ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2018-ENST-0062

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité « Computer Science and Multimedia »

présentée et soutenue publiquement par

Julien PLU

le 21 12 2018

Knowledge Extraction in Web Media :

At The Frontier of NLP, Machine Learning and Semantics

Directeur de thèse : **Dr. Raphaël TRONCY**
Co-encadrement de la thèse : **Dr. Giuseppe RIZZO**

Jury
Pierre-Antoine CHAMPIN, Dr HDR, LIRIS, University Claude Bernard Lyon 1, France, reviewer
Harald SACK, Prof, FIZ Karlsruhe, Leibniz Institute for Information Infrastructure, Germany, reviewer
Anna Lisa GENTILE, Dr, IBM Research Almaden, USA, examiner
Andrea TETTAMANZI, Prof, Inria Sophia Antipolis, France, examiner
Christian BONNET, Prof, EURECOM, France, president

TELECOM ParisTech
école de l'Institut Télécom - membre de ParisTech

THÈSE

Dedication

I dedicate this thesis to my family, and especially to my grandfather Bernard who always pushed me and encouraged me even when I did not believe in myself anymore.

Acknowledgements

Thanks to my interested, encouraging and enthusiastic father: he was always keen to know what I was doing and how, although it was difficult to explain him what my work was all about!

I am grateful to all my family members and friends who have supported me until the end.

Thanks a million to my two great supervisors Prof. Raphaël Troncy and Dr. Giuseppe Rizzo for their amazing supervision and always valuable advice to continue to improve this work again and again.

A special thanks goes to the 3cixty project that has provided the funding for this thesis.

A special mention to my group members: Jose, Ghislain, Enrico and Pasquale who became good friends along the years. It was fantastic to have the opportunity to work at your side.

I would like, as well, to address all my gratitude to my sister of heart, Natacha. I will miss our long talks in my office.

Thanks also to my jury members for all their great comments: Dr. Pierre-Antoine CHAMPIN, Prof. Harald SACK, Dr. Anna Lisa GENTILE, Prof. Andrea TETTAMANZI and Prof. Christian BONNET.

And finally, last but by not least, also to everyone at EURECOM, it was great to share the laboratory with all of you during these years.

Abstract

The Web offers a vast amount of structured and unstructured content from which more and more advanced techniques are developed for extracting entities and relations between entities, one of the key elements for feeding the various knowledge graphs that are being developed by major Web companies as part of their product offerings. Most of the knowledge available on the Web is present as natural language text enclosed in Web documents aimed at human consumption. A common approach for obtaining programmatic access to such a knowledge uses information extraction techniques. It reduces texts written in natural languages to machine readable structures, from which it is possible to retrieve entities and relations, for instance obtaining answers to database-style queries. This thesis aims to contribute to the emerging idea that entities should be a first class citizen on the Web. A common research line consists in annotating texts such as users' posts, item descriptions, video subtitles, with entities that are uniquely identified in some knowledge bases as part of the Global Giant Graph. The Natural Language Processing (NLP) community has been addressing this crucial task for the past few decades. As a result, the community has established gold standards and metrics to evaluate the performance of algorithms in important tasks such as co-reference Resolution, Named Entity Recognition, Entity Linking and Relationship Extraction, just to mention few examples. Some of these topics overlap with research that the Database Systems and more recently the Knowledge Engineering communities have been addressing also for decades, such as Identity Resolution (deduplication, entity resolution, record linkage), Schema Mapping (schema mediation, ontology matching) and Data Fusion (instance matching, data interlinking). Meanwhile the Semantic Web and Linked Data communities have been addressing questions related to how to model, serialize and share such information on the Web, as well as how to use knowledge described in more expressive formalisms for a variety of integration, retrieval and discovery tasks. Finally, the Information Retrieval community has been increasingly paying attention to the intersection of structured and unstructured data, with topics encompassing entity-oriented search and semantic search.

This thesis sits at the intersection of several research fields including semantic web, natural language processing and information retrieval, with an emphasis on how to disambiguate entities when machine reading texts. Two tasks have been researched:

Word Sense Disambiguation to infer the sense of ambiguous words and Entity Linking which is used to explore the correct reference of Named Entities occurring in documents from referent knowledge bases. We identify four main challenges 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. Our main contribution is ADEL, an adaptable entity recognition and linking hybrid framework using linguistic, information retrieval, and semantics-based methods. ADEL is a modular system 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 numerous benchmark datasets. 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. The solution to these problems impacts other related domains such as discourse summary, search engine relevance improvement, anaphora resolution and question answering.

Contents

List of Figures

List of Tables

Listings

Part I

Thesis Content

The true sign of intelligence is not knowledge but imagination.

Albert Einstein

1 Introduction

1.1 Context and Motivations

The age of modern artificial intelligence has roughly started in the middle of the 1940s. In 1950, Alan Turing has stated the earliest artificial intelligence problem that was natural language processing oriented, called the Turing test [173]. The goal of this test, as formulated by Turing, can be seen as a game where a human is talking to two different interlocutors through a computer and he has to determine which one is human and which one is artificial. If the human cannot make the difference, then we can assume that a machine can behave like a human. Later in 1966, we see appearing the first chatbot, ELIZA [188], being also the first natural language processing application developed to try to answer to the Turing test. ELIZA was supposed to act like a psychotherapist, and was working with language pattern recognition manually written in a script. The Figures 1.1 and 1.2 show some of the most important dates related to natural language processing. From 1978, people have started to talk about structuring knowledge in order to make machines smarter. From 1991 [141], we see the need to automatically extract important facts from textual content by focusing on recognizing named entities. Once we have started to have usable knowledge bases, we see that people have focused their attention on linking these named entities, and one of the first approach was to link medical entities. Finally, the knowledge bases became more an more complete which allowed people to create more sophisticated applications based on real world knowledge such as Google Home or IBM Watson. To sum up this chronology, one can see that the more we

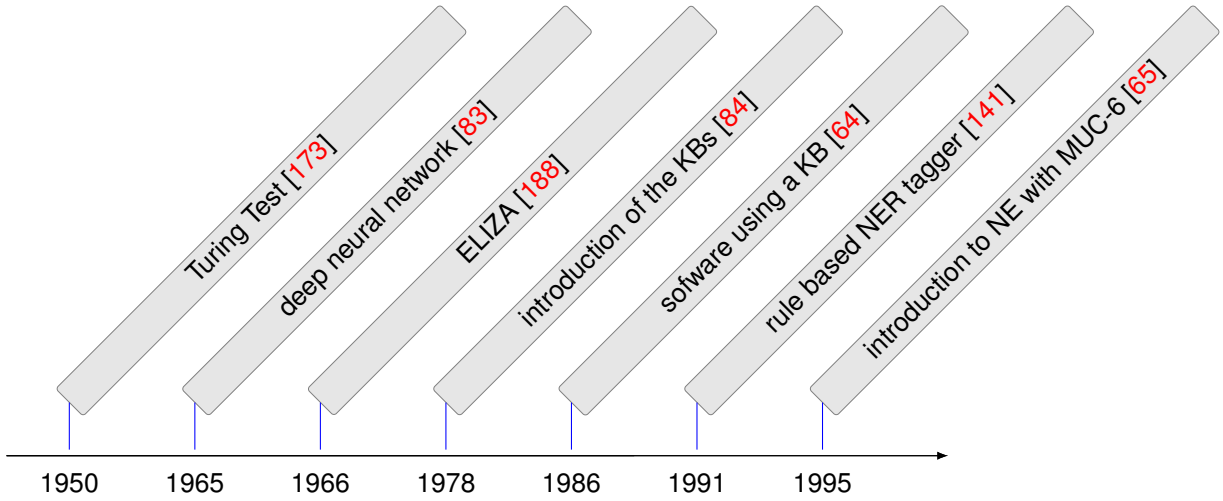

Figure 1.1 – Important dates for natural language processing (Part.1)

advance to the current days, the more we focus on applications that need structured knowledge and that are based on machine learning approaches. Therefore, the need of world knowledge to accomplish natural language processing tasks is exponentially growing, and the performance of these tasks highly depends on the real world entities knowledge they ingest. IBM Watson is often referenced as a good example [169], making the knowledge base a crucial resource for several high level Natural Language Processing tasks such as question answering, chatbots or personal assistants.

The Web offers a vast amount of structured and unstructured content coming in multiple forms: text, images, videos or sounds. The amount of data available on the Web is also constantly growing, making the manual creation and maintenance of knowledge bases a very costly effort. This has lead researchers to investigate how to automatically acquire knowledge from text in order to feed knowledge bases, a task which is often called knowledge base population. It requires to extract relevant information from textual content using an information extraction pipeline, and consists often of the following sub tasks:

1. named entity recognition: extracting mentions occurring in target documents that can be classified into entities of different types such as PERSON, ORGANIZATION or LOCATION.

2. co-reference resolution: grouping a set of at least two phrases and other anaphoric mentions in a document that refers to the same real world entity.

3. relation extraction: determining one or multiple relations between entities occurring in a document.

4. entity linking: determining the identity of entities mentioned in a text against

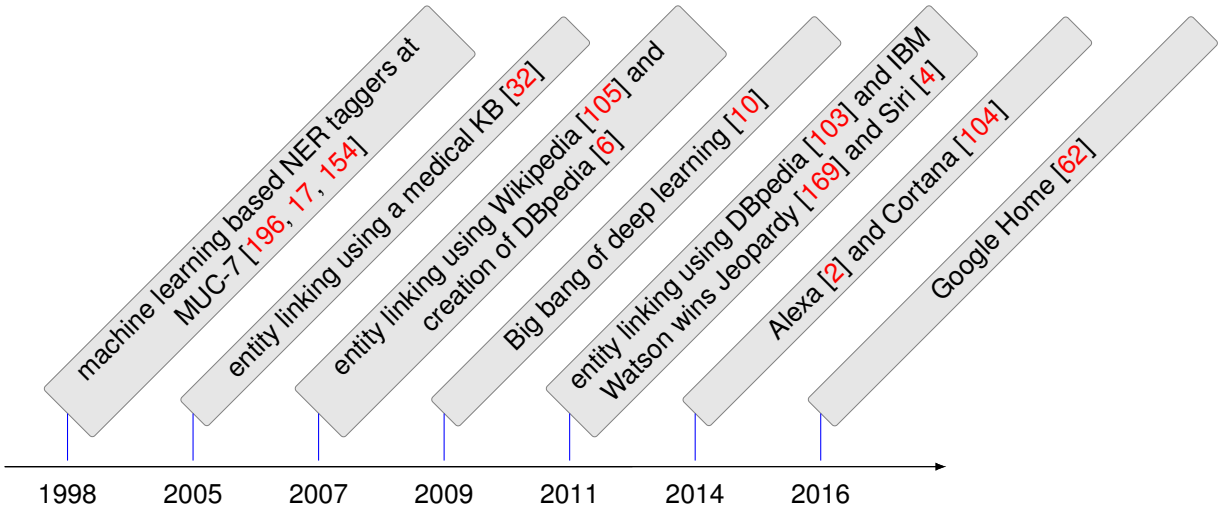

Figure 1.2 – Important dates for natural language processing (Part.2)

a referent knowledge base.

This broad overview of an information extraction pipeline for knowledge base population has received a considerable amount of attention the last decade [160, 42, 168, 6, 22, 117, 38, 187]. The main goal of a knowledge base population task is to use this pipeline to merge the extracted information from the text with the initial content of a knowledge base and then to update the knowledge base, or if the information does not exist, to create a new corresponding entry in this same knowledge base. As said earlier, entity linking is also at the core of many different applications:

Chatbots[1][2][3]**.** In the 2010's, chatbots have gradually evolved into virtual assistant capable of accomplishing tasks and making proposals based on the preferences of the person they serve. Apple (with Siri), Google (with Google Assistant), Microsoft (with Cortana), and Facebook have invested heavily in the development of conversational agents to make them as performing as possible. To realize such exploit, the assistant has to understand about what a conversation is about (the intents), and then needs to recognize and disambiguate the involved entities to be able to perform actions.

Security threats [11]. Recently, the activity of extremist groups became a real threat for multiple countries. Social media platforms are often used to spread hate messages, making the need of national agencies to deploy automatic solution in order to detect any threat. Entity linking is used to characterize those threats.

Entity popularity [37, 49]. Business experts look for customers feedback to help their decision making. The e-commerce web sites and social media are an important source of information about customers attitude and opinion. For example, if the

[1] https://wit.ai/
[2] https://dialogflow.com/
[3] https://dev.botframework.com/

Orange French telecommunication company wants to gather information about its user satisfaction, it needs to distinguish texts mentioning the term *Orange* referring to the company name from the ones talking about the fruit which is achieved using entity linking techniques.

Question answering [195, 40]. Since the earliest age of the computers, one human fantasy was to make machines capable of answering any questions. The *The Hitchhiker's Guide to the Galaxy* book popularizes the computer Deep Thought that was able to answer to the Ultimate Question of Life, the Universe, and Everything. More recently, in 2011, IBM won the game Jeopardy! with Watson [169], beating humans at the task of question answering. Approaches for question answering include information retrieval (and in particular passage retrieval) and queries to knowledge base which require to perform entity linking [21].

1.2 Entity Linking at The Age of Media Diversity

The scope of the work presented in this document is focused on entity linking over different type of media content: textual, acoustic and visual. We consider the acoustic media that can be turned to some textual representation using speech-to-text techniques [28], and visual media that can be turned into textual representation using optical character recognition [82] or image captioning techniques [12]. To this end, we can say that media such as pictures and sounds can be turned to a textual content, making the text the most representative content of information available on the Web. This textual content reveal multiple challenges:

1. the nature of the text, considering that we distinguish between two different categories of text: *i)* formal texts that are generally edited (e.g. official documents, books, news reports, articles) ; *ii)* informal texts, that are used in everyday conversations, personal letters, social media or search queries [39]. Each category of textual content has its own peculiarities. For example, tweets are often written in such a way that grammar rules are not followed or contain slangs; the text is often mixed with Web links and hashtags.[4] This requires to process a tweet differently than a newspaper or 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, widely-used languages versus low-resource languages).

3. the entity types: it 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

[4] A hashtag is a string preceded by the character # and used to give a topic or a context to a message

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 depends on the knowledge base being used. First, the *coverage*: if a text describes movies 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. In contrast, a general purpose or movie specific knowledge base is likely to be more suited. 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.

5. personalization: the links given by an entity linking system can be personalized depending of the preferences of the person who is using the system. For example, one might prefer to have the entity *Tim Berners-Lee* linked with https://www.w3.org/People/Berners-Lee/ than with http://dbpedia.org/resource/Tim_Berners-Lee.

6. temporality: the date from when a document has been written is an important aspect to take into account for linking entities, because results might change depending on this temporal context. For example, if the mention *President Bush* occurs in a document written in 1992, it must be associated to the entity *George H. W. Bush*, whereas if this mention occurs in a document written in 2008, it is more likely to be associated to the entity *George W. Bush* except perhaps in December 2018, when this mention generally refers again to the former president who just died.

This thesis aims to tackle the first four aforementioned challenges: nature of the text, language, entity types and knowledge base used, while we acknowledge that the other two challenges are part of future work.

1.3 Hypothesis and Research Questions

The hypothesis behind this work is based on the fact that the definition of an entity does not change across the language, the type of text, the category it belongs or

the knowledge base. The structure and the context surrounding the mention of an entity is the changing variable. To this end, this work has the objective to harmonize the entity linking process by providing an approach that can be adapted to these different structures and context. To realize this objective, this work aims to address the following research questions:

- How can entities be extracted and typed, using multiple taxonomy of entity types, for various kind of textual content?
- How can different knowledge bases and their corresponding index be used to leverage the linking of the extracted entities?
- How to set an entity linking process in order to create an adaptive pipeline?

1.4 Contributions

After having identified the aforementioned challenges of the textual data available on the Web, this work proposed to tackle them with a generic solution that is to make the process of entity linking adaptable to each of the challenges by proposing a generic framework named ADEL. The key contributions are as follows:

1. The most important contribution of this thesis is that we have introduced a new and unique angle to improve current entity linking approaches for Web media content: conducting an adaptable information analysis for entity linking. Through all our case studies, we show that adaptable information analysis is also powerful for many sub-tasks of entity linking. This is crucial since previous successful approaches were often adaptable for one or two challenges at most. In this thesis, we explore and construct an adaptivity which is involved with content and tools that tend to include a lot of noise. Thus this thesis demonstrates the potential application of adaptable approaches in the field of entity linking.

2. Another important contribution is that we have enhanced natural language processing for coreference resolution. It helps the approach to identify salient information, provides richer background knowledge, and resolves the anaphora to their regular referents entities to make them easier to understand. Our work can also benefit many down-streaming natural language processing tasks such as information retrieval and text classification.

3. We propose, explore, and adapt various approaches including supervised classification, unsupervised graph regularization, ensemble learning and deep neural networks to model entity linking, entity recognition, coreference, and entity

similarity measurement. We achieved the state-of-the-art performance in several natural language processing tasks. For instance, we advanced the standard concept of relatedness, which is adopted in many existing entity link systems by proposing a novel direction based on different contexts.

4. We propose methods to construct and optimize an index of a knowledge base in order to improve the candidate generation process of an entity linking system.

5. We develop the ADEL framework relying on a generic architecture, that can be adaptable to many criteria in order to fit as many entity linking use cases as possible. It is available both as a standalone application and as a REST API.

.

1.5 Outline

The rest of this document is organized as follows:

- Chapter 2: this chapter starts by introducing the needed background knowledge to understand the word done in this thesis, and then gives a thorough state-of-the-art on entity recognition, co-reference and entity linking.
- Chapter 3: this chapter details the entity recognition part of our approach. We explain our recognition approach including a new deep neural network for co-reference, and then our overlap resolution that aims to gather multiple recognition methods as input in order to give more accurate results.
- Chapter 4: this chapter details the entity linking part of our approach. We first detail how we generate candidate from an index created with a generic method. Next, we explain all the linking methods that we have developed, and finally our NIL clustering process.
- Chapter 5: This chapter details the entity linking framework we have develop during this thesis, named ADEL. The details include the architecture, the usage and how to extend it.
- Chapter 6: This chapter details the evaluations and the experiments done during this thesis of our approach over multiple benchmark datasets.
- Finally, the last Chapter reflects our conclusion and highlights the future works of the research carried out in this thesis.

Success is not final, failure is not fatal: it is the courage to continue that counts.

Winston Churchill

2 State of the Art

2.1 Preliminaries For Information Extraction

Information extraction aims to get structured information from unstructured text by attempting to interpret natural language. In particular, information extraction aims to detect entities, relationships between entities and links between entities and external referents. We provide below some preliminaries and common definitions useful to understand the context of this thesis and the techniques that are being researched.

2.1.1 Definitions

In this section, we provide some definitions about:

- what is an entity?
- what is linked data?
- what is a knowledge base?
- what is named entity recognition and entity linking?
- what means coreference resolution?
- what are deep neural networks?
- what are word embeddings?

- what is a string similarity?

2.1.1.1 Entity

There is currently no agreed upon definition of what is an entity. We identify two cases: *i)* named entities as defined in [65] 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 over these two, making this as a requirement (an external entry) and not as a feature to select.

In this thesis, 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. This work defines *novel entities* as 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.

2.1.1.2 Linked Data

The Figure 2.1 summarizes how the current Web is represented. This representation brings to the question: what is the link between the home page of these three web sites? Someone who knows me personally may know this, but a machine or a person who does not know me can not know it. These are the home pages of my personal website, the university where I got my master and the laboratory that is attached to that same university and where I got an internship during my master. This brief example shows one of the problems of the current Web: the data is aimed at human consumption but is generally very poorly structured for machine processing. However, a human, after reading these pages, can make the link between these three documents, while a machine, is not able to do that. One of the aims of linked data is to make the machine able to understand this on its own.

Linked data is based on a set of recommendations developed within the vision of a Semantic Web, as an evolution of the Web of documents:

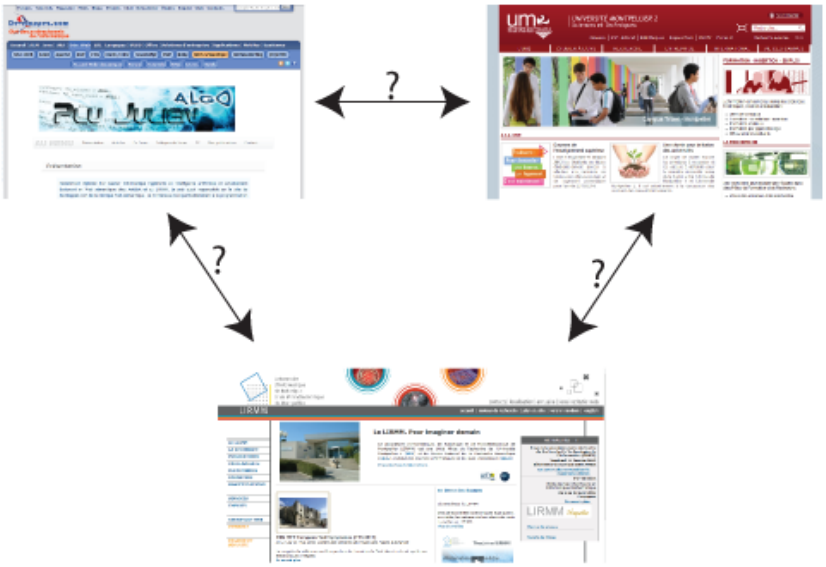

Figure 2.1 – Representation diagram of the current Web. With this diagram it is not possible to detect what is the relation among these three different pages, or why these pages are related.

1. Resource Description Framework (RDF) [185], being a way to represent data as triple statement linking a subject to an object via a relationship property;
2. HyperText Transfer Protocol (HTTP) [182] to access to the data through the Web;
3. Internationalized Resource Identifier (IRI) [184] to uniquely represent something on the Web;
4. Hyperlinks [183] to link the entities to each other.

RDF. Resource Description Framework is a standard describing: *i)* the resources being a unique IRI that might represent anything (person, place, animal, document, concept, etc...); *ii)* the description of these resources through attributes and relations; and *iii)* the framework containing a model of data, languages and syntaxes. It is true that RDF is mainly a data model, since the data are structured by triples (subject, predicate, object), where a *subject* is a resource, a *predicate* is a property, and an *object* is a resource or a literal. For example:

- (eiffel_tower, is_created, 1887)
- (eiffel_tower, is_located, Paris)

Here, *eiffel_tower* is the subject, *is_created* and *is_located* are the predicates, and *1887* and *Paris* are the objects. With this, one can easily deduce that the structure

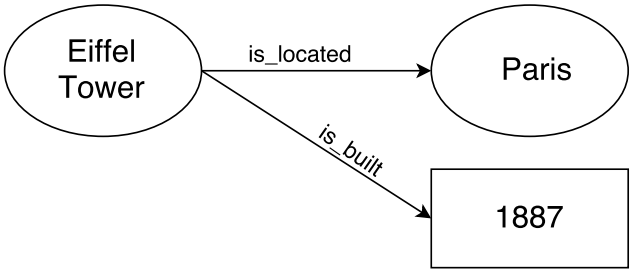

Figure 2.2 – RDF representation with a graph of triples.

created by this model is a directed graph, whose subjects and objects are the nodes and whose predicates are the directed edges. A small example describing the two triplets above is depicted in Figure 2.2 where the oval form represents a resource and the rectangular form represents a literal.

IRI. Internationalized Resource Identifier extends the principle of what is Uniform Resource Locator (URL) principle, since it is usual to use a URL to uniquely identify a Web document. This is the main difference between the two: an IRI represents a single object (a person, a place, a book, etc.) and a URL a Web document. As an example, there are three ways to represent a person on the Web:

1. The person himself: `http://jplu.developpez.com/julien#JP`
2. The RDF document describing this person: `http://jplu.developpez.com/julien`
3. The HTML document describing this person: `http://jplu.developpez.com/julien.html`

Only the last address is a URL, because it represents a Web document. What should also be known is that subjects are IRIs, properties are IRIs and objects can be IRIs or literals. To explain this, let's take the example of the Eiffel Tower. To correctly represent these triples, it would have been necessary to write it in this way:

- (`http://dbpedia.org/resource/Eiffel_Tower`, `http://dbpedia.org/ontology/location`, `http://dbpedia.org/resource/Paris`)
- (`http://dbpedia.org/resource/Eiffel_Tower`, `http://dbpedia.org/property/startDate`, *1887*))

The two triples described above gives the graph depicted in Figure 2.3. Now that we have a RDF graph, there is still a small possibility of improvement or, rather, a shortcut to improve the reading of the graph. It consists in shortening the IRIs: this is called Compact IRI. It is an extremely simple principle to assign a prefix to the IRIs. In the above example, we have IRIs with three different

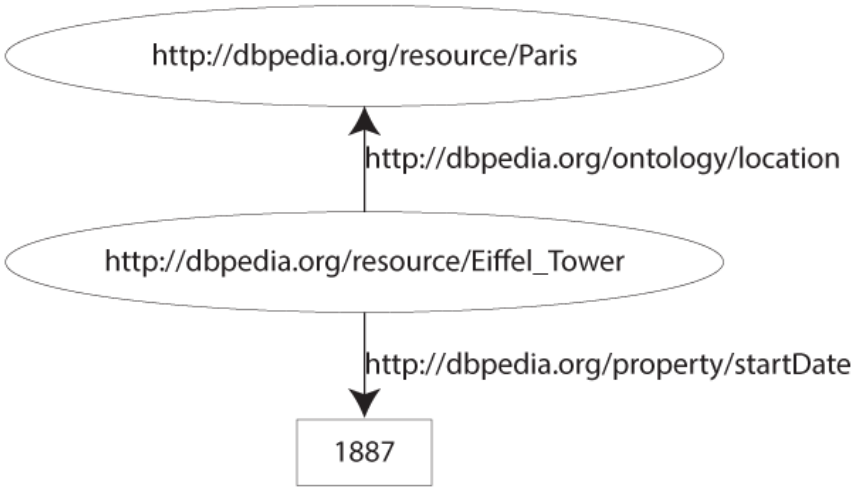

Figure 2.3 – First version of the RDF graph for the Eiffel Tower example.

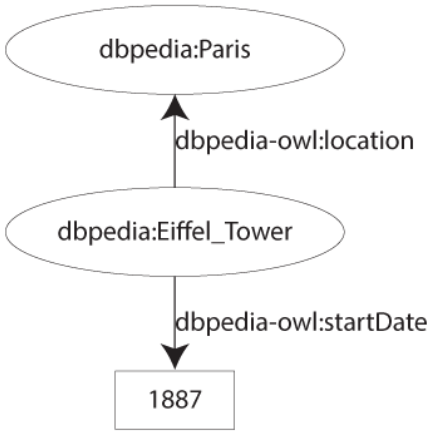

Figure 2.4 – Second version of the RDF graph for the Eiffel Tower example.

namespaces: *i) http://dbpedia.org/ontology/* that can be shortened as *dbpedia-owl*; *http://dbpedia.org/resource/* that can be shortened as *dbpedia*; and *http://dbpedia.org/property* that can be shortened as *dbpprop*. The shortened version of the RDF graph is represented in Figure 2.4.

Literal. Literals are always represented as object in a triple and are represented by a string of characters. These character strings correspond to the so-called *datatypes*: in general, these datatypes are the XSD data types (*xsd:date*, *xsd:anyURI*, *xsd:integer*, etc.). If no datatype is specified, then, by default, the data is considered to be an *xsd:string* type. Literal can also be assigned with a language (*@fr*, *@en*, etc.). An example is given in Figure 2.5.

RDF serialization formats. It exists multiple serialization formats that are used

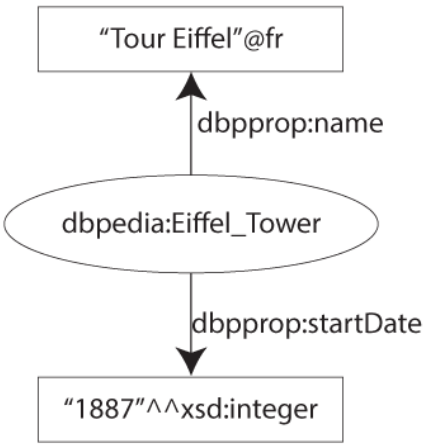

Figure 2.5 – third version of the RDF graph for the Eiffel Tower example.

to serialize RDF such as *JSON-LD*[1], *Turtle*[2] or *RDF/XML*[3]. One of the simplest and most used serialization format is Turtle mostly because it is easy for a human to read it. It supports the compact IRIs via the term *prefix* which associates a namespace with a prefix. The Listing 2.1 is a small example of a set of statements expressed in Turtle. To further improve reading, two other shortcuts can be used as shown in Listing 2.2. If a resource is described by several properties, it is possible to not repeat this resource at the beginning of each line, simply by putting a semi-colon instead of a dot. The second shortcut consists of separating the objects corresponding to the same property by a simple coma.

```
@prefix dbpedia-owl: <http://dbpedia.org/ontology/> .
@prefix dbpprop: <http://dbpedia.org/property/> .
@prefix dbpedia: <http://dbpedia.org/resource/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

dbpedia:Eiffel_Tower dbpedia-owl:location dbpedia:Paris
dbpedia:Eiffel_Tower dbpprop:startDate "1887"^^xsd:integer
dbpedia:Eiffel_Tower dbpprop:name "Tour Eiffel"@fr
dbpedia:Eiffel_Tower dbpprop:name "Eiffel Tower"@en
```

Listing 2.1 – Turtle example

Blank node. In an RDF graph, a blank node represents an anonymous resource or, more simply, an IRI that does not exists or has no identification. A blank node, also, cannot be referenced from outside the graph where it has been defined. In Turtle, we note a blank node by prefixing it with *_:* followed by any identifier. In a graph, a blank node is represented by an empty oval form. The Figure 2.6 represents the graphic version of a blank node, and the Listing 2.3 the Turtle version. A little

[1]https://json-ld.org/spec/latest/json-ld/
[2]https://www.w3.org/TR/turtle/
[3]https://www.w3.org/TR/rdf-syntax-grammar/

```
@prefix dbpedia-owl: <http://dbpedia.org/ontology/> .
@prefix dbpprop: <http://dbpedia.org/property/> .
@prefix dbpedia: <http://dbpedia.org/resource/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

dbpedia:Eiffel_Tower dbpedia-owl:location dbpedia:Paris ;
            dbpprop:startDate "1887"^^xsd:integer ;
            dbpprop:name "Tour Eiffel"@fr, "Eiffel Tower"@en .
```

Listing 2.2 – Turtle example with shortcut

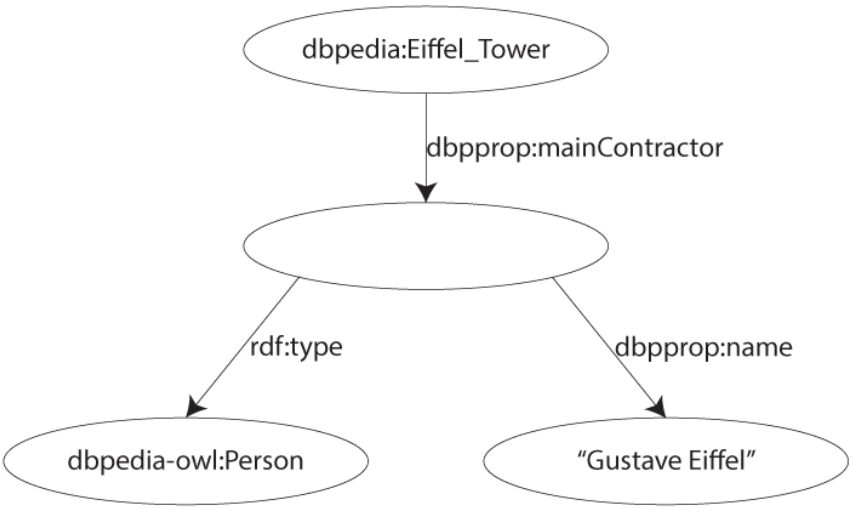

Figure 2.6 – RDF graph for the Eiffel Tower example with an anonymous node.

novelty here is the *rdf:type* property. It is used to describe the type of the associated resource. Here, we describe a blank node that corresponds to a person. Besides, it is possible in Turtle, to write such triple without writing this property and to replace it with the shortcut: *a* as represented in Listing 2.4.

```
@prefix dbpprop: <http://dbpedia.org/property/> .
@prefix dbpedia: <http://dbpedia.org/resource/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.

dbpedia:Eiffel_Tower dbpprop:mainContractor _:GustaveEiffel .

_:GustaveEiffel rdf:type dbpedia-owl:Person ;
                dbpprop:name "Gustave Eiffel" .
```

Listing 2.3 – Turtle example with anonymous node

```
@prefix dbpprop: <http://dbpedia.org/property/> .
@prefix dbpedia: <http://dbpedia.org/resource/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.

dbpedia:Eiffel_Tower dbpprop:mainContractor _:GustaveEiffel .

_:GustaveEiffel a dbpedia-owl:Person ;
                dbpprop:name "Gustave Eiffel" .
```

Listing 2.4 – Turtle example with shortcut for an anonymous node

2.1.1.3 Knowledge Base

A knowledge base gathers specific knowledge to a form that is usable by a computer. It may contain rules (in this case, we speak of a basis of rules), facts or other representations. If it contains rules, an inference engine - simulating logical deductive reasoning - can be used to deduce new facts. Another way of defining a knowledge base is to say that it is an ontology populated by individuals. A knowledge base is used to gather - in a centralized way - the knowledge that is generally formalized in a declarative way.

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 [145], 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 adding 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 [51].

2.1.1.4 Named Entity Recognition and Entity Linking

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 (e.g. Barack Obama), or an abstract concept (e.g. Santa Clause) with a specific identifier. Named Entity Recognition and Entity Linking are non-trivial tasks because when processing natural language text we have to face two main problems: ambiguity and synonymy.

AMBIGUITY. One mention could denote more than one distinct entity. Its interpretation is done depending on the context in which the entity appears. As an example, on *LOCATION* entities, we can consider the following small text:
Last January, I went to visit Paris for the first time. After lunch I started my city walk from my hotel. After walking for an hour, I reached the Eiffel Tower.

In this example, it is impossible to know which *Paris* we are talking about, is it the capital of France or the city in Texas, and the term *Eiffel Tower* cannot help because there is this monument in both cities. We can even imagine more challenging cases such as when an entity is represented by a pseudonym, a nickname or an acronym. For example, the acronym *WWW* might represents hundreds of different entities such

as *Wild Wild West*, *World Wide Web* or *Wallace Welch & Willingham Inc.*.Finally, two mentions may be overlap each other. For example, in the following text:

The University of Montpellier *(LOCATION / ORGANIZATION)* ***is a French public research university in Montpellier*** *(LOCATION)* ***in south-east of France. Established in 1289,*** ***Montpellier*** *(ORGANIZATION)* ***is one of the oldest university of the world.***

Here, *University of Montpellier* is an overlap that refers to both *University of Montpellier* as an *ORGANIZATION*, and to *Montpellier* as a *LOCATION* entity. The second and third occurrences of *Montpellier* refer to a *LOCATION* entity and to an *ORGANIZATION* entity, respectively. We call this case, nested entities. The level of ambiguity can be again increased with social media content, such as tweets because the text is very short, often written without following any natural language rules (e.g. grammar-free and slangs) and might be mixed with short Web links, hashtags and user mentions.

SYNONYMY. One entity in the knowledge base can be referred by multiple mentions. For example, in the following sentence:

Joanne Rowling (a.k.a J. K. Rowling and Robert Galbraith) is a British novelist and screenwriter who wrote the Harry Potter fantasy series.

Here, *J. K. Rowling* and *Robert Galbraith* are synonym and refer both to *Joanne Rowling*. Besides, *metonymy* is a kind of synonymy and can also be a problem. A metonymy is when an entity is not called by its own name but by something associated to the meaning of this entity. For example, in the following sentence:

The White House will be announcing the decision today.

Here, *The White House* is used as a metonym for the U.S. President. We can illustrate both problems of ambiguity and synonymy in an example depicted in Figure 2.7. First, the mention *Manning* may correspond to at least three entities: *Eli Manning*, *Payton Manning* and *Christopher Manning*. Second, the mention *Denver* may correspond to at least height entities *The city*, *The Rockies*, *The Nuggets*, *The Broncos*, *The Crush*, *The Rapids*, *The Avalanche* and *The Mammoth*. The need to have a knowledge base with Linked Data is crucial in order to properly disambiguate this example, and helps to deduce these facts:

1. *Christopher Manning* has no relations with these entities then can be skipped.
2. Only *Denver* the city and *The Broncos* share relations with *Eli Manning* and *Payton Manning*, then all the others can be skipped.
3. *Eli Manning* is playing in NFL and then shares of lot of relations with *The Broncos*. *Payton Manning* has played in NFL for *The Broncos* and shares also a lot of relations with this entity. Then, the entity *Denver* the city can be skipped because the relations with it are too few.

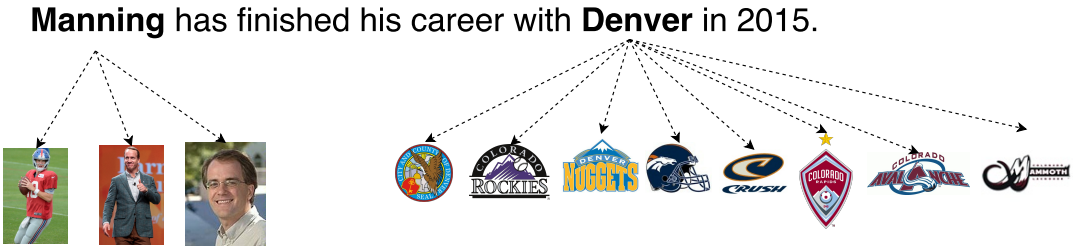

Figure 2.7 – This is a figure that shows how the Entity Linking task can be complex.

4. The last team for which *Payton Manning* has played is *The Broncos* and *Eli Manning* is still playing after 2015.

Finally, with these facts, we can disambiguate the entities to *Payton Manning* and *The Broncos*.

A last problem that an entity linking approach will face is the not-in-knowledge-base (NIL) entities (previously defined as novel entities). They are entities that are not present in the referent knowledge base. Even though, these entities do not belong to the referent knowledge base, an entity linking approach must detect them, and each mention that refers to the same NIL entities must be clustered together.

2.1.1.5 Coreference

We introduce the terminology used to define a coreference. Some of the linguistic units appearing in textual contents have the function of representing physical or conceptual objects. Linguists often call such units "referring expressions", while the objects are called "referents" and the relations that unite a referring expression and its referent are called "references". In the following example: So Jesus said again, *"I assure you, I am the gate for the sheep. All those who came before me were thieves and robbers. [...] I have other sheep too. They are not in this flock here"*, the referring expressions are:

- noun phrases (NPs) and pronouns referring to people (e.g. *Jesus*; *all those who came before me*), things (e.g. *the gate*), classes (e.g. *sheep*; *they*) or that designate interlocutors (e.g. *I*; *you*).
- clauses that name facts (e.g. *I am the gate for the sheep*; *I have other sheep too*; *they are not in this flock here*).
- the adverb *here* that designates a location.

Linguists often distinguish coreference from anaphora [178]. The difference between the two concepts is subtle and is explained in the following. We have a coreference

every time two (possibly different) referring expressions denote the same referent, that is, the same entity. For example, in the sentence *"Abraham Lincoln, the first president of the USA, died in 1865"*, *Abraham Lincoln* and *the first president of the USA* refer to the same entity, thus, they co-refer. We have an anaphora every time the reference of an expression E2, called anaphoric expression, is function of a previous expression E1, called antecedent, so that one needs E1 to interpret E2. For example, in the sentence *"I like dragons! Those animals are really cute!"*, the mention *those animals* is an anaphoric expression and the reader needs to know that it refers to *dragons* (the antecedent) in order to understand the sentence. Finally, the two concepts can be combined:

- The sentence *"You have a cat? I donâĂŹt like them"* is a case of anaphora without coreference since the pronoun *them* needs the antecedent *a cat* to be interpreted (it is the anaphoric), but the two references do not designate the same object (*a cat* = an individual / *them* = the entire species).
- The sentence about *Abraham Lincoln* we presented above is an example of coreference without anaphora, since if we remove *Abraham Lincoln* one can still understand the sentence.
- The sentence *"The dragon is coming. It is going to burn the city!"* is an example of anaphora and coreference since one needs an antecedent to resolve *It*, and both *It* and *the dragon* refer to the same entity.

