. Doser - a knowledge-base-agnostic framework for entity disambiguation using semantic embeddings. In *13th Extended Semantic Web Conference (ESWC)*, 2016.