2.1.1.6 Basic Concepts for Deep Neural Networks

A feed-forward neural network [174] is a classification algorithm. It consists of a simple number of processing units called neurons, organized in layers. Every neuron in a layer is connected with all the neurons in the previous layer. These connections are not all equal: each connection may have a different strength or weight. The weights on these connections encode the knowledge of a network. Often, the neurons in a neural network are also called nodes. Data enters at the inputs and passes through the network, layer by layer, until it arrives at the outputs. The name feed-forward comes from the fact that there is no feedback between the layers. In traditional feed-forward neural networks, a hidden layer neuron is a neuron where the output is connected to the inputs of other neurons and is not visible as a network output. Architecturally, a deep feed-foward network has an input layer, an output layer and one or more hidden layers connecting them. A special kind of deep feed-forward networks are autoencoders, which aim to learn a representation (encoding) for a set of data, mostly for reducing the representation of a vector.

A convolutional neural network [174] is a type of deep feed-forward neural network that contains one or more convolutional layers with a subsampling step and then

followed by one or more fully connected layers being usual layers in deep feed-forward neural networks. The architecture of a convolutional neural network is designed to take advantage of a 2D structure. This is achieved with local connections and linked weights followed by subsampling steps. Another benefit of these neural networks is that they are easier to train and have fewer parameters than feed-forward neural network with the same number of hidden units. There are two main subsampling steps in a convolutional neural network: max pooling and average pooling. Max pooling aims to down-sample an input representation (image, hidden-layer output matrix, etc.) by reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions. This is done by applying a max filter over the non-overlapping sub-regions of the initial representation. Average pooling is an alternative, where instead of doing a max, we do an average over the sub-regions. These pooling strategies mostly help against over-fitting by providing an abstracted form of the representation and reduce the computational cost by reducing the number of parameters to learn.

A recurrent neural network [174], contrarily to deep feed-forward neural network, take as input not just the current example they see, but also what they have perceived previously in time. The decision a recurrent neural network reached at time step $t-1$ affects the decision it will reach one moment later at time step t. So recurrent neural networks have two sources of input, the present and the recent past, which combine to determine how they respond to new data, much as we do in life. Recurrent neural networks are distinguished from feed-forward networks by that feedback loop connected to their past decisions, ingesting their own outputs moment after moment as input. It is often said that recurrent networks have memory. Adding memory to neural networks has a purpose: There is information in the sequence itself, and recurrent neural networks use it to perform tasks that feed-forward networks cannot. That sequential information is preserved in the recurrent network hidden state, which manages to span many time steps as it cascades forward to affect the processing of each new example. It is finding correlations between events separated by many moments, and these correlations are called *long-term dependencies*, because an event downstream in time depends upon, and is a function of, one or more events that came before.

2.1.1.7 Word Embeddings

Word embeddings is an unsupervised machine learning method that focus on learning word representation in a space vector. This method allows to represent each word from a dictionary into a vector of real numbers. We need such representation because none machines learning or deep learning methods allow to use raw form of plain text or strings as they require mathematical representations such as numbers or vectors.

Therefore, with the huge quantity of data represented as text, we had to find a way to represent raw strings and documents with numbers. In other words, a word embedding approach, generally tries to map a word using a dictionary to a vector. For example, as a simple word embedding approach we can take the following sentence:

Word embeddings are super cool

Each word can be mapped to numbers in a vector, and have a dictionary that is composed of all the words from the example, such as: *[Word,embeddings,are,super,cool]*, then to represent the word *super* we get the vector *[0,0,0,1,0]*. Such vectors are called *one-hot encoded* where 1 stands for the position where the word exists and 0 everywhere else. Now, we can extend the representation of a single word to a sentence with a list of one-hot encoded vectors: *[[1,0,0,0,0],[0,1,0,0,0],[0,0,1,0,0],[0,0,0,1,0],[0,0,0,0,1]]*.

This is just a very simple approach to represent a word or a sentence as a vector and facilitates the semantic analysis of words. This new representation has the particularity that words appearing in similar contexts possess corresponding vectors which are relatively close. For example, one might expect the words *dog* and *cat* to be represented by vectors relatively distant in the vector space where these vectors are defined. It exists multiple other word embeddings approaches, the most popular are, Word2Vec [106], FastText [15], GloVe [121] and Starspace [191].

Word2Vec. This approach is the most popular and can be seen as a prediction-based approach. Prediction, because from surrounding words it will predict the missing word (Continuous Bag Of Words) or from a word, it will predict the surrounding words (Skip-Gram). These two kind of prediction are a three layers neural network: one input, one hidden and one output. The logic of the continuous bag of word architecture is depicted in Figure 2.8. The input layer corresponds to the signals that represent the context (surrounding words) in a fixed window size and the output layer correspond to the signals for the predicted target word. Let's take the same example than previously with a context window equal to 2, then two words before and two words after. The aim is to learn the representation of each word that compose the sentence. To this end, the neural network will learn the features by looking at the words in a window, here *Word embeddings super cool* and try to predict the missing word *are*. Therefore, with the input *Word, embeddings, super, cool*, the training process adjusts the weights of its network to make the probability to output the word *are*. The more the training process will repeat this logic, the more stable will be the weights, and then the predictions. However, this architecture has some disadvantages like if it is not properly optimized, the training can take forever, or because it takes the average of the context of a word. For example, *Orange* can be both a fruit and a company but the architecture takes an average of the two contexts and places it in between a cluster for fruits and companies. Then, in order to solve these two problems, there is another architecture called skip-gram and depicted in Figure 2.9. It is the exact opposite of the bag of words architecture, from the word *are* it has to

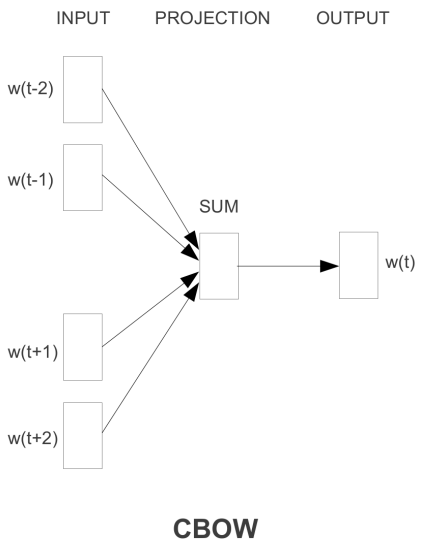

Figure 2.8 – CBOW architecture representation taken from [106].

predict *word, embeddings, super, cool*. It has multiple advantages such as capturing two semantics for a single word (it will have two vector representations of Orange one for the company and other for the fruit) and generally outperform any other architectures. The final representation of continuous bag of words, or skip-gram, is the learned weights as the vectorized words. Finally, in order to know how much two words are close, we take their vectorized representation and apply a cosine similarity. This website[4] proposes a nice visualization of the two architectures.

GloVe. This approach is also based on skip-gram architecture but instead of using a local context it uses a global context (the whole sentence) in order to get a broader representation of the words.

FastText. This approach also uses a skip-gram architecture but instead of being based on the words itself, it uses a bag of character n-grams that compose each word. For example, with a window of three characters, the word *house* is represented as *<ho,hou,ous,use,se>*. This approach allows the model to be robust against words that never appear in the training data.

Starspace. This approach is totally different from the others because it allows to learn any kind of embeddings (sentence, word, entities, graph, etc...). The strength behind this approach is that the model consists of learning things, where each thing is described by a set of discrete features (bag-of features) coming from a fixed-length dictionary. A thing, such as a document or a sentence can be described by a bag of words or n-grams, a user can be described by a bag of documents, movies or items they have liked, and so forth. To this end, this model is free to compare things of

[4]https://ronxin.github.io/wevi/

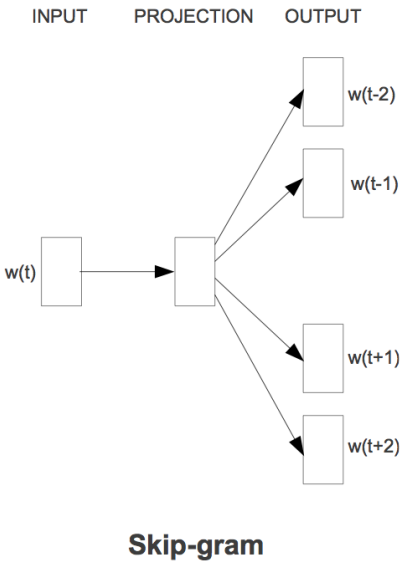

Figure 2.9 – Skip-Gram architecture representation taken from [106].

different kinds.

2.1.1.8 String Similarity

In mathematics, a distance is an application that formalizes the intuitive idea of distance, that is, the length that separates two points. Here, it is the same principle, we want to measure how much two strings are close or far to each other. To do so it exists multiple methods such as Dice [61], Jaro-Winkler [61], Jaccard [61] or Levenshtein [61]. The most popular being the Levenshtein distance.

We say that a Levenshtein distance is equal to the minimum number of characters to be deleted, inserted or replaced to move from one string to another. Each of these operations is associated with a cost. The cost is always equal to 1, except in the case of a substitution of identical characters. For example, if $W_1 =$ entity and $W_2 =$ entity then $LevenshteinDistance(W_1, W_2) = 0$ because they are indeed strictly equal. Another example, if $W_1 =$ entity and $W_2 =$ entiti then $LevenshteinDistance(W_1, W_2) = 1$ because there has been a replacement (change from y to i), and that one cannot do less.

All the other methods are based on the same principle, giving a score representing a distance between two strings. The score an also be normalized, between 0 and 1, where 0 represents the absence of similarity and 1, the equality of the chains compared.

2.1.2 Textual Content

In [145], the authors classify a textual content in two categories: short and long text. This work proposes a different orthogonal categorization where textual content is divided between formal text and informal text. Formal texts are used in serious content and situations (e.g. official documents, books, news reports, articles). 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 [53].

On the contrary, informal texts are used in everyday conversations, personal letters, social media or search queries. 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 can be both (formal or informal), because they can come from an automatic speech recognition (ASR) system that will introduce errors and non-existing words or generate awkward sentences that will make them informal. Similarly, if the video is a stream coming from Twitch[5], it is likely that the subtitles are informal texts. Nevertheless, if we listen an official discourse from a politician, or the news on TV, the subtitles will be formal.

2.1.3 Linked Data and Non-Linked Data Knowledge Bases

Many knowledge bases can be used for doing entity linking: DBpedia[6], Freebase[7], Wikidata[8] 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

[5] https://www.twitch.tv
[6] http://wiki.dbpedia.org
[7] https://www.freebase.com
[8] https://www.wikidata.org

Geonames[9] for geography, Musicbrainz[10] for music, or LinkedMDB[11] for movies and each of them have their own characteristics. Most popular linked data knowledge bases, as they are in December 2017, are:

- DBpedia [92] 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.

- 3cixty KB [146] 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.

- Wikidata [48] 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 [168] 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 [112] 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.

[9] http://www.geonames.org
[10] https://musicbrainz.org
[11] http://www.linkedmdb.org

All these linked data knowledge bases are modelled using different ontologies. *DBpedia* uses the DBpedia Ontology[12]; Freebase uses its own data model[13] that has been mapped into RDF by keeping the same property names; YAGO uses its own data model [168]; Babelnet implements the lemon vocabulary[14]; Wikidata has developed its own ontology [48]. 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. Finally, there is also knowledge bases that are not represented as linked data, some of them are:

- Wikipedia[15] 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.

- Freebase [16] 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[16] (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.

- Musicbrainz[17] is a project that aims to create an open data music relational database. It captures information about artists, their recorded works, the relationships between them. Musicbrainz is maintained by volunteer editors and contains over 53 million entities. A linked data version of Musicbrainz named LinkedBrainz[18] is also regularly generated.

[12] http://wiki.dbpedia.org/services-resources/ontology
[13] https://developers.google.com/freebase/guide/basic_concepts
[14] http://lemon-model.net/lemon
[15] http://www.wikipedia.org
[16] https://discourse.cayley.io/t/query-languages-tour/191
[17] http://www.wikipedia.org
[18] https://wiki.musicbrainz.org/LinkedBrainz

- WordNet [107] is a lexical database developed by linguists from the cognitive science laboratory at Princeton University over the past twenty years. Its purpose is to list, classify and relate in various ways the semantic and lexical content of the English language. Versions of WordNet for other languages exist, but the English version is the most complete and up to date.
- JeuxDeMots [87] is a lexical network for French and created through a two player blind game based on agreement. Blind implies that the other player is unknown until the end. Agreement means that both players will always score the same amount of points according to what they have proposed in common. Then, the network if built according to what the players propose, the more they share answers the strongest will be the relations between the words.

While these knowledge bases are not using linked data to represent their content, they can also be used to perform the task of entity linking.

2.2 Entity Recognition Architectures

Since the last 20 years we have seen two kind of approaches for entity recognition: handcrafted rules-based and learning approaches.

2.2.1 Rules-based Approaches

The first entity recognition approach proposed in 1991 [141], was handcrafted rules-based and has been developed to recognize company names. Currently, this kind of approach is not very popular and is mostly used for specific domains [52, 170, 122] or specific entities [25]. Despite the oldness of this approach, such systems are still being used, even for a broad set of types, especially in LIMA [13]. Rules are usually regular expressions that combine information from terminological resources and characteristics of the entities of interest. We take two examples of rules understood by LIMA [13], one being very simple (Listing 2.5, and another one being more complex (Listing 2.6).

```
Liberation:::ORGANIZATION:
Monde:Le:Diplomatique:ORGANIZATION:
Monde:Le:de l'Education:ORGANIZATION:
Monde:Le::ORGANIZATION:
Courrier::International:ORGANIZATION:
Canard::Enchaine:ORGANIZATION:
```

Listing 2.5 – Simple LIMA rules

The rules in Listing 2.5 recognize the names of French newspapers. The rules in Listing 2.6 recognize the persons where the first line is triggered on a first name (the list of first names must be also defined in the rule file). The left context optionally

```
@Firstname:[(@Title|@FunctionTitle)?]:((de|da|le)? t_capital_1st){1-2}:PERSON:
t_capital_1st:[(@Title|@FunctionTitle)]:t_capital_1st{0-2}:PERSON:
```

Listing 2.6 – Complex LIMA rules

contains a title (Mr, Mrs, Dr, ...) or a function name (Chairman, MP, ...), which is not kept in the rule result. The right context must be composed of one or two capitalized words, possibly preceded by *de*, *da* or *le*. The second line recognizes the names of persons introduced by titles or function names, but without a first name. These two examples are taken from the LIMA documentation [19].

The main disadvantage of these methods is the manual construction of the rules, which is a time-consuming task. Mainly for this reason, most of the recent approaches are based on learning.

2.2.2 Learning Approaches

Most efficient learning approaches are based on supervised approaches. In this section we do make the difference between: machine and deep learning. Machine learning are approaches that are based on feature engineering, and can be associated as an equivalent of a single hidden layer neural network.

Deep learning techniques represents a huge step forward for machine learning. It is based on the way the human brain process information and learns. It consist in a machine learning model composed by a several levels of representation, in which every level use the information from the previous level to learn deeply. Each level correspond, in this model, to a different area of the cerebral cortex, and every level abstract more the information in the same way of the human brain. Thus, no need anymore of feature engineering.

2.2.2.1 Supervised Approaches

Until three years ago and during fifteen years, the most popular approaches was based on common classification algorithms [111].

Support Vector Machine. Support Vector Machines are a class of learning algorithms initially defined for discrimination, or more formally, for the prediction of a qualitative binary variable. They were then generalized to the prediction of a quantitative variable. In the case of discrimination of a dichotomous variable, they are based on the search for the optimal margin hyperplane which, when possible, classifies or separates the data correctly while being furthest away of all observations. The principle is therefore to find a classifier, or a discrimination function, where the capacity of generalization (forecast quality) is as big as possible. The basic principle of

[19] https://github.com/aymara/lima/wiki/LIMA-User-Manual

support vector machines is to reduce the problem of discrimination to a linear search for an optimal hyperplane. Two ideas or tricks make it possible to achieve this: *i)* defining the hyperplane as a solution to a problem of constrained optimization whose objective function is not expressed by using scalar products between vectors and in which the number of "active" constraints or support vectors controls the complexity of the model, and *ii)* the transition to the search for nonlinear separating surfaces is obtained by introducing a kernel function into the scalar product implicitly implying a non-linear transformation of the data to a feature space of s greater dimension. This lands to the commonly encountered name of kernel machine. On the theoretical level, the kernel function defines a Hilbert space being self-reproducing and isometric by the nonlinear transformation of the initial space and in which the linear problem is solved. Few named entity recognition approaches that use a support vector machine are [5, 159, 192]

Decision Trees. A decision tree classifies data by posing a series of questions about the features associated with the data. Each question is contained in a node, and every internal node points to one child node for each possible answer to the question. The questions thereby form a hierarchy encoded as a tree. Each example is sorted into a class by following the path from the topmost node, the root, to a node without children, a leaf, according to the answers that apply to the example under consideration. An example is assigned to the class that has been associated with the leaf it reaches. In some variations, each leaf contains a probability distribution over the classes that estimates the conditional probability that an example reaching the leaf belongs to a given class. Few named entity recognition approaches that use decision trees are [154, 120].

Ensemble Learning. Ensemble learning is an approach that creates multiple models and then combine them by different heuristics to produce improved results. Ensemble learning usually produces more accurate solutions than a single learning approach would. It exists some widely known methods for doing ensemble learning: voting, averaging, stacking, bagging and boosting. **Voting and averaging** are two of the easiest ensemble methods. They are both easy to understand and implement. Voting is used for classification and averaging is used for regression. In both methods, the first step is to create multiple classification/regression models using some training dataset. Each base model can be created using different splits of the same training dataset and same algorithm, or using the same dataset with different algorithms, or any other method. There are four different voting and averaging approaches: *i)* majority voting is when every model makes a prediction (votes) for each test example and the final output prediction is the one that receives more than half of the votes. If none of the predictions get more than half of the votes, we may say that the ensemble method could not make a stable prediction for this example. Although this is a widely used technique, you may try the most voted prediction (even if that is less than half

of the votes) as the final prediction ; *ii)* weighted voting is when each model has different rights, unlike majority voting, and we can increase the importance of one or more models ; *iii)* averaging is when for every example of test dataset, the average predictions are calculated ; *iv)* weighted averaging is a slightly modified version of averaging, where the prediction of each model is multiplied by a weight and then their average is calculated. Few named entity recognition approaches that use ensemble learning are [163, 47, 86]

Bayesian Naive Classification. Bayesian naive classification is a Bayesian type of simple probabilistic classification based on the theorem of Bayes with a strong (or naive) independence of hypotheses. A naive Bayesian classifier assumes that the existence of a characteristic for a class is independent of the existence of other characteristics. An animal can be considered as a cat if it has four paws, a whiskers, and a tail. Even if these characteristics are related in reality, a naive Bayesian classifier will determine that the animal is a cat by considering these characteristics independently. In many practical applications, the estimation of parameters for naive Bayes models relies on maximum likelihood, which means that it is possible to work with a Bayesian naive model without worrying about Bayesian probability or using any Bayesian methods. A named entity recognition approaches that use Bayesian naive classification is [149].

Maximum Entropy Models. Maximum entropy classifier is a probabilistic classifier which belongs to the class of exponential models. Unlike the Bayesian Naive classifiers that we discussed in the previous paragraph, the maximum entropy does not assume that the features are conditionally independent of each other. The maximum entropy is based on the principle of maximum entropy, which means that it selects the model that has the largest entropy. The principle of maximum entropy consists, when one wants to represent an imperfect knowledge of a phenomenon by a law of probability, we have to identify the constraints that this distribution must address and choosing from all the distributions corresponding to these constraints the one with the greatest entropy in the sense of Shannon (a mathematical function which, intuitively, corresponds to the quantity of information contained or delivered by a source of information). A named entity recognition approaches that use maximum entropy is [35].

Hidden Markov Models. The hidden Markov models is a sequence classifier where its job is to assign a label or class to each unit in a sequence, thus mapping a sequence of observations to a sequence of labels. An hidden Markov model is probabilistic: given a sequence of units (words, letters, morphemes, sentences, etc.), it computes a probability distribution over possible sequences of labels and choose the best label sequence, finally, the sequences are always oriented from left to right. Being oriented is one of the reasons of why these models are derived from what we call the Markov chain, it is a special case of a weighted automaton in which weights

are probabilities (the probabilities on all arcs leaving a node must sum to 1) and in which the input sequence uniquely determines which states the automaton will go through. Few named entity recognition approaches that use the hidden Markov models are [45, 203].

Multilayer Perceptron. Multilayer perceptron is a kind a feed-forward deep neural network as detailed in 2.1.1.6, where the number of hidden layers is equal to one. A named entity recognition approaches that use a multilayer perceptron is [57].

Conditional Random Fields. Conditional random fields is a probabilistic approach for labeling and segmenting sequential data, based on a conditional approach. Conditional models are used to label an observation sequence by selecting a label sequence which maximizes the conditional probability that the observation sequence defines a label sequence. To this end, a conditional random field is a form of undirected graph model that defines a single log-linear distribution over label sequences given a particular observation sequence. These models have been developed in order to answer to the issues raised by the Markov-based models like hidden Markov models and have been for several years the best method to achieve the named entity recognition task such as[54, 101] until deep learning approaches starts to become popular.

Although all these type of classifiers work differently, they have one thing in common, the features they use to be trained that are often based on:

1. Word-level features:
    - case: lowercase, uppercase, normalized
    - punctuation: hyphen, internal period
    - digit: numbers, dates
    - character: Greek letters
    - morphology: plural, lemma
    - part-of-speech: adjectives, proper nouns
    - function: multi-tokens expression
2. gazetteers: list of common entities, usually generated from knowledge bases
3. contextual features:
    - window: a number of words before and after the current word
    - topic of the document
    - occurrences of the entities

In the cited supervised approaches for named entity recognition, the authors do not change the algorithm of the classifier itself but they all use different features

based on the list above. To this end, it is easy to train multiple classifiers with a set of features and to compare the results and see which classifier works better than another on such or such dataset. The biggest part of the work with these approaches is then the feature engineering, which means selecting the best features. This is the main difference with the new deep learning-based approaches where we do not need anymore a feature engineering process and that are mostly based on word embeddings.

2.2.2.2 Deep Learning Approaches

One of the most well-known weaknesses of the previous generation of NLP approaches is that they are unable to model semantic similarity between two words. For example, for a CRF model, *New York* and *Turin* differ as much in meaning as *New York* and *rabbit*. In order to overcome this problem, people use word embeddings to have an understanding of semantic similarities in Entity Recognition. Word embeddings are one of the main drivers behind the success of deep learning in NLP. Usually, the first layer in a neural network for text processing is mostly an embedding layer that maps one-hot word vectors to their dense embeddings. Recently, we have seen multiple techniques that use deep neural networks to tackle Entity Recognition even though some people have started to use these neural networks quite some time ago [71]. This first approach uses a Long Short Term Memory (LSTM). They are a type of recurrent neural network, which means that they are ideal for dealing with sequential data. The great thing about these kind of neural networks is that they are able to learn long-term dependencies, as their structure allows the network to pass or block information from one step to the other. Unfortunately, the performances were not very convincing [152]) due to the usage of very small neural networks because the resources available at that period was not allowing people to train large neural networks, and performing word embeddings techniques were not existing yet.

These last three years, a large amount of approaches have been released and they are based on the same kind of neural networks, LSTMs with word embeddings. Nevertheless, a simple LSTM has a problem, it uses the context only from one direction, the left one. To overcome this issues, people has started to use Bidirectional LSTM (BiLSTM). This new version process the data with two separate LSTM layers. One of them reads the sentence from left to right, while the other one receives a reversed copy of the sentence to read it from right to left. Both layers produce an output vector for every single word, and these two output vectors are concatenated to a single vector that now models both the right and the left context [186]. The last techniques, surprisingly, replace the final softmax layer of a BiLSTM networks, with a CRF layer and have shown to yield to improvements [80, 99, 88]. In [99], the authors go even further and develop a CNN on the top of the BiLSTM in order to look at the individual characters of a word and collect important information present in the character

sequences.

2.3 Coreference Resolution

In [114], the author describes two categories of coreference resolution methods: *i)* machine learning-based, and *ii)* semantics - world knowledge-based. The first category of methods propose multiple different models to solve this task:

1. Mention-pair models are certainly the most common coreference model. A mention-pair model is a binary classifier that determines if a pair of mentions is co-referring or not. To train such a model, each training example is a pair of mentions represented by local features encoding each of the two mentions and their relationships. Any learning algorithm can be used to train a mention-pair model. However, these models can violate the transitivity, which is a native property of the coreference relation (see Section 2.1.1.5).

2. Mention-ranking models. A weakness of the mention-pair models is to consider each candidate antecedent of an anaphoric mention be resolved independently of other candidate antecedents. As a result, they can only determine how good a candidate antecedent is relative to the anaphoric mention, but not how good it is relative to other candidate antecedents. Mention-ranking models tackle this weakness by allowing candidate antecedents of a mention to be ranked simultaneously.

3. Entity-based models. Another weakness of the mention-pair models is their lack of expressiveness: they can only employ local features that cannot be defined on more than two mentions. However, the information extracted from the two mentions alone may not be sufficient for making a proper coreference decision, especially if the candidate antecedent is a pronoun or a mention without any descriptive information such as gender or speaker. Entity-based models aim to fix this expressiveness problem.

4. Graph-based models. The goal of such models is to cast coreference as a graph partitioning problem. Given a test document, a graph is first constructed, where the nodes and edges typically correspond to the mentions and their compatibility.

5. Joint models. These models put multiple models together by making joint classification decisions. Joint models allow also the integration of external knowledge that encodes task-specific constraints.

6. Easy-first models. These coreference models aim to make easy linking decisions first. Like entity-based models, easy-first models can employ cluster-level features: the information extracted from the clusters established and can be used to help identify the difficult links.

7. Antecedent structure-based models. This kind of model predicts for each test document the most probable antecedent structure, which is a vector of antecedents storing the antecedent chosen for each mention (empty if the mention is non-anaphoric) in the document. Effectively, it is a mention-ranking model, but it is trained to maximize the conditional likelihood of the correct antecedent structure given a document. Inference is easy because the most probable candidate antecedent of a mention is selected to be its antecedent independently of the other mentions.

Most of these different models are combined together in current deep neural networks models such as Stanford deep-coref [31]. As a starting point, the deep-coref pre-trains a cluster-ranking model that takes advantage of entity-level information, with a neural mention-ranking model inspired from [190]. In [189], the authors extend their previous mention-ranking model [190] by integrating entity-level information taken from the output of a recurrent neural network running over the candidate antecedent-clusters. Nevertheless, this is a simple change in their original mention-ranking model, and then not a true clustering model as the deep-coref cluster ranker is. Coreference resolution systems such as joint inference [102, 135, 68] and those that construct coreference clusters incrementally [97, 194, 138] integrate entity-level information. Stanford deep-coref takes inspiration from the second kind of system and particularly from a combination of cluster-ranking [139, 98] and easy-first clustering strategies [166, 29]. While most of the previous systems used handcrafted features to integrate linguistic constraints, Stanford deep-coref, in addition, uses a learning-to-search approach inspired from [26] in order to learn from data the entity-level distributed representation.

Finally, the second category of coreference resolution approaches that uses structural knowledge from external datasources [167, 134, 18, 175] such as Wikipedia, YAGO or WordNet.
SANAPHOR [137] belongs to that category and uses DBpdia and YAGO to help disambiguate the different entities that are involved into a coreference cluster. SANAPHOR is plugged to the output of the Stanford decoref [91] coreference resolution system, and improves its results by linking the different entities by deciding if a cluster has to be merged with another one, or if it has to be split based on the type, and the link (from YAGO or DBpedia) of the disambiguated entities.

In general, the difficulty of coreference resolution stems from its reliance on sophisticated knowledge sources and inference mechanisms [109]. Despite its difficulty,

coreference resolution is a core task in information extraction as it is the fundamental part for consolidating the textual information about an entity.

2.4 Entity Linking Architectures

The architectures for doing entity linking task may vary depending of how we want to link entities. Regardless of the different entity linking components that intervene in typical workflows [145], 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*.

Except the fourth workflow, all the others have at least one step in common: candidate generation.

2.4.1 Candidate Generation

A referent knowledge base such as DBpedia, contains a huge number of entities, and then it becomes impractical to consider all the entities it contains as candidate. Therefore, we have to find a solution in order to retrieve only the most relevant entities that can potentially be candidates for the searched mention. This solution is called *candidate generation*. We describe in the following two common approaches for generating entity candidates.

Dictionary-based. Currently, this method is the most popular and is adopted by numerous approaches [66, 60, 19, 34, 176]. The methods that use a dictionary, have created it with Wikipedia article pages, redirect pages, disambiguation pages and in-article hyperlinks. Some of them also, extend the dictionary with name variations, abbreviations, confusable names, spelling variations, nicknames, etc. Wikipedia is not the only one knowledge base to use as dictionary, one can also use Babelnet [110]. It is represented as a key-value storage, where the key is the mention to search against, and the value is the list of entity candidates corresponding to the mention (key).

See Table 2.1 to have an example. Once the dictionary is built, we use a string matching approach over it. It exists two different kind of string matching: strict and fuzzy. Strict matching is when the mention and the candidate entity label are strictly the same. Fuzzy matching is when the mention and the candidate entity label are similar. This similarity is based on string comparison metrics (see 2.1.1.8) such as Levenshtein, Dice or Jaccard. Therefore, given the score retrieved from the approach and a threshold, we can decide if a candidate entity is similar enough to the mention or not to become a potential entity candidate. This candidate generation method is very costly in computing time and can mainly be applied on small strings due to the complexity of these algorithms that are based on the maximum length of the two strings being compared, and the process has to be done over the entire entities contained in the knowledge base.

Paris	*db:Paris,db:Paris_(Texas),db:Paris_Hilton,db:Paris_Saint_Germain*

Table 2.1 – Example of entities dictionary

Information Retrieval Systems. A more sophisticated approach is to use information retrieval systems that are specifically developed to resolve such problem [24, 53, 123]. The goal of those systems is to act like a search engine, for a given mention retrieving the top K entity candidates, K being a threshold chosen a priori. These systems allows to index an entire document, such as a Wikipedia article, and allow to make a search very efficiently on top of them. Therefore, we are not fixed to use the labels and/or their augmentation to retrieve the candidates, but we can also use the entire document from which they come from. Such systems are Lucene [55] or Elasticsearch [46] and SolR [56] that use Lucene underneath.

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 [123, 53, 103, 153, 24]. 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 [131].

2.4.2 Entity Selection or Ranking

In order to select the most probable entity among the candidates, we have to apply an approach to rank them. Currently, it exists two different approaches: independent or collective. Each of the two can be realized with supervised or unsupervised methods.

1. *independent*: these methods consider that the mentions to be linked in a document are independent, and do not leverage the relations between all the mentions into a same document to help to select the best entity. In order to rank

the candidate entities, they mainly leverage the context similarity between the text around the mention and the document associated with the candidates.

2. *collective*: contrarily to the independent methods, these methods assume that a document largely refers to common entities from one or a few related topics, and entity assignments for the mentions in one document are interdependent with each other. Thus, in these methods, the mentions in one document are collectively linked by exploiting this *commonness*.

2.4.3 Independent Approach

Common features used for this kind of approach are:

- *string similarity*: the logic here is the same than the one explained in Section 2.4.1. We use a string similarity to give a syntactic score to the candidates.
- *entity popularity*: the logic here is to assume that each candidate has a different popularity, meaning that a mention will be more likely linked to a popular entity candidate than others. For example, in most cases when people talked about the mention *Paris*, there is a higher chance they talk about the French city, than any others. The entity popularity might be computed with different techniques: prior probability [76, 53, 24], Google PageRank [43], Wikipedia statistics such as page views, number of editions or page length [66, 60].
- *entity type*: the logic here is to indicate if the type of the mention such as PERSON, ORGANIZATION, or LOCATION in text is consistent with the type of the candidate entity in the knowledge base, and apply a filter in case they are different [66].

2.4.3.1 Unsupervised Independent Approach For Entity Linking

Unsupervised method for an independent approach are mainly based on information retrieval techniques. These techniques rely on mathematic formula where each part of the formula is based on statistics computed with the features above [63, 181]. Therefore, the features for each candidate entity is indexed as a separate document, and for each mention, they generate a search query from the mention. Finally, the search query is given to the index and the candidate entity which has the highest relevant score is retrieved as the linked entity for the mention.

2.4.3.2 Supervised Independent Approach For Entity Linking

Some methods propose to solve the problem of entity linking with a binary classifier [73, 124], where they decide the best couple mention-entity from a list of pairs

of mentions and candidate entities. The examples in the training and test datasets was composed of positive and negative examples, the positive examples are all the that already exists from an annotated document, and the negative examples are automatically generated by all the combinations of pairs that were not already written as positive. The features used to represent a pair are all described at the beginning of this section: string similarity, entity type and entity popularity. The main used classifiers are Support Vector Machine, Naïve Bayes or k-Nearest Neighbors. In the case a same mention is annotated as positive with two different entity candidate, another classifier is used to determine which one will be the proper pair.

Another popular supervised method to solve this problem is based on ensemble learning [199, 27]. These apply an ensemble method over the results of multiple binary classifiers, or other published approaches to get the results.

2.4.4 Collective Approach

All these approaches are mostly based on trying to represent a collective context among all the mentions, either from the document point of view or from the knowledge base point of view.

- *document context*: this context aims to represent the mentions from the document point of view. The two most common features are *bag of words* and *semantic vectors*. They both have one thing in common, being a vector representation allowing to use standard vector similarities such as cosine, Jaccard, dot-product or Dice.
  - *bag of words*: this is an approach often used in information extraction that counts how many times a word appears in a document. Those word counts allow to compare documents and measure their similarity. They lists words with their word counts per document. Their representation is like a table where the words and documents become vectors, and each row is a word, each column is a document and each cell is a word count. Each of the documents in the corpus is represented by columns of equal length. The length corresponds to the number of distinct words across all documents and the columns are what we call word count vectors. Therefore, for each mention, the document context is represented as a bag of words collected from the input document where the mentions appears, and for each candidate entity, the context is usually represented as a bag of words from the whole Wikipedia page [66, 199, 93].
  - *semantic vectors*: this is an approach that aims to build vectors that represents the semantic of the mentions and the candidate entities. This semantic vector is a concatenation of multiple features such as keyphrases,

anchor texts, categories and concepts. For the mentions, this representation is built with the input document, and for the candidate entities with their Wikipedia article [34, 157, 43].

- *knowledge base context*: this context aims to represent the candidate entities from the knowledge base point of view. The main approach to represent a knowledge base context is what we call the commonness. This is a way to measure how close to each other two entities are in the knowledge base. For that, it exists multiple fashions, the most popular being Milne&Witten [108] or Deep Semantic Relatedness Model [78]. The former being the first published where they use common incoming Wikipedia links, and the later being the first to introduce entity embeddings and deep neural networks to compute the commonness. These methods will be detailed in the following sections.

2.4.4.1 Unsupervised Collective Approach For Entity Linking

The most popular unsupervised methods for a collective linking approach, are graph-based. The earliest published work are [74] and [76]. The former proposed to model the global topical independence between different entity linking decisions in a document as a graph (referent graph) which could model the document context and the knowledge base context in same time, and finally apply an inference algorithm (similar to the PageRank) in order to find the best solution. The latter propose to build a mention-entity graph with three different features: entity popularity, document context and knowledge base context. Given this constructed graph, their goal is to compute a dense subgraph that contains exactly one mention-entity edge for each entity mention. To do so, they use a method that is able to find strongly interconnected, size-limited groups in a graph, derived from [161]. Another graph based approach is Babelfy [110], they use a method inspired from word sense disambiguation domain, applied to entity linking with a random walk algorithm. The most efficient graph-based approach is what is called graph regularization [78, 77]. The nodes of the graph are mention-entity pairs and each node has a ranking, which determines the probability of the couple being the best solution, the initial ranking is determined by selecting seed nodes, and finally the regularization determines the final rankings for each node, the highest ranked node as the correct solution.

2.4.4.2 Supervised Collective Approach For Entity Linking

The earliest supervised methods for a collective approach used are with a learning to rank model. Learning to rank is a type techniques that aims to automatically construct a ranking model from a training dataset. This kind of training data is composed of a list of examples where each of them is an ordered list composed of

the features described above. Basically, for each mention we have an ordered list composed of feature vectors of each candidate entity from the most likely to the less likely. This kind of representation allows to have a single ranking example for each possible mention, and then we just need to select the candidate entity which achieves the highest score as the linked entity [200, 140, 157]. The most popular classifier used to proceed a learn to rank is Support Vector Machine. Although, learning to rank is a popular method, few other entity linking methods use a more probabilistic process. In [72], they use a generative model to collectively link the entities from a document. A generative model describes how data is generated, in terms of a probabilistic model, and is most of the time used to automatically generate data, but sometimes also to make predictions, like in our case. In this method, they use exclusively knowledge base context features to create their probabilistic model. Another interesting method is based on crowdsourcing [36]. This method takes advantages of human predictions with a set of annotations done by humans in order to improve the results of usual methods. More precisely, they have created a probabilistic model that takes into account human decisions from annotated data on a crowdsourcing platform and automatic decisions made by machines, we can also see this approach as a kind of ensemble learning, but instead of mixing only machine learning techniques, they use also human annotations.

2.4.4.3 Joint Recognition-Linking

Since recently, few methods are doing what we call *joint recognition-linking*. The goal of these methods is to recognize and link the entities in same time [115, 95, 44, 158]. They are all mostly based on the same approach, on a supervised, non-linear graphical model, derived from Conditional Random Fields, that combines multiple per-sentence models into an entity coherence-aware global model. The global model detects mention spans, tag them with coarse grained types, and map them to entities in a single joint-inference step based on the Viterbi algorithm (for exact inference) or Gibbs sampling (for approximate inference). In order to label an input of tokens with output labels (types and entities) they use a family of linear-chain and tree shaped probabilistic graphical models. These models are used to better encode the distribution of multiple probability. These per-sentence models are optionally combined into a global factor graph by adding also cross-sentence dependencies. These cross-sentence dependencies are added whenever overlapping sets of entity candidates are detected among the input sentences. The search space of candidate entities for the models depends of the mention spans as they are determined independently for each sentence. They use pruning heuristics to restrict this space such as spans of mentions that are derived from dictionaries, and they consider only the top-20 entity candidates for each mention. In order to generate linguistic features (tokenization, sentence detection, POS tagging, lemmatization, and dependency parsing) they use

Stanford CoreNLP [100], they build an entity repository and name-entity dictionary using YAGO2 to detect the potential mentions.

2.4.4.4 Deep Learning

Since 2015 we start to see deep learning based methods coming out. The earliest method being [78] where they developed the notion of entity embeddings, by using data about entity-entity co-occurences in a knowledge base such as relations or types as feature for a deep neural network called deep semantic relatedness model. This model is based on the so-called Deep Semantic Similarity Model [79] which aims to be able to create embeddings of any kind of objects and then compute a similarity between these embeddings. Other approaches [193, 50, 205] of entity embeddings also use only entity-entity features. A most recent method [59] proposes to compute entity embeddings without taking into account any data from the knowledge base but essentially from the canonical Wikipedia entity pages and the local context of their hyperlink annotations, in order to allow a more efficient training, avoid to compute co-linking statistics or any heuristics, and avoid hand-engineered features. While [78] use a graph-based approach for linking entities, and the others [193, 50, 205] are only based on embeddings, the method in [59] use a context attention in order to select words that are informative for the disambiguation decision with a learned combination of the context-based entity scores and a mention-entity prior to get a final score. Finally, they use a conditional random field inference layer to determine the best entity to be linked, inspired from the named entity recognition methods depicted in Section 2.2.2.2.

2.4.5 NIL Clustering

Entities are tagged as NIL when no correspondence to them in the knowledge base (or we are not able to select the appropriate one). Then, NIL entities imply that a new entry could be included into the knowledge base. In a set of documents, several mentions may refer to a same NIL entity, which means that all the mentions related to the same NIL entity should be grouped in clusters with the same id (e.g. NIL001, NIL002, etc.). Therefore, each NIL cluster could correspond to a new entry in the reference knowledge base. Several techniques have been applied to this task, and they all belong to approaches that have participated to the TAC-KBP campaign[20] as it is a requirement in the guidelines of their challenges.

String Matching. This technique consists in grouping queries by mention. For instance, two NIL entities with the name *NAME* would be clustered together and, so, would be given the same NIL ID. A mention may contain fragments of other mentions. To this end, the fuzzy string matching is applied. It clusters all NIL entities with a

[20]Text Analysis Conference-Knowledge Base Population

distance metric between the mentions higher than a predefined threshold. Systems that use this approach are: [20, 23, 75].

Hierarchical Agglomerative Clustering. In this approach, those mentions referring to the same NIL entity are clustered using a Hierarchical Agglomerative Clustering algorithm. Hierarchical clustering can be approached by top-down and bottom-up algorithms. The bottom-up algorithm treats each mentions as a singleton cluster at the outset and then sequentially merge the pairs of clusters as long as two clusters exceeded a similarity threshold. Systems that use this approach are: [125, 201]. In contrary, top-down approach works by starting with a root cluster, where all the candidates are placed and then recursively splitting the clusters based on the most likely partition in each stop.

Graph-based Clustering. Using this approach, a graph structure is generated for the clustering. In [202], they employed a graph-based approach to cluster the NIL entities. They used Spectral Graph Partitioning [113] to generate the globally optimized entity clusters. The results obtained by the spectral graph partitioning usually outperform the traditional clustering algorithms such as k-means or minimum-cut [202].

Topic Modeling. Topic modeling is a statistical model to explore the topics underlying the documents. The topic modeling approaches are widely used in different NLP applications [202]. Latent Dirichlet Allocation is a common topic model approach that was first presented as graphical model for topic discovery [14]. The topics are a probability distributions over words. Systems that use this approach are: [202, 85].

2.5 Approaches Summary

This section shows a summary of several state-of-the-art systems that will be used to compare our results for evaluation purpose. These approaches are divided in two tables, the Table 2.2 details the extraction or recognition techniques adopted, and the Table 2.3 details the linking techniques. Some of the approaches referred in the second table do not appear in the first one because they are only able to link entities. The approaches are: AIDA [76], Babelfy [110], DBpedia Spotlight [103], Dexter [24], Entityclassifier.eu [41], FOX [176, 162], FRED [33], FREME[21], KEA [165], TagMe 2 [53], WAT [123], X-LiSA [198], AGDISTIS [176], DoSeR [204], NERFGUN [70] and PBOH [58].

The two tables share two columns (*Recognition* and *Candidate Generation*) which indicate if the corresponding system does recognition or generate candidates at the step represented by the table. For example, if there is a *yes* in Table 2.2 for the column *Candidate Generation*, it means that the candidates are generated during the

[21] https://freme-project.github.io/api-doc/full.html

entity extraction process and not during the linking. The same logic holds with the *Recognition* column.

The systems in the tables are all ordered by chronological order, from the older to the newer. In Table 2.2, we can see that the trend is to rely on external supervised natural language processing tools. The few others are based on a dictionary. The work described in this document rely on both, for the main reason that labeled data in most of the languages to properly train a supervised approach such as (part-of-speech tagger or named entity recognition tagger) are rare, and for this case using a dictionary is useful. In Table 2.3, we can see that the trend is more oriented to a collective approach with an equal distribution between graph-based and unsupervised. Independent approaches are equally distributed among supervised and unsupervised. Also, doing *NIL clustering* is not often handled by these systems including the most recent. The work described in this document proposes collective and independent approaches for linking entities, including NIL entities with a *NIL clustering* method.

The Table 2.4 provides details on the possibility to address the four challenges mentioned in Chapter 1, Section 1.2 that we propose to tackle in this work: text independency, knowledge base independency, language independency and entity type independency. We can see that the systems have difficulties to tackle these challenges altogether, as they address at most two challenges and sometimes none. Systems without a symbol in a column represent the fact that they do not perform entity extraction or recognition. The work described in this document proposes an adaptable approach to tackle each of these challenges at the same time.

Entity Extraction							
Year	System	External Tool	Main Features	Method	Language Resource	Recognition	Candidate Generation
2010	TagMe 2	-	N-Grams	lexical similarity	Wikipedia gazetteer	no	yes
2011	AIDA	StanfordNER	-	-	NER Dictionary	yes	no
2011	DBpedia Spotlight	LingPipePOS	syntactic features	lexical similarity	DBpedia gazetteer	no	yes
2013	KEA	-	syntactic features	-	DBpedia gazetteer	no	yes
2013	Entityclassifier.eu	GATE	Syntactic features	-	Wikipedia gazetteer	no	yes
2013	Dexter	-	N-Grams	lexical similarity	Wikipedia gazetteer	no	yes
2014	WAT	OpenNLP	syntactic features	Collective agreement, Wikipedia statistics and SVM	Wikipedia gazetteer	no	yes
2014	X-LiSA	-	syntactic features	-	Wikipedia gazetteer	yes	no
2014	Babelfy	-	syntactic features	Lexical Similarity	Babelnet	no	yes
2014	FOX	StanfordNER, OpenNLP, Illinois NE Tagger, Ottawa Baseline IE	-	Ensemble Learning	-	yes	no
2015	FRED	TagMe	-	-	-	yes	yes
2016	FREME	-	syntactic features	Conditional Random Field	-	yes	no

Table 2.2 – Analysis of Named Entity Extraction and Recognition systems

EL (Entity Linking)							
Year	System	Main Features	Method	Knowledge Base(s)	NIL Clustering	Recognition	Candidate Generation
2010	TagMe 2	collective approach	unsupervised	Wikipedia	no	no	no
2011	AIDA	collective approach	graph-based	YAGO2	no	yes	yes
2011	DBpedia Spotlight	independent approach	unsupervised	DBpedia	no	no	no
2013	KEA	independent approach	unsupervised	DBpedia	yes	no	no
2013	Entityclassifier.eu	independent approach	unsupervised	DBpedia	no	no	no
2013	Dexter	collective approach	unsupervised	Wikipedia	no	no	no
2014	WAT	independent approach	supervised	Wikipedia	no	no	no
2014	X-LiSA	collective approach	unsupervised	DBpedia	yes	no	yes
2014	Babelfy	collective approach	graph-based	Babelnet	no	no	no
2014	AGDISTIS	collective approach	graph-based	DBpedia	no	no	yes
2014	FOX	collective approach	graph-based	DBpedia	yes	no	yes
2015	FRED	collective approach	unsupervised	Wikipedia	no	no	no
2016	FREME	independent approach	supervised	DBpedia	no	yes	yes
2016	DoSeR	collective approach	unsupervised	Wikipedia, Freebase	no	no	yes
2016	PBOH	independent approach	supervised	Wikipedia	no	no	yes
2016	NERFGUN	collective approach	graph-based	DBpedia	no	no	yes

Table 2.3 – Analysis of Entity Linking systems

System	text independency	knowledge base independency	language independency	entity type independency
TagMe 2	✓	✗	✗	✗
AIDA	✓	✗	✗	✗
DBpedia Spotlight	✗	✗	✓	✗
KEA	✓	✗	✗	✓
Entityclassifier.eu	✗	✗	✗	✗
Dexter	✓	✗	✗	✗
WAT	✓	✗	✗	✗
X-LiSA	✓	✗	✗	✓
Babelfy	✗	✗	✓	✗
AGDISTIS	-	✗	✓	-
FREME	✗	✗	✗	✗
FRED	✗	✗	✗	✗
FOX	✗	✗	✗	✓
DoSeR	-	✗	✗	-
PBOH	-	✗	✗	-
NERFGUN	-	✗	✗	-

Table 2.4 – Availability of the systems for the four challenges tackle in this thesis

People who think they know everything are a great annoyance to those of us who do.

Isaac Asimov

3 Entity Recognition

In this chapter, we describe our approach to recognize mentions from texts that are likely to be selected as entities, and their pre-processing when they are coming from social media. After having identified the mentions, we resolve their potential overlaps using a so-called *Overlap Resolution* module.

3.1 Social Media Preprocessing

One of the challenge this work proposes to tackle is textual content diversity. This includes texts coming from social media platforms which need particular preprocessing to be handled by an entity linking approach, because they are very challenging, and often contain two typical characteristic: hashtags and user mentions. Before processing such textual content, we propose to deal with such characteristics.

3.1.1 Hashtag Segmentation

In social network like Twitter or Instagram, the hashtags are single words or unspaced phrases preceded by the symbol '#'. The content of a hashtag can be composed by abbreviations, wrongly spelled words, made-up words containing arbitrary character sequences, numeric digits, and entities [156]. They are often used to give a context to a message. This allows to create subsets of tweets over millions of them by grouping them according to their topic. In order to properly identity the entities that a hashtag might contain, we have first to properly segment the hashtag. For example, turn the hashtag *#Iloveparisandrome* into *I love paris and rome*, to be able to recognize *paris*

and *rome* as *LOCATION* entities. We can also imagine more complex cases, where there are more than a single segmentation, for example *#goodweather*, that can be segmented into *good weather* or into *good we at her*. Both are a correct segmentation but the former is more meaningful.

Our approach follows the one proposed by [142], which is a dictionary based approach. The hashtag is segmented through multiple heuristics. Each heuristic takes as input the hashtag and return a possible segmentation, next, the segmentation of each heuristic is scored in a scoring function, and the segmentation that has the highest score is taken as the most probable proper segmentation. As resource we use two different dictionaries: *1)* a single word American-English dictionary that one can find in any Linux distribution located at /usr/share/dict/american−english, and *2)* a two-grams dictionary available online[1] based on the largest publicly-available, genre-balanced corpus of English, *the Corpus of Contemporary American English (COCA)*, and it is composed of the first 1 million two-grams scored based on their frequency into the corpus. We now list below the available heuristics.

Entire hashtag checking. The first heuristic we apply is to check if the entire hashtag is itself a single. To know that, we check if the hashtag has a full match in the single word dictionary. We do that with three casing logic of the hashtag: as it is, all lower case, and capitalized. As soon as the hashtag is found the heuristic returns the entire hashtag as a correct segmentation.

Camel case. The hashtag is segmented based on capitalization i.e. each time a capital letter is encountered the hashtag will be segmented. For example, *#GoodToMe* becomes *#Good To Me*. Once we have this segmentation, we check if the two by two tokens appear in the two-grams dictionary. For checking in the dictionary we apply the same casing logic than with the previous heuristic for each token. In this example, we find in the dictionary *good to* and *to me*. Each two-grams in the dictionary has a score, and we use as segmentation score the sum of the score of all of the found two-grams occurrences, otherwise, if no occurrences are found, we score the segmentation with -1.

Two-grams. This heuristic explores all the possible combination of letters that can be created with the hashtag. For example, the hashtag *goodtome* is turned into "*goodtom e*", "*goodto me*", "*good tome*", "*goo dtome*", "*g oo dtome*", etc. The number of possible segmentations are $2^n - 1$, where n is equal to the number of letters of the hashtag. In order to reduce the number of possibilities, we take the assumption that all the segmentation that has more than $n/2 - 1$ are not a proposer segmentation. This is an empiric statement done across thousands of different hashtags retrieved from Twitter. Therefore, segmentations such as: "*g o o d t o m e*" or "*go o d t o me*" will be pruned and not evaluated. The scoring logic and the output are the same than for the camel case heuristic.

[1] http://www.ngrams.info/

Maximum matching. The segmentations explored by this heuristic are the same than for the two-grams heuristic, only the scoring part is changing. The idea of the scoring function is to maximize the average word length. Note that average word length is inversely proportional to the number of words in a segmentation. The scoring function of a segmentation is:

$$score(s) = \sqrt[i]{\sum_{k=1}^{i} len(w_k)^2}$$

. This formula is the one proposed in [142]. Each segmentation will be scored only if they are found into one of the dictionaries, otherwise -1 is retrieved.

The final scoring takes as input the best segmentation retrieved by the heuristics. We have put one exception in the final scoring, if the first heuristic (entire hashtag checking) gives a result, this segmentation will be automatically accepted to be the final segmentation. Otherwise, we have to harmonize the scores given by the other heuristics, and we use the formula proposed for the maximum matching to re-score each segmentation, and the segmentation that have the highest score is chosen as final segmentation. We can also define a threshold, in order to know if it might be a good or bad segmentation, and according to our test, the best threshold is 1. If no score is above the threshold, no segmentation is finally retrieved.

3.1.2 User Mention Dereferencing

On several social medias, users are mentioned with names starting by the character and are often representing user nicknames. To have the real name of the user, we have to dereference the user mention against the API from where the textual content comes from. For example, the user mention *julienplu*, after a call to the Twitter API, is replaced by the mention *Julien Plu*.

3.2 Aggregation of Multiple Named Entity Extractors

The main intuition behind our recognition process is to associate multiple type of extractors altogether and to get the entities from each of them in order to get a recall as high as possible. Currently, we make use of four different type of extractors: *1)* dictionary tagger, *2)* Part-Of-Speech Tagger, *3)* Named Entity Recognition Tagger and *4)* Coreference Tagger. If two or more of these extractors are activated, they run in parallel.

3.2.1 Dictionary Tagger

This approach can be associated to a rules-based named entity recognition approach, and needs to have a file in order to know which pattern to recognize. The simplest rule file has two tab-separated columns on a line where the first column has the text to match and the second column has the entity category to assign. We can see the Listing 3.1 as a very simple example. In a little more detail now, the first column is not just a string, but might also be a sequence of one or more space-separated patterns. The approach divides text into tokens and each whitespace-separated pattern in the first column has to match against successive tokens in the text. Each pattern is a standard regular expression. If the regular expression match a sequence of tokens, the tokens will be relabeled as the category in the second column. Next, we might also want to match a variety of universities, then rather than writing many strings, we can compress them into one regular expression like in Listing 3.2, or like in Listing 3.3 if we already know the ones we want to recognize.

```
University of Montpellier        ORGANIZATION
University of Nice-Sophia-Antipolis      ORGANIZATION
```

Listing 3.1 – Example to recognize a couple of universities

```
University of [A-Za-z-]*          ORGANIZATION
```

Listing 3.2 – Example to recognize a multiple universities that share the same pattern

```
University of (Montpellier|Nice-Sophia-Antipolis)          ORGANIZATION
```

Listing 3.3 – Example to recognize two universities that share the same pattern

This regular expression approach allows to save a significant amount of time by creating smaller dictionaries with this flexible expressiveness than writing every mention we want to extract one by one. Nevertheless, creating a dictionary still requires a non negligible time, for this reason we have developed an approach in order to automatically create a dictionary based on a knowledge base, and finding entities that share the same spelling and merge them into a same regular expressions. This approach relies on a query where we can specify the type of the entities we want to create a dictionary with. More details on this query system is available in Chapter 5. For example, for the entities of type *Person* we merge all the entities that share the same family name into one single regular expression pattern that is ([A-Z][a-zA-Z]*[\s]XXXX)+ where *XXXX* corresponds to the family name.

Using a dictionary as extractor, gives the possibility to be very flexible in terms of entities to extract with their corresponding type, allows to handle multiple languages.

However, it does not allows to recognize entities that are not in it, do not respect the regular expressions, or respect the regular expression but they are not entities. To this end, we use other extractors in order to solve this problem.

3.2.2 Part-Of-Speech Tagger

In the Latin languages, words can be considered as the smallest elements that have distinctive meanings in written text, otherwise for speaking it is phonemes. Based on their use and functions, words are categorized into several types or parts-of-speech. While there are above 30 parts-of-speech in the grammar of Latin languages, only 8 are mostly used in texts: noun, pronoun, verb, adverb, adjective, conjunction, preposition, and interjection. This list is taken when you do an average of the parts-of-speech of the words contained in a book. For the purpose of extracting entities, we are mostly interested by two part-of-speech: the nouns (and more specifically proper-nouns) because this part-of-speech refers to words that are used to name persons, things, animals, places, organizations, or events; and the numbers in order to help to extract simple numbers, money expressions or dates. In this thesis, we propose a neural network for sequences tagging problem inspired from [99]. The high level system architecture can be summarized with the following Figure 3.1.

Our approach is similar with some variations. In order to fully understand the variations we first explain the network itself and then the variations. The network is composed of three layers:

1. first layer: This layer can be seen as a word representation. For each word of the sentence we merge its corresponding word embedding and the representation of its characters. The characters representation is itself a sub-neural network of three layers:
   (a) first sub-layer: each character is represented as an embedding. These embeddings are randomly generated, using the LeCun uniform method [90].
   (b) second sub-layer: three by three character embeddings are given to a convolution layer
   (c) third sub-layer: a maximum pooling is applied at the end of the convolution layer. The output of this maximum pooling layer is what we call the characters representation.
2. second layer: the word representation is given to a bidirectional long short term memory layer. This layer is composed of a forward and a reverse long short term memory. This layer aims to represent the sequence of words in sentence.
3. third layer: the last layer corresponds to the classification part of the network. As classification method, conditional random field has been shown to be the

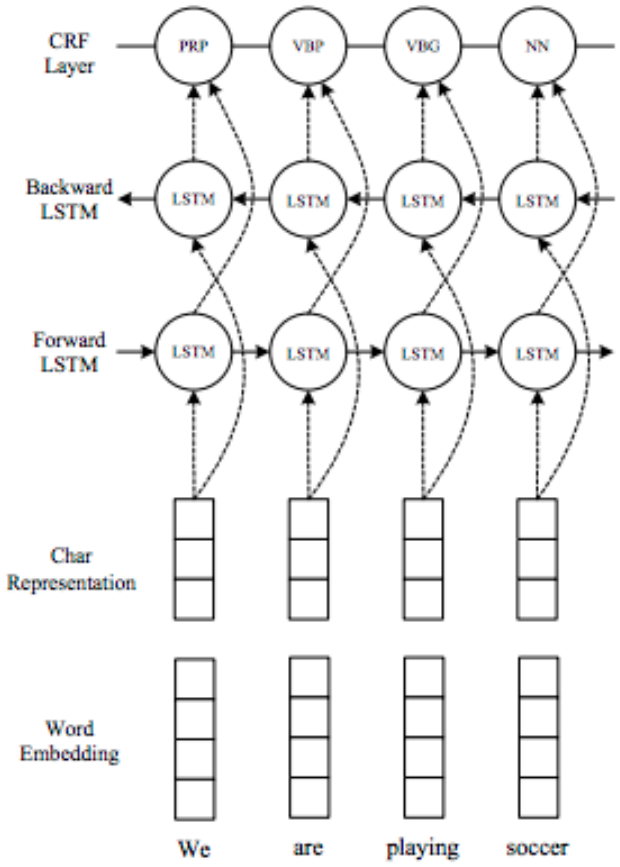

Figure 3.1 – Neural network architecture diagram taken from [99]

most efficient for sequences tagging problems (see Chapter 2, Section 2.2), for this reason we use this classification approach as last layer.

Once we have this network, we can modify some parts. The part we have modified is the conditional random field layer. In a conditional random field classifier, the usual inference approach used a called Viterbi, a dynamic programming algorithm, to infer the most likely hidden state sequence given the input and the model. As shown in [54], they proved that using another inference approach, the authors show that using a Monte Carlo with a Gibbs sampling approach is more efficient than the standard Viterbi algorithm. Following this assumption, we have reimplemented their approach and replaced the usual Viterbi algorithm in our conditional ranfom field layer by this new one. We have also added a new feature vector based on casing information about the words (isLower, isUpper, isNumber, hasNumber, isCapitalized, isDate, isUserMention, isHashtag), create a word embedding vector for special tokens that are not in the models (dates, numbers, user mentions and hashtag) created randomly in order to take also into account some additional features proposed in [54]. The last change is to add an extra bidirectional long short term memory layer.

3.2.3 Named Entity Recognition Tagger

As detailed in Chapter 2, Section 2.2, a named entity recognition tagger aims to recognize entities that belongs to given types such as *PERSON*, *LOCATION*, or *ORGANIZATION*. We use the same approach detailed for the part-of-speech tagger also to make a named entity recognition tagger, because the approach can be extended to any sequence tagging problem, and only needs to be adapted to the input training data. This named entity recognition tagger outperforms the current state-of-the-art approaches with a F1 score of 92.18% over the CoNLL2003 benchmark dataset, previous best score was 91.69% in [171]. More details on this evaluation in Chapter 6. Contrarily to state-of-the-art approaches, we use a model combination method that aims to jointly make use of different named entity recognition models into one single

named entity recognition tagger as described in the Algorithm 1.

Algorithm 1: Algorithm used to combine multiple named entity recognition models

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

```
1  begin
2      finalTuples ← EmptyList();
3      foreach model in M do
           /* tmpTuples contains the tuples {token, label} got from model
              */
4          tmpTuples ←apply model over Txt;
5          foreach {token, label} in tmpTuples do
6              if token from {token, label} not in finalTuples then
7                  add {token, label} in finalTuples;
8              end
9          end
10     end
11 end
```

We explain the logic of this named entity recognition model combination using the following example: *William Bradley Pitt (born December 18, 1963) is an American actor and producer.*. We assume to have trained two models with a different training dataset for each. The first model know how to recognize the types *PERSON* and *MISC*, while the second model know how to recognize the types *PERSON* and *DATE*. If we only apply the first model, we get the following result: *William Bradley Pitt* as *PERSON*, and *American* as *MISC*. If we only apply the second 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 labeling problem. Let's imagine two models trained on two different datasets, where in one dataset a location is labeled as *LOC* but in the other dataset, it is labeled as *Place*. Therefore, if we apply a combination of these two models, the results will contain labeled 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 labeling problem, we propose to do not mix models that have been trained with different labels to represent

the same entity type but, instead, create two instances of a named entity recognition tagger where each one has a combination of compatible models.

3.2.4 Coreference Tagger

The Coreference Tagger aims to extract all the coreferences that might be related to an entity inside a same document but not across documents. In this thesis we have developed a co-reference approach that succeed to leverage a deep neural network with semantic knowledge, called Sanaphor++. We detail the different steps of Sanaphor++ and, in particular, how we extend Stanford deep-coref and SANAPHOR into one single method. The new Sanaphor++ pipeline is depicted in Figure 3.4 together with the previous SANAPHOR pipeline 3.2, and Stanford deep-coref 3.3 in order to show their differences. In the three figures, the ellipses represent an input/output and the rectangles represent a process. The first step is to create a new logic that extends the SANAPHOR one to take into account ambiguous and novel or emergent entities. The second step is to extend Stanford deep-coref to handle the new logic represented with the first step.

3.2.4.1 Ambiguous entities

Sanaphor++ is able to handle coreference for ambiguous entities. In SANAPHOR, an ambiguous entity is a mention for which the basic entity linking method finds multiple candidates, such as *Paris* that might refers to: *Paris in France*, *Paris in Texas*, *Paris Hilton*, *Paris the movie* or even *Paris the band*. To be able to handle those cases, we rely on the linking part of the work of this thesis, detailed in the next chapter.

In a nutshell, we devise a three-step approach: *i)* mention extraction, which detects mentions in text using multiple NER extractors and that are likely to denote entities; *ii)* overlap resolution, that takes the output of the mention extraction step and decides on a single output with no overlaps among the extracted mentions; and *iii)* linking, that generates a list of potential entities, called entity candidates, that might disambiguate the extracted mentions, and then selecting the best one. In the new Sanaphor++ logic, only the third step of ADEL is being used since mentions from the text are already given (Section 3.2.4.3 details how we get those mentions). Finally, the DBpedia type of the retrieved entity is mapped against its YAGO types. It is important to notice that, if the original linking approach and ADEL do not find any entity, it probably means that the mention is either a novel entity or a referent mention such as *he* or *him*. The second case is normal, while the first one has to be tackle, because even if the entity has no counter-part in DBpedia, it does not mean we cannot get its type.

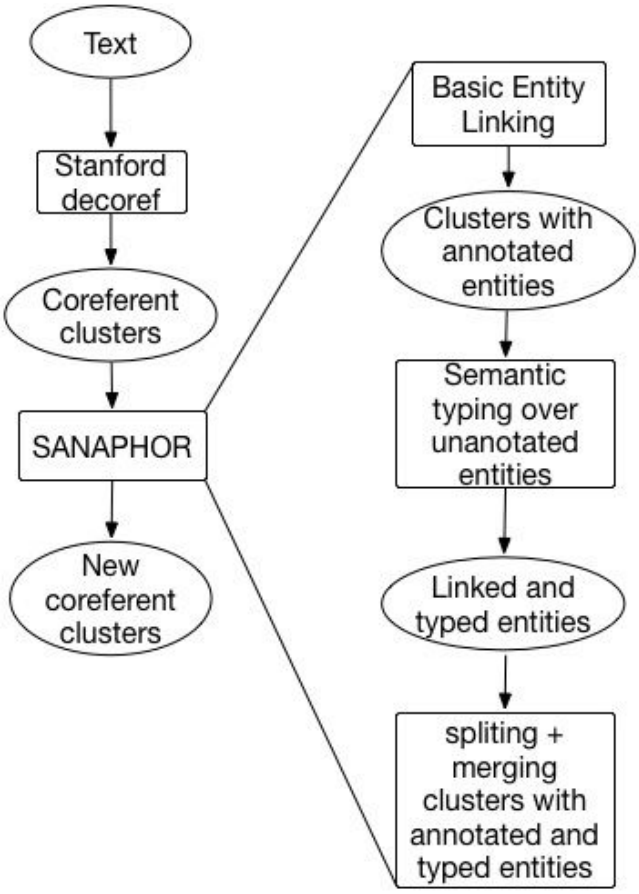

Figure 3.2 – The SANAPHOR process

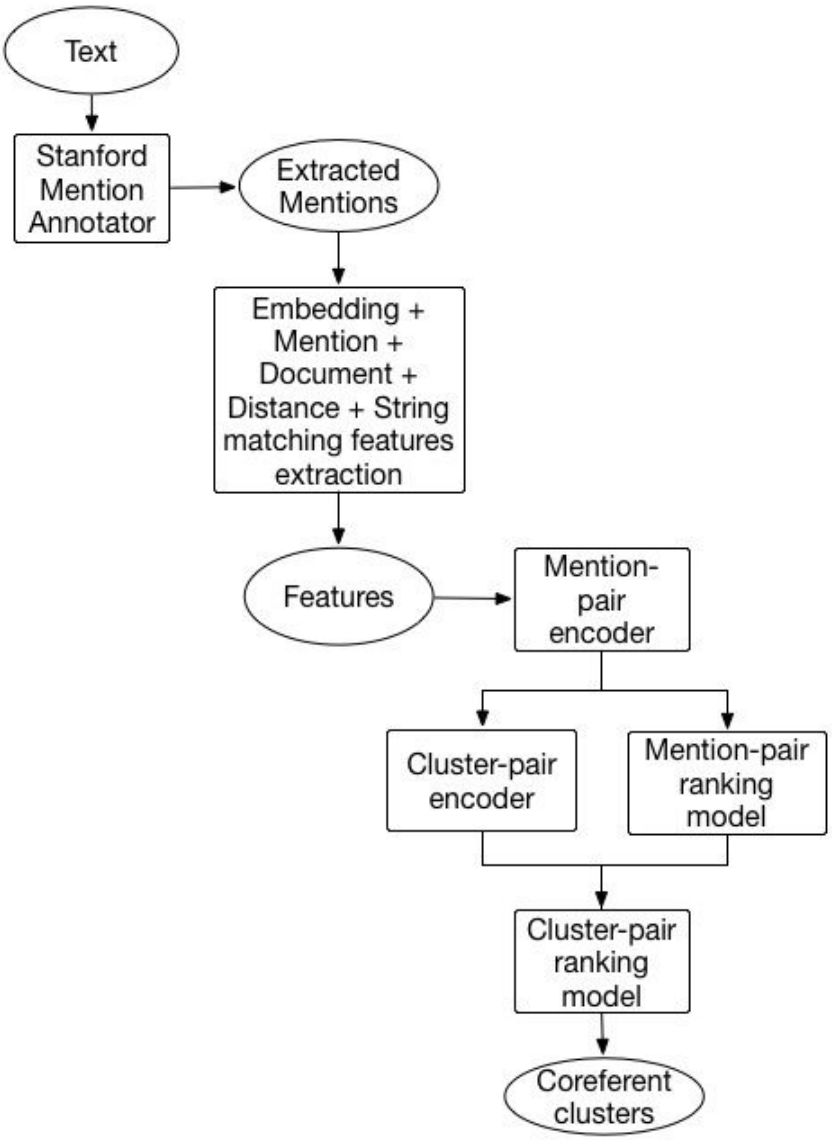

Figure 3.3 – The Stanford deep-coref process

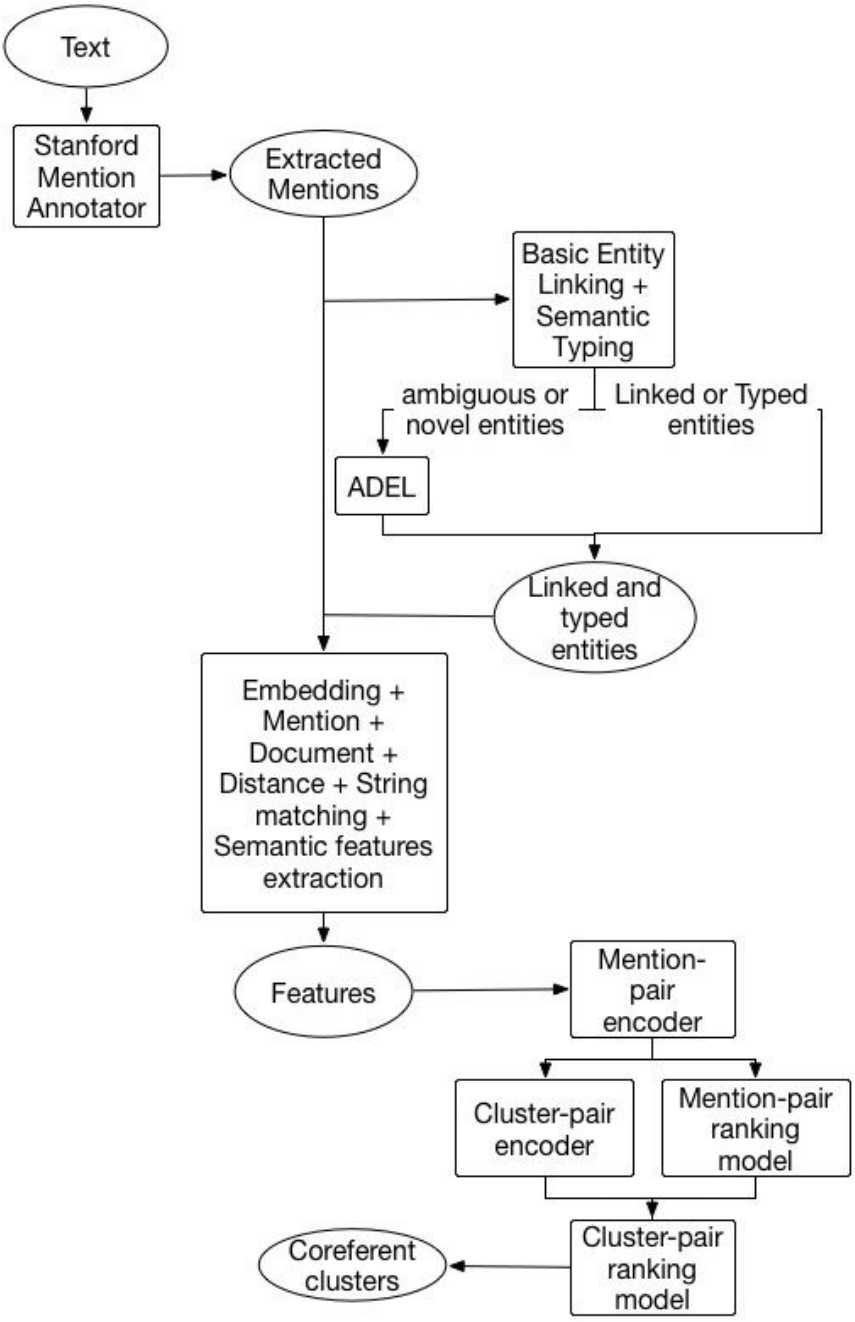

Figure 3.4 – The Sanaphor++ process

3.2.4.2 Novel Entities

The second stage is to handle novel or emergent entities, that is, entities that do not exist (yet) in the knowledge base being used, in our experiment, DBpedia. The case of a novel entity occurs when both the SANAPHOR original entity linking method and ADEL give an empty result. In that case, we rely on the Stanford NER annotator that is integrated into the mention extraction process detailed in 3.2.4.3. In that process, mentions come with their NER type attached. The used model to attach these types has been trained with the CoNLL2012 Shared Task dataset that contains the types of mentions defined as entities. Finally, like for the linking that maps DBpedia types to YAGO types, we have defined a mapping that links the NER types to YAGO types. Nevertheless, these NER types are high level types such as *PERSON* or *ORGANIZATION* and then do not give a lot of details on the semantic behind. Despite this lack of information on the semantic of these entities, it is still worth to handle because it helps the system to split clusters where cases that mix, for example, *PERSON* and *ORGANIZATION*. This is often the case when a company is named according to the family name of the owner. Once these two new cases are handled, we need to find a way to modify Stanford deep-coref in order to make it able to take into account this extended logic: handling ambiguous and novel entities.

3.2.4.3 New Mention-pair Ranking Model

The final step is to make Stanford deep-coref able to handle the new logic seen before, to have the final Sanaphor++ pipeline as detailed in Figure 3.4. Thus, after a thorough study of Stanford deep-coref, we have found that it shares two common parts with SANAPHOR: *i)* the cluster merging part and the cluster-ranking model, and *ii)* the cluster splitting part and the mention-ranking model.

The cluster-ranking model has a very complex structure and modifying it to take into account the merging features was impractical, such that we decided to leave this as future work. Therefore, we decided to create a new three-dimensional vector, one dimension for each semantic feature: *i)* if the entities of the two mentions are the same, *ii)* if the type of the two mentions are the same, and *iii)* if the type of one mention is included into the hierarchy of the other one. They are then concatenated with the original vector in the input layer of the mention-pair encoder and then to the mention-pair ranking model.

The extraction of the new features must be implemented in Stanford deep-coref. To do so, we have to put the new process to extract the semantic features directly into the one that take care of the Stanford deep-coref features. The CoNLL2012 Shared Task dataset is composed of three datasets: train, dev and test. The extraction of the mentions for the train and the set {dev, test} datasets is done differently. About the training dataset, the mentions are directly extracted from the annotated gold

standard, whereas for the set {dev, test} datasets, the mentions are extracted using the Stanford mention annotator, where its goal is basically to extract mentions from the text. It means that the annotations in the {dev, test} datasets are not used at all, because they are the evaluation datasets. Once the mentions have been extracted, either via the dataset or via the Stanford mention annotator, they are all put into different feature computation process, one for each kind of feature: embedding, mention, document, distance, string matching and now the semantic feature, i.e. the 3-dimensional vector. In order for the neural network to take into account these new features, we must modify its training objective and more precisely the mistake-specific cost function. The new mistake-specific cost function is represented in Equation 3.1.

$$\Delta(a, m_i) = \begin{cases} \alpha_{FN} & \text{if} \quad a = NA \wedge \Gamma(m_i) \neq \{NA\} \\ \alpha_{FA} & \text{if} \quad a \neq NA \wedge \Gamma(m_i) = \{NA\} \\ \alpha_{WL} & \text{if} \quad a \neq NA \wedge a \notin \Gamma(m_i) \\ 0 & \text{if} \quad a \in \Gamma(m_i) \vee e_a \in \Omega(m_i) \vee t_a \in \Phi(m_i) \vee T(e_a) \in \Phi(m_i) \end{cases} \tag{3.1}$$

Where NA indicates an empty antecedent; $\Gamma(m_i)$ denotes the set of true antecedents of m_i (i.e. mentions preceding m_i that are coreferent with it or $\{NA\}$ if m_i has no antecedent); a is a possible antecedent of m_i; e_a denotes the entity of a; $\Omega(m_i)$ denotes the set of the entities of the true antecedents of m_i; t_a denotes the type of a; $\Phi(m_i)$ denotes the set of the types of the true antecedents of m_i (i.e. this set includes also all the types that belongs to the hierarchy of a type); $T(e_a)$ denotes the type of the entity of a. The goal of this new mistake-specific cost function, is to make understand to the network if two mentions are likely to be compatible together and being a pair or not. We detail the clauses of this new function:

1. The first clause stands for the *false new* antecedents. It means that if the antecedent is an empty antecedent and if the set of true antecedents is not equal to empty, then this antecedent is likely to be a wrong pair;

2. The second clause stands for the *false anaphoric* antecedents. It means that if an antecedent is not an empty antecedent and if the set of true antecedents is equal to empty, then this antecedent is likely to be a wrong pair;

3. The third clause stands for the *wrong link* antecedents. It means that if an antecedent is not empty and this antecedent does not belong to the set of true antecedents, then this antecedent is likely to be a wrong pair;

4. The fourth clause stands for a correct coreferent decision. It means that if an antecedent is in the set of the true antecedents, or if the entity of the antecedent

is in the set of the true entities, or if the type of the antecedent is in the set of the true types, or if one type belonging to the hierarchy of type of the antecedent is in the set of the true types, then this antecedent is likely to be a good pair.

The error penalties α_{FN}, α_{FA} and α_{WL} are hyperparameters that must be defined at the beginning of the training. The evaluation of this new co-reference approach outperform the current state-of-the-art approaches with an averaged F1 score of 61.96% over the CoNLL2012 benchmark dataset, previous best performance was 60.83% in [30]. More details on this evaluation in Chapter 6.

3.3 Overlap Resolution

This feature aims to resolve the overlaps among the outputs of the extractors and to give one output without overlaps. The logic behind is organized as follows: given two overlapping mentions, *e.g.* `States of America` from the named entity recognition tagger and `United States` from the part-of-speech tagger, we only take the union of the two phrases. We obtain the mention `United States of America` and the type provided by the named entity recognition tagger is selected. The overlaps in terms of text are easy to resolve, because we take the assumption to always take the largest entity, 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. In order to solve this typing problem, we have developed a two step approach: types mapping and majority voting.

3.3.1 Types Mapping

The first step is what we call types mapping. The goal here is to map types from different sources that represents the same real world entities but with a different label. To realize this task we have decided to go for a solution that map one or multiple types from one source to one type in another source, and make a *n to 1* relationship between types. As example we propose in Listing 3.4 a mapping between some types from the DUL ontology[2] and the CoNLL2003 types [152]. More details on the used format to represent a mapping is available in Chapter 5.

```
dul:Place              LOCATION
dul:Person             PERSON
dul:Organization       ORGANIZATION
dul:Role,dul:Event     MISC
```

Listing 3.4 – Types mapping between the CoNLL2003 types and the DUL ontology

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

With this types mapping feature we are now able to harmonize the used types from any source to the types from any other source. In the work of this thesis, this is very useful for mostly two reasons:

1. This is very useful when we want to apply a named entity recognition model that has been trained with a specific dataset with specific types, to another kind of dataset that uses different labels but contains the same types of real world entities. No need to retrain a model.

2. Our recognition process makes use of multiple different extractors, and the need to harmonize the types among these extractors is very important to help to solve the typing issue raised by the overlap resolution, as it allows us to apply machine learning algorithms such as ensemble in order to select the most representative type across all the types proposed by these extractors for the same entity.

3.3.2 Majority Voting

Now that we have harmonized the types across all the extractors, we are able to apply an ensemble machine learning algorithm in order to resolve the typing issue raised by the overlap resolution process. We have decided to apply a majority voting which is an ensemble machine learning algorithm that aims to select the type that has been proposed the most across all the extractors. Once the types have been harmonized the logic becomes as the following:

1. aggregating the harmonized types proposed by each extractor

2. count the number of time a same type is proposed

3. select the one that has the majority

While this approach works for most of the cases, sometimes it is not enough and we might encounter a draw score between at least two different types. To solve this draw score issue, we have decided to propose a way to rank the extractors by attributing a confidence order, because we can think that such or such extractor works better than another one. As each used extractor is ranked accordingly to its confidence, in case of a draw score among the extractors, we take the type proposed by the highest ranked extractor. Finally, in order to have a better idea of how this overlap resolution works, we can apply it over a simple example. We assume to have the following three extractors with a specific setting:

1. A named entity recognition extractor that uses a model trained with the CoNLL2003 types

2. A named entity recognition extractor that uses a model trained with the DUL ontology types

3. A dictionary extractor that uses a dictionary generated from DBpedia entities and providing DBpedia types

4. A coreference extractor

We assume to have the piece of text: *J. K. Rowling is a famous author. She has written the series of books: Harry Potter.* to process. The confidence order of the extractors is: 3-1-2. Which means that the dictionary extractor is the one we trust the most. The coreference extractor do not give any entity type, so it does not belong to the confidence list. The output of each extractors is:

1. *J. K. Rowling* (*PERSON*) *is a famous author. She has written the series of books:* *Harry Potter.* (*PERSON*)
2. *J. K.* *Rowling* (*dul:Place*) *is a famous author. She has written the series of books: Harry Potter.*
3. *J. K. Rowling* (*dbo:Person*) *is a famous author. She has written the series of books:* *Harry Potter* (*dbo:TelevisionShow*) .
4. *J. K. Rowling is a famous author.* *She* *has written the series of books: Harry Potter.*

We can see that each extractor has extracted similar entities, the entity *Rowling* from the extractor 2, is a substring of *J. K. Rowling* from the three others extractors, then we select the largest of the three versions of this entity represented by *J. K. Rowling*, same thing for *Harry Potter*. As the coreference extractor has also extracted the mention *She*, it will become an entity with the exact same type than emphJ. K. Rowling. Now that the mentions have been reconciled, we have to reconcile the types. We assume that the targeted set of types is CoNLL2003, then in the mapping file, the types *dul:Person* and *dbo:Person* are mapped to *PERSON*, the type *dbo:TelevisionShow* is mapped to *MISC*, and the type *dul:Place* is mapped to *LOCATION*. For the entity *J. K. Rowling* we have two different types associated *PERSON* and *LOCATION*, but *PERSON* has been raised twice, then *PERSON* is kept as final type. The entity *Harry Potter* has been detected by two extractors and both have tagged the entity with a different type. The majority voting cannot take a decision, so we use the confidence order of the extractors making the type from the dictionary, *MISC*, the definitive type. Finally, the final result can be summarized as:

J. K. Rowling (**PERSON**) **is a famous author.** **She** (**PERSON**) **has written the series of books:** **Harry Potter.** (**MISC**)

This entity recognition process allows us to handle a large set of languages, document types and type of real world entities 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 sub-task for the entity linking task. The evaluation of our new recognition approach proposes a recall always above 90% at extraction level, and also with scores that most of the time outperform the state-of-the-art systems at recognition level. More details on the evaluation over multiple benchmark datasets in Chapter 6.

Science is organized knowledge. Wisdom is organized life.

Immanuel Kant

4 Entity Linking

In this chapter, we describe our method for indexing a knowledge base and for optimizing the search of candidate entities, as well as the linking and NIL clustering approaches we have developed during this thesis.

4.1 Candidate Generation

This section is focusing on the indexing and the search optimization of a knowledge base. As detailed in Chapter 2, Section 2.1.3, there are multiple differences across the existing knowledge bases that make the indexing and the search process very complex. The following approach can be applied to any knowledge base that uses linked data, or not. We will detail what are the minimum requirements that a knowledge base should comply with, but also the extra other data that they might contain. We will first detail the approach with linked data knowledge bases: DBpedia (encyclopedic content) and Musicbrainz (vertical content), and then over a non linked data knowledge base: JeuxDeMots (lexical network). The indexing process propose to always give the same output, does not matter what is the knowledge base we want to index (linked data or not) in order to harmonize the input of the index optimization process.

4.1.1 Linked Data Indexing: Use Cases with DBpedia and Musicbrainz

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 require-

ments 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 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[1] 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 4.1 for DBpedia or Query 4.3 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 4.2 and the Query 4.4 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 4.1 – SPARQL query that filters the entities we would like to index

The result of this second query is then used to obtain a file that will be the final index of the knowledge base. As main indexing tool we use Elasticsearch [2] or Lucene [3].

4.1.2 Non Linked Data Indexing: Use Cases With The Lexical Network JeuxDeMots

The case of JeuxDeMots is special because the available dumps[4] are written in a specific format created by the authors and cannot be directly queried without a processing to turn them into a format that can be load into a queryable data storage. To this end, we have developed a script to pre-process a dump of JeuxDeMots, and to turn it into the exact same output format and structure used with the Linked Data process. In this thesis, we have developed a linking approach based on graph-theory

[1] http://jung.sourceforge.net/
[2] https://www.elastic.co
[3] http://lucene.apache.org/
[4] http://www.jeuxdemots.org/JDM-LEXICALNET-FR/?C=M;O=D

```
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 4.2 – 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 {
        ?s foaf:name ?label .
    } UNION {
        ?s dc:title ?label .
    }
}
```

Listing 4.3 – 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 4.4 – 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

theorems for disambiguating entities (see Section 4.2.2) that uses the graph structure of JeuxDeMots. Consequently, our indexing tool will be specifically created for graph manipulations, Neo4J [5].

4.1.3 Index Optimization

Once we have the 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 by using datasets that have been annotated with the proper targeted knowledge base.

The DBpedia index of the 2015-10 version has 4726950 rows (entities) and 281 columns (entity properties). Given the benchmark datasets: OKE2015 (Task 1), OKE2016 (Task 1), OKE2017 (Task 1 and 2), NEEL2014, NEEL2015 and NEEL2016, we parse their content in order to extract a list of distinct pairs (mention, link). The Musicbrainz index of the 2016-12 version has 18306792 rows (entities) and 7 columns (datatype properties). The benchmark dataset that makes use of Musicbrainz is OKE2017 (Task 3). 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 7 queries for Musicbrainz), 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. With DBpedia, at the end, we end up with a file that looks like the excerpt depicted in the Listing 4.5.

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 4.5, 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`

[5] https://neo4j.com/

```
{
  "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 4.5 – Excerpt of the result file for the optimization process

6. `foaf_name`
7. `dbo_wikiPageDisambiguates`
8. `dbo_longName`
9. `dbo_slogan`

Fortunately, for Musicbrainz, with the union we directly get the optimized list of properties reducing the list from 7 to 4 properties to query, being:

1. `foaf_name`
2. `dc_title`
3. `skos_altLabel`
4. `terms_sortLabel`

Nevertheless, 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.

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_pseudonym`
4. `rdfs_label`

If run over Musicbrainz, the optimization algorithm gives the exact same list of properties than the one we got with the union seen above. This optimization reduces

the time of the query to generate the entity candidates by a factor 4. 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. This indexing and optimizing process allows to reach every time over 92% of recall for the candidate generation process with a computing time always below the second, for each benchmark dataset we have used to evaluate this approach. More details on the evaluation in Chapter 6.

4.2 Linking Methods

This section lists all the linking approaches we have developed during this thesis.

4.2.1 Text Similarity Weighted With PageRank

This approach is an empirically assessed function represented by Equation 4.1 that ranks all possible candidates given by the candidate generation.

$$r(l) \;=\; (a \cdot L(m, title) + b \cdot max(L(m, R)) + c \cdot max(L(m, D))) \cdot PR(l) \quad (4.1)$$

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.

This approach has the advantage to be fully compliant with each challenge raised in the Chapter 1. The results of this approach proposes a broad set of results and is highly dataset dependent. More details in Chapter 6.

4.2.2 JeuxDeLiens: Path Finding In A Semantic-lexical Network

Our entity linking approach is designed in two steps: *i)* word embeddings, and *ii)* path-based similarity. As a running example, we will use the following three sentences:

- Paris compte au 1er janvier 2013 plus de 2,2 millions d'habitants.[6] *Paris* stands for *the French capital.*

[6] As of 1 January 2013, Paris has more than 2.2 million inhabitants

- En matière de commerce, Paris a clairement affiché sa volonté.[7] *Paris* stands for *the French government.*
- Paris fait ses premiÃÍres apparitions comme mannequin dans plusieurs évènements de charité.[8] *Paris* refers to *Paris Hilton.*

Entities are annotated beforehand in the text between double square brackets (e.g. [[Paris]]). We perform a preprocessing that consists in the following three steps:

1. tokenizing the input text by respecting the annotated entities between double square brackets;
2. removing the stopwords.
3. once the text is properly tokenized, we link each word to a node in the JeuxDeMots network (if it exists, otherwise the word is removed), and we generate all the possible entity candidates for the entities. In the three sentences, the candidate entities for Paris are: *i)* Paris Hilton, *ii)* Paris, capitale[9], *iii)* Paris, gouvernement français[10], *iv)* Paris, prénom[11], and *v)* Paris, nom[12]. We also map the words that are multitokens expression such as *table de chevet*[13].

After this preprocessing, we have a list of words that represent the context of the document and where each of them has its node mapped in the JeuxDeMots network. The next step is to select the best entity candidate for each mention in the text.

4.2.2.1 Word Embeddings

In order to properly disambiguate the entities, we use word2vec [106] as word embeddings method with a model trained over the frWac corpus [8]. Word2vec learns a numerical embeddings space of a vocabulary which would result in similar words ending up close to each other. For example, *cat* will be placed close to *kitten*, where *dog* will be placed further than *kitten* and *iphone* even further. By learning such numerical representation of the words of the vocabulary, one can do various vectorized operations producing interesting results. For example, operation *kitten* - *cat* + *dog* should result in an embedded vector very close to *puppy*. This method has been chosen since a surface form might have more than one meaning as shown in our running example. A simple string comparison between surface forms, e.g. using the Levenshtein distance, is not efficient since the two contexts of the entity and into

[7] In terms of trade, Paris has clearly stated its will.
[8] Paris made her first appearances as a model in several charity events.
[9] capital city
[10] French government
[11] firstname
[12] familyname
[13] nightstand

JeuxDeMots shall be compared. In JeuxDeMots, the context of an entity is represented by his gloss (e.g. gouvernement français, prénom, capitale, etc.). For the first sentence, instead of comparing the surface form *Paris* with the words of the context, we use *gouvernement français* and the others gloss such as in Equation 4.2. The set W represents the following context: [compte, janvier, millions, habitants].

$$\begin{aligned} &w2v([gouvernement, francais], W) \\ &w2v([capitale], W) \\ &w2v([Paris, Hilton], W) \\ &w2v([prénom], W) \\ &w2v([nom], W) \end{aligned} \tag{4.2}$$

The entity candidate that got the best score is taken as the proper linked entity. If an entity does not have any candidate, it is what we call a novel entity, and it will be linked to NIL. We also propose to cluster the novel (*NIL*) entities that may identify the same real-world thing. We attach the same *NIL* value within and across documents. For example, if we take two different documents that share the same emergent entity, this entity will be linked to the same *NIL* value. We can then imagine different *NIL* values, such as *NIL_1*, *NIL_2*. We perform a string matching over the surface form between each novel entities that have been linked to *NIL* (or between each token if it is a multiple token mention). Concerning our running example, with this approach, all the *Paris* entities have been properly linked to their corresponding meaning. We use a path-based similarity method to find the type of the entity.

4.2.2.2 Path-Based Similarity

At this stage, we need to assign a type to an entity. From the JeuxDeMots network, we can gather the context words W, the entity e and the set of possible classes C. Although, we got W during the mapping step and e with word2vec, the class nodes are simply hand-picked nodes describing the class. Here, we only use *lieu*[14] for LOC, *organisation* for ORG and *personne*[15] for PER as classes.

The logic is to find the best path in the JeuxDeMots network from W to each element of C passing by e. An example is represented in Figure 4.1 for the first sentence. In case e is a novel entity, we try to directly find the best path from W to each element of C. The JeuxDeMots network has two characteristics that we have to take into account: *1)* a direct link from the entity to the class may not exist and

[14] place
[15] person

could be indirect, for example, *Paris (capitale)* $\xrightarrow{\text{is-a}}$ métropole[16] $\xrightarrow{\text{is-a}}$ lieu, and *2)* the JeuxDeMots taxonomy has no constraints, can have multiple words to express the same meaning of a class and can have loops.

In order to choose the class of the entities, we propose a path-based similarity measure inspired from LP-index [94]. First, the score of a path μ is defined as:

$$score(\mu) = \delta^{|\mu|-1}. \sum_{\forall e=(u,v)\in\mu} w_e.|R^+_{T_e}(u)|^{-1} \tag{4.3}$$

With $\delta \in [0,1]$ a length malus, $|\mu|$ the length of the path, $R^+_{T_e}(u)$ the set of outgoing relations of u with the same type T_e as e and w_e its weight.

The similarity between two nodes u and v can then be expressed as:

$$sim(u,v) = \frac{1}{|M_{u,v}|}. \sum_{\forall \mu\in M_{u,v}} score(\mu) \tag{4.4}$$

With $M_{u,v}$ the set of paths between u and v. If the entity was found in the network we choose the class $c = \arg\max_{c\in C} sim_{ctx}(W,e,c)$ with:

$$sim_{ctx}(W,e,c) = sim_s(W,e) + sim(u,c) \tag{4.5}$$

And:

$$sim_s(W,e) = \frac{1}{|W|}. \sum_{\forall w\in W} sim(w,e) \tag{4.6}$$

If the network does not contain the entity, then we only use its context to try to choose the class $c = \arg\max_{c\in C} sim_s(W,c)$. In practice, we limit the length of the paths to 2 and only the most relevant types of relations are considered. This approach proposes good results over our French dataset, reaching 77.8% as F1 score over pretty much ambiguous news texts. More details on the evaluation in Chapter 6.

4.2.3 Graph Regularization

A majority of entity linking methods compute a similarity taking into account the document context and the knowledge base context but they rarely use these two different contexts as individual features in order to provide flexibility on how those contexts are computed which is required for adapting or even personalizing the entity linking process. For example, [76] use a measure called Mention-Entity similarity (ME-sim) which mixes keyphrases similarity (keyphrases from the processed document and keyphrases from the Wikipedia article of each entity candidate) and syntax

[16] metropolis

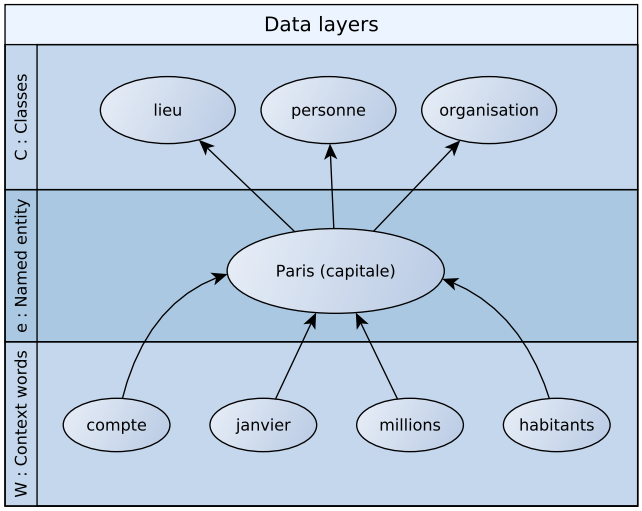

Figure 4.1 – The 3 data layers: classes, named entity and context words.

similarity (syntactic information such as whether the mention is a subject or an object). To the best of our knowledge, only [78] has developed a method that brings the possibility to use these contexts individually without making one dependent of the other. This work is a first step that we aim to generalize in order to propose a framework that will enable to truly experiment with different methods computing document and knowledge base contexts and their combination that could be used as input in a neural network.

In order to be able to better assess the importance of different textual context similarity measures, it is beneficial to opt for an approach that does not combine the two in a unique similarity measure, but use them separately in the disambiguation process. For this reason, we reproduce the architecture proposed in [78]. We start from the GraphRegu model firstly proposed in [77], which is a semi-supervised collective entity linking algorithm based on graph regularization. The nodes of the graph n_i are mention-candidate pairs $n_i = (m, c_i)$ and each node has a ranking r_i, which determines the probability of m being the referent entity of c_i. We adopt the heuristic proposed in [78] to select seed nodes and determine the initial ranking r_i^0, obtaining a completely unsupervised approach. Then, the regularization determines the final rankings r_i and, for each mention, we select the highest ranked node as the correct candidate entity. We assume that the extraction of the mentions has already been performed, taking the true mentions from the gold standard and focusing on the disambiguation. The approach can be summarized as represented in Figure 4.2 in the following order.

Candidate generation. We query an index to get the candidates and we select the 30 first in terms of prior probability $PR(m, c_i)$ as good candidates. The prior probability is computed as the frequency of the occurrences of a mention m as an

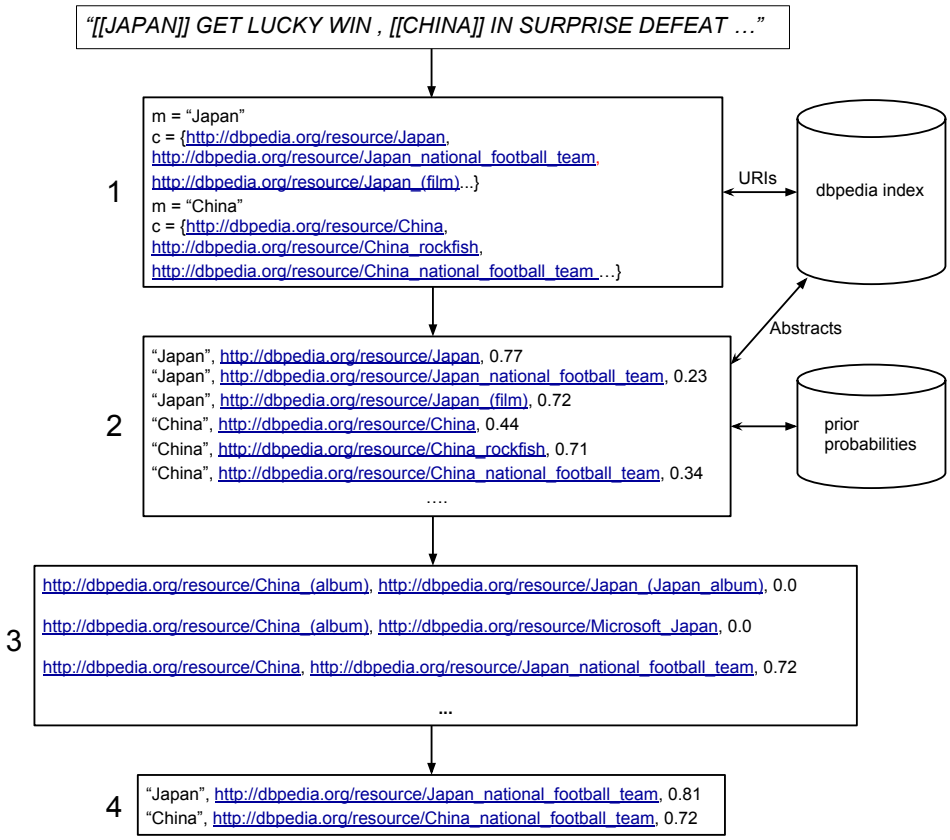

Figure 4.2 – Examples of the steps of the regularization process with a short document containing two mentions

anchor link pointing to the entity c_i in Wikipedia [76]. In order to be able to better evaluate the disambiguation performance mitigating the influence of the candidate generation strategy on the final results, which is not the core point of this work, we always make sure that the correct link is among the set of generated candidates and, if not, we retrieve it from the gold standard and add it, since the core of this experimentation is to evaluate the disambiguation and not the candidate generation process.

Seed selection and ranking initialization. The seed selection strategy aims to find nodes, i.e. mention-entity pairs, which we can consider as true links with a very high confidence in a pure unsupervised way. Following what has been proposed in [78], a node is a seed if it has $PR(m, c_i) > 0.95$ and if it is the first ranked both in terms of PR and textual context similarity $s(m, c_i)$. If a node is a seed, its initial ranking will be $r_i = 1.0$ and the other nodes corresponding to other candidate links will have ranking $r_j = 0.$, these rankings do not change. The initial ranking for all non-seed nodes is given by the average of prior popularity and document context similarity $r_i^0 = (PR(m, c_i) + s(m, c_i))/2$. We denote the set of all rankings with $R = R_1^l \cup R_{l+1}^n$, where R_1^l are the l seed nodes and the R_{l+1}^N are the other $N - l$ nodes.

Similarities. The similarity between nodes is used to create weighted edges, i.e. $W_{ij} = \sigma(n_i, n_j)$. $\sigma(n_i, n_j)$ can take into account different features, such as the similarity between the mentions and the semantic relatedness of the candidate entities. $\sigma(n_i, n_j)$ represents the knowledge base context measure. To prune the connectivity matrix, we adopt a k-nearest neighbor (k-NN) approach, meaning that if n_j is not among the k nearest neighbors of n_i with respect to the measure σ, we set $W_{ij} = 0.$. We set $k = 20$, similarly to what is done in [78].

Regularization. The regularization is based on the minimization of the loss function $F(R) = \mu \sum_{i=l+1}^{N} (r_i - r_i^0)^2 + \frac{1}{2} \sum_{ij} W_{ij} (r_i - r_j)^2$, which we compute using the closed form solution [77]. The work described in this section is still under evaluation and has been evaluated only over AIDA, where it achieves a F1 score of 53.3%. More details on this evaluation in Chapter 6 and further evaluations are considered as future work.

4.2.4 Heuristic based Candidate Re-Ranking

In this section, we apply an approach we call ReCon,[17] which acts as a contextual adaptation layer over ADEL. ReCon adapts the entity linking task to the textual content that is being analyzed, by leveraging genre and topic domain information about the text. In the current version, ReCon focuses on entity linking in news articles.

ReCon uses the entity linking annotations performed by ADEL as a starting point

[17] ReCon stands for Reranking with Context

and applies a set of heuristics which consider aspects of news articles that are not taken into account by ADEL. ReCon makes a reasonable assumption that the order of expressing information in news articles is somewhat systematic, i.e. the introductory sentences of an article tend to be written more explicitly, while the latter ones often contain abbreviations and incomplete mentions which are coreferential with previous explicit mentions. Furthermore, the title of an article is often intentionally ambiguous and uses figurative speech, thus requiring a reader to understand the running text of the article first. Finally, the entities mentioned in domain-specific news articles usually stem from that domain, e.g. in the basketball domain, it is customary to refer to the Chicago Bulls' mega-star Michael Jordan with *Jordan*.

ReCon applies four article-wide heuristics which consider these properties of the newswire text and, additionally, allows them to be tunable to the topic domain of an article. Our heuristics take the form of binary rules. We detail them as follows:

H1: Order of processing. The first Recon heuristic, H1, focuses on the relation between the title and the running text of a news article. Concretely, we observe that the title of a news article is often ambiguous and uses figurative speech, probably because of the intention of the writer to make the title attractive for a generic reader. On the other hand, the running text of a news article, especially the introductory sentences, are written in a clear, informative and contextually rich manner, allowing a shallowly informed reader to grasp as much as possible from the presented news. As a reaction to these observations, we introduce our heuristic H1 which concerns the order of information extraction in a news article. Following H1, we first process the running text of a news article and disambiguate the title of the article at the end.

H2: Co-reference. Our second heuristic, H2, follows a related rationale. We observe that surface forms that refer to an entity earlier in text are surrounded by richer verbal context, compared to the latter mentions of the same entity. This is understandable from a practical perspective: once the writer has clearly introduced an entity, he can use abbreviations or more ambiguous ways to refer to it further in the text. As a consequence, our second heuristic also operates on a cross-sentence level and takes advantage of the order in which sentences and entities appear in text. Concretely, ReCon's H2 tries to detect earlier mentions of the same entity, i.e. H2 checks if an entity mention is in a same co-referential chain as another entity mention which occurred earlier in the same article. This co-reference relation is established based on the textual similarity of these mentions, in particular it is based on the following three rules:

1. An entity mention M_j is co-referential with a previously occurred mention M_i if M_j is an abbreviation of M_i

2. An entity mention M_j is co-referential with a previously occurred mention M_i if M_j is identical to M_i

3. An entity mention M_j is co-referential with a previously occurred mention M_i if M_j is a sub-string of M_i and M_i was linked to an entity of type `Person`

H3: Domain relevance. If no pre-occurring co-referential entity is found by H2, we apply a topic modeling heuristic, H3, in which we exploit a contextual knowledge base about the topic of interest. We rely on this knowledge base to examine whether a mention has been frequently and dominantly associated with a certain entity disambiguation link within the specific topical domain. The frequency and dominance for each surface form are expressed through threshold values: a mention is resolved to an entity link according to H3 only if the mention appears sufficiently (with frequency above the threshold frequency) in the domain knowledge base, and a certain entity link is dominantly associated with that mention (the frequency distribution for that entity needs to be above the dominance threshold).

H4: Semantic typing. Whenever either of H2 or H3 proposes an entity disambiguation link for a mention, we apply a fourth heuristic, H4, to check whether its semantic type [18] corresponds to the textual context of that surface form. In practice, this is done by matching its semantic type from the knowledge base against the entity type as specified by the hybrid entity linker in the first step.[19] If this comparison is successful and the proposed entity corresponds to the type constraint set by the hybrid module, then the mention is linked to the proposed entity. Otherwise, we decline the proposed entity link. When none of the ReCon heuristics is able to resolve an entity mention, then we refrain to the linking provided in the first, hybrid step.

The ReCon approach improves the results by 9% the F1 score, when applied over the output of the *text similarity weighted with PageRank* approach to achieve 52.62%. More details on this evaluation in Chapter 6.

4.2.5 Deep-Similarity Network

DSRM [78] proposes an interesting approach to compute the similarity between two entities from a same knowledge base with a deep neural network. This deep neural network is composed of two fully connected hidden layer and takes as input four features:

1. the connected entities: a list of entities that are directly connected to a given entity through a property and each of these connected entities are represented as a bag of words with their label.

2. the properties: a list of properties that a given entity holds

[18] The `rdf:type` of the entity

[19] The incompleteness of semantic typing in our knowledge bases poses a challenge to H3. In the implementation of this heuristic, we assume that entities which do not belong to any of the main classes fit the type constraint posed by the hybrid step.

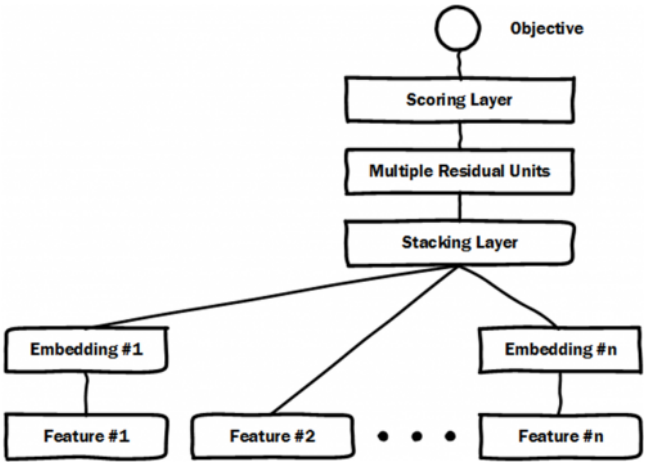

Figure 4.3 – Deep-crossing architecture from [155]

3. entity types: a list of attached entity types of a given entity

4. description: a description attached to a given entity

The first and fourth features are translated into a letter tri-grams vector. Each word is cut into a list of its letter tri-grams components (i.e. *cat* is turned into *#ca,cat,at#*), where the character # represents the start and the end of the word (see [79] for more details). The second and third features are represented as a set of one-hot vectors. Finally, the network itself learns the similarity between a pair of entity and the learning algorithm optimizes an objective function based on cosine distance over the output of the last hidden layer of each entity representation.

The disadvantage of this approach is that it delays the feature interaction to the late stage of the forward computation. Before reaching the cosine distance computation, the input features are fully embedded through multiple layers of transformations on two separate routes loosing the real meaning that each feature can bring at the beginning of the learning. Second, the size of the concatenated input vector is huge (105K columns) increasing significantly the time to train a model. Recently, to solve this problem of features combination in a neural network, Microsoft has developed what they call *deep-crossing* [155]. The role of this new generation of neural network is to *cross* the features at the beginning of the learning by adopting at most one layer of single-feature embedding. More precisely, each feature is put into an embedding layer and this embedding layer will reduce the size and catch the importance of each input feature, next the output of these embedding layers will be stacked into a stacking layer. Next, this stacking layer is followed by several residual units layers and finally a scoring layer. The Figure 4.3 shows the architecture of this new architecture.

A last fact on this architecture is that the embedding layers are used (and mostly useful) on sparse vectors, the dense vectors (much smaller) can be directly feed into

the stacking layer.

We have decided to re-create this architecture with the same features than DSRM and then put this new similarity measure into the graph regularization approach for entity linking. We let as future work the full evaluation of this network over the entire Ceccarelli dataset [24] and to integrate it into the graph regularization approach. The longest part being the creation of the training set with the entire set of entities in DBpedia. However, we have been able to implement it, to test it, and to compare it against DSRM (that we have also re-implemented) over a small part of the Ceccarelli dataset and a training set of few dozens of examples. The preliminary results show indeed a better score for computing the similarity between two entities than DSRM but despite this encouraging observation we cannot extend it to say that it outperforms DSRM and let the real experiment as part of the future work.

4.2.6 NIL Clustering

Many mentions refer to the entities that are not present in the reference knowledge base (NIL entities). For those mentions, the system should cluster them into groups, each referring to a same NIL entity (NIL Clustering). To this objective, a clustering method is applied to cluster such entities. Each mention that has been detected as NIL, forms a cluster and a unique id is assigned to it. Afterwards, we attach to each cluster the mention of the NIL entity and its type. Once done we compute a Jaccard similarity between the new coming cluster and the others that have already been created. If the similarity score is higher than a given threshold the two clusters are merged and obtain the cluster id of the matched cluster, otherwise, the new cluster is kept as it is and put into the list of clusters. For example, let's take the following piece of text:

When his brother is killed in a robbery, paraplegic Marine Jake Sully decides to take his place in a mission on the distant world of Pandora. There Jake learns of greedy corporate figurehead Parker Selfridge's intentions of driving off the native humanoid "Na'vi" in order to mine for the precious material scattered throughout their rich woodland.

None of the *PERSON* entities belonging to this piece of text belongs to DBpedia. Hence, *Jake Sully*, *Jake* and *Parker Selfridge* are all typed as *PERSON* and are NIL entities. The first NIL entity, *Jake Sully*, becomes the first NIL cluster with the $id = 1$. Next, the NIL entity, *Jake*, is compared with the cluster $id = 1$ and the Jaccard similarity gives a score higher than the threshold, then the NIL entity *Jake* is integrated into the cluster $id = 1$. Finally, the last NIL entity, *Parker Selfridge*, is compared with the cluster $id = 1$ and the similarity is lower than the threshold, and then becomes a second NIL cluster with $id = 2$. At the end, in the results, we will have the links *NIL1* and *NIL2* where the former is attached to the entities *Jake Sully*

and *Jake*, and the latter to the entity *Parker Selfridge*.

5

ADEL: A Scalable Architecture For Entity Linking

In this chapter, we describe the architecture of the ADEL framework, its implementation and its usage. We detail every single components and its configuration possibilities as well as its extension points.

5.1 Architecture

The architecture of ADEL is depicted in Figure 5.1.

5.1.1 Formats

ADEL uses and handles the following formats:

- **TSV**: the Tabulation Separated Values [1] format is used to define the mapping between classes from different datasource. It is also used to represent the single word and two-grams dictionaries for the hashtag segmentation process.
- **NIF**: the NLP Interchange Format [2] is used to exchange data between the extractors and ADEL. It is also used as possible input to feed ADEL and output format for the results.

[1] http://www.iana.org/assignments/media-types/text/tab-separated-values
[2] http://persistence.uni-leipzig.org/nlp2rdf/

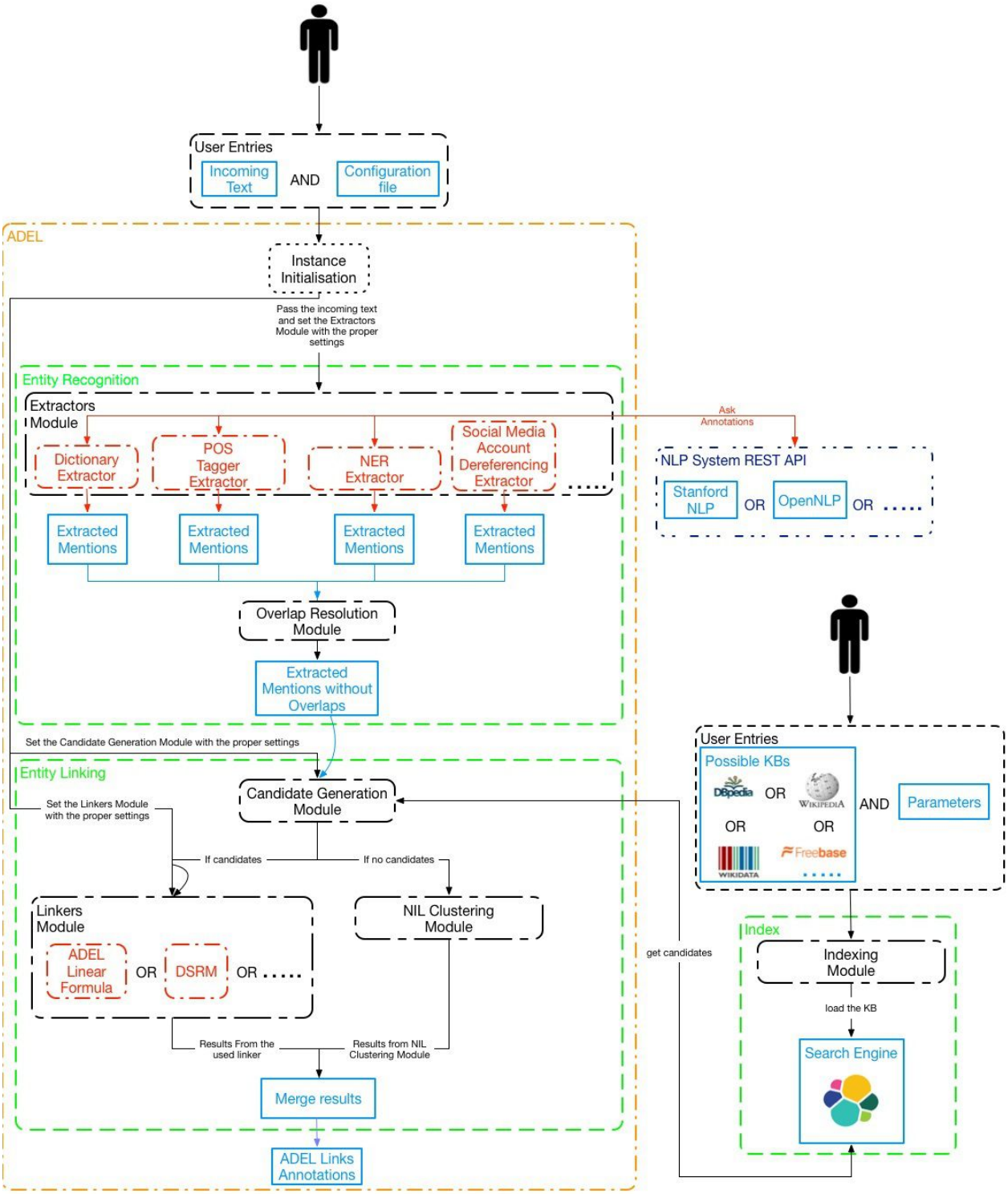

Figure 5.1 – The ADEL architecture

- **TAC**: this format is the official format used to represent the annotations for the TAC-KBP challenges [3], and is a possible output format for the results given by ADEL.
- **CoNLL**: this format is the official format used to represent the annotations for the CoNLL challenges [4]. This format is used by ADEL as a possible output for the entity recognition.
- **JSON**: this format [5] is used to query the extractors and the Elasticsearch indexes. It is also the format used to output the indexing process.
- **SRT**: this format [6] is used to represent the video subtitles and ADEL can be feed with it.
- **TTML**: this format [7] is also used to represent the video subtitles and as for SRT, it can be used as input for ADEL.

5.1.2 Technologies

ADEL makes use of the following technologies:

- **Java**: ADEL is entirely coded in Java[8] version 8.
- **Maven**: ADEL uses Maven[9] 3 as building tool.
- **Hazelcast**: ADEL uses Hazelcast[10] as in-memory, caching and key-value store system.
- **Docker**: ADEL uses Docker[11] as a possible deployment system.
- **Elasticsearch**: ADEL mainly uses Elasticsearch[12] as indexing system.
- **Dropwizard**: ADEL uses Dropwizard[13] as REST API and command line toolset.
- **Python**: ADEL uses Python[14] 2.7 and 3.4+ as scripting language.

[3] http://nlp.cs.rpi.edu/kbp/2016/taskspec.pdf
[4] http://universaldependencies.org/docs/format.html
[5] https://www.rfc-editor.org/rfc/rfc7159.txt
[6] https://matroska.org/technical/specs/subtitles/srt.html
[7] https://www.w3.org/TR/ttml1/
[8] http://www.oracle.com/technetwork/java/javase/terms/products/index.html
[9] http://maven.apache.org/
[10] https://hazelcast.org/
[11] https://www.docker.com/
[12] https://www.elastic.co/
[13] http://www.dropwizard.io/1.2.0/docs/
[14] https://www.python.org/

- **Neo4j**: ADEL uses Neo4j[15] for manipulating graphs, specifically for the JeuxDeLiens approach detailed in Chapter 4 Section 4.2.2.
- **Swagger**: ADEL uses Swagger[16] for documenting and providing a user interface to use the ADEL REST API.

5.1.3 Project Structure

The structure of ADEL is represented by the folder tree in Figure 5.2. Each of them has a specific role:

- **conf**: contains the Dropwizard and Swagger configuration files.
- **datasets**: contains the bencharmark datasets against which ADEL is tested.
- **dictionaries**: contains the single word and two-grams dictionaries used for the hashtag segmentation.
- **local-maven-repo**: contains the locally installed Java packages for Maven that are not listed in Maven central repository.
- **logs**: contains the logs of ADEL
- **mappings**:
  - **index**: contains the mappings between the properties in an index to the fixed properties used in ADEL
  - **types**: contains the mappings between sets of types
- **profiles**: contains the ADEL profiles
- **queries**:
  - **elasticsearch**: contains the JSON queries for Elasticsearch
  - **sparql**: contains the SPARQL queries for creating the indexes
- **scripts**: contains different scripts for using ADEL such as evaluation, annotation debugging or data sanity check
  - **indexing**: contains different scripts that helps for indexing different knowledge bases, index optimization or evaluation
- **src**: ADEL sources
- **target**: Maven compiled files and binaries

[15] https://neo4j.com/
[16] https://swagger.io/

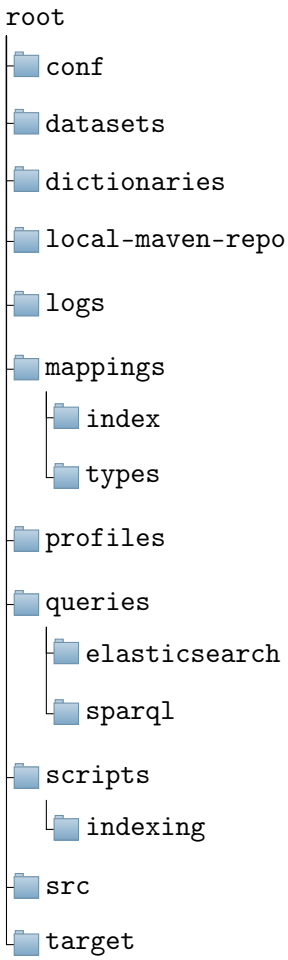

Figure 5.2 – ADEL project structure tree

5.2 Usage

This section describes how to use ADEL with its different possibilities: standalone, REST API or via Gerbil. ADEL makes use of a specific configuration format.

5.2.1 Configuration File

The configuration of ADEL is done with Dropwizard, and then needs a configuration file expressed in YAML that we call *profile* (Listing 5.1). This configuration aims to adapt an ADEL workflow to a specific case. In the reminder of this section, we will detail how each component of a configuration file works.

```
extract:
  tokenize: <http address of a tokenization service>
  to: <targeted types>
  priority: <ordered list of extractors>
  <extractor>:
    - address: <http address of the extractor service>
      name: <name of the extractor>
      tags: <list of labels to take into account>
      profile: <extractor profile>
      from: <source types>
      method: <ADEL extractor API to use>

index:
  name: <index name>
  from: <source types of the index>
  <index object>:
        <index properties>

link:
  to: <targeted types>
  method: <ADEL linking method to use>

nerd:
  to: <targeted types>
  priority: <ordered list of extractors and link>
```

Listing 5.1 – An example of an ADEL profile

Extract. In Listing 5.1, the object *extract* configures the entity recognition component. The property *tokenize* refers to the address where to find a tokenizer. The property *to* indicates into which set of types the entities must be mapped. The property *priority* represents the priority list of the declared extractors. These two last property are taken into account **only** when the extract process is called, and ignored for link and nerd process (see the paragraphs below). Finally, it is composed of at least one extractor object (ner, pos, coref, gazetteer, date and number), the value of these objects being a list of instances. An instance is composed of five mandatory properties: *address*, *name*, *profile*, *from*, *method*, 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 when it is queried. The property *method* is the name given to the Java class that has to be used internally to run the extractor. The single optional property, *tags*, represents the list of tags that have to be extracted (all if empty or not present).

Index. In Listing 5.1, 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, Lucene and Neo4j. 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 5.2. The property *strict* can have two values: *true* if we want a strict search, or *false* if we want a fuzzy search.

```
lucene:
  folder: <where to find the Lucene folder>
  fields: <list of fields to query>
  size: <number of candidates to retrieve>
```

Listing 5.2 – Lucene index object

```
elasticsearch:
  address: <Elasticsearch HTTP address>
  strict: <if use strict search or not>
  query: <file containing the JSON query to use>
```

Listing 5.3 – Elasticsearch index object

```
neo4j:
  address: <Neo4J HTTP address>
  query: <file containing the Cypher query to use>
```

Listing 5.4 – Neo4J index object

Link. In Listing 5.1, the object *link* configures the entity linking component. The property *method* is the name given to the Java class that has to be used internally to run the linking. The property *to* specify to which set of types the entities in the results must be mapped. The property *to* is taken into account **only** when ADEL is used for the link process, and ignored for extract and nerd process.

Nerd. In Listing 5.1, the object *nerd* configures the nerd part. Nerd means to run extract + link. The property *to* indicates into which set of types the entities must be mapped. The property *priority* represents the priority list of the declared extractors, and also the type attached to the URI. These two last property are taken into account **only** when the nerd process is called, and ignored for link and extract process

Now that we have seen how a configuration file looks like, we gonna detail the default configuration file as example in Listing 5.5.

```
extract:
  tokenize: http://localhost/v4/tokenize
  to: Stanford
  priority: stanfordner
  ner:
    - address: http://localhost/v4/ner
      name: stanfordner
      profile: none
      from: Stanford
      method: NER API
  pos:
    - address: http://localhost/v4/pos
      name: stanfordpos
      tags: NNP
      profile: none
      from: none
      method: POS API

index:
  name: dbpedia201510
  from: DBpedia
  elasticsearch:
    address: http://localhost:9200
    strict: true
    query: dbpedia-with-filter

link:
  to: Stanford
  method: ADEL Formula

nerd:
  to: Stanford
  priority: stanfordner,uri
```

Listing 5.5 – ADEL default configuration

Extract. The extract part of this example involves two extractors, a NER and a POS, both based on Stanford CoreNLP. The tokenization is also based on Stanford CoreNLP. We want all the entity types mapped to the Stanford ones, and the NER will have the highest priority in case of draw majority vote during the overlap resolution (if only the recognition process is called). For the NER, the default configuration of the extractor is used, the internal class corresponding to the name *NER API* will be used to communicate with the extractor, and we say that the types provided by this extractor are all based on Stanford. Finally, for the POS, the default configuration of the extractor is also used, the internal corresponding to the name *POS API* will be used to communicate with the extractor, no types are provided by the extractor, and only the tokens tagged as *NNP* will be kept. As the *from* and the *to* properties have the same value, there is no need to pass by a mapping before the overlap resolution.

Index. The index part of this examples, makes use of an Elasticsearch index. The name of the index is *dbpedia201510* where each entity is typed with the *DBpedia* ontology. We want a strict search and the JSON query pattern to use is available in the file *queries/elasticsearch/dbpedia-with-filter.qry*.

Link. We want all the entity types mapped to the Stanford ones (if only the link process is called). The linking method will be *ADEL Formula* which is the method described in Chapter 4, Section 4.2.1.

Nerd. We want all the entity types mapped to the Stanford ones, and the NER will have the highest priority over the URI type in case of draw majority vote during the overlap resolution (if only the nerd process is called).

5.2.2 ADEL Standalone

ADEL can be used as a standalone JAR archive, but we must be sure to have all the external resources (index and extractors) already up and running before to run ADEL itself. ADEL can be run with three different process: *extract* (Listing 5.6), *link* (Listing 5.7) and *nerd* (Listing 5.8). The listing of each process represent its help message. The *extract* and *nerd* process share the same parameters and work the same way except with one difference, the *conll* output format does not exists for *nerd* and the *tac* output format does not exists for *extract*. Here several usage examples:

1. Recognize entities from the sentence *I like Paris* with default values for each parameters.

   java -jar adel.jar extract -t "I like Paris"

2. Recognize entities from the sentence *I like Paris* with the default profile and with a CoNLL output.

   java -jar adel.jar extract -t "I like Paris" -of conll

3. Recognize entities from the content of a NIF file with the default profile and with French language.

   java -jar adel.jar extract -i file.ttl -l fr -if nif

4. Recognize entities from the content of the Google homepage with default values for each parameters.

   java -jar adel.jar extract -u http://google.fr

5. Recognize entities from a tweet with the tweet profile.

   java -jar adel.jar extract -t "This is the thesis of @julienplu #longwork" -s tweet

6. Recognize entities from a list of sentences in a TSV file with default profile. Each line has two columns, the first column is the ID of the sentence and the second column is the sentence itself. Get the results also into a file in TSV format where the first column is the ID of the sentence and the second columns is the sentence itself with annotated entities.

   java -jar adel.jar extract -i file.tsv -if tsv -of tsv -o output.tsv

7. Link the entity *Paris* from the sentence *I like Paris* with default values for each parameters.

java -jar adel.jar link -t "I like [[Paris]]"

8. Link the annotated entity *Paris* from the sentence *I like Paris* with default values for each parameters.

java -jar adel.jar link -t "I like [[Paris]]"

9. Link the annotated entities from a TSV file (same format than for the example 5) with the perfect candidate generation parameter. The format of the gold standard file must be in TAC.

java -jar adel.jar link -i dataset.tsv -g goldstandard.tsv -pcg

10. Link the annotated entities from a TSV file (same input format than for the output of the example 6) with the perfect linking parameter. The format of the gold standard file must be in TAC.

java -jar adel.jar link -i dataset.tsv -g goldstandard.tsv -pl

```
usage: java -jar adel.jar
       extract [-of {nif,brat,conll,naf,tsv}] [-if {raw,srt,ttml,tsv,nif}] [-s SETTING] [-l LANG] [-o OFILE] [-h] (-t TEXT | -i IFILE | -u URL
           ) [file]

Only extract and type entities

positional arguments:
  file                  application configuration file

optional arguments:
  -of {nif,brat,conll,naf,tsv}, --output-format {nif,brat,conll,naf,tsv}
                        the output format for the annotations (default: nif)
  -if {raw,srt,ttml,tsv,nif}, --input-format {raw,srt,ttml,tsv,nif}
                        the input format for the text (default: raw)
  -s SETTING, --setting SETTING
                        Select the setting (default: default)
  -l LANG, --language LANG
                        Select the language (default: en)
  -o OFILE, --output-file OFILE
                        Output file name which will contain the annotations
  -h, --help            show this help message and exit

inputs:
  -t TEXT, --text TEXT  text to analyse
  -i IFILE, --input-file IFILE
                        Input file name which contain the text to process
  -u URL, --url URL     URL to process
```

Listing 5.6 – ADEL help message for the extract process

```
usage: java -jar adel.jar
       link [-of {nif,brat,tac,naf}] [-if {raw,tsv,nif}] [-s SETTING] [-l LANG] [-o OFILE] [-g G] [-h] (-t TEXT | -i IFILE | -u URL) [-pl | -
           pcg] [file]

Only linking entities

positional arguments:
  file                  application configuration file

optional arguments:
  -of {nif,brat,tac,naf}, --output-format {nif,brat,tac,naf}
                        the output format for the annotations (default: tac)
  -if {raw,tsv,nif}, --input-format {raw,tsv,nif}
                        the input format for the text (default: raw)
  -s SETTING, --setting SETTING
                        Select the setting (default: default)
  -l LANG, --language LANG
                        Select the language (default: en)
  -o OFILE, --output-file OFILE
                        Output file name which will contain the annotations
  -g G, --gold-standard G
                        Gold standard in case of perfect linking or candidate generation if TSV dataset
  -h, --help            show this help message and exit

inputs:
  -t TEXT, --text TEXT  text to analyse
  -i IFILE, --input-file IFILE
                        Input file name which contain the text to process
  -u URL, --url URL     URL to process

perfect-option:
  -pl, --perfect-linking
                        If perfect linking. Perfect linking aims to always give the right link if it belongs to the list of candidates
                            retrieved by the index (default: false)
  -pcg, --perfect-candidate-generation
                        If perfect candidate generation. Perfect candidate generation aims to always have the right link in the list of
                            candidates retrieved by the index (default: false)
```

Listing 5.7 – ADEL help message for the link process

```
usage: java -jar adel.jar
       nerd [-of {nif,brat,tac,naf,neel2014}] [-if {raw,srt,ttml,tsv,nif}] [-s SETTING] [-l LANG] [-o OFILE] [-h] (-t TEXT | -i IFILE | -u URL
            ) [file]

Extract, type and link entities

positional arguments:
  file                  application configuration file

optional arguments:
  -of {nif,brat,tac,naf,neel2014}, --output-format {nif,brat,tac,naf,neel2014}
                        the output format for the annotations (default: tac)
  -if {raw,srt,ttml,tsv,nif}, --input-format {raw,srt,ttml,tsv,nif}
                        the input format for the text (default: raw)
  -s SETTING, --setting SETTING
                        Select the setting (default: default)
  -l LANG, --language LANG
                        Select the language (default: en)
  -o OFILE, --output-file OFILE
                        Output file name which will contain the annotations
  -h, --help            show this help message and exit

inputs:
  -t TEXT, --text TEXT  text to analyse
  -i IFILE, --input-file IFILE
                        Input file name which contain the text to process
  -u URL, --url URL     URL to process
```

Listing 5.8 – ADEL help message for the nerd process

5.2.3 ADEL REST API

Another possibility to use ADEL is through its REST API and we also must be sure to have all the external resources (index and extractors) already up and running before to run the API. Like the standalone version, the API has three methods: *extract*, *link* and *nerd*, and for these methods the same two parameters: *setting* and *lang*. The parameter *setting* correspond to the option *-s* (profile selection) in the standalone version. The parameter *lang* correspond to the option *-l* (language selection) in the standalone version. The other properties are set with the JSON input query as detailed in Listing 5.9 for the extract method, and in Listing 5.10 with an explanation for each property:

```
{
  "content": "string",
  "url": "string",
  "input": "raw",
  "output": "nif"
}
```

Listing 5.9 – ADEL JSON extract and nerd intput API query

```
{
  "content": "string",
  "input": "raw",
  "output": "nif"
}
```

Listing 5.10 – ADEL JSON link intput API query

- *content*: Content of the processed document (content XOR URL must be specified)
- *url*: Valid URL of a HTML document (content XOR URL must be specified)
- *input* (for extract and nerd): The content must respect one of these format. If *raw* the content is a raw text; if *nif* the content is in Turtle respecting the NIF vocabulary; if *html* the content is regularly marked up; if *ttml* or *srt* the content correspond to subtitles in their respecting format
- *input* (for link): The content must respect one of these format. If *raw* the content is a raw text, if *nif* the content is in Turtle respecting the NIF vocabulary.
- *output* (for extract): The response can only be in one of the format. *nif*, *conll*, *brat*, *naf* or *tsv*.
- *output* (for link and nerd): The response can only be in one of the format. *nif*, *tac*, *brat*, *naf*.

We propose to reproduce the same examples than for the standalone version but the operations implying a file that is not in NIF format are not possible through the API:

1. Recognize entities from the sentence *I like Paris* with default values for each parameters (Listing 5.11).
2. Recognize entities from the sentence *I like Paris* with the default profile and with a CoNLL output (Listing 5.12).
3. Recognize entities from the content of a NIF file with the default profile and with French language (Listing 5.13).
4. Recognize entities from the content of the Google homepage with default values for each parameters (Listing 5.14).
5. Recognize entities from a tweet with the tweet profile (Listing 5.15).
6. Link the entity *Paris* from the sentence *I like Paris* with default values for each parameters (Listing 5.16).

```
curl -X POST "http://localhost:7000/v1/extract?setting=default&lang=en" -H "accept: text/
    plain;charset=utf-8" -H "content-type: application/json;charset=utf-8" -d "{ \"content
    \": \"I like Paris\"}"
```

Listing 5.11 – Apply example 1 of the API usage

```
curl -X POST "http://localhost:7000/v1/extract?setting=default&lang=en" -H "accept: text/
    plain;charset=utf-8" -H "content-type: application/json;charset=utf-8" -d "{ \"content
    \": \"I like Paris\", \"output\": \"conll\"}"
```

Listing 5.12 – Apply example 2 of the API usage

```
curl -X POST "http://localhost:7000/v1/extract?setting=default&lang=fr" -H "accept: text/
    plain;charset=utf-8" -H "content-type: application/json;charset=utf-8" -d @file.ttl"
```

Listing 5.13 – Apply example 3 of the API usage

```
curl -X POST "http://localhost:7000/v1/extract?setting=default&lang=en" -H "accept: text/
    plain;charset=utf-8" -H "content-type: application/json;charset=utf-8" -d "{ \"url\":
    \"http://google.fr\"}"
```

Listing 5.14 – Apply example 4 of the API usage

The ADEL API can also be used through GERBIL [177], one need to give the API URL as new annotator by specifying the proper method and parameters, but it

```
curl -X POST "http://localhost:7000/v1/extract?setting=tweet&lang=en" -H "accept: text/
    plain;charset=utf-8" -H "content-type: application/json;charset=utf-8" -d "{ \"content
    \": \"This is the thesis of @julienplu #longwork\"}"
```

Listing 5.15 – Apply example 5 of the API usage

```
curl -X POST "http://localhost:7000/v1/link?setting=default&lang=en" -H "accept: text/plain
    ;charset=utf-8" -H "content-type: application/json;charset=utf-8" -d "{ \"content\":
    \"I like [[Paris]]\"}"
```

Listing 5.16 – Apply example 6 of the API usage

is not possible to modify the default properties given by the JSON queries because GERBIL does not allows to send such query. The example 3 allows also to simulate a local usage through GERBIL. A public version of the API is available[17].

5.2.4 ADEL Routines

ADEL comes with several useful integrated command line routines:

- *gendic*: this routine creates a dictionary as described in Chapter 3 Section 3.2.1 from a given index.
- *neel20142tac*: this routine turns a dataset in NEEL2014 format into a TAC format.
- *neel20142nif*: this routine turns a dataset in NEEL2014 format into a NIF format.
- *nif-stats*: this routine gives statistics about a NIF dataset (number of sentences, average length of sentences, number of entities per type, number of NIL entities, etc.)
- *nif2conll*: this routine turns a dataset in NIF format into a CoNLL format.
- *nif2tac*: this routine turns a dataset in NIF format into a TAC format.
- *pagerank*: this routine computes the PageRank of each entity contained in a RDF file.
- *tac-stats*: this routine gives statistics about a TAC dataset (number of sentences, average length of sentences, number of entities per type, number of NIL entities, etc.)
- *tac2conll*: this routine turns a dataset in TAC format into a CoNLL format.
- *tac2nif*: this routine turns a dataset in TAC format into a NIF format.

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

5.3 Adaptation

This section propose to detail how to customize the usage of ADEL by adding its own extractors, linking methods or index.

5.3.1 Add a New Extractor

A good example of how to develop an extractor is proposed with the Stanford CoreNLP toolkit[18]. This implementation proposes an extractor for each Stanford annotator with several configurations. As we can see in the complete documentation[19], the fashion to query an extractor is similar to the one used for the ADEL API in order to avoid difficulties and keep an harmony among the tools. Such as with ADEL, the answer is in NIF and each developed extractor has to respect the exact same protocol in order to be used through ADEL. The technology used to develop an API is let to the developer, the only one requirement is to respect the protocol.

As detailed in Chapter 3 there are four types of extractors, and each of them as to respect a specific NIF answer, while they all share the same JSON input query represented in Listing 5.17. There are three different answer: entity (Listing 5.18), coreference (Listing 5.19) and part-of-speech(Listing 5.20). A named entity recognition tagger and a dictionary tagger have both an entity answer. The examples for each type of answer are given by the Stanford CoreNLP toolkit for the following toy text: *J. K. Rowling is a famous author. She has written the series of books: Harry Potter.*.

```
{
  "content": "string",
  "url": "string"
}
```

Listing 5.17 – Extractor JSON input query

5.3.2 Add a New Linker

Currently, the only way to extend ADEL with a new linker is to modify the source code. In order to add a new linking approach, one has to create a new class that implements the Java interface *fr.eurecom.adel.tasks.linking.Linking* and to give it a name with the annotation *fr.eurecom.adel.annotations.Name* that will be used in a profile to identify the new linking approach to use. We have started to develop a similar approach than for the extractors to integrate new linking methods via a REST API which will be pursued as future work.

[18] https://github.com/jplu/stanfordNLPRESTAPI
[19] https://github.com/jplu/stanfordNLPRESTAPI/wiki/API-documentation

```
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix nif:   <http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core#> .
@prefix local: <http://localhost:7000/stanfordnlp/ontology/> .

<http://localhost:7000/stanfordnlp/entity#char=0,13>
        a                       nif:Phrase , nif:RFC5147String , nif:String ;
        local:type              "PERSON" ;
        nif:anchorOf            "J. K. Rowling" ;
        nif:beginIndex          "0"^^xsd:nonNegativeInteger ;
        nif:endIndex            "13"^^xsd:nonNegativeInteger ;
        nif:referenceContext    <http://localhost:7000/stanfordnlp/context#char=0,84> ;
        nif:sentence            <http://localhost:7000/stanfordnlp/sentence#char=0,33> .

<http://localhost:7000/stanfordnlp/context#char=0,84>
        a                 nif:Context , nif:RFC5147String , nif:String ;
        nif:beginIndex    "0"^^xsd:nonNegativeInteger ;
        nif:endIndex      "84"^^xsd:nonNegativeInteger ;
        nif:isString      "J. K. Rowling is a famous author. She has written the series of
             books: Harry Potter." .

<http://localhost:7000/stanfordnlp/entity#char=71,83>
        a                       nif:Phrase , nif:RFC5147String , nif:String ;
        local:type              "PERSON" ;
        nif:anchorOf            "Harry Potter" ;
        nif:beginIndex          "71"^^xsd:nonNegativeInteger ;
        nif:endIndex            "83"^^xsd:nonNegativeInteger ;
        nif:referenceContext    <http://localhost:7000/stanfordnlp/context#char=0,84> ;
        nif:sentence            <http://localhost:7000/stanfordnlp/sentence#char=34,84> .

<http://localhost:7000/stanfordnlp/sentence#char=34,84>
        a                       nif:String , nif:RFC5147String , nif:Sentence ;
        local:entity            <http://localhost:7000/stanfordnlp/entity#char=71,83> ;
        nif:anchorOf            "She has written the series of books: Harry Potter." ;
        nif:beginIndex          "34"^^xsd:nonNegativeInteger ;
        nif:endIndex            "84"^^xsd:nonNegativeInteger ;
        nif:previousSentence    <http://localhost:7000/stanfordnlp/sentence#char=0,33> ;
        nif:referenceContext    <http://localhost:7000/stanfordnlp/context#char=0,84> .

<http://localhost:7000/stanfordnlp/sentence#char=0,33>
        a                       nif:String , nif:RFC5147String , nif:Sentence ;
        local:entity            <http://localhost:7000/stanfordnlp/entity#char=0,13> ;
        nif:anchorOf            "J. K. Rowling is a famous author." ;
        nif:beginIndex          "0"^^xsd:nonNegativeInteger ;
        nif:endIndex            "33"^^xsd:nonNegativeInteger ;
        nif:nextSentence        <http://localhost:7000/stanfordnlp/sentence#char=34,84> ;
        nif:referenceContext    <http://localhost:7000/stanfordnlp/context#char=0,84> .
```

Listing 5.18 – Example of an entity output

```
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix nif:   <http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core#> .
@prefix local: <http://localhost:7000/stanfordnlp/ontology/> .

<http://localhost:7000/stanfordnlp/head#char=0,13>
        a                       nif:Phrase , nif:RFC5147String , nif:String ;
        nif:anchorOf            "J. K. Rowling" ;
        nif:beginIndex          "0"^^xsd:nonNegativeInteger ;
        nif:endIndex            "13"^^xsd:nonNegativeInteger .

<http://localhost:7000/stanfordnlp/context#char=0,84>
        a                 nif:Context , nif:RFC5147String , nif:String ;
        nif:beginIndex    "0"^^xsd:nonNegativeInteger ;
        nif:endIndex      "84"^^xsd:nonNegativeInteger ;
        nif:isString      "J. K. Rowling is a famous author. She has written the series of
            books: Harry Potter." .

<http://localhost:7000/stanfordnlp/coref#char=34,37>
        a                       nif:Phrase , nif:RFC5147String , nif:String ;
        local:head              <http://localhost:7000/stanfordnlp/head#char=0,13> ;
        nif:anchorOf            "She" ;
        nif:beginIndex          "34"^^xsd:nonNegativeInteger ;
        nif:endIndex            "37"^^xsd:nonNegativeInteger ;
        nif:referenceContext    <http://localhost:7000/stanfordnlp/context#char=0,84> .
        nif:sentence            <http://localhost:7000/stanfordnlp/sentence#char=34,84> .

<http://localhost:7000/stanfordnlp/sentence#char=34,84>
        a                       nif:String , nif:RFC5147String , nif:Sentence ;
        local:coref             <http://localhost:7000/stanfordnlp/coref#char=34,37> ;
        nif:anchorOf            "She has written the series of books: Harry Potter." ;
        nif:beginIndex          "34"^^xsd:nonNegativeInteger ;
        nif:endIndex            "84"^^xsd:nonNegativeInteger ;
        nif:previousSentence    <http://localhost:7000/stanfordnlp/sentence#char=0,33> ;
        nif:referenceContext    <http://localhost:7000/stanfordnlp/context#char=0,84> .

<http://localhost:7000/stanfordnlp/sentence#char=0,33>
        a                       nif:String , nif:RFC5147String , nif:Sentence ;
        local:entity            <http://localhost:7000/stanfordnlp/entity#char=0,13> ;
        nif:anchorOf            "J. K. Rowling is a famous author." ;
        nif:beginIndex          "0"^^xsd:nonNegativeInteger ;
        nif:endIndex            "33"^^xsd:nonNegativeInteger ;
        nif:nextSentence        <http://localhost:7000/stanfordnlp/sentence#char=34,84> ;
        nif:referenceContext    <http://localhost:7000/stanfordnlp/context#char=0,84> .
```

Listing 5.19 – Example of a coreference output

```
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix nif:   <http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core#> .
@prefix local: <http://localhost:7000/stanfordnlp/ontology/> .

<http://localhost:7000/stanfordnlp/token#char=17,18>
        a                       nif:Word , nif:RFC5147String , nif:String ;
        nif:anchorOf            "a" ;
        nif:beginIndex          "17"^^xsd:nonNegativeInteger ;
        nif:endIndex            "18"^^xsd:nonNegativeInteger ;
        nif:nextWord            <http://localhost:7000/stanfordnlp/token#char=19,25> ;
        nif:posTag              "DT" ;
        nif:previousWord        <http://localhost:7000/stanfordnlp/token#char=14,16> ;
        nif:referenceContext    <http://localhost:7000/stanfordnlp/context#char=0,84> ;
        nif:sentence            <http://localhost:7000/stanfordnlp/sentence#char=0,33> .

<http://localhost:7000/stanfordnlp/context#char=0,84>
        a               nif:Context , nif:RFC5147String , nif:String ;
        nif:beginIndex  "0"^^xsd:nonNegativeInteger ;
        nif:endIndex    "84"^^xsd:nonNegativeInteger ;
        nif:isString    "J. K. Rowling is a famous author. She has written the series of
             books: Harry Potter." .

<http://localhost:7000/stanfordnlp/token#char=14,16>
        a                       nif:RFC5147String , nif:String , nif:Word ;
        nif:anchorOf            "is" ;
        nif:beginIndex          "14"^^xsd:nonNegativeInteger ;
        nif:endIndex            "16"^^xsd:nonNegativeInteger ;
        nif:nextWord            <http://localhost:7000/stanfordnlp/token#char=17,18> ;
        nif:posTag              "VBZ" ;
        nif:previousWord        <http://localhost:7000/stanfordnlp/token#char=6,13> ;
        nif:referenceContext    <http://localhost:7000/stanfordnlp/context#char=0,84> ;
        nif:sentence            <http://localhost:7000/stanfordnlp/sentence#char=0,33> .

<http://localhost:7000/stanfordnlp/token#char=0,2>
        a                       nif:Word , nif:String , nif:RFC5147String ;
        nif:anchorOf            "J." ;
        nif:beginIndex          "0"^^xsd:nonNegativeInteger ;
        nif:endIndex            "2"^^xsd:nonNegativeInteger ;
        nif:nextWord            <http://localhost:7000/stanfordnlp/token#char=3,5> ;
        nif:posTag              "NNP" ;
        nif:referenceContext    <http://localhost:7000/stanfordnlp/context#char=0,84> ;
        nif:sentence            <http://localhost:7000/stanfordnlp/sentence#char=0,33> .

<http://localhost:7000/stanfordnlp/sentence#char=0,33>
        a                       nif:Sentence , nif:RFC5147String , nif:String ;
        nif:anchorOf            "J. K. Rowling is a famous author." ;
        nif:beginIndex          "0"^^xsd:nonNegativeInteger ;
        nif:endIndex            "33"^^xsd:nonNegativeInteger ;
        nif:firstToken          <http://localhost:7000/stanfordnlp/token#char=0,2> ;
        nif:lastToken           <http://localhost:7000/stanfordnlp/token#char=32,33> ;
        nif:nextSentence        <http://localhost:7000/stanfordnlp/sentence#char=34,84> ;
        nif:referenceContext    <http://localhost:7000/stanfordnlp/context#char=0,84> ;
        nif:word                <http://localhost:7000/stanfordnlp/token#char=0,2> , <http://
             localhost:7000/stanfordnlp/token#char=3,5> , <http://localhost:7000/
             stanfordnlp/token#char=26,32> , <http://localhost:7000/stanfordnlp/token#char
             =17,18> , <http://localhost:7000/stanfordnlp/token#char=6,13> , <http://
             localhost:7000/stanfordnlp/token#char=32,33> , <http://localhost:7000/
             stanfordnlp/token#char=14,16> , <http://localhost:7000/stanfordnlp/token#char
             =19,25> .

<http://localhost:7000/stanfordnlp/token#char=32,33>
        a                       nif:String , nif:Word , nif:RFC5147String ;
        nif:anchorOf            "." ;
        nif:beginIndex          "32"^^xsd:nonNegativeInteger ;
        nif:endIndex            "33"^^xsd:nonNegativeInteger ;
        nif:posTag              "." ;
        nif:previousWord        <http://localhost:7000/stanfordnlp/token#char=26,32> ;
        nif:referenceContext    <http://localhost:7000/stanfordnlp/context#char=0,84> ;
        nif:sentence            <http://localhost:7000/stanfordnlp/sentence#char=0,33> .

<http://localhost:7000/stanfordnlp/token#char=26,32>
        a                       nif:Word , nif:RFC5147String , nif:String ;
        nif:anchorOf            "author" ;
        nif:beginIndex          "26"^^xsd:nonNegativeInteger ;
        nif:endIndex            "32"^^xsd:nonNegativeInteger ;
        nif:nextWord            <http://localhost:7000/stanfordnlp/token#char=32,33> ;
        nif:posTag              "NN" ;
        nif:previousWord        <http://localhost:7000/stanfordnlp/token#char=19,25> ;
        nif:referenceContext    <http://localhost:7000/stanfordnlp/context#char=0,84> ;
        nif:sentence            <http://localhost:7000/stanfordnlp/sentence#char=0,33> .
```

Listing 5.20 – Example of a part-of-speech output. This output is very large and for this reason we put only an excerpt

5.3.3 Add a New Index

In order to create a new index, we have developed several scripts to help the creation and the deployment. Currently, only the indexing of linked data knowledge base are provided. We let the automation of an index for non-linked data knowledge base as future work, in order to index such knowledge base, for now, one need to code specific scripts to create an index file. The default indexing system is Elasticsearch, then all the scripts are configured to create and use an Elasticsearch index.

Let's take an example by indexing and optimizing a particular DBpedia knowledge base snapshot (version 2016-10):

1. Download the data files from the DBpedia server:
   - wget http://downloads.dbpedia.org/2016-10/core-i18n/en/anchor_text_en.ttl.bz2
   - wget http://downloads.dbpedia.org/2016-10/core-i18n/en/disambiguations_en.ttl.bz2
   - wget http://downloads.dbpedia.org/2016-10/core-i18n/en/instance_types_en.ttl.bz2
   - wget http://downloads.dbpedia.org/2016-10/core-i18n/en/instance_types_transitive_en.ttl.bz2
   - wget http://downloads.dbpedia.org/2016-10/core-i18n/en/labels_en.ttl.bz2
   - wget http://downloads.dbpedia.org/2016-10/core-i18n/en/long_abstracts_en.ttl.bz2
   - wget http://downloads.dbpedia.org/2016-10/core-i18n/en/mappingbased_literals_en.ttl.bz2
   - wget http://downloads.dbpedia.org/2016-10/core-i18n/en/page_ids_en.ttl.bz2
   - wget http://downloads.dbpedia.org/2016-10/core-i18n/en/page_links_en.ttl.bz2
   - wget http://downloads.dbpedia.org/2016-10/core-i18n/en/pages_articles_en.xml.bz2
   - wget http://downloads.dbpedia.org/2016-10/core-i18n/en/redirects_en.ttl.bz2
   - wget http://downloads.dbpedia.org/2016-10/core-i18n/en/transitive_redirects_en.ttl.bz2
2. Compute PageRank scores for the dataset (this task needs a consequent amount of RAM, around 100GB):
   - bzip2 -d page_links_en.ttl.bz2
   - git clone https://github.com/jplu/adel
   - cd adel
   - mvn clean package
   - java -Xmx128g -jar target/adel.jar tools pagerank --input ../page_links_en.ttl --output ../pagerank.ttl
   - cd ..
   - bzip2 pagerank.ttl
   - bzip2 page_links_en.ttl
3. Create the JSON index file:

- java −Xmx32g −jar adel/target/adel.jar index create −−folder . −−pattern .ttl.bz2 −−dataset dbpedia −−output ./index_dbpedia201610.jsonl

4. Load the JSON index file into Elasticsearch:

   - java −jar adel/target/adel.jar index load −−file index_dbpedia201610.jsonl.gz −−address http://localhost:9200 −−name dbpedia201610

5. Optimize the index (the output of the last command will be the list of the optimized columns):

   - python3 scripts/create_dbpedia_dataset.py
   - java −jar adel/target/adel.jar index matrix −−address http://localhost:9200/dbpedia201610 −−dataset dbpedia_dataset.json −−output dbpedia_matrix.json
   - java −jar adel/target/adel.jar index optimize −−address http://localhost:9200/dbpedia201610 −−dataset dbpedia_matrix.json

It is not enough to have a good mind; the main thing is to use it well.

Rene Descartes

6 Evaluation

In this Chapter, we provide a thorough evaluation of ADEL over different entity linking benchmark datasets namely OKE2015 [118], OKE2016 [119], OKE2017 [164], NEEL2014 [9], NEEL2015 [143], NEEL2016 [148], AIDA [76] and LeMonde, each of these datasets have its own characteristics detailed in Table 6.1. In addition to entity linking benchmark datasets, we have also a coreference benchmark dataset (CoNLL2012 [136]), named entity recognition benchmark datasets (CoNLL2002 [150] and CoNLL2003 [152]), a part-of-speech benchmark dataset (CoNLL2009 [69]) and a chunking benchmark dataset (CoNLL2000 [151]). All the scores for the entity linking benchmark datasets are computed with GERBIL, except the ones for the best participant of the NEELs challenges that have been computed with the respective official scorer (neleval [67] for NEEL2015 and NEEL2016, and neeleval[1] for NEEL2014), since those scores are not available through GERBIL. The scores for the named entity recognition, coreference, part-of-speech and chunking benchmark datasets are computed with their official scorer, respectively conlleval[2], CoNLL2012 scorer[3] and CoNLL2009 scorer[4]. Following the terminology proposed by GERBIL, extraction is defined as *Entity Recognition*, typing as *Entity Typing*, recognition as *RT2KB*, and linking as *D2KB*.

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

[2] http://www.cnts.ua.ac.be/conll2000/chunking/conlleval.txt (For CoNLL2000, CoNLL2002 and CoNLL2003

[3] http://conll.cemantix.org/2012/software.html

[4] http://ufal.mff.cuni.cz/conll2009-st/scorer.html

Datasets	Coreferences	Entity Classification	Entity Types (Provenance)	Dates and Numbers Entities	NIL Entities	Document Types	Knowledge Bases	Languages
OKE2015	✓	✓	Person, Organization, Place, Role (DUL ontology)	✗	✓	news	DBpedia 2014	English
OKE2016	✓	✓	Person, Organization, Place, Role (DUL ontology)	✗	✓	news	DBpedia 2015-10	English
OKE2017 Task 1	✗	✓	Person, Organization, Place (DBpedia ontology)	✗	✓	news	DBpedia 2016-04	English
OKE2017 Task 2	✗	✓	Any (DBpedia ontology)	✗	✓	news	DBpedia 2016-04	English
OKE2017 Task 3	✗	✓	MusicArtist, SignalGroup, MusicalWork (Music ontology)	✗	✓	news	Musicbrainz 2016-12	English
NEEL2014	✗	✗	Any (DBpedia)	✓	✗	tweets	DBpedia 3.9	English
NEEL2015	✗	✓	Person, Location, Organization, Character, Event, Thing (NEEL Taxonomy)	✗	✓	tweets	DBpedia 2014	English
NEEL2016	✗	✓	Person, Location, Organization, Character, Event, Thing (NEEL Taxonomy)	✗	✓	tweets	DBpedia 2015-04	English
AIDA	✗	✗	Person, Organization, Location, Miscellaneous (CoNLL Taxonomy)	✗	✗	news	Wikipedia, Freebase, YAGO	English
LeMonde	✗	✓	Person, Organization, Location (CoNLL Taxonomy)	✗	✓	news	JeuxDeMots	French

Table 6.1 – Characteristics for each entity linking benchmark dataset

6.1 Evaluation Metrics

The assessment of entity linking (recognition) systems is usually performed in terms of evaluation measures, named: precision, recall and F1-measure. The precision of an entity linking (recognition) system is computed as the division of the correctly linked (recognized) entities that are generated by the system by the total number of linked (recognized) entities by the system:

$$\text{Precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}}$$

The precision takes into account all the entities that are linked (recognized) by the system and determines how correct these linked (recognized) entities are. The recall is the division of the correctly linked (recognized) entities that should be linked (recognized) by the total number of linked (recognized) entities contained in the gold standard:

$$\text{Recall} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}}$$

The recall takes into account all the entities that should be linked (recognized) and determines how correct the linked (recognized) entities are with regard to the total amount of entities that should be linked (recognized). These two measures are used together to compute the F1-measure to provide a single measurement of a system. F1-measure is defined as the harmonic mean of the precision and the recall:

$$\text{F1-measure} = 2.\frac{\text{precision} \ . \ \text{recall}}{\text{precision} + \text{recall}}$$

Among the compared systems AGDISTIS, DoSeR, PBOH and NERFGUN take as input the entities that should be linked and not the text itself, so the number of linked entities generated by these systems is equal to the number of entities that should be linked and then in this situation, $Precision = Recall = F1\text{-}measure$.

Another useful metric is accuracy. The accuracy measure how much the predictions of an approach is true and is measured with the division of the true predictions plus the false predictions by the total number of elements that must be predicted (population):

$$\text{Accuracy} = \frac{\text{true positive} + \text{true negative}}{\text{population}}$$

Many metrics have been proposed to evaluate the performance of coreference resolution systems, such as *MUC* [179], B^3 [7], or *CEAF* [96], including multiple variants of them. *MUC* counts the minimum number of links between mentions to be inserted or deleted when mapping a system response to a gold standard key set. B^3 overcomes the shortcomings of the *MUC* score, instead of looking at the links, it computes precision and recall for all mentions in the document, which are then combined to produce the final precision and recall numbers for the entire output. *CEAF* is calculated based on the best mapping between coreference expressions or entities, thus results in two types of *CEAF*: expression-based (*CEAF-M*) and entity-based (*CEAF-E*). Finally, *Avg F1* is an average of the F1 scores of the three previous ones.

6.2 Datasets

This section aims to analyze the characteristics of the used benchmark datasets in order to show how much they can be different and difficult to adapt an approach being compliant to each of them. First we give an introduction for each of them.

6.2.1 Datasets Presentation

OKE 2015 Dataset. The Open Knowledge Extraction Challenge 2015 (OKE2015) corpus consists of 196 documents from Wikipedia articles. The annotation task focused on extraction (including co-reference), classification according to the Dolce Ultra Lite classes, and linking of named entities to DBpedia. The annotation set was created using the Wikipedia links found in the articles, extended with automatic anaphora resolution, and detection of emerging entities. The annotation set was manually reviewed and fixed. The corpus was split into a train and test set: 95 in the training set, and 101 in the test set.

OKE 2016 Dataset. The Open Knowledge Extraction Challenge 2016 (OKE2016) corpus consists of 251 documents from Wikipedia articles. The annotation task focused on extraction (including co-reference), classification according to the Dolce Ultra Lite classes, and linking of named entities to DBpedia. The annotation set was created using the Wikipedia links found in the articles, extended with automatic anaphora resolution, and detection of emerging entities. The annotation set was manually reviewed and fixed. The corpus was split into a train and test set: 196 in the training set, and 55 in the test set.

OKE 2017 Dataset. The Open Knowledge Extraction Challenge 2017 (OKE2017) has 3 tasks, where the corpus has respectively: 116 sentences for the task1, 107 sentences for the task2 and 200 sentences for the task3 from Wikipedia articles. The documents for the first two tasks are from different news websites such as BBC, NYTimes or Reuters, but contains also movie descriptions from IMDB. The document of the third task are from LastFM. The annotation task focused on extraction, classification according to the DBpedia classes for the first two tasks and according to the Music ontology for the third task, and linking of named entities to DBpedia for the first two tasks and to Musicbrainz for the third task. For each task, the corpus was split into a train and test set: 60 in the training set, and 58 in the test set for Task1, 56 in the training set, and 53 in the test set for task2 and finally 100 in the training set, and 100 in the test set for task3.

NEEL2014. The Named Entity rEcognition and Linking Challenge 2014 (NEEL2014) corpus consists of 3,505 tweets extracted from a much larger collection of over 18 million tweets. The tweets were provided by the Redites project, which covers event-annotated tweets collected for a period of 31 days between July 15th, 2011 and August 15th, 2011. It includes multiple noteworthy events, such as the death of Amy

Winehouse, the London Riots, and the Oslo bombing. The corpus was created to benchmark automatic extraction and linking entities. The corpus is split into a train and a test set: 2340 in the training set, and 1165 in the test set.

NEEL2015. The Named Entity rEcognition and Linking Challenge 2015 (NEEL2015) corpus consists of 5,661 tweets. This corpus covers noteworthy events from 2011 and 2013 (e.g. the Westgate Shopping Mall shootout), as well as tweets extracted from the Twitter firehose in 2014. The training set is built on top of the corpus of the NEEL2014 challenge. It was further extended to include entity types, NIL references and remove dates and numbers entities. The corpus is split into a train, a development and a test set: 3498 in the training set, 500 in the development set, and 1663 in the test set.

NEEL2016. The Named Entity rEcognition and Linking Challenge 2016 (NEEL2016) corpus consists of 6,421 tweets. This corpus covers events from 2015 (e.g. the Star Wars Episode 7 cinema release and the Americans elections). The training set is built on top of the corpus of the NEEL2015 challenge. It includes the same guideline than the 2015 challenge. The corpus is split into a train, a development and a test set: 6025 in the training set, 100 in the development set, and 296 in the test set.

AIDA. The AIDA dataset is an extension of the CoNLL 2003 entity recognition task dataset, more precisely the English one, that consists of 1393 documents. It is based on news articles published between August 1996 and August 1997 by Reuters. Each entity is identified by its YAGO2 entity name, Wikipedia URL, and, if available, by Freebase Machine ID. The corpus is split into a train, a development and a test set: 946 in the training set, 216 in the development set, and 231 in the test set.

LeMonde. The LeMonde corpus consists of 15 documents. It is based on news articles from the French Le Monde newspaper of different topics (e.g. political, technological, economical). All the entities are annotated with the JeuxDeMots lexical semantic network. The corpus is composed of one set.

6.2.2 Datasets Characteristics

The documents which comprise benchmarking datasets can vary along several dimensions:

- **Type of discourse:** news articles, tweets, transcriptions, blog articles, scientific articles, government/medical reports
- **Topical domain:** science, sports, politics, music, catastrophic events, general (cross-domain)
- **Document length (in terms of number of tokens)**: long, medium, short
- **Format:** document format (TSV, CoNLL, NIF), stand-off vs. in inline annotation

- **Character encoding:** Unicode, ASCII, URL-encoding
- **Licensing**: open, closed, open via agreement

The Table 6.2 summarizes the document type characteristics of each corpora we use, already exposing notable diversity among the datasets with respect to the considered set of aspects.

Corpus	Type	Domain	Doc. Length	Format	Encoding	License
AIDA	news	general	long	TSV	ASCII	open via agreement
NEEL2014	tweets	general	short	TSV	ASCII	open via agreement
NEEL2015	tweets	general	short	TSV	UTF8	open via agreement
NEEL2016	tweets	general	short	TSV	UTF8	open via agreement
OKE2015	encyclopedia	general	medium	NIF/RDF	UTF8	open
OKE2016	encyclopedia	general	medium	NIF/RDF	UTF8	open
OKE2017 Task1/2	encyclopedia	general	medium	NIF/RDF	UTF8	open
OKE2017 Task3	encyclopedia	musical	long	NIF/RDF	UTF8	open
LeMonde	news	general	long	TSV	UTF8	Closed

Table 6.2 – General characteristics for analyzed datasets

6.2.3 Datasets Analysis

We also analyze the coverage of entities and mentions in these different datasets along three dimensions: entity overlap, entity distribution, and entity types.

Overlap. In Table 6.3, we present the entity overlap between the different benchmark datasets. Each row in the table represents the percentage of unique entities present in that dataset that are also represented in the other datasets. As the table illustrates, most of the benchmark datasets share common entities sometime a small, and sometime a large amount of entities. The consequent overlap between the NEEL2014, NEEL2015 and NEE2016 datasets, and also between the OKE2015 and OKE2016 datasets, is explained by the fact that each edition is an extension of the previous one. The OKE2017 Task3 and LeMonde datasets do not share any entities with the others datasets because they are not linked against DBpedia but respectively to Musicbrainz and JeuxDeMots.

	AIDA	NEEL2014	NEEL2015	NEEL2016	OKE2015	OKE2016	OKE2017 Task1	OKE2017 Task2	OKE2017 Task3	LeMonde
AIDA (5626)	-	337 (6)	455 (8.1)	463 (8.2)	69 (1.2)	89 (1.6)	103 (1.8)	100 (1.8)	0 (0)	0 (0)
NEEL2014 (2373)	337 (14.2)	-	1690 (71.2)	1693 (71.3)	53 (2.2)	66 (2.8)	77 (3.2)	93 (3.9)	0 (0)	0 (0)
NEEL2015 (2799)	455 (16.3)	1690 (60.4)	-	2798 (99.9)	48 (1.7)	62 (2.2)	97 (3.5)	109 (3.9)	0 (0)	0 (0)
NEEL2016 (2991)	463 (15.5)	1693 (56.6)	2798 (93.5)	-	51 (1.7)	65 (2.1)	108 (3.6)	121 (4)	0 (0)	0 (0)
OKE2015 (520)	69 (13.3)	53 (10.2)	48 (9.2)	51 (9.8)	-	520 (100)	19 (3.7)	31 (6)	0 (0)	0 (0)
OKE2016 (650)	89 (13.7)	66 (10.2)	62 (9.5)	65 (10)	520 (80)	-	26 (4)	38 (5.8)	0 (0)	0 (0)
OKE2017 Task1 (364)	103 (28.3)	77 (21.2)	97 (26.6)	108 (29.7)	19 (5.2)	26 (7.1)	-	335 (92)	0 (0)	0 (0)
OKE2017 Task2 (463)	100 (21.6)	93 (20.1)	109 (23.5)	121 (26.1)	31 (6.7)	38 (8.2)	335 (72.4)	-	0 (0)	0 (0)
OKE2017 Task3 (3197)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	-	0 (0)
LeMonde (143)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	-

Table 6.3 – Entity overlap in the analyzed benchmark datasets. Behind the dataset name in each row the number of unique entities present in that dataset is given. For each datasets pair the overlap is given in number of entities and percentage (in parentheses).

Confusability. Let the true confusability of a surface form s be the number of meanings that this surface form can have. As new places, organizations and people are named every day, without access to an exhaustive collection of all named entities in the world, the true confusability of a surface form is unknown. However, we can estimate the confusability of a surface form through the function $A(s) : S \rightarrow \mathbb{N}$ that maps a surface form to an estimate of the size of its candidate mapping, such that $A(s) = |C(s)|$.

The confusability of, for example, a place name offers only a rough *a priori* estimate of how difficult it may be to disambiguate that surface form. Observation of annotated occurrences of this surface form in a text collection allows us to make more informed estimates. We show the average number of meanings denoted by a surface form, indicating the confusability, as well as complementary statistical measures on the datasets in Table 6.4. In this table, we observe that most datasets have a low number of average meanings per surface form, but there is a fair amount of variation, i.e. number of surface forms that can refer to a meaning. In particular, the OKEs datasets stand out in their high number of maximum meanings per surface form and standard deviations.

Dataset	Average	Min.	Max.	σ
AIDA	1.10	1	12	0.42
NEEL2014	1.02	1	3	0.16
NEEL2015	1.03	1	4	0.19
NEEL2016	1.03	1	4	0.20
OKE2015	1.12	1	30	1.49
OKE2016	1.14	1	33	1.69
OKE2017 Task1	1.01	1	5	0.09
OKE2017 Task2	1.01	1	3	0.11
OKE2017 Task3	1.03	1	4	0.18
LeMonde	1	1	264	0

Table 6.4 – Confusability stats for analyzed datasets. Average stands for average number of meanings per surface form, Min. and Max. stand for the minimum and maximum number of meanings per surface form found in the corpus respectively, and σ denotes the standard deviation.

Given a surface form, some senses are much more *dominant* than others – e.g. for the name 'berlin', the resource `dbpedia:Berlin` (Germany) is much more 'talked about' than `Berlin,_New_Hampshire` (USA). Therefore, we also take into account the *dominance*.

Dominance. Let the true dominance of a resource r_i for a given surface form s_i be a measure of how commonly r_i is meant with regard to other possible meanings when s_i is used in a sentence. Let the dominance estimate $D(r_i, s_i)$ be the relative

frequency with which the resource r_i appears in Wikipedia links where s_i appears as the anchor text. Formally:

$$D(r_i, s_i) = \frac{|WikiLinks(s_i, r_i)|}{\forall_{r \in R}|WikiLinks(s_i, r)|}$$

The dominance statistics for the analysed datasets are presented in Table 6.5. The dominance scores for all corpora are quite high and the standard deviation is low, meaning that in vast majority of cases, a single resource is associated with a certain surface form in the annotations, creating a low of variance for an automatic disambiguation system.

Corpus	Dominance	Max	Min	σ
AIDA	0.96	1	1	0.14
NEEL2014	0.99	1	1	0.08
NEEL2015	0.99	1	1	0.09
NEEL2016	0.99	1	1	0.09
OKE2015	0.99	1	1	0.10
OKE2016	0.98	1	1	0.11
OKE2017 Task1	1	2	1	0.05
OKE2017 Task2	1	2	1	0.05
OKE2017 Task3	0.99	1	1	0.08
LeMonde	1	1	1	0

Table 6.5 – Dominance stats for analyzed datasets.

Corpora that contain resources with high confusability and low dominance are considered more difficult to disambiguate. This is due to the fact that such corpora require a more careful examination of the context of each mention before algorithms can choose the most likely disambiguation. In cases with low confusability and high dominance, simple popularity-based baselines that ignore the context of the mention can already perform quite accurately.

Entity types. Table 6.1 shows that it is hard to build NER tools that perform well in all benchmarks. While for the other benchmarks, annotations of general concepts would be punished as false positives, NEEL2014 and OKE2017 Task 2 would expect them and punish their absence as false negatives.

Mention annotation characterization. When annotating entity mentions in a corpus, several either implicit or explicit decisions are being made by the dataset creators, that can influence evaluations on, and the comparison between, those datasets:

- **Mention boundaries**: inclusion of determiners (“the pope” vs “pope”), annotation of inner or outer entities (“New York’s airport JFK” vs “JFK”), tokenization decisions (“New York’s” vs “New York ’s”), sentence-splitting heuristics.

- **Handling redundancy**: annotating only the first vs. annotating all occurrences of an entity.
- **Inter-annotation agreement**: one annotator vs multiple annotators, low agreement vs high agreement.
- **Mention ambiguity**: when is there enough context for the entity to be considered non-ambiguous?
- **Offset calculation**: using 0 vs. using 1 as the initial identifier.
- **IRI vs. URI**: character support for ASCII vs. Unicode.
- **Nested entities**: does the annotation allow for annotation of multiple entity layers e.g. is 'The President of the United States of America' one entity in its entirety, two entity mentions ('President' and 'United States of America') or three ('President', 'United States of America', 'The President of the United States of America')?

In the analyzed datasets, there is only limited variety on the entity boundaries and offsets, but each dataset was generated using different annotation guidelines, resulting in major differences between types of classes annotated, and which entities are (not) to be included. The 2014/2015/2016 NEEL annotation guidelines, for example, are based on the CoNLL 2003 annotation guidelines (which also apply to AIDA-YAGO2) but where the CoNLL guidelines consider names of fictional characters as mentions of type *PERSON*, the NEEL guidelines consider this as a mention of type *CHARACTER*.

Many research papers treat benchmarks as black boxes, making it difficult to interpret efficacy improvements in terms of the individual contributions of algorithms and data. For system evaluations to provide generalisable insights, we must understand better the details of the entity linking task that a given dataset sets forth. Here we have analyzed a number of entity linking benchmark datasets where their characteristics are depicted in Table 6.1 and Table 6.2 that can help the community to interpret how such or such dataset is defined.

6.3 Experiments

We present in this section all the evaluations that we have done during this thesis. We split the evaluations in two part: one part being focus on Natural Language Processing tasks (chunking, part-of-speech tagging, named entity recognition and coreference resolution) in order to evaluate our deep-sequence-tagger and Sanaphor++ approaches detailed in Chapter 3, and finally we evaluate ADEL over well known entity linking benchmarks.

6.3.1 Chunking Evaluation

We evaluate our deep-sequence-tagger on the standard dataset CoNLL2000 on Chunking. We compare our approach with the current best published approach [197] on this dataset. We have tested multiple hyperparameters for this evaluation and we got the best results with a dropout of 0.25 before and after each Bidirectional-Long Short Term Memory layer and a NADAM optimizer As word embeddings approach we used GloVe with the 840B model. The training took around 2 hours with a NVIDIA K80 GPU.

Method	Accuracy
Zhai et al. [197]	94.72
deep-sequence-tagger	**96.4**

Table 6.6 – Chunking results over the CoNLL2000 dataset.

As shown in Table 6.6 our approach performs better than the last published work over the CoNLL2000 dataset by 1.68%. We explain this better result because in [197] they take the chunks as an ensemble of tokens and create a chunk representation with a Convolutional Neural Network which leads the network to be confused when the chunks are large (composed of at least three tokens). Contrarily to them, we have decided to use a Convolutional Neural Network for representing characters in order to have a more specific representation per word. Also, while they used a Bidirectional-Long Short Term Memory layer for encoding the sentence and a simple Long Short Term Memory layer for decoding it, we have decided to have two Bidirectional-Long Short Term Memory layers and a final Conditional Random Field layer.

6.3.2 Named Entity Recognition Evaluation

We evaluate our deep-sequence-tagger on the standard dataset CoNLL2002 and CoNLL2003 on named entity recognition. We compare our approach with the current best published approaches [88, 171] on these datasets. Each of these datasets are composed of 2 languages: Dutch and Spanish for CoNLL2002 and English and German for CoNLL2003. We used the exact same hyperparameters than for the CoNLL2000 and CoNLL2009 datasets. As word embeddings approach we used GloVe 840B model for English and the FastText Wikipedia models for the Spanish, Dutch and German languages. The training took around 6 hours with a NVIDIA K80 GPU for each language.

As shown in Table 6.7 our approach performs better than the last published work over the CoNLL2002 and CoNLL2003 datasets for all the languages. We explain our better performance over [88] because they use an Long Short Term Memory layer in order to represent a sequence of characters, while we used a Convolutional Neural Network, a Conditional Random Field inference based on [54] instead of the classic

	English	German	Dutch	Spanish
Lample et al. [88]	90.94	78.76	81.74	85.75
Tran et al. [171]	91.66	-	-	86.24
deep-sequence-tagger	**92.18**	**82.44**	**86.54**	**88.77**

Table 6.7 – Named Entity Recognition results over the CoNLL2002 and CoNLL2003 datasets. Numbers represent the respective F1 of each approach for a given language. Tran et al. [171] do not propose an evaluation for the German and Dutch languages.

Viterbi method, and a second Bidirectional-Long Short Term Memory layer. Next, we explain our better performance over [171] because they use an approach based on [88] where instead of seeing the sequence of layers Bidirectional-Long Short Term Memory layer and Conditional Random Field layers as the final architecture they have decided to stack them. It is true that stacking similar set of layers brings successful improvements in several tasks but it suffers of something called *degradation problem* due to the difficulty in training a lot of stacked layers when the amount of data are small. Basically, we are beating the approach in [171], mostly because the training datasets are small compare to what they should be to bring successful performances with stacked layers.

More fine grained results are given in Table 6.8, Table 6.9, Table 6.10 and Table 6.11. The first thing we can see is that entities of type *PERSON* is the most well recognized while the entities of type *MISC* is the less well recognized across all the languages. The entities of type *LOCATION* and *ORGANIZATION* are more or less well recognized across the languages. Except for English, the scores of the recall are lower than the precision, which means that for these languages there are a lot of false negative across all types.

Entity Types	Precision	Recall	F1
PERSON	97.03	97.76	97.40
LOCATION	91.77	93.87	92.81
ORGANIZATION	89.38	90.67	90.02
MISC	79.54	82.57	81.03
OVERALL	91.45	92.94	92.18

Table 6.8 – Fine grained results for the CoNLL2003 English dataset

6.3.3 Part-of-speech Tagging Evaluation

We evaluate our deep-sequence-tagger on the standard dataset CoNLL2009 on part-of-speech tagging. We compare our approach with the current best published approach [3] on this dataset, namely Google SyntaxNet. This dataset is composed of 7 languages: English, German, Catalan, Spanish, Chinese, Czech and Japanese. Be-

Entity Types	Precision	Recall	F1
PERSON	92.50	90.72	91.60
LOCATION	88.40	87.18	87.78
ORGANIZATION	74.85	77.27	76.04
MISC	72.51	62.41	67.08
OVERALL	83.61	81.42	82.44

Table 6.9 – Fine grained results for the CoNLL2003 German dataset

Entity Types	Precision	Recall	F1
PERSON	94.84	94.54	94.69
LOCATION	90.57	87.48	89
ORGANIZATION	83.49	84.72	84.10
MISC	83.84	72.45	77.73
OVERALL	88.35	84.96	86.54

Table 6.10 – Fine grained results for the CoNLL2002 Dutch dataset

Entity Types	Precision	Recall	F1
PERSON	95.62	95.76	95.69
LOCATION	87.24	86.37	86.80
ORGANIZATION	90.07	87.98	89.01
MISC	80.81	80.36	80.58
OVERALL	89.31	88.23	88.77

Table 6.11 – Fine grained results for the CoNLL2002 Spanish dataset

cause of licensing issue we did not succeed to get the Japanese dataset, so we have evaluated our approach only over the six other languages. We used the exact same hyperparameters than for the CoNLL2000 dataset. As word embeddings approach we used GloVe 840B model for English and the FastText Wikipedia models for the other languages. The training took between 6 hours and 2 days with a NVIDIA K80 GPU depending of the language.

Method	English	German	Catalan	Spanish	Chinese	Czech
Google SyntaxNet [3]	97.65	97.52	99.03	98.97	**94.72**	99.02
deep-sequence-tagger	**98.09**	**98.04**	**99.15**	**99.12**	88.47	**99.13**

Table 6.12 – Part-of-speech tagging results over the CoNLL2009. Numbers represent the respective accuracy of each approach for a given language.

As shown in Table 6.12 our approach performs better than the Google SyntaxNet approach for 5 over 6 languages. The explanation is because Google SyntaxNet takes the features on a window of 3 tokens centered at the current focus token: word, cluster and character n-gram up to length 3. They also extract the tag predicted for the previous 4 tokens. While in our case we take the tokens one by one without taking into account the tokens in a specific window, which explains why our approach gives a lower accuracy for Chinese, because this language highly depends on words that occur before and after the current token.

6.3.4 Coreference Resolution Evaluation

We evaluate Sanaphor++ on the standard dataset CoNLL2012 on Coreference Resolution [136], originally this dataset is split in three languages English, Arabic and Chinese languages, but we have decided to focus only on English and put the evaluation on the other languages as future work. We compare Sanaphor++ with the most recent version of Stanford deep-coref based on a deep reinforcement learning [30] (being the current best published coreference approach over the CoNLL2012 dataset) in Table 6.13. The Sanaphor++ model has been exported into a Stanford CoreNLP Framework compliant format, in order to be interoperable and foster the usage through the Stanford CoreNLP Framework. We have run the Sanaphor++ model and the Stanford deep-coref model over the CoNLL2012 test dataset. The CoNLL2012 dataset contains specific features for coreference such as speaker or document genre, as we plan to use Sanaphor++ for *real-world* usage where those features are not usually given, we have decided to train both approaches (Sanaphor++ and Stanford deep-coref) without taking into account those coreference specific features. This explain why the results reported in Table 6.13 for Stanford deep-coref are lower than from the published paper. As hyperparameters of the network we used the exact same ones than in the original Stanford deep-coref paper [31]. We update the network

policy by using RMSProp and apply dropout with a rate of 0.5 to the input layer. For the experiment, we initialize the mention-pair encoder component of the cluster ranking model with the learned weights from the mention-ranking model, which we find to greatly improve performance. A negative effect of adding our new features into the network is that it increases the training time from 3 days to 5 days over a NVIDIA K80 GPU.

	MUC			B^3			CEAF-E			
	P	R	F1	P	R	F1	P	R	F1	Avg F1
Sanaphor++	**65.81**	**74.65**	**69.95**	**58.84**	**62.37**	**60.55**	**52.47**	**58.64**	**55.39**	**61,96**
Stanford deep-coref	64.3	72.93	68.34	57.46	60.91	59.14	52.11	58.24	55	60,83

Table 6.13 – Sanaphor++ and Stanford deep-coref results over the CoNLL2012 dataset for the English language. **P** stands for **Precision** and **R** for **Recall**

As shown in Table 6.13, the new semantic logic brought by Sanaphor++ allows to compute a better mention-pair score, as all the scores are improved compared to Stanford deep-coref. While the results are promising, the new logic provides also wrong clusters, such as in the piece of text: *How does {the copyright thing} work on a Live365 stream? [...] About {the JW thing}, I work with a couple of them...*, the mentions *the copyright thing* and *the JW thing* have a high pair score since they share common surface forms and the same type, both are novel entities and have been typed to *THING*. The case also appears with ambiguous entities, such as in the piece of text: *It is she who asks if {Michael} and John could come too. [...] or that can't be tied to {Michael Jackson}*, the mentions *Michael* and *Michael Jackson* have a high pair score, because they share a common surface form, the same type and the same link. The first *Michael* refers to the character in Peter Pan which is completely different from the artist *Michael Jackson*. The problem is that *Michael* is highly ambiguous in this sentence.

6.3.5 Entity Linking Evaluation

Depending the guideline of a challenge we evaluate ADEL at different level:

- **extraction (Entity Recognition in GERBIL)**: the annotator gets a text and shall extract entities in this text.
- **recognition (RT2KB in GERBIL)**: the annotator gets a text and shall extract and type entities in this text.
- **typing (Entity Typing in GERBIL**: the annotator gets a text with the entities already extracted and shall give a proper type to these entities.
- **extraction+linking (A2KB in GERBIL)**: the annotator gets a text and

shall extract entities inside and link them to a knowledge base or to NIL if the entities do not have a corresponding entry in the knowledge base.

- **linking (D2KB in GERBIL)**: the annotator gets a text with the entities already extracted and shall link them to a knowledge base or to NIL if the entities do not have a corresponding entry in the knowledge base.

Our multiple linking approaches are evaluated only over the AIDA dataset and their evaluation over the other benchmark datasets is let as a future work. To this end, for all the datasets except AIDA, the linking approach for the extraction+linking or linking levels is always *Text Similarity Weighted With PageRank*.

We propose to evaluate several configurations of ADEL in order to show the powerness of its adaptability. Due to the high dimensionality of possible configurations we take only the combinations of extractors that are the most representative to properly evaluate ADEL. The named entity recognition model combination approach will be based on Stanford CoreNLP whereas the usage of only one model is done through deep-sequence-tagger, because the performances of deep-sequence-tagger is higher than Stanford CoreNLP and then using using one single model trained with Stanford CoreNLP will not bring anything compared to deep-sequence-tagger. The used part-of-speech tagger will be the one provided by Stanford CoreNLP because we did not implement the API over the part-of-speech version of deep-sequence-tagger.

To this end, we define an ADEL configuration as a combination of one or multiple of the following extractors:

- *MC* (named entity recognition model combination): Use one named entity recognition tagger with a model combination setting where the models are the 3 default Conditional Random Fields models provided by Stanford CoreNLP.
- *SM* (named entity recognition single model): Use one named entity recognition tagger with a model trained with the respective training data of the benchmark dataset via deep-sequence-tagger.
- *POS* (part-of-speech): Use one part-of-speech 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.
- *DT* (date): Use one named entity recognition tagger with a model specifically trained to recognize dates provided by Stanford CoreNLP.
- *NUM* (number): Use one named entity recognition tagger with a model specifically trained to recognize numbers provided by Stanford CoreNLP.
- *COREF* (coreference): Use Sanaphor++ via Stanford CoreNLP.

- *DIC* (dictionary): Use a dictionary respectively built for the corresponding benchmark dataset with DBpedia or Musicbrainz depending of the dataset.

The permanent GERBIL links for each run in this document are publicly available[5].

6.3.5.1 OKE 2015

	extraction			recognition			extraction+linking		
	P	R	F1	P	R	F1	P	R	F1
MC	89.95	57.4	70.08	51.94	51.94	51.94	40.91	26.11	31.87
SM	74.67	77.86	76.23	60.14	60.14	60.14	27.53	28.7	28.1
MC+SM	79.44	82.6	80.99	65.66	65.66	65.66	29.52	30.69	30.09
MC+POS	78.35	58.02	66.67	47.37	47.37	47.37	35.67	26.41	30.35
SM+POS	72.82	80.15	76.31	58.4	58.4	58.4	26.77	29.47	28.05
MC+SM+POS	74.65	80.92	77.66	60.84	60.84	60.84	27.32	29.62	28.42
POS	62.92	51.3	56.52	-	-	-	28.65	23.36	25.74
MC+COREF	91.83	72.06	80.75	51.94	51.94	51.94	38.33	30.08	33.7
SM+COREF	74.71	78.02	76.33	60.14	60.14	60.14	28.8	30.08	29.42
MC+SM+COREF	79.47	82.75	81.08	65.66	65.66	65.66	30.79	32.06	31.41
MC+POS+COREF	81.93	72.67	77.02	47.37	47.37	47.37	34.25	30.38	32.2
SM+POS+COREF	72.85	80.31	76.4	58.4	58.4	58.4	27.98	30.84	29.34
MC+SM+POS+COREF	74.68	81.07	77.75	60.84	60.84	60.84	28.55	30.99	29.72
POS+COREF	68.57	65.95	67.24	-	-	-	28.41	27.33	27.86
MC+DIC	97.24	83.05	89.84	77.66	77.66	77.66	43.88	37.25	40.3
SM+DIC	80.55	85.34	82.88	68.61	68.61	68.61	28.82	30.53	29.65
MC+SM+DIC	84.7	89.62	87.09	74.24	74.24	74.24	30.74	32.52	31.6
MC+POS+DIC	89.23	79.69	84.19	69.22	69.22	69.22	39.15	34.96	36.94
SM+POS+DIC	78.7	86.87	82.58	66.63	66.63	66.63	27.94	30.84	29.32
MC+SM+POS+DIC	79.86	87.18	83.36	68.71	68.71	68.71	28.39	30.99	29.64
POS+DIC	76.23	73.44	74.81	22.11	22.11	22.11	33.12	31.91	32.5
MC+COREF+DIC	**98.16**	**97.71**	**97.93**	**77.66**	**77.66**	**77.66**	**41.41**	**41.22**	**41.32**
SM+COREF+DIC	80.58	85.5	82.96	68.61	68.61	68.61	30.07	31.91	30.96
MC+SM+COREF+DIC	84.73	89.77	87.18	74.24	74.24	74.24	31.99	33.89	32.91
MC+POS+COREF+DIC	90.75	94.35	92.51	69.08	69.08	69.08	37.44	38.93	38.17
SM+POS+COREF+DIC	78.73	87.02	82.67	66.63	66.63	66.63	29.14	32.21	30.6
MC+SM+POS+COREF+DIC	79.89	87.33	83.44	68.71	68.71	68.71	29.61	32.37	30.93
POS+COREF+DIC	79.37	88.09	83.5	22.11	22.11	22.11	32.32	35.88	34.01
DIC	77.48	26.26	39.22	23.69	23.69	23.69	32.88	11.15	16.65
COREF	100	14.66	25.57	-	-	-	27.08	3.97	6.92

Table 6.14 – Results over the OKE2015 dataset at different levels for different ADEL configurations. Scores in bold represent the best ADEL configuration. **P** stands for **Precision** and **R** for **Recall**

Regarding the Table 6.14, it is interesting to notice that the deep-sequence-tagger extractor (SM) is the best single extractor setting for ADEL, whereas once we combine it with other extractor(s) (SM + extractor(s)) the results are always lower than with model combination extractor (MC) combined with the same other extractor(s) (MC +

[5] https://docs.google.com/spreadsheets/d/1Oaoc1mtmdX8HVzUUf7YG25GmB3I_iWOHMUbB5ligJvs/edit?usp=sharing

	extraction			typing			linking		
	P	R	F1	P	R	F1	P	R	F1
AGDISTIS	-	-	-	-	-	-	59.69	59.69	59.69
AIDA	68.48	61.92	65.04	-	-	-	71.36	47.18	56.8
Babelfy	30.46	48.49	37.42	-	-	-	75.8	50.69	60.75
DBpedia Spotlight	48.45	51.51	49.93	0	0	0	46.57	23.82	31.52
Dexter	54.73	44.38	49.02	-	-	-	91.88	32.82	48.37
Entityclassifier.eu	49.28	56.16	52.5	-	-	-	58.23	22.14	32.08
FOX	76.65	70.14	73.25	26.37	39.54	31.64	79.08	44.43	56.89
FRED	11.8	65.48	20	0	0	0	23.08	14.66	17.93
FREME	The annotator caused too many single errors.								
Kea	40.99	36.16	38.43	-	-	-	63.62	58.47	60.94
PBOH	-	-	-	-	-	-	**64.12**	**64.12**	**64.12**
TagMe 2	The annotator caused too many single errors.								
WAT	60.69	52.88	56.52	-	-	-	74.28	47.18	57.7
xLisa-NER	0	0	0	-	-	-	0	0	0
xLisa-NGRAM	0	0	0	-	-	-	0	0	0
ADEL	**98.16**	**97.71**	**97.93**	**79.08**	**79.08**	**79.08**	38.02	38.02	38.02

Table 6.15 – Comparison over the OKE2015 dataset. **P** stands for **Precision** and **R** for **Recall**

extractor(s)). This is due to a low amount of training data, the deep-sequence-tagger needs more data to performs better. Furthermore, having a dictionary can significantly improve the results (+13% in average). By analyzing the results, we have seen that the coreference tagger is very useful for extracting entities. We can also notice that the COREF extractor do not bring much to the SM extractor at extraction level, this is due to the fact that our model succeed to learn how to recognize a coreference, and properly type it, however, the coreference tagger is important as it links these mentions to their proper reference, what the other taggers cannot do because it is not possible for such taggers 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.*, another 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. However, we notice that in Table 6.14 the recognition scores (precisions, recall and f1) given by GERBIL are always exactly the same. In order to understand better why this occur, we have decided to evaluate ADEL with another scorer (namely neleval) with one config (namely *SM*) in order to possibly explain this. Indeed, with neleval, the extraction and extraction+linking scores are exactly the same than the ones given by GERBIL but the recognition scores for the *SM* config are much bigger: from P=R=F1=59.79 (with GERBIL) to P=71.7 R=76.5 F1=74 (with neleval). The recognition scores given by neleval show that GERBIL has a bug for its recognition scores, and they are working on a fix[6].

[6] https://github.com/dice-group/gerbil/issues/226

Despite this, we have decided to stick with GERBIL as scorer because we are not able to score all the other systems with neleval. To this end, we remind the readers that the recognition scores given by GERBIL might not represent the full potential of our approach. This remark is also true for all the following evaluations. Finally, the linking score drops a lot, this is due to the used linking approach that has two noticeable issues: *1)* the string similarity that always promotes the shortest string (the string distance score over the title, the redirect and the disambiguation pages between the mention *GM* and the entity candidate *db:Germany* (0.32879817) is higher than with the entity candidate *db:General_Motors* (0.21995464)), and *2)* the pagerank that always promotes the most popular entity despite a low string similarity score (the mention *Harry Potter* will always be linked to *db:Harry_Potter* and never to *db:Harry_Potter_(film_series)*, *db:Harry_Potter_(character)* or *db:Harry_Potter_(journalist)*). The same remark for the linking can be applied for any of the following evaluations.

In Table 6.15 we can see that ADEL has the best extraction and recognition score while PBOH has the best linking score.

6.3.5.2 OKE 2016

As we can see in Table 6.16, the comments we have done for the OKE2015 dataset are also valid for the OKE2016 dataset (they both share the exact same guideline). The main difference between the two is a better performance of the settings implying the SM extractor, this can be explained by a bigger training dataset, which means that the bigger is the training set the better will be the combinations implying the *SM* extractor. To this end, we notice that the best recognition setting implies the SM extractor (MC+SM+DIC) and is different from the best extraction and extraction + linking setting (MC+COREF+DIC). Nevertheless, these two different ADEL setting beat all the other systems in Table 6.17 in term of extraction and recognition, and once again PBOH has the best linking results.

6.3.5.3 OKE 2017 Task1

The SM extractor cannot be used here because recognition is not part of the guideline and then we do not have a training set that includes the entity types. The Table 6.18 shows a really good extraction score for the *MC + DIC* ADEL setting compared to the other systems (or ADEL setting) while it is also interesting to notice that with a simple part-of-speech extractor we can reach a score at extraction level that beats most of the systems listed in Table 6.19. For this dataset, AIDA gives the best results for extraction + linking, and PBOH, once again, get the best results at linking.

	extraction			recognition			extraction+linking		
	P	R	F1	P	R	F1	P	R	F1
MC	88.27	48.77	62.82	42.32	42.32	42.32	34.64	19.14	24.65
SM	82.93	83.95	83.44	68.95	68.95	68.95	25	25.31	25.15
MC+SM	85.11	86.42	85.76	72.39	72.39	72.39	25.84	26.23	26.03
MC+POS	17.31	88.27	28.95	11.95	11.95	11.95	5.33	27.16	8.91
SM+POS	78.51	84.57	81.43	65.16	65.16	65.16	24.07	25.93	24.96
MC+SM+POS	79.77	85.19	82.39	67.51	67.51	67.51	24.28	25.93	25.07
POS	14.56	77.47	24.51	-	-	-	4.16	2.16	6.84
MC+COREF	90.23	59.88	71.99	42.32	42.32	42.32	34.42	22.84	27.46
SM+COREF	82.93	83.95	83.44	68.95	68.95	68.95	26.52	26.85	26.69
MC+SM+COREF	85.11	86.42	85.76	72.39	72.39	72.39	27.36	27.78	27.57
MC+POS+COREF	17.31	88.27	28.95	11.95	11.95	11.95	5.51	28.09	9.21
SM+POS+COREF	78.51	84.57	81.43	65.16	65.16	65.16	25.5	27.47	26.45
MC+SM+POS+COREF	79.77	85.19	82.39	67.51	67.51	67.51	25.14	26.85	25.97
POS+COREF	14.56	77.47	24.51	-	-	-	4.23	22.53	7.13
MC+DIC	98.93	85.8	91.9	80.12	80.12	80.12	34.88	30.25	32.4
SM+DIC	88.24	92.59	90.36	78.85	78.85	78.85	25.59	26.85	26.2
MC+SM+DIC	90.06	95.06	92.49	**82.4**	**82.4**	**82.4**	26.32	27.78	27.03
MC+POS+DIC	19.5	96.6	32.45	15.59	15.59	15.59	6.04	29.94	10.06
SM+POS+DIC	85.31	93.21	89.09	75.8	75.8	75.8	25.14	27.47	26.25
MC+SM+POS+DIC	86.36	93.83	89.94	78.23	78.23	78.23	25.28	27.47	26.33
POS+DIC	16.64	8.58	27.87	7.05	7.05	7.05	4.73	24.38	7.92
MC+COREF+DIC	**98.42**	**96.3**	**97.35**	79.03	79.03	79.03	**34.7**	**33.95**	**34.32**
SM+COREF+DIC	88.24	92.59	90.36	78.85	78.85	78.85	27.06	28.4	27.71
MC+SM+COREF+DIC	90.06	95.06	92.49	**82.4**	**82.4**	**82.4**	27.19	28.7	27.93
MC+POS+COREF+DIC	19.5	96.6	32.45	15.59	15.59	15.59	6.23	30.86	10.37
SM+POS+COREF+DIC	85.31	93.21	89.09	75.8	75.8	75.8	25.99	28.4	27.14
MC+SM+POS+COREF+DIC	86.36	93.83	89.94	78.23	78.23	78.23	26.14	28.4	27.22
POS+COREF+DIC	16.64	8.58	27.87	7.05	7.05	7.05	4.91	25.31	8.22
DIC	80.54	37.04	50.74	33.71	33.71	33.71	24.83	11.42	15.64
COREF	100	11.11	20	-	-	-	33.33	3.7	6.67

Table 6.16 – Results over the OKE2016 dataset at different levels for different ADEL configurations. Scores in bold represent the best ADEL configuration. **P** stands for **Precision** and **R** for **Recall**

	extraction			typing			linking		
	P	R	F1	P	R	F1	P	R	F1
AGDISTIS	-	-	-	-	-	-	53.4	53.4	53.4
AIDA	84.71	44.44	58.3	-	-	-	66.83	42.28	51.8
Babelfy	45.98	44.14	45.04	-	-	-	63.82	48.46	55.09
DBpedia Spotlight	57.65	34.88	43.46	0	0	0	56.98	31.48	40.56
Dexter	67.53	32.1	43.51	-	-	-	92	28.4	43.4
Entityclassifier.eu	50.39	39.51	44.29	-	-	-	62.5	24.69	35.4
FOX	87.13	45.99	60.2	20.99	34.57	26.12	81.41	39.2	52.92
FRED	31.42	76.23	44.5	0	0	0	21.29	16.36	18.5
FREME	The annotator caused too many single errors.								
Kea	49.58	54.32	51.84	-	-	-	66.32	59.57	62.76
PBOH	-	-	-	-	-	-	**67.59**	**67.59**	**67.59**
TagMe 2	The annotator caused too many single errors.								
WAT	72.73	44.44	55.17	-	-	-	73.28	52.47	61.15
xLisa-NER	0	0	0	-	-	-	0	0	0
xLisa-NGRAM	0	0	0	-	-	-	0	0	0
Challenge Winner	74.03	81.05	77.38	63.07	62.58	62.83	71.82	51.63	60.08
ADEL	**98.42**	**96.3**	**97.35**	**91.05**	**91.05**	**91.05**	30.86	30.86	30.86

Table 6.17 – Comparison over the OKE2016 dataset. **P** stands for **Precision** and **R** for **Recall**
. The extraction score for ADEL correspond to the MC+COREF+DIC setting, and the typing score correspond to the MC+SM+DIC setting

	extraction			extraction + linking		
	P	R	F1	P	R	F1
MC	78.51	72.05	75.14	27.16	24.93	26
MC+POS	60.95	73.97	66.83	19.86	24.11	21.78
POS	58.2	70.96	63.95	18.65	22.74	20.49
MC+DIC	**92.47**	**94.25**	**93.35**	**31.18**	**31.78**	**31.48**
MC+POS+DIC	73.44	90.14	80.93	23.88	29.32	26.32
POS+DIC	71.11	87.67	78.53	23.11	28.49	25.52
DIC	91.49	23.56	37.47	27.66	7.12	11.33

Table 6.18 – Results over the OKE2017 Task1 dataset at different levels for different ADEL configurations. Scores in bold represent the best ADEL configuration. **P** stands for **Precision** and **R** for **Recall**

	extraction			linking			extraction + linking		
	P	R	F1	P	R	F1	P	R	F1
AGDISTIS	-	-	-	64.93	64.93	64.93	-	-	-
AIDA	68.48	61.92	65.04	78.28	57.26	66.14	**54.37**	**46.03**	**49.85**
Babelfy	30.46	48.49	37.42	73.28	49.59	59.15	24.1	38.36	29.6
DBpedia Spotlight	48.45	51.51	49.93	68.55	29.86	41.6	42.01	44.66	43.29
Dexter	54.73	44.38	49.02	90.13	37.53	53	50.18	37.53	42.95
Entityclassifier.eu	49.28	56.16	52.5	56.1	31.51	40.35	27.64	31.51	29.45
FOX	76.65	70.14	73.25	68.5	47.67	56.22	51.8	47.4	49.5
FRED	11.85	65.75	20.08	33.74	22.47	26.97	4	22.19	6.78
FREME	The annotator caused too many single errors.								
Kea	41.14	35.62	38.18	71.72	58.36	64.35	45.73	29.32	35.73
PBOH	-	-	-	**69.32**	**69.32**	**69.32**	-	-	-
TagMe 2	The annotator caused too many single errors.								
WAT	60.69	52.88	56.52	78.01	51.51	62.05	52.5	40.27	45.58
xLisa-NER	0	0	0	0	0	0	0	0	0
xLisa-NGRAM	0	0	0	0	0	0	0	0	0
Task Winner [164]	95.90	65.80	78.05	61.96	41.47	49.69	56.15	38.53	45.70
ADEL	**92.47**	**94.25**	**93.35**	33.42	33.42	33.42	31.18	31.78	31.48

Table 6.19 – Comparison over the OKE2017 Task1 dataset. **P** stands for **Precision** and **R** for **Recall**

	extraction			extraction + linking		
	P	R	F1	P	R	F1
MC	74.23	62.34	67.76	32.22	27.06	29.41
MC+POS	66.74	69.05	67.87	28.66	29.65	29.15
POS	67.67	63.42	65.47	29.33	27.49	28.38
MC+DIC	**88.87**	**95.02**	**91.84**	**38.4**	**41.13**	**39.75**
MC+POS+DIC	81.25	92.86	86.67	34.66	39.61	36.97
POS+DIC	82.26	88.31	85.18	34.88	37.45	36.12
DIC	92.53	34.85	50.63	38.51	14.5	21.07

Table 6.20 – Results over the OKE2017 Task2 dataset at different levels for different ADEL configurations. Scores in bold represent the best ADEL configuration. **P** stands for **Precision** and **R** for **Recall**

	extraction			linking			extraction + linking		
	P	R	F1	P	R	F1	P	R	F1
AGDISTIS	-	-	-	58.01	58.01	58.01	-	-	-
AIDA	74.61	51.52	60.95	70.91	55.41	62.21	57.89	38.1	45.95
Babelfy	39.08	49.57	43.7	67.34	50.43	57.67	30.68	35.06	32.73
DBpedia Spotlight	60.79	53.03	56.65	70.05	32.9	44.77	**51.86**	**45.24**	**48.32**
Dexter	65.1	41.99	51.05	87.57	35.06	50.08	57.71	34.85	43.45
Entityclassifier.eu	60.77	54.33	57.37	56.18	30.52	39.55	34.14	30.52	32.23
FOX	81.25	56.28	66.5	70.31	38.96	50.14	56.25	38.96	46.04
FRED	16.07	67.75	25.98	27.07	18.4	21.91	4.31	18.18	6.97
FREME	The annotator caused too many single errors.								
Kea	44.9	42.86	43.85	75.06	63.85	69.01	42.01	37.01	39.36
PBOH	-	-	-	**68.18**	**68.18**	**68.18**	-	-	-
TagMe 2	The annotator caused too many single errors.								
WAT	63.01	49.78	55.62	73.67	53.9	62.25	58.91	35.06	43.96
xLisa-NER	0	0	0	0	0	0			
xLisa-NGRAM	0	0	0	0	0	0			
Task Winner [164]	95.90	65.80	78.05	63.42	35.28	45.34	56.15	38.53	45.70
ADEL	**88.87**	**95.02**	**91.84**	43.72	43.72	43.72	38.4	41.13	39.75

Table 6.21 – Comparison over the OKE2017 Task2 dataset. **P** stands for **Precision** and **R** for **Recall**

6.3.5.4 OKE 2017 Task2

The Task2 shares the same requirement that the Task1 about the recognition, then once again, no SM extractor. The Table 6.20 shows similar results than for the OKE2017 Task1 dataset, and the observations over ADEL we can do are exactly the same as well. Table 6.21 shows for this dataset that DBpedia Spotlight gives the best results for extraction + linking, and Kea, get the best results at linking.

6.3.5.5 OKE 2017 Task3

Task 3 provides fine-grained and specific entity types (artists, songs and albums), which brings a major issue: the name of an artist, a song or an album can be anything and include, for instance, a punctuation mark[7]. In this dataset the entities are typed with the Music Ontology types, and linked to Musicbrainz entries. The Table 6.22 shows how much the SM extractor can perform well over formal textual content with a training set having a decent size. The training set is 100 documents and each of them has an average of 400 tokens. The compared systems in Table 6.23 (except FOX) are not able to handle these types or knowledge base, but GERBIL has a mapping feature that succeed to map the DBpedia links to their Musicbrainz equivalent and same thing for the DBpedia types to the Music Ontology types. To this end, it is possible to score these systems against this dataset.

This time the SM extractor brings the best performance, and the results are even

[7] https://musicbrainz.org/work/25effd3c-aada-44d1-bcbf-ede30ef34cc0

	extraction			recognition			extraction + linking		
	P	R	F1	P	R	F1	P	R	F1
MC	82.55	48.01	60.71	43.25	43.25	43.25	82.55	48.01	60.71
SM	83.28	90.14	86.57	76.32	76.32	76.32	83.28	90.14	86.57
MC+SM	80.65	91.67	85.81	75.14	75.14	75.14	80.65	91.67	85.81
MC+POS	54.4	71.5	61.79	30.17	30.17	30.17	54.4	71.5	61.79
SM+POS	63.61	92.92	75.52	59.43	59.43	59.43	63.61	92.92	75.52
MC+SM+POS	63.88	92.92	75.71	60.24	60.24	60.24	63.88	92.92	75.71
POS	52.57	69.14	59.72	-	-	-	52.57	69.14	59.72
MC+DIC	84.79	60.02	70.29	54.19	54.19	54.19	84.79	60.02	70.29
SM+DIC	**85.04**	**93.95**	**89.27**	**80.63**	**80.63**	**80.63**	**85.04**	**93.95**	**89.27**
MC+SM+DIC	82.23	95.11	88.21	78.9	78.9	78.9	82.23	95.11	88.21
MC+POS+DIC	57.77	77.34	66.14	38.53	38.53	38.53	57.77	77.34	66.14
SM+POS+DIC	64.91	95.15	77.17	62.28	62.28	62.28	64.91	95.15	77.17
MC+SM+POS+DIC	65.13	95.11	77.32	62.86	62.86	62.86	65.13	95.11	77.32
POS+DIC	56.08	75.23	64.26	10.06	10.06	10.06	56.08	75.23	64.26
DIC	82.63	14.58	24.79	14.15	14.15	14.15	82.63	14.58	24.79

Table 6.22 – Results over the OKE2017 Task3 dataset at different levels for different ADEL configurations. Scores in bold represent the best ADEL configuration. **P** stands for **Precision** and **R** for **Recall**

	extraction			recognition			linking		
	P	R	F1	P	R	F1	P	R	F1
AGDISTIS	-	-	-	-	-	-	31.69	31.69	31.69
AIDA	57.74	38.15	45.95	-	-	-	0	0	0
Babelfy	39.29	43.37	41.23	-	-	-	0.99	0.7	0.82
DBpedia Spotlight	49.25	52.03	50.6	4.53	4.53	4.53	0	0	0
Dexter	56.87	49.92	53.17	-	-	-	0	0	0
Entityclassifier.eu	38.55	43.16	40.73	-	-	-	0	0	0
FOX	63.02	49.21	55.27	0	0	0	22.07	10.85	14.55
FRED	The annotator caused too many single errors.								
FREME	The annotator caused too many single errors.								
Kea	0	0	0	-	-	-	0	0	0
PBOH	The annotator caused too many single errors.								
TagMe 2	The annotator caused too many single errors.								
WAT	The annotator caused too many single errors.								
xLisa-NER	0	0	0	-	-	-	0	0	0
xLisa-NGRAM	0	0	0	-	-	-	0	0	0
ADEL	**85.04**	**93.95**	**89.27**	**80.63**	**80.63**	**80.63**	**100**	**99.96**	**99.98**

Table 6.23 – Comparison over the OKE2017 Task3 dataset. **P** stands for **Precision** and **R** for **Recall**

better when it is combined to the dictionary. Furthermore, our linking formula that promotes the popular entities works very well for the Musicbrainz knowledge base, and largely outperforms any other method used in the compared systems, and then ADEL becomes the best system for this dataset.

6.3.5.6 NEEL2014

	extraction			extraction + linking		
	P	R	F1	P	R	F1
MC	66.08	30.68	41.91	29.2	13.56	18.52
MC+POS	59.33	66.23	62.59	21.84	24.38	23.04
POS	58.47	62.67	60.5	22.49	24.11	23.17
MC+DIC	75.37	52.4	61.82	31.53	21.92	25.86
MC+POS+DIC	**66.99**	**89.93**	**76.78**	24.69	33.15	28.3
POS+DIC	66.58	87.6	75.66	**25.09**	**33.01**	**28.51**
MC+DT	48.52	30.41	37.39	21.53	13.49	16.59
MC+POS+DT	51.23	64.32	57.03	19.09	23.97	21.26
POS+DT	50.56	61.71	55.58	19.36	23.63	21.28
MC+DT+DIC	60.75	51.85	55.95	25.68	21.92	23.65
MC+POS+DT+DIC	59.42	87.74	70.85	22.17	32.74	26.44
POS+DT+DIC	59.18	86.3	70.21	22.31	32.53	26.47
MC+NUM	57.49	40.48	47.51	22.28	15.68	18.41
MC+POS+NUM	55.26	65.14	59.79	20.4	24.04	22.07
POS+NUM	54.59	61.92	58.02	20.95	23.77	22.27
MC+NUM+DIC	67.09	62.12	64.51	25.96	24.04	24.96
MC+POS+NUM+DIC	63.31	88.77	73.91	23.4	32.81	27.32
POS+NUM+DIC	62.97	86.78	72.98	23.71	32.67	27.48
MC+NUM+DT	48.82	39.73	43.81	19.19	15.62	17.22
MC+POS+NUM+DT	50	64.25	56.24	18.66	23.97	20.98
POS+NUM+DT	49.18	61.58	54.68	18.87	23.63	20.99
MC+NUM+DT+DIC	59.26	61.16	60.2	23.22	23.97	23.59
MC+POS+NUM+DT+DIC	58.25	87.74	70.02	21.74	32.74	26.13
POS+NUM+DT+DIC	57.95	86.16	69.29	21.88	32.53	26.16
NUM	38.92	11.3	17.52	8.49	2.47	3.82
DT	11	2.26	3.75	1.67	0.34	0.57
DIC	79.57	22.67	35.29	31.01	8.84	13.75

Table 6.24 – Results over the NEEL2014 dataset at different levels for different ADEL configurations. Scores in bold represent the best ADEL configuration. **P** stands for **Precision** and **R** for **Recall**

Contrarily to the previous datasets, NEEL2014 is on tweets and not on encyclopedic or news documents which increases the difficulty of entity extraction, recognition and linking. Furthermore, this dataset is particularly difficult because the guideline ask to the systems to extract and link (but not type) all the entities (including the hashtags and the user mentions) that have an entry in DBpedia does not matter to what they correspond (animals, dates, numbers, books, etc.) but not the novel entities. According to the guideline, an entity can also be a mix of usual surface form with a hashtag or a user mention, for instance *season 2 of #Sherlock* has to be taken as one

	extraction			linking			extraction + linking		
	P	R	F1	P	R	F1	P	R	F1
AGDISTIS	-	-	-	33.7	33.7	33.7	-	-	-
AIDA	77.86	22.88	35.36	62.29	33.15	43.27	63.97	17.88	27.94
Babelfy	32.92	32.47	32.69	63.38	38.29	47.74	29.13	19.93	23.67
DBpedia Spotlight	54.81	41.37	47.15	72.42	40.82	52.21	46.46	35.07	39.97
Dexter	50.16	31.78	38.91	84.05	26.71	40.54	42.16	26.71	32.7
Entityclassifier.eu	40.77	45.68	43.09	67.92	31.03	42.6	27.69	31.03	29.26
FOX	73	23.15	35.15	68.1	16.23	26.22	49.68	15.75	23.92
FRED	6.78	56.16	12.1	4.99	2.81	3.59	0.34	2.81	0.61
FREME	The annotator caused too many single errors.								
Kea	27.11	40.68	32.54	72.93	62.19	67.13	22.01	32.6	26.28
PBOH	-	-	-	**72.74**	**72.74**	**72.74**	-	-	-
TagMe 2	The annotator caused too many single errors.								
WAT	71.7	26.03	38.19	0	0	0	56.98	20.68	30.35
xLisa-NER	0	0	0	0	0	0	0	0	0
xLisa-NGRAM	0	0	0	0	0	0	0	0	0
Challenge Winner	-	-	-	-	-	-	**77.1**	**64.2**	**70.06**
ADEL	**66.58**	**87.6**	**75.66**	36.92	36.92	36.92	25.09	33.01	28.51

Table 6.25 – Comparison over the NEEL2014 dataset. **P** stands for **Precision** and **R** for **Recall**

single entity. As there is no typing, it is not possible for us to train a named entity recognition model with the training set, which makes the part-of-speech extractor becoming an important extractor as we can see in Table 6.24. We have used a number extractor (NUM) and a date extractor (DT) through a named entity recognition model trained with the MUC-7 dataset provided by default in Stanford CoreNLP, and this approach do not bring anything due to the difficulty for the named entity recognition extractor to parse tweets, including the model combination extractor that has difficulties to perform well alone for the same reasons. During this evaluation we have noticed a new bug in GERBIL that had difficulties to properly handle tweets containing emoticons, which might biased the results of ADEL and of the other systems, and they are working on a fix[8]. To this end, all the results over tweets dataset are done with this bug. Despite this comment, the best setting implies the model combination and the dictionary extractors such as for the other datasets, but this time with the addition of the part-of-speech extractor. Another thing to notice is that the best ADEL setting for extraction + linking is not the same than extraction because the model combination extractor brings too much false positive links. Once again, as shown is Table 6.25, ADEL has the best extraction process, while the system that has the best extraction + linking is the official challenge winner. Finally, once again PBOH gets the best linking score.

	extraction			recognition			extraction + linking		
	P	R	F1	P	R	F1	P	R	F1
MC	85.03	30.34	44.72	28.8	28.8	28.8	28.85	10.35	15.23
SM	81.71	70.34	75.6	60.77	60.77	60.77	25.34	21.81	23.45
MC+SM	82.33	74.9	78.44	64.53	64.53	64.53	24.94	22.7	23.77
MC+POS	87.65	72.3	79.23	65.61	65.61	65.61	23.45	19.31	21.18
SM+POS	80.46	77.48	78.94	65.21	65.21	65.21	23.19	22.34	22.76
MC+SM+POS	**87.29**	**88.22**	**87.75**	**78.18**	**78.18**	**78.18**	22.95	23.2	23.07
POS	89.53	41.88	57.07	-	-	-	19.39	9.07	12.36
MC+DIC	87.18	43.08	57.67	40.51	40.51	40.51	31.43	15.53	20.79
SM+DIC	84.14	77.72	80.8	67.79	67.79	67.79	26.92	24.86	25.85
MC+SM+DIC	84.52	81.7	83.09	71.07	71.07	71.07	**26.6**	**25.72**	**26.16**
MC+POS+DIC	88.5	86.66	87.57	77.87	77.87	77.87	25.69	25.15	25.41
SM+POS+DIC	82.8	85.82	84.28	72.84	72.84	72.84	24.76	25.67	25.21
MC+SM+POS+DIC	83.17	89.76	86.34	75.96	75.96	75.96	24.54	26.48	25.47
POS+DIC	88.16	59.6	71.12	55.19	55.19	55.19	25.13	16.99	20.28
DIC	84.25	16.03	26.93	15.56	15.56	15.56	38.9	7.4	12.44

Table 6.26 – Results over the NEEL2015 dataset at different levels for different ADEL configurations. Scores in bold represent the best ADEL configuration. **P** stands for **Precision** and **R** for **Recall**

.

	extraction			recognition			extraction + linking		
	P	R	F1	P	R	F1	P	R	F1
AIDA	88.24	26.79	41.1	-	-	-	68.68	20.17	31.18
Babelfy	30.52	27.21	28.77	-	-	-	21.83	17.49	19.42
DBpedia Spotlight	56.05	36.59	44.28	-	-	-	49.38	32.24	39.01
Dexter	56.45	25.91	35.51	-	-	-	49.57	22.75	31.19
Entityclassifier.eu	40.4	36.02	38.08	-	-	-	26.19	23.35	24.69
FOX	83.53	28.15	42.11	-	-	-	57.15	19.26	28.81
FRED	10.34	47.17	16.96	-	-	-	3.95	18.03	6.49
FREME	The annotator caused too many single errors.								
Kea	29.58	34.43	31.82	-	-	-	24.11	25.46	24.77
TagMe 2	The annotator caused too many single errors.								
WAT	77.94	29.37	42.67	-	-	-	56.3	21.19	30.79
xLisa-NER	0	0	0	-	-	-	0	0	0
xLisa-NGRAM	0	0	0	-	-	-	0	0	0
Challenge Winner	-	-	-	**85.7**	**76.1**	**80.7**	**81**	**71.9**	**76.2**
ADEL	**87.29**	**88.22**	**87.75**	(78.18)	(78.18)	(78.18)	26.6	25.72	26.16

Table 6.27 – Comparison over the NEEL2015 dataset. **P** stands for **Precision** and **R** for **Recall**
. The extraction and recognition scores for ADEL correspond to the MC+SM+POS setting, and the extraction+linking score correspond to the MC+SM+DIC setting

6.3.5.7 NEEL2015

The Table 6.26 reveals, such as OKE2016 dataset, that the best setting might be different depending of which level we want to have the best results, however in both we can see the combination of the SM and MC extractors. For the best extraction and recognition it is MC+SM+POS setting, while for the best extraction + linking it is MC+SM+DIC. Despite that the SM extractor belongs to the two best settings, we can see that its performance are much lower over tweets than over formal texts, which means that the network has difficulties to handle tweets. The training set of NEEL2015 contains over 3000 tweets, which is a decent number of examples to train such network but apparently, after analyzing the results, it is difficult for the network to properly identify entities that are hashtags or user mentions. Nevertheless, in the Table 6.27 we can see that we still do better at extraction than the other systems, and we put the recognition scores between parenthesis because of the identified bug in GERBIL that influence the recognition score, which means that we might do better than the challenge winner as well.

6.3.5.8 NEEL2016

	extraction			recognition			extraction + linking		
	P	R	F1	P	R	F1	P	R	F1
MC	79.52	6.3	11.67	6.2	6.2	6.2	26.19	2.1	3.89
SM	82.84	55.73	66.63	49.96	49.96	49.96	26.73	18.03	21.54
MC+SM	82.61	56.2	66.89	50.26	50.26	50.26	26.79	18.23	21.69
MC+POS	95.6	70.42	81.1	68.21	68.21	68.21	41.85	31.11	35.69
SM+POS	84.86	72.71	78.31	64.3	64.3	64.3	29.51	25.29	27.24
MC+SM+POS	84.64	73.09	78.44	64.53	64.53	64.53	29.61	25.57	27.44
POS	97.69	64.6	77.77	-	-	-	44.01	29.1	35.04
MC+DIC	91.45	20.42	33.39	20.04	20.04	20.04	28.21	6.3	10.3
SM+DIC	86.36	65.84	74.72	59.64	59.64	59.64	27.91	21.28	24.15
MC+SM+DIC	85.88	65.55	74.35	59.17	59.17	59.17	27.87	21.28	24.13
MC+POS+DIC	**96.34**	**92.94**	**94.61**	**89.77**	**89.77**	**89.77**	**37.98**	**36.64**	**37.3**
SM+POS+DIC	87.64	84.54	86.06	75.53	75.53	75.53	30.2	29.1	29.64
MC+SM+POS+DIC	87.27	84.35	85.78	75.11	75.11	75.11	30.21	29.2	29.69
POS+DIC	96.99	89.22	92.94	86.82	86.82	86.82	38.38	35.31	36.78
DIC	93.51	16.51	28.06	16.32	16.32	16.32	27.03	4.77	8.11

Table 6.28 – Results over the NEEL2016 dataset at different levels for different ADEL configurations. Scores in bold represent the best ADEL configuration. **P** stands for **Precision** and **R** for **Recall**

The NEEL2015 and NEEL2016 datasets share the same guideline. The main characteristic of the NEEL2016 test dataset is that it contains a majority of hashtags and user mentions annotated as entities in the test dataset. Therefore, this explains why the SM and MC extractors are performing so badly and why the POS extractor is

[8] https://github.com/dice-group/gerbil/issues/227

	extraction			recognition			extraction + linking		
	P	R	F1	P	R	F1	P	R	F1
AIDA	84.62	5.25	9.88	-	-	-	59.65	3.24	6.15
Babelfy	22.47	9.73	13.58	-	-	-	16.08	6.97	9.72
DBpedia Spotlight	44.25	17.27	24.85	-	-	-	34.72	13.55	19.49
Dexter	46.28	10.69	17.36	-	-	-	44.34	8.97	14.92
Entityclassifier.eu	38.6	28.91	33.06	-	-	-	21.27	15.94	18.22
FOX	75	5.44	10.14	-	-	-	43.42	5.87	3.15
FRED	10.42	27.67	15.14	-	-	-	3.05	8.11	4.44
FREME	The annotator caused too many single errors.								
Kea	41.41	54.77	47.17	-	-	-	30.88	40.84	35.17
TagMe 2	The annotator caused too many single errors.								
WAT	The annotator caused too many single errors.								
xLisa-NER	0	0	0	-	-	-	0	0	0
xLisa-NGRAM	0	0	0	-	-	-	0	0	0
Challenge Winner	-	-	-	45.3	49.4	47.3	**45.4**	**56**	**50.1**
ADEL	**96.34**	**92.94**	**94.61**	**89.77**	**89.77**	**89.77**	37.98	36.64	37.3

Table 6.29 – Comparison over the NEEL2016 dataset. **P** stands for **Precision** and **R** for **Recall**

performing so well as showed in Table 6.28. Once again ADEL performs the best at extraction and recognition level compared to any other systems proposed in Table 6.29.

6.3.5.9 AIDA

	extraction			extraction + linking		
	P	R	F1	P	R	F1
MC	72.77	92.91	81.62	23.02	29.39	25.82
SM	**77.61**	**92.69**	**84.48**	**24.68**	**29.48**	**26.87**
MC+SM	74.1	93.47	82.67	23.81	30.03	26.56
MC+POS	63.58	85.6	72.96	20.67	27.83	23.72
SM+POS	64.15	85.02	73.12	20.93	27.74	23.86
MC+SM+POS	63.59	85.37	72.89	20.66	27.74	23.68
POS	59.17	72.75	65.26	20.67	25.42	22.8
MC+DIC	67.38	92.91	78.11	21.31	29.39	24.71
SM+DIC	71.46	92.8	80.74	22.7	29.48	25.65
MC+SM+DIC	68.6	93.58	79.17	22.03	30.06	25.43
MC+POS+DIC	59.92	85.57	70.49	19.48	27.83	22.92
SM+POS+DIC	60.41	84.99	70.63	19.71	27.74	23.05
MC+SM+POS+DIC	60.16	84.41	70.25	19.58	27.47	22.86
POS+DIC	55.43	72.73	62.91	19.37	25.42	21.99
DIC	0.73	0.09	0.16	0	0	0

Table 6.30 – Results over the AIDA dataset at different levels for different ADEL configurations. Scores in bold represent the best ADEL configuration. **P** stands for **Precision** and **R** for **Recall**

We had an hardware difficulty with the AIDA dataset, the size of the documents (an average of 600 tokens) has raised sometime a memory issue in ADEL and we

	extraction			linking			extraction + linking		
	P	R	F1	P	R	F1	P	R	F1
AGDISTIS	-	-	-	54.98	54.98	54.98	-	-	-
AIDA	**87.83**	**88.16**	**87.99**	81.87	80.82	81.34	**73**	**73.2**	**73.1**
Babelfy	43.56	83.41	57.23	80.46	79.6	80.03	38.02	67	48.51
DBpedia Spotlight	67.1	78.66	72.42	78.34	51.44	62.1	57.26	66.58	61.57
Dexter	73.25	60.45	66.24	78.3	47.71	59.3	58.2	46.98	51.99
Entityclassifier.eu	56.92	79.18	66.23	67.41	52.95	59.32	39.19	52.95	45.05
FOX	76.7	79.84	78.24	66.52	53.87	59.53	51.38	53.47	52.4
FRED	The annotator caused too many single errors.								
FREME	The annotator caused too many single errors.								
Kea	45.65	60.74	52.12	73.75	73.47	73.61	36.83	44.66	40.37
PBOH	-	-	-	**88.12**	**88.12**	**88.12**	-	-	-
TagMe 2	The annotator caused too many single errors.								
WAT	83.68	86.4	85.02	86.2	84.1	85.14	73.85	72.02	72.92
xLisa-NER	0	0	0	0	0	0	0	0	0
xLisa-NGRAM	0	0	0	0	0	0	0	0	0
ADEL	**77.61**	**92.69**	**84.48**	53.3	53.3	53.3	24.68	29.48	26.87

Table 6.31 – Comparison over the AIDA dataset. **P** stands for **Precision** and **R** for **Recall**

were not able to extend the memory dedicated to it because we have reached the maximum of our machine. Therefore, when such error occurs in ADEL, it returns the already computed annotations without continuing the others which brings incomplete set of results. As showed in Table 6.30 the DIC extractor performs very bad, and we were not able to explain why, yet. Another fact is that similarly to NEEL2014, the AIDA dataset do not contain NIL entities whereas ADEL does annotate them and then brings a lot of false positive, explaining the low precision scores compared to the recall. The SM extractor brings the best results. This time we are not the best system at extraction level and AIDA is the best system for extraction and extraction + linking, and for linking PBOH has again the best score.

For AIDA we have tested our other linking approaches as we can see in Table 6.32. The graph regularization approach is the most encouraging. In this evaluation we have used M&W [108] as entity similarity measure and Doc2Vec [89] as document similarity measure in our graph regularization approach. We think that with a better entity similarity measure such as the one proposed in Chapter 4, Section 4.2.5 we can reach better results.

6.3.5.10 Lemonde

To evaluate JeuxDeLiens, we use the neleval scorer [67] with the following metrics:

- *strong_typed_all_match*: performs a micro-averaged evaluation of all mentions. A mention is counted as correct if it is a correct link or a correct nil. A correct link must have the same span, entity type, and KB identifier as a gold link. A

	linking		
	P	R	F1
Text Similarity Weighted With PageRank	38.75	38.75	38.75
Graph Regularization	53.3	53.3	53.3
Heuristic based Candidate Re-Ranking	55.58	49.97	52.62

Table 6.32 – Results over the AIDA dataset for different linking approaches. Scores in bold represent the best linking approach. **P** stands for **Precision** and **R** for **Recall**

	Precision	Recall	F1
strong_typed_all_match	40.2	40.2	40.2
entity_ceaf	74.2	74.9	74.6
strong_typed_mention_match	46.1	46.1	46.1
strong_link_match	77.8	77.8	77.8
strong_nil_match	91.7	91.7	91.7
strong_all_match	82.6	82.6	82.6

Table 6.33 – Results for JeuxDeLiens

correct NIL must have the same span as a gold NIL.

- *entity_ ceaf*: performs an evaluation based on a one-to-one alignment between system and gold entity clusters for both KB identifier and NIL across documents.
- *strong_ typed_ mention_ match*: performs a micro-averaged evaluation of entity mentions. A system span must match a gold span exactly to be counted as correct and additionally requires the correct entity type.
- *strong_ link_ match*: performs a micro-averaged evaluation of links. A system link must have the same span and KB identifier as a gold link to be counted as correct.
- *strong_ nil_ match*: performs a micro-averaged evaluation of NIL entities. A system NIL must have the same span as a gold NIL to be counted as correct.
- *strong_ all_ match*: performs a micro-averaged link evaluation of all mentions. A mention is counted as correct if is either a link match or a NIL match as defined above.

In Table 6.33, it emerges that the heterogeneity of the network plays an important role for entity linking. Indeed, the knowledge is unevenly distributed across the network and while some domains are well-supplied, some are not. The entity coverage is very satisfying (77.8% recall for the *strong_ link_ match* score) but some entities have a very few incident edges making a path difficult to find. For instance, this is the case for *Boston Dynamics* and *BigDog*, and symmetrically, human-related nodes

(*man*, *woman*, *human*...) are heavily well-supplied as they are high-degree nodes, and more paths are found favoring a misclassification to the class *PER*. This explains why some entities have been mistyped. The *entity_ ceaf* score shows that the approach can succeed to give the same link for the entities that have the same meaning, including the ones that have been linked to same *NIL* across the documents. Nevertheless, the precision is a bit low because of a weakness of our *NIL* clustering method that might link unrelated entities if they share a same token in their mention. The score *strong_ nil_ match* reveals that when we link an entity to *NIL*, it is a good guess. However, our system still proposes some candidates for several entities that should be linked to *NIL*. Our word2vec measure has also a problem when processing a word that does not belong to the vocabulary since no similarity is computed.

6.3.6 Candidate Generation Evaluation

Table 6.34 shows that 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 10000 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 6.34). For example, it is hard to retrieve the proper candidate link[9] for the mention corresponding to the hashtag *#TRUMP2016*. We tackle this problem with our proposed hashtag segmentation approach in Chapter 3. 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, such as *007's*[10].

6.3.7 Computing Performance Evaluation

The performance of ADEL in terms of time needed to process a document is low (from 100ms to 1.5sec across all the possible settings and all the datasets). Despite this good processing time we have identified two significant bottlenecks that increase the processing time: *i)* the network latency, and *2)* the candidate generation. The first is due to a high usage of external systems via HTTP queries (all the extractors and Elasticsearch), the sum of the latency of each HTTP query penalizes the runtime of ADEL. Unfortunately, we cannot really do something to solve this as it is an ADEL

[9] *db:Donald_ Trump_ presidential_ campaign,_ 2016*
[10] http://dbpedia.org/resource/James_Bond_(literary_character)

	Recall
OKE2015	98.38
OKE2016	97.34
OKE2017 T1	99.12
OKE2017 T2	96.45
OKE2017 T3	100
NEEL2014	93.35 (61.91)
NEEL2015	93 (61.84)
NEEL2016	93.55 (60.68)
AIDA	99.62

Table 6.34 – Indexing optimization evaluation: measure if the correct entity is among the list of entity candidates retrieved by the index. *T1* stands for Task1, *T2* stands for Task2 and *T3* stands for Task3. The score between parenthesis represent the index optimization recall without the hashtag segmentation and user mention dereferencing features

requirement to use external systems. Finally, the second bottleneck is Elasticsearch, arriving to a certain number of queries, our ADEL instance gets stuck and starts to queue the queries. As a possible solution, adding a load balancer system on top of our Elasticsearch cluster might allow us to increase the number of queries run in same time without being queued, and then reducing the time to get our candidates.

6.3.8 Evaluation Summary

As an overall overview of these per level evaluations, 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 also interesting to notice that the best ADEL setting across the datasets are different and makes the possibility to create multiple pipeline in ADEL, a key feature. Here a recall of the best setting per dataset:

- OKE2015: MC+COREF+DIC
- OKE2016: MC+COREF+DIC for extraction and extraction+linking, and MC+SM+DIC for recognition
- OKE2017 Task1: MC+DIC
- OKE2017 Task2: MC+DIC
- OKE2017 Task3: SM+DIC
- NEEL2014: MC+POS+DIC for extraction, and POS+DIC for extraction+linking
- NEEL2015: MC+SM+POS for extraction and recognition, and MC+SM+DIC for extraction+linking

- NEEL2016: MC+POS+DIC
- AIDA: SM

Except for the AIDA dataset, ADEL propose every time the best results for extraction and recognition while the linking can be highly improved. The DIC and MC extractors are a recurrent extractor that most of the time belongs to the best setting for such or such dataset, but never alone where it performs in general quite poorly. An interesting experiment would be to integrate the PBOH linking approach into ADEL, since most of the time, PBOH gives the best linking scores.

However difficult life may seem, there is always something you can do and succeed at.

Stephen Hawking

7 Conclusions And Future Directions

7.1 Conclusions

In this thesis, we aimed to improve knowledge extraction methods usable for different Web media by studying three important challenges related to entity recognition, data source indexing and entity linking. We propose a method and an implemented framework for extracting and disambiguating entities that is adaptable to the language being used in the documents, the nature of the documents, the type of identities that should be recognized and disambiguated and the targeted knowledge base. We have contributed with a series of methods relying on Semantic Web, Natural Language Processing and Machine Learning approaches to achieve these goals. We have also showed that a part of our work can have an impact on other Natural Language Processing tasks such as Chunking or Part-of-Speech tagging. Numerous experiments have been conducted on formal documents (i.e. newspapers or encyclopedic documents) in English with DBpedia as targeted knowledge base. However, many of our findings proposed in this thesis have also been tested in other languages (i.e. French), type of documents (i.e. tweets) or knowledge base (i.e. Musicbrainz) and can be easily applied and adapted to untested contexts such as Chinese tweets or Spanish video subtitles with IMDB[1]. Our findings can be summarized as follows:

- For entity recognition, combining different extractors is more effective than using a single one. Performing a combination between different extractors im-

[1] http://www.imdb.com/

proves the entity recognition over different kind of documents, for different type of entities and in different languages. Combining multiple extractors might bring overlap among the recognized entities. To solve this issue, we have proposed a method that successfully resolves these overlaps in order to continue to increase the efficiency of the recognition process and to have the best recognition scores on several benchmark dataset according to our evaluations.

- We have improved the coreference resolution state-of-the-art by successfully combining different approaches from different domains. Deep neural networks are known to be very efficient for performing Natural Language Processing tasks, and coreference resolution is one of them. Semantic approaches are known to be very efficient to model and represent the real world information, and helps to improve several Natural Language Processing tasks, including coreference resolution. Therefore, in order to enhance the performance of a corefence resolution, we have found an approach that makes deep neural networks able to understand real world knowledge represented with an ontology. This approach performs better than the current state-of-the-art according to our evaluation.

- Name entity recognition, part-of-speech tagging and chunking are three challenging Natural Language Processing tasks, and we have succeed to improve the current state-of-the-art of each of these tasks on several languages with one single approach. Our deep-sequence tagger is a neural network that succeed to be agnostic to the kind of document to process and the language by mixing a Convolutional Neural Network to represent the words, a Bidirectional Long Short Term Memory network to represent the sentences and a Conditional Random Field to classify each token.

- It exists multiple knowledge bases against which we can link entities (i.e. DBpedia, Musicbrainz or Freebase), and these sources have their own characteristics such as model, content or storage. We have successfully develop an approach that can be adapted to each of these characteristics and turn their content into a homogeneous and optimized index in order to have a powerful candidate generation process.

- We have proposed fifth encouraging entity linking approach based on adaptability of a given document and knowledge base. The first is a collaborative approach based on a graph representation called graph regularization that use both the context from the document and the knowledge base to link the entities. The second is an approach based on few heuristics that help to better understand in which document context the entities are used, called ReCon. The third is an approach that leverage the popularity of the entities from a given knowledge base with the local similarity of the mentions of these entities. The

fourth is an approach that compute the shortest path between two nodes in a graph depending of the lexical properties of the nodes. Finally, our last approach proposes a deep neural network in order to compute the similarity across the entities of a same knowledge base, and uses the surrounding context of the entities (i.e. types or relations) as feature. This last approach aims to be used in the graph regularization approach as a knowledge base context computation measure.

- We performed a thorough set of evaluations of our work in multiple different contexts: different languages, knowledge bases, type of entities or type of documents.

- Finally, we propose ADEL, our adaptable framework that one can use to do entity linking or entity recognition. This framework implements the proposed architecture and the theoretical work described in this thesis.

7.2 Future Directions

The work described in this thesis leaves naturally to numerous new open research directions. This section proposes to collect future lines of research to pursue this work, grouping them by duration of production from the shortest to the longest. We estimate *short term* to be at most a duration of one year, *middle term* to be at most a duration of two years and finally, *long term* to be at most a duration of three years and can even be seen as a new PhD topic:

- short term:
  - To integrate the missing linking approaches in ADEL: ReCon and graph regularization.
  - To evaluate ADEL over these missing approaches with all the other possible ADEL setting they might be integrated in.
  - To implement a part-of-speech extractor that uses our deep-sequence-tagger approach.

- middle term:
  - To create a consequent training dataset in order to properly train our deep-similarity approach and evaluate the model over the entire Ceccarelli dataset [24].
  - To integrate in ADEL our deep-similarity model into the graph regularization linking approach.

  - To evaluate all the possible ADEL settings with the graph regularization using the deep-similarity model.
  - The project NLP2RDF[2] proposes several datasets in different languages for entity recognition and entity linking. To evaluate ADEL over these datasets would help to strength our claims on the adaptability of ADEL.
  - From the same manner we combine the extractors for improving entity recognition, we can also extend this logic to the linking approaches. It would be interesting to see how much combining linking approaches can improve the current state-of-the-art.

- long term:
  - Entity linking is useful for many applications but it is often scoped to link mentions from a document to their referent into a knowledge base. What if we broaden this vision to a related task of instance matching in knowledge bases? In ADEL, we propose an approach to index multiple different knowledge base, but to use only one at a time as referent. Therefore, if we generate candidate not from a single knowledge base but from multiple ones and use the result of the linking to state that the final links found from different knowledge base are potential duplicates. For example, given the following sentence: *Eminem is an American rapper, record producer and actor*. The mention to link being *Eminem*, we can imagine generating candidates from both DBpedia and Musicbrainz, and select the best final candidate from each of these two knowledge bases, that is *db:Eminem* for DBpedia and *mb:b95ce3ff-3d05-4e87-9e01-c97b66af13d4* for Musicbrainz respectively. Finally, we could deduce that the triple *db:Eminem owl:sameAs mb:b95ce3ff-3d05-4e87-9e01-c97b66af13d4* has a high probability and represents the similarity of these two entities described in two different knowledge bases.
  - 3cixty[3] is an application that aims to help tourists to visit a city by giving them recommendations on events, hotels or restaurants. Behind this application, a knowledge base has been created from multiple sources such as Facebook, Evensi[4], eventful[5] or local data sources given by the city itself. An interesting improvement would be to add more intelligence by making ADEL able to automatically populate the 3cixty knowledge base from tourist documents, user reviews or social media. To fully comply with this vision, ADEL has to be extended with an entity relations extraction

[2] http://wiki-link.nlp2rdf.org/abstracts/
[3] https://www.3cixty.com/
[4] https://www.evensi.fr/
[5] http://www.eventful.com

process. Indeed, with an entity relations extraction and an entity linking process in the same framework, ADEL will truly be able to perform knowledge base population to increase the content of a knowledge base.

Part II

Résumé de Thèse

8

Résumé de la thèse

8.1 Introduction

Le contenu textuel représente la plus grande partie du contenu disponible sur le Web, mais il se prÃľsente sous différentes formes (commentaires sur les réseaux sociaux comme les tweets, les revues ou les statuts Facebook, les sous-titres de vidéos, les articles journalistiques, etc.) et dans différentes langues ce qui représente un ensemble de défis pour les chercheurs en traitement automatique du langage naturel. L'utilisation de contenu textuel requiert d'analyser et d'interpréter les informations qu'ils contiennent. La reconnaissance et la désambiguïsation d'entité nommées ont largement ÃľtÃľ utilisées dans deux projets que sont NexGenTV[1] et ASRAEL[2] oÃź nous Ãľtions impliquÃľ.

Dans le projet NexGentTV, nous créons des outils performants qui permettent de développer des applications deuxième écran qui facilitent la télévision sociale. En particulier, il y a un besoin pour une analyse automatique proche temps réel pour facilement identifier les passages pouvant susciter l'intérêt des téléspectateurs, décrire leur contenu, et faciliter leur enrichissement ainsi que de le partager [1]. Dans ce contexte, nous analysons les sous-titres des programmes télévisuels en français pour extraire et désambiguïser les entités nommées et les sujets pertinents. Dans le projet ASRAEL, nous analysons un volume de données très conséquent de documents journalistiques en anglais et français afin d'induire un schéma précis qui décrit les

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

évènements qui sont transcrits dans ces documents. Plus précisémment, on extrait et désambiguïse les entités nommées qui sont les plus représentatives afin d'en identifier les attributs qui décrivent le mieux un évènement d'une manière complètement non supervisée [116].

8.1.1 Motivation

À la base de ces deux projets, il y a un besoin d'extraction de l'information qui a pour but de récupérer de l'information structurée à partir de texte non structuré en essayant d'interpréter la langue naturelle afin d'extraire de l'information à propos des entités, des relations parmi ces entités et désambiguïser ces entités vis à vis de références externes. Plus précisémment, la reconnaissance d'entités nommées a pour but de localiser et de classifier les entités dans un texte dans des classes prédéfinies comme Personne, Lieu ou Organisation. La désambiguïsation d'entités nommées a pour but de désambiguïser les entités dans un texte à partir de leur référence correspondante, représentant une ressource contenue dans une base de connaissances. Chaque ressource représente une entité du monde réel avec un identifiant qui lui est propre.

Dans cette thèse, nous dénotons une *mention* comme la forme de surface textuelle extraite à partir d'un texte. Une entité comme une annotation qui varie en fonction de la tâche: *i)* quand on fait seulement une tâche de reconnaissance d'entité, une entité est la paire *(mention, classe)*; *ii)* quand on fait seulement une tâche de désambiguïsation d'entité, une entité est la paire *(mention, lien)*; *iii)* quand on fait les deux tâches, la reconnaissance et la désambiguïsation d'entité, une entité est le triplet *(mention, classe, lien)*. Une *entité candidate* est une entité possible que l'on génère afin de désambiguïser la mention extraite. Les *entités nouvelles* sont des entités qui ne sont pas encore apparues dans la base de connaissances utilisée. Ce phénomène apparaît principalement dans les tweets et quelques fois dans les articles journalistiques: typiquement, les personnes qui commencent juste à devenir populaires mais qui n'ont pas encore leur article dans Wikipedia.

Beaucoup de bases de connaissances peuvent être utilisées pour faire de la désambiguïsation d'entité: DBpedia[3], Freebase[4], Wikidata[5] pour en nommer quelques-unes. Ces bases de connaissances sont connues pour être larges en termes de couverture, alors que les bases de connaissances verticales existent aussi pour des domaines spécifiques, telles que Geonames[6] pour la géographie, Musicbrainz[7] pour la musique or LinkedMDB[8] pour le cinéma.

[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

Les deux problèmes principaux lorsque l'on analyse du texte en langue naturelle sont l'ambiguïté et la synonymie. Une entité peut avoir plus d'une mention (synonymie) et une mention peut représenter plus d'une entité (ambiguïté). Par exemple, les mentions *HP* et *Hewlett-Packard* peuvent faire référence à la même entité (synonymie), mais la mention *Potter* peut faire référence à plusieurs entités[9] (ambiguïté) comme des lieux, des personnes, des groupes de musique, des films ou des bateaux. Ce problème peut être étendu à n'importe quelle langue. Ainsi, la désambiguïsation d'entité est aussi prévue pour résoudre les problèmes de synonymie et d'ambiguïté intrinsèque au langage naturel.

8.1.2 Verrous scientifiques

En mettant l'accent sur le contenu textuel, nous pouvons énumérer quatre principaux défis auxquels la communauté du traitement automatique du langage naturel se penche pour effectuer un traitement aussi intelligent et auxquel les systèmes de reconnaissance d'entités et de désambiguïsation d'entités sont confrontés. Ces défis affectent principalement la stratégie utilisée pour comprendre le texte, afin d'extraire des informations utiles et les lier à des référents externes.

1. la nature du texte, faisant référence aux faits qu'il y a deux différentes catégories de texte: *i)* les textes formels, habituellement bien écris et provenant de sources de confiance telles que les journaux, les magazines ou les encyclopédies; *ii)* les textes informels qui sont des textes provenant des plateformes de médias sociaux ou de requêtes de recherche. Chaque catégorie de contenu textuel a ses propres particularités. Par exemple, les tweets sont souvent écrits sans suivre de règles d'écritures particulières (grammaire, orthographe, etc.) et le texte est mélangé avec des liens Web et des hashtags. Un hashtag est une chaîne de caractères précédée par le caractère # et utilisé pour donner un sujet ou un contexte à un message. C'est pourquoi on n'analyse pas un tweet comme on analyse un article Wikipedia.

2. la langue utilisée: le contenu textuel sur le Web est disponible dans de multiples langues et ces langues ont quelques particularités qui les rendent plus ou moins difficiles à analyser (par exemple, les langues d'origines latines par rapport aux langues asiatiques).

3. les types d'entités: il peut exister de multiples classes (types) dans lesquelles une entité peut être catégorisée et où chaque type à une définition. La définition d'un type peut varier selon les personnes. Par exemple, dans le texte *Rendez-vous au Starbucks sur la* 42^{eme} *rue*, l'un peut reconnaître *Starbucks* comme une

[9] https://en.wikipedia.org/wiki/Potter

Organisation alors que d'autres peuvent vouloir considérer que *Starbucks* est un *Lieu* où la branche locale d'un café fait des affaires. Les deux annotations seront correctes conformément à une configuration, mais avec deux définitions différentes.

4. la base de connaissances utilisée: nous pouvons facilement imaginer que les résultats d'un système de désambiguïsation d'entité dépendent hautement de la base de connaissance utilisée. D'abord, la *couverture*: si un texte est à propos d'un film et que l'on utilise une base de connaissances contenant des descriptions de lieux géographiques (comme Geonames), le nombre d'entités pouvant être désambiguïsé sera très petit, contrairement à l'utilisation d'une base de connaissance généraliste (comme DBpedia) ou spécialisée (comme LinkedDB). Ensuite, le *modèle de données*: les bases de connaissances peuvent utiliser différents vocabulaires ainsi qu'une modélisation différente, ce qui empêche la possibilité d'uniformiser les requêtes (e.g. Freebase vs DBpedia). Ils peuvent aussi utiliser différentes technologies de modèlisation de données (e.g. base de données relationnelle vs données liées). Enfin, la *fraîcheur*: si on utilise une version de DBpedia datant de cinq ans, il ne sera pas possible de trouver l'entité *Star Wars: Le réveil de la force* et rendra la désambiguïsation de chaque occurrence de cette entité très difficile.

8.1.3 Hypothèses et questions de recherches

L'hypothèse derrière ce travail est basée sur le fait que la définition d'une entité ne change pas en fonction de la langue, du type de texte, de la catégorie à laquelle elle appartient ou de la base de connaissances, mais que seule la structure et le contexte entourant cette entité changent. À cette fin, nous avons décidé d'harmoniser le processus de désambiguïsation d'entité en fournissant une approche qui peut être adapté à ces différentes structures et contextes.

Pour atteindre cet objectif, nous présentons les questions de recherche suivantes que nos contributions visent à résoudre:

- Comment les entités peuvent-elles être extraites et typées, en utilisant de multiple taxonomies de types d'entités, pour différentes variétés de contenu textuel ?

- Comment différentes bases de connaissances et leur index correspondant peuvent-ils être utilisés pour influencer la désambiguïsation des entités extraites ?

- Comment définir un processus de désambiguïsation d'entité afin de créer un pipeline adaptatif ?

8.1.4 Contributions de la thèse

Après avoir identifié les défis susmentionnés des données textuelles disponibles sur le Web, nous avons proposé de nous attaquer à eux avec une solution générique qui permet de rendre le processus de désambiguïsation adaptable à chaque défi. Le résultat est le framework nommé ADEL qui adresse les quatre défis susmentionnés. Nous résumons nos contributions clés comme ce qui suit:

1. La contribution la plus importante de cette thèse est que nous avons introduit un angle nouveau et unique pour améliorer les approches de désambiguïsation d'entités actuelles pour le contenu multimédia Web: effectuer une analyse d'information adaptable pour la désambiguïsation d'entités. À travers toutes nos études de cas, nous montrons que l'analyse adaptable de l'information est aussi puissante pour de nombreuses sous-tâches de la désambiguïsation d'entités. Ceci est crucial, car les approches à succès précédentes obtenues étant adaptables dans le domaine de la désambiguïsation d'entités étaient principalement basées sur un ou deux défis au maximum. Dans cette thèse, nous explorons et construisons une adaptabilité qui est impliquée dans le contenu et les outils qui ont tendance à inclure beaucoup de bruit. Ainsi, cette thèse démontre l'application potentielle d'approches adaptables dans le domaine de la désambiguïsation d'entités.

2. Une autre contribution importante est que nous avons amélioré le traitement du langage naturel des coréférences pour les machines. Elle aide l'approche automatique à identifier les informations saillantes, fournit des connaissances de base plus riches, et résout l'anaphore à leurs entités de référence régulières pour les rendre plus faciles à comprendre. Notre travail peut également bénéficier de nombreuses tâches de traitement du langage naturel en aval telles que la recherche d'informations et la classification de texte.

3. Nous proposons, explorons et adaptons diverses approches, notamment la classification supervisée, la régularisation non supervisée des graphes, l'apprentissage d'ensemble et les réseaux neuronaux profonds pour modéliser la désambiguïsation d'entités, la reconnaissance d'entités, la coréférence et la mesure de similarité d'entités. Nous avons atteint l'état de l'art dans plusieurs tâches de traitement du langage naturel. Par exemple, nous avons avancé le concept standard de parenté, qui est adopté dans de nombreux systèmes de désambiguïsation d'entités existants en proposant une nouvelle direction basée sur des contextes différents.

4. Nous proposons des méthodes pour construire un index d'optimisation, directement à partir de n'importe quelle base de connaissance afin d'améliorer le processus de génération de candidat d'une tâche de désambiguïsation d'entité.

5. Nous proposons un tout nouveau framework, ADEL, réalisé au cours de cette thèse. ADEL propose une architecture générique, adaptable à de nombreux critères afin de s'adapter au plus grand nombre de cas d'utilisation possible. Il est disponible à la fois en tant qu'application autonome et API REST.

8.2 Travaux connexes

Dans cette section, nous décrivons les différentes méthodes pour chaque composant utilisé dans les approches de pointe: l'extraction de la mention, la désambiguïsation d'entité et la reconnaissance conjointe.

8.2.1 Composants communs pour la désambiguïsation d'entités

Indépendamment des différentes entités reliant les composants qui interviennent dans les workflow classiques [145], il existe différentes manières d'utiliser ces composants. Nous avons identifié quatre workflow différents:

1. les systèmes composés de deux étapes indépendantes: l'extraction de la mention et la désambiguïsation d'entité. Pour l'étape d'extraction de la mention, cela consiste généralement en une détection de mention et un typage d'entité. Pour la phase de désambiguïsation d'entité, il y a souvent une génération d'entité candidate, une sélection d'entité candidate et une classification NIL;

2. les systèmes qui donnent un type à l'entité à la fin du workflow en utilisant les types de l'entité sélectionnée à partir de la base de connaissances lorsqu'ils existent;

3. les systèmes qui génèrent les d'entités candidates en utilisant un dictionnaire pendant le processus d'extraction, et, par conséquent, qui ne seront pas en mesure de traiter des entités NIL;

4. les systèmes qui sont une fusion de toutes ces étapes en une seule, appelée reconnaissance et désambiguïsation conjointe.

8.2.2 Comparaisons

Cette section montre un résumé de plusieurs systèmes représentants l'état de l'art qui seront utilisés pour comparer nos résulats dans un but d'évaluation. Ces approches sont divisées en deux tableaux : le tableau 2.2 détaille les techniques d'extraction ou de reconnaissance adoptées, et le tableau 2.3 détaille les techniques de désambiguïsation. Certaines des approches visés dans le second tableau n'apparaissent pas dans le premier tableau car elles ne sont capables que de faire de la désambiguïsation. Les approches sont : AIDA [76], Babelfy [110], DBpedia Spotlight [103], Dexter [24],

Entityclassifier.eu [41], FOX [176, 162], FRED [33], FREME[10], KEA [165], TagMe 2 [53], WAT [123], X-LiSA [198], AGDISTIS [176], DoSeR [204], NERFGUN [70] et PBOH [58].

Les deux tableaux partagent deux colonnes : *Reconnaissance* et *Génération de candidats*. Les deux disent si le système correspondant fait de la reconnaissance ou génére des candidats á l'étape représenté par le tableau. Par exemple, s'il y a un *oui* dans le tableau 2.2 pour la colonne *Génération de Candidats* cela veut dire que les candidats sont générés durant le processus d'extraction d'entités et non pas durant la désambiguïsation. La même logique s'applique á la colonne *Reconnaissance*.

Les systèmes présentés dans les tableaux sont tous classés par ordre chronologique, du plus ancien au plus récent. Dans le tableau 2.2, nous pouvons voir que la tendance est de s'appuyer sur des outils externes de traitement du langage naturel supervisé. Les quelques autres sont basés sur un dictionnaire. Les travaux décrits dans ce document s'appuient sur les deux, pour la principale raison que les données étiquetées dans la plupart des langues pour former correctement une approche supervisée sont rares, et dans ce cas, l'utilisation d'un dictionnaire est utile. Dans le tableau 2.3, nous pouvons voir que la tendance est davantage orientée vers une approche collective avec une distribution égale entre les approches orientÃĺes graphes et les approches indépendantes. Les approches indépendantes sont réparties également entre les approches supervisées et non supervisées. De plus, faire du *regroupement de NIL* n'est pas souvent géré par ces systèmes, y compris les plus récents. Les travaux décrits dans le présent document proposent des approches collectives et indépendantes pour relier les entités, y compris les entités NIL avec une méthode de *regroupement de NIL*.

Le tableau 2.4 donne des détails sur la possibilité de résoudre les quatre défis mentionnés au chapitre 1, section 1.2, que nous proposons d'aborder dans ce travail : indépendance du texte, indépendance de la base de connaissances, indépendance linguistique et indépendance du type d'entité. Nous constatons que les systèmes ont des difficultés à proposer un moyen de résoudre ces défis, car ils n'en relèvent au maximum que deux et parfois aucun. Les systèmes sans symbole dans une colonne représentent le fait qu'ils ne font pas l'extraction ou la reconnaissance d'entités. Les travaux décrits dans le présent document proposent une approche adaptative pour relever chacun de ces défis en même temps.

8.3 Approche

L'objectif d'une approche de désambiguïsation des entités est de reconnaître et de relier toutes les mentions figurant dans un texte à des entrées existantes de la base de données et d'identifier de nouvelles entités non encore incluses dans la base de connaissances. ADEL est livré avec une nouvelle architecture (Figure 5.1) par rapport à

[10] https://freme-project.github.io/api-doc/full.html

l'état de l'art. Ces architectures sont typiquement statiques et montrent peu de flexibilité pour extraire et désambiguïser les entités. Ils ne peuvent généralement pas être prolongés sans apporter des changements importants qui nécessiteraient beaucoup de temps en termes d'intégration. Par exemple, pour l'extraction, il n'est pas possible d'ajouter un moteur d'extraction de dictionnaire à AIDA [76] ou une extraction NER à TagME [53]. Ensuite, le processus de désambiguïsation est également fixé car, par exemple, on ne peut pas ajouter à Babelfy [110] une méthode basée sur une formule linéaire qui utilise une approche orienté graphes. Enfin, la base de connaissances utilisée est souvent pré-établie : elle est difficile à modifier car on ne peut pas demander à Babelfy [110] de se passer de Babelnet [112] au profit d'une autre base de connaissances appartenant au *Linked Open Data Cloud*.

ADEL a été conçu pour permettre tous ces changements. L'architecture d'ADEL est modulaire et les modules se répartissent en trois grandes catégories. La première partie, (*Reconnaissance des entités*), contient les modules *Extracteurs* et *Résolution de chevauchement*. La deuxième partie, (*Index*), contient le module *Indexation*. Enfin, la troisième partie, (*Désambiguïsation des entités*), contient les modules *Génération de candidats*, *Regroupement de NIL* et les *Désambiguïseurs*. L'architecture fonctionne avec ce que nous appelons des *modules* définis comme une partie de l'architecture configurable via un fichier de configuration et où chaque composant d'un module (en rouge sur le schéma) peut être activé ou désactivé en fonction d'une configuration que l'on souhaite utiliser. Chaque module est décrit plus en détail dans les sections 8.3.1, 8.3.2 et 8.3.3. Un pipeline général peut également être configuré automatiquement pour certains modules.

8.3.1 Reconnaissance des entités

Dans cette section, nous décrivons comment nous reconnaissons les mentions présentes dans les textes qui sont susceptibles d'être sélectionnés comme entités avec le *Module Extracteurs*. Après avoir identifié les mentions candidats, nous résolvons leurs chevauchements potentiels à l'aide du *Module de Résolution des Chevauchements*.

Module Extracteurs. Actuellement, nous utilisons six extracteurs différents : *1)* Étiqueteuse dictionnaire, *2)* Étiqueteuse POS, *3)* Étiqueteuse NER, *4)* Étiqueteuse de date, *5)* Étiqueteuse de numéros et *6)* Étiqueteuse de coréférences. Si deux ou plusieurs de ces extracteurs sont activés, ils fonctionnent en parallèle. Le processus de reconnaissance est basé sur des systèmes TAL externes tels que Stanford CoreNLP [100], GATE, NLTK ou OpenNLP. Pour être conforme à tout système NLP externe, nous avons basé notre processus de reconnaissance sur une interface API Web qui utilise NIF comme format d'échange de données. Par conséquent, en utilisant ce module, il est possible de passer d'un système NLP à un autre sans changer quoi que ce soit dans le code ou de combiner différents systèmes. Un exemple est disponible

avec Stanford CoreNLP[11].

1. L'extracteur Étiqueteuse dictionnaire s'appuie sur le traitement intégré proposé dans les systèmes TAL tels que *RegexNER*[12] de Stanford CoreNLP, *DictionaryNameFinder*[13] pour OpenNLP ou le *Dictionary Setup*[14] pour GATE. Nous proposons également un moyen automatisé de générer un dictionnaire en envoyant des requêtes SPARQL à une base de connaissance de données liée qui s'inspire de la façon dont GATE génère ses dictionnaires. Tout en utilisant un dictionnaire comme extracteur, il donne la possibilité d'être très flexible en termes d'entités à extraire et de leur type correspondant, et permet de gérer plusieurs langues.

2. L'extracteur Étiqueteuse POS est configuré pour extraire les noms propres singuliers et pluriels et pour attacher le type générique *THING*.

3. L'extracteur Étiqueteuse NER vise à extraire les entités nommées qui sont classées à travers les taxonomies utilisées par Stanford CoreNLP, OpenNLP, GATE ou autres systèmes TAL. Afin de gérer les tweets, nous entraînons un modèle en utilisant les données du Challenge NEEL [145].

4. L'extracteur Étiqueteuse de Date vise à reconnaître toutes les formes de surface qui représentent une expression temporelle comme *Aujourd'hui*, *18 décembre 1997* ou *1997/12/18* et s'appuie sur les systèmes temporels actuels tels que *SUTime*[15], *ManTIME*[16] ou *HeidelTime*[17].

5. L'extracteur Étiqueteuse de Numéro vise à reconnaître les chiffres (par exemple 15, 1, 35) ou leur représentation textuelle (par exemple un, trente), et peut être fait soit par une étiqueteuse NER (avec Stanford NER), une étiqueteuse POS (avec l'étiquette CD[18]) ou des expressions régulières.

6. L'extracteur Étiqueteuse de co-référence vise à extraire les co-références à l'intérieur d'un même document mais pas à travers les documents. Les annotateurs fournis par Stanford CoreNLP, OpenNLP, GATE ou autres systèmes TAL peuvent être utilisés.

[11] https://github.com/jplu/stanfordNLPRESTAPI
[12] http://stanfordnlp.github.io/CoreNLP/regexner.html
[13] http://opennlp.apache.org/documentation/apidocs/opennlp-tools/opennlp/tools/namefind/DictionaryNameFinder.html
[14] https://gate.ac.uk/sale/tao/splitch13.html#x18-34700013.9.2
[15] https://nlp.stanford.edu/software/sutime.shtml
[16] https://github.com/filannim/ManTIME/
[17] https://github.com/HeidelTime/heideltime/releases
[18] https://sites.google.com/site/partofspeechhelp/#TOC-CD-

Nous avons la possibilité de combiner tous ces extracteurs, mais aussi de combiner les différents modèles NER en un seul extracteur Étiqueteuse NER. Plus précisément, nous utilisons une méthode de combinaison de modèles qui vise à utiliser conjointement différents modèles CRF dans Stanford NER comme décrit dans l'algorithme 1. Cet algorithme montre que l'ordre dans lequel les modèles sont appliqués est important. Dans Stanford NER, il s'appelle *NER Classifier Combiner*. Cette logique peut être étendue à n'importe quel autre étiqueteuse NER. Nous expliquons la logique de cette combinaison de modèles NER à l'aide de l'exemple suivant : *William Bradley Pitt (né le 18 décembre 1963) est un acteur et producteur américain.*. Les détails des modèles utilisés sont disponibles dans la documentation de Stanford NER[19]. Si nous n'appliquons que le modèle à 4 classes, nous obtenons le résultat suivant : *William Bradley Pitt* en *PERSONNE*, et *américain* en *DIVERS*. Si nous n'appliquons que le modèle de 7 classes, nous obtenons le résultat suivant : *William Bradley Pitt* en *PERSONNE* et le *18 décembre 1963* en *DATE*. Si nous appliquons les deux modèles en même temps en utilisant la logique de combinaison de modèles, on obtient le résultat suivant : *William Bradley Pitt* en *PERSONNE*, le *18 décembre 1963* en *DATE* et *américain* en *DIVERS*.

Cette combinaison de différents modèles peut toutefois entraîner un problème d'étiquetage. Imaginons deux modèles entrainés sur deux ensembles de données différents, où dans un ensemble de données, une localisation est étiqueté *LOC* mais dans l'autre ensemble de données, il est étiqueté *Lieu*. Par conséquent, si nous appliquons une combinaison de ces deux modèles, les résultats contiendront des entités étiquetées qui représentent une localisation, mais certaines d'entre elles avec l'étiquette *LOC* et d'autres avec l'étiquette *Lieu* et certaines mentions pourraient avoir une étiquette ou l'autre selon l'ordre dans lequel les modèles ont été appliqués. Dans ce cas, les classes ne sont plus harmonisées car nous mélangeons des modèles qui ont été entrainés avec des étiquettes différentes pour représenter le même type d'entités. Pour résoudre ce problème d'étiquetage, nous proposons une solution en deux étapes : *i)* ne pas mélanger des modèles qui ont été entrainés avec des étiquettes différentes pour représenter le même type d'entité, mais plutôt créer deux instances d'un extracteur NER où chacun a une combinaison de modèles compatibles ; et *ii)* utiliser un module de résolution de chevauchement qui résout les chevauchements entre les mentions extraites de chaque extracteur et harmoniser les étiquettes provenant des modèles des différentes instances de cet extracteur dans une même définition de marquage.

Module de résolution des chevauchements. Ce module vise à résoudre les chevauchements entre les sorties des extracteurs et à obtenir une sortie sans chevauchement. La logique de ce module est la suivante : compte tenu de deux mentions qui se chevauchent, par exemple, `Etats-Unis d'Amérique` de l'étiqueteuse NER et

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

`Etats-Unis` d'étiqueteuse POS, nous ne prenons que l'union des deux phrases. Nous obtenons la mention `Etats-Unis d'Amérique` et le type fourni par l'étiqueteuse NER est sélectionné. Les chevauchements en termes de texte sont faciles à résoudre, mais cela devient beaucoup plus difficile pour les types lorsqu'il s'agit de décider quel type conserver lorsque deux types proviennent de deux extracteurs différents.

Un premier cas est celui où deux étiquettes représentent la même catégorie, par exemple *LOC* du modèle Stanford 3 classes et *dul:Place* d'un modèle formé avec le jeu de données OKE2015[20]. Afin de résoudre cette ambiguïté, nous avons développé un alignement représenté en SKOS entre les types de sources multiples dont les sources sont les étiquettes données par les trois modèles par défaut de Stanford NER, l'ontologie DUL[21], l'ontologie Schema.org[22], l'ontologie DBpedia[23], l'ontologie Music Ontology, l'ontologie NERD et la taxonomie NEEL [145]. Un exemple de cette association pour le type *Personne* est fourni à https://gist.github.com/jplu/74843d4c09e72845487ae8f9f201c797 et la même logique est appliquée pour les autres types. Avec cette association, il est alors possible de passer d'une source à l'autre avec une requête SPARQL. Nous utilisons également la notion de correspondances larges et étroites de SKOS afin d'introduire une hiérarchie entre les types permettant d'avoir un parent ou une sous-catégorie si une catégorie équivalente n'existe pas.

Ce processus de reconnaissance nous permet de traiter un grand nombre de langues et de types de documents en *i)* combinant intelligemment différents annoteurs provenant de systèmes externes multiples, et *ii)* fusionnant leurs résultats en résolvant leurs chevauchements et en alignant leurs types. Une fois que nous avons réussi à reconnaître les entités, nous générons des candidats d'entités extraits de la base de connaissances. Dans la section suivante, nous décrivons en détail le processus d'indexation d'une base de connaissances comme une tâche essentielle pour la désambiguïsation.

8.3.2 Indexation des données liées

Dans cette section, nous décrivons comment nous indexons une base de connaissances et comment nous optimisons la recherche à l'aide du module d'indexation. Le module est composé de deux étapes : *i)* l'indexation et *ii)* l'optimisation de la recherche. Il existe de multiples différences entre les bases de connaissances existantes qui rendent le processus d'indexation très complexe. Le processus suivant peut s'appliquer à toute base de connaissances qui utilise des données liées. Nous détaillerons quelles sont les exigences minimales en matière de données liées qu'une base de connaissances doit respecter, mais aussi les autres données liées supplémentaires qu'elle peut contenir.

[20] https://ckan.project-hobbit.eu/fr/dataset/oke2015_task1
[21] http://www.ontologydesignpatterns.org/ont/dul/DUL.owl
[22] http://schema.org
[23] http://mappings.dbpedia.org/server/ontology/classes/

Indexation. La première étape consiste à extraire toutes les entités qui seront indexées à l'aide d'une requête SPARQL. Cette requête définit autant de contraintes que nécessaire. Les exigences minimales pour qu'une entité soit indexée sont d'avoir un identifiant, une étiquette et un score. Ce score peut correspondre au PageRank de l'entité, ou à tout autre moyen de noter les entités dans une base de connaissance de données liée. Par exemple, avec DBpedia, les extractions nécessaires correspondantes[24] sont : Labels, Page Ids et Page Links. L'extraction Page Links n'est utilisé que pour calculer le PageRank des entités DBpedia et ne sera pas chargé. Nous utilisons une bibliothèque dédiée de graphes[25] pour calculer le PageRank et générer un fichier RDF qui contient le score du PageRank pour toutes les entités. En général, il faut générer un fichier qui ne contient que les liens entre les entités d'une même source afin de calculer leur PageRank. Pour DBpedia, nous utilisons également d'autres extractions : textes d'ancres, types d'instance, type d'instance transitive, liens de désambiguïsation, longs résumés, littéraux basés sur les associations et les redirections. Une fois fait, nous chargeons toutes les données dans un système de stockage de triplets RDF et nosu utilisons une requête SPARQL (Requête 4.1 pour DBpedia ou Requête 4.3 pour Musicbrainz) qui récupère les entités recherchées. Dans le cas de DBpedia, nous ajoutons une contrainte supplémentaire telle que ne pas être une redirection ou une page de désambiguïsation. Ensuite, pour chaque entité que nous avons obtenue via cette première requête, nous exécutons une seconde requête SPARQL qui a pour rôle de récupérer toutes les données que nous voulons indexer. La requête 4.2 et la requête 4.4 sont respectivement utilisées pour DBpedia et Musicbrainz. Le résultat de cette seconde requête est ensuite utilisé pour obtenir un index de la base de connaissances.

Optimisation. Une fois que nous avons cet index, nous pouvons rechercher une mention et retrouver des candidats pour les entités. La recherche dans toutes les colonnes a un impact négatif sur la performance de l'index en termes de temps de calcul. Afin d'optimiser l'index, nous avons développé une méthode qui maximise la couverture de l'index tout en interrogeant un nombre minimum de colonnes (ou de propriétés d'entité). Par conséquent, nous avons besoin de savoir à l'avance sur quelles colonnes chercher. Nous avons expérimenté une logique d'optimisation pour les ensembles de données de référence suivants : AIDA et NEEL2015. Ces ensembles de données doivent être annotés avec la base de connaissances ciblée appropriée. Pour cette raison, nous prenons comme exemple comment optimiser un index DBpedia mais la logique proposée peut être étendue à toute autre base de connaissances.

L'index DBpedia comporte 4 726 950 lignes (entités) et 281 colonnes (propriétés qui ont un litéral comme valeur). Compte tenu de certains ensembles de données de référence tels que OKE2015, OKE2016, NEEL2014, NEEL2014, NEEL2015 et

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

NEEL2016, nous analysons leur contenu afin d'extraire une liste de paires distinctes (mention, lien). Ensuite, pour chaque paire, nous interrogeons l'index pour chaque colonne (dans le cas de dbpedia, cela représente 281 requêtes pour chaque paire), et pour chaque requête, nous vérifions si le lien approprié de la paire est parmi les résultats ou non. Si oui, nous mettons la propriété dans une liste blanche, et si non, la propriété est ignorée comme n'étant pas utile pour récupérer le bon lien candidat. À la fin, nous obtenons un fichier qui ressemble à l'extrait représenté dans l'énumération 4.5.

Ce fichier indique les colonnes qui doivent être interrogées pour obtenir le lien approprié pour chaque paire. Nous remarquons que la plupart des paires partagent des colonnes similaires. Par conséquent, nous faisons une union de toutes ces colonnes pour obtenir une liste de colonnes uniques à utiliser pour interroger l'index. Pour l'extrait de l'énumération 4.5, l'union distincte donne la liste suivante de 9 propriétés :

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`

Dans le cas de dbpedia, cela réduit le nombre de colonnes à interroger de 281 à 72, mais cette liste est encore trop longue. Si nous vérifions attentivement cet extrait, nous remarquons que la colonne *dbo_ wikiPageWikiLinkText* appartient à chaque liste, ce qui signifie qu'avec une seule colonne (au lieu de 9) nous pouvons récupérer toutes les paires sauf la paire*AnotherYou—-http://dbpedia.org/resource/Another_ You.* La logique sous-jacente est que nous devons maximiser le nombre de paires que nous récupérons pour chaque colonne, et l'objectif est alors de minimiser le nombre de colonnes. À la fin, nous terminons avec une liste minimale de colonnes qui maximisent la couverture des paires. Cette optimisation peut se faire avec l'Algorithme 2. Le code source est également disponible[26].

[26] https://gist.github.com/jplu/a16103f655115728cc9dcff1a3a57682

Au terme de cette optimisation, nous produisons une liste réduite de 4 propriétés qui sont nécessaires pour maximiser la couverture des paires dans l'ensemble de données de référence :

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

Cette optimisation réduit le temps de la requête pour générer les entités candidates d'environ 4 secondes à moins d'une seconde. Ce ratio est un temps moyen calculé sur l'ensemble des réponses aux requêtes. Le processus d'indexation nous permet d'indexer un grand nombre de bases de connaissances qui utilisent des données liées et d'optimiser la recherche en fonction de celles-ci. Ce dernier est possible à la condition d'avoir au moins un ensemble de données de référence utilisant la base de connaissances ciblée.

8.3.3 Désambiguïsation d'entité

Le composant de désambiguïsation des entités commence par le *module de génération de candidats* qui interroge l'index et génère une liste des entités candidates pour chaque entité extraite. Si l'index renvoie une liste de candidats d'entité, alors le *module des désambiguateurs* est invoqué. Alternativement, si une liste vide de candidats d'entités est retournée, alors le *module des regroupements de NIL* est invoqué.

Module des regroupements de NIL. Nous proposons de regrouper les entités *NIL* qui peuvent identifier la même chose dans le monde réel. Le rôle de ce module est d'attacher la même valeur *NIL* dans et entre les documents. Par exemple, si nous prenons deux documents différents qui partagent la même entité émergente, cette entité sera liée à la même valeur NIL. On peut alors imaginer différentes valeurs NIL, telles que *NIL_1*, *NIL_2*, etc. Nous effectuons un appariement strict des chaînes de caractères sur chaque entité *NIL* possible (ou entre chaque signe s'il s'agit d'une mention composée de plusieurs signes). Par exemple, deux mentions : "Sully" et "Marine Jake Sully" seront liés à la même entité *NIL*.

Module des désambiguateurs. Tout comme le *module des extracteurs*, ce module peut gérer plus d'une méthode de dÃĺsambiguïsation. La fonction détaillée dans ce document est une fonction évaluée empiriquement représentée par l'équation 1 qui classe tous les candidats possibles donnés par le *module de génération de candidats*.

La fonction $r(l)$ utilise la distance Levenshtein L entre la mention m et le titre, la distance maximale entre la mention m et chaque élément (titre) de l'ensemble des pages de redirection R de Wikipedia et la distance maximale entre la mention m et

chaque élément (titre) de l'ensemble des pages de désambiguation D de Wikipedia, pondérées par le PageRank PR, pour chaque entité candidate l. Les poids a, b et c sont une combinaison convexe qui doit satisfaire : $a+b+c = 1$ et $a > b > c > 0$. Nous partons du principe que la mesure de la distance entre une mention et un titre est plus importante que la mesure de distance avec une page de redirection qui est elle-même plus importante que la mesure de distance avec une page de désambiguïsation.

8.4 Implémentation

Le framework ADEL est implémenté en Java et est accessible au public via une API REST[27]. ADEL traite les quatre défis susmentionnés en s'adaptant à la langue et au type de texte à traiter, aux types d'entités à extraire et à la base de connaissances à utiliser pour fournir des identificateurs aux entités.

ADEL a besoin d'un fichier de configuration exprimé en YAML que nous appelons *profile* (liste 5.1) afin d'adapter son workflow. Dans le reste de cette section, nous détaillerons le fonctionnement de chaque composant.

Extract. Dans le Listing 5.1, l'objet *extract* configure le composant de reconnaissance d'entité. Il est composé d'un objet pour chaque extracteur utilisé (NER, POS, COREF, dic, date et numéro.), la valeur de ces objets étant une liste d'instances. Par exemple, dans la liste 5.1, il y a deux extracteurs : *ner* et *pos*, où chaque extracteur génère une instance. Une instance est composée de quatre propriétés obligatoires : *address*, *name*, *profile*, *className*, et facultatives : *tags*. La propriété *address* est l'adresse HTTP de l'API Web utilisée pour interroger l'extracteur. La propriété *name* est un nom unique donné à l'instance de l'extracteur. La propriété *profile* est le profil que l'extracteur doit adopter[28]. La propriété *className* est le nom complet de la classe Java (package + classe) qui doit être utilisée en interne pour exécuter l'extracteur. Cette propriété permet à quiconque de gérer le comportement de l'extracteur via la réflexion de Java[29]. La propriété optionnelle unique, *tags*, représente la liste des tags qui doivent être extraits (tous si vides ou non présents). Il est également composé de deux autres propriétés obligatoires qui sont *mapping* et *reference*. Le premier est l'emplacement du fichier de d'association SKOS pour les types, et le second est la source qui sera utilisée pour typer les entités.

Index. Dans le Listing 5.1, l'objet *index* configure l'index qui est composé de quatre propriétés obligatoires : *type*, *address*, *strict* et *name*. La propriété *address* est l'adresse HTTP de l'API Web ou l'adresse de dossier utilisée pour localiser l'index.

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

[28] La liste disponible des profils existants pour l'extracteur NER commençant par le préfixe *ner_* est décrite à l'adresse suivante https://github.com/jplu/stanfordNLPRESTAPI/tree/develop/properties

[29] La réflexion permet d'examiner, d'introspecter et de modifier la structure et le comportement du code au moment de l'exécution.

La propriété *type* définit le type d'index à utiliser. Actuellement, nous ne traitons que Elasticsearch et Lucene mais notre processus d'indexation peut être étendu à tout autre système d'indexation. Dans le cas d'un index Elasticsearch, les propriétés *query* et *name* sont obligatoires, le premier est le fichier où trouver le modèle de requête Elasticsearch et le second est le nom de l'index. Dans le cas de Lucene, ces propriétés sont remplacées par deux autres propriétés obligatoires qui sont *fields* et *size*, la première étant la liste des champs qui seront interrogés et la seconde étant le nombre maximum de candidats à récupérer 5.2. La propriété *strict* peut avoir deux valeurs : *true* si on veut une recherche stricte, ou *false* si on veut une recherche floue.

Link. Dans le Listing 5.1, l'objet *link* configure le module de dÁĺsambiguïsation. Cette propriété contient le nom complet de la classe Java (package + classe) qui doit être utilisée en interne pour exécuter la méthode de dÁĺsambiguïsation correspondante.

8.5 Évaluation

Nous proposons d'évaluer plusieurs configurations d'ADEL afin de montrer la puissance de son adaptabilité. En raison de la grande dimensionnalité des configurations possibles, nous ne prenons que les combinaisons d'extracteurs qui sont les plus représentatives pour évaluer correctement ADEL. L'approche de combinaison de modèles de reconnaissance d'entités nommées sera basée sur Stanford CoreNLP alors que l'utilisation d'un seul modèle se fait au moyen d'un marqueur de séquence profonde, car les performances du marqueur de séquence profonde sont supérieures à celles de Stanford CoreNLP et l'utilisation d'un seul modèle formé avec Stanford CoreNLP ne donnera rien par rapport au Deep-sequence-tagger. Le marqueur de partie du discours utilisé sera celui fourni par Stanford CoreNLP parce que nous n'avons pas implémenté l'API sur la version de marqueur de partie du discours de Deep-sequence-tagger. A cette fin, nous définissons une configuration ADEL comme une combinaison d'un ou plusieurs des extracteurs suivants :

- *MC* (combinaison de modèles pour l'extraction d'entités nommées) : utilise un étiqueteur de reconnaissance d'entité nommée avec un paramètre de combinaison de modèles où les modèles CRF sont ceux fournis par défaut par Stanford CoreNLP.

- SM *(modèle unique de reconnaissance des entités nommées) : utilise un étiqueteur de reconnaissance d'entité nommée avec un modèle formé avec les données d'entraînement d'un jeu de données de référence via deep-sequence-tagger.*

- POS *(partie de discours) : utilise un étiqueteur de partie de discours avec le bon modèle, pour les tweets si l'ensemble de données de référence est basé sur*

des tweets ou général si l'ensemble de données de référence est basé sur du texte s'apparentant à des articles de presse.

- DT *(date) : utilise un étiqueteur de reconnaissance d'entité nommée avec un modèle spécialement formé pour reconnaître les dates fournies par Stanford CoreNLP.*
- NUM *(nombre) : utilise un étiqueteur de reconnaissance d'entité nommée avec un modèle spécialement formé pour reconnaître les numéros fournis par Stanford CoreNLP.*
- COREF *(coréférence de base) : utilise Sanaphor++ via Stanford CoreNLP.*
- DIC *(dictionnaire) : utilise un dictionnaire construit respectivement pour l'ensemble de données de référence correspondant avec DBpedia ou Musicbrainz selon l'ensemble de données.*

Les liens permanents GERBIL pour chaque exécution de ce document sont accessibles au public en note de bas de page https://docs.google.com/spreadsheets/d/1Oaoc1mtmdX8HVzUUf7YG25GmB3I_iW0HMUbB5ligJvs/edit?usp=sharing.

8.5.1 OKE 2015

En ce qui concerne le tableau 6.14, il est intéressant de noter que l'extracteur deep-sequence-tagger (SM) est le meilleur réglage d'extracteur simple pour ADEL, alors que si on le combine avec d'autres extracteurs (SM + extracteur(s)), les résultats sont toujours inférieurs à ceux obtenus avec un extracteur combiné modèle (MC) combiné avec les mêmes autres extracteurs (MC + extracteur(s)). Ceci est dû à une faible quantité de données d'entraînement, le deep-sequence-tagger a besoin de plus de données pour être plus performant. De plus, disposer d'un dictionnaire permet d'améliorer significativement les résultats (+13% en moyenne). En analysant les résultats, nous avons vu que le l'extracteur COREF est très utile pour extraire les entités. On peut aussi remarquer que l'extracteur COREF n'apporte pas grand-chose à l'extracteur SM au niveau de l'extraction. Ceci est dû au fait que notre modèle réussit à apprendre à reconnaître une coréférence, et à la classifier correctement, cependant, l'extracteur COREF est important car il lie ces mentions à leur référence correcte, ce que les autres extracteurs ne font pas car il est impossible pour ces extracteurs de faire une relation entre les entités extraites. Par exemple, dans la phrase *Barack Obama était le président des États-Unis. Il est né à Hawaï*, un autre extracteur pourrait extraire *Barack Obama* et *Il* et les typer en tant que *Personne*, mais ne fera jamais la relation que *Il* fait référence à *Barack Obama* et puis que *Barack Obama* doit être utilisé pour désambiguïser *Il*. C'est pourquoi nous avons besoin d'un extracteur de coréférence qui fournit cette relation. Cependant, nous remarquons que dans le tableau 6.14 les

scores de reconnaissance (précision, rappel et f1) donnés par GERBIL sont toujours exactement les mêmes. Afin de mieux comprendre pourquoi cela se produit, nous avons décidé d'évaluer ADEL avec un autre évaluateur (à savoir neleval) avec une configuration (à savoir *SM*) afin de proposer une explication. En effet, avec neleval, les scores d'extraction et d'extraction+lien sont exactement les mêmes que ceux donnés par GERBIL mais les scores de reconnaissance pour la configuration *SM* sont beaucoup plus importants : de P=R=F1=59.79 (avec GERBIL) à P=71.7 R=76.5 F1=74 (avec neleval). Les scores de reconnaissance donnés par neleval montrent que GERBIL a un bug pour ses scores de reconnaissance, et ils travaillent sur une note de correctionhttps://github.com/dice-group/gerbil/issues/226. Malgré cela, nous avons décidé de nous en tenir à GERBIL en tant que évaluateur car nous ne sommes pas en mesure de noter tous les autres systèmes avec neleval. À cette fin, nous rappelons aux lecteurs que les notes de reconnaissance attribuées par GERBIL pourraient ne pas représenter le plein potentiel de notre approche. Cette remarque vaut également pour toutes les évaluations suivantes. Enfin, le score de désambiguïsation baisse beaucoup, ceci est dû à l'approche de désambiguïsation utilisée qui a deux problèmes notables : *1)* la similarité de chaîne de caractères qui favorise toujours la chaîne de caractères la plus courte (le score de distance de chaîne de caractères sur le titre, les pages de redirection et de désambiguïsation entre la mention *GM* et l'entité candidate *db:Allemagne* (0.32879817) est supérieure à celle du candidat entité *db:General_Motors* (0.21995464))), et *2)* le pagerank qui promeut toujours l'entité la plus populaire malgré un faible score de similarité de chaîne de caractères (la mention *Harry Potter* sera toujours liée à *db :Harry_Potter* et ne mettra jamais l'accent sur *db:Harry_Potter_(film_series)*, *db:Harry_Potter_(character)* ou *db:Harry_Potter_(journalist)*). La même remarque pour la désambiguïsation peut s'appliquer à n'importe laquelle des évaluations suivantes.

Dans le tableau 6.15 nous pouvons voir que ADEL a le meilleur score d'extraction et de reconnaissance tandis que PBOH a le meilleur score de désambiguïsation.

8.5.2 OKE 2016

Comme nous pouvons le voir dans le tableau 6.16, les commentaires que nous avons faits pour le jeu de données OKE2015 sont également valables pour le jeu de données OKE2016. La principale différence entre les deux est une meilleure performance des réglages impliquant l'extracteur *SM*, ce qui peut s'expliquer par un jeu de données d'entraînement plus important, ce qui signifie que plus le jeu d'entraînement est grand, meilleures seront les combinaisons impliquant l'extracteur *SM*. A cette fin, nous remarquons que le meilleur réglage de reconnaissance implique l'extracteur SM (MC+SM+DIC) et est différent du meilleur réglage d'extraction et d'extraction + désambiguïsation (MC+COREF+DIC). Néanmoins, ces deux réglages ADEL différents

battent tous les autres systèmes du tableau 6.17 en termes d'extraction et de reconnaissance, et une fois de plus PBOH a les meilleurs résultats de désambiguïsation.

8.5.3 OKE 2017 Task1

L'extracteur SM ne peut pas être utilisé ici parce que la reconnaissance ne fait pas partie de la tâche et que nous n'avons donc pas un jeu de données qui inclut les types d'entités. Le tableau 6.18 montre un très bon score d'extraction pour la configuration *MC + DIC* de ADEL par rapport aux autres systèmes (ou réglage ADEL) alors qu'il est également intéressant de noter qu'avec un simple extracteur de partie du discours nous pouvons atteindre un score au niveau de l'extraction qui bat la plupart des systèmes énumérés dans le tableau 6.19. Pour ce jeu de données, AIDA donne les meilleurs résultats pour l'extraction + désambiguïsation, et PBOH, encore une fois, obtient les meilleurs résultats pour la désambiguïsation.

8.5.4 OKE 2017 Task2

La Tâche 2 partage la même exigence que la Tâche 1 sur la reconnaissance, donc encore une fois, pas d'extracteur SM. Le tableau 6.20 montre des résultats similaires à ceux du jeu de données OKE2017 Task1, et les observations sur ADEL que nous pouvons faire sont exactement les mêmes. Le tableau 6.21 montre pour ce jeu de données que DBpedia Spotlight donne les meilleurs résultats pour l'extraction + désambiguïsation, et Kea, obtient les meilleurs résultats pour la désambiguïsation.

8.5.5 OKE 2017 Task3

La tâche 3 fournit des types d'entités précises et spécifiques (artistes, chansons et albums), ce qui soulève un problème majeur : le nom d'un artiste, d'une chanson ou d'un album peut être n'importe quoi et inclure, par exemple, un signe de ponctuation[30]. Dans ce jeu de données, les entités sont créés avec les types de Music Ontology et liées aux entrées Musicbrainz. Le tableau 6.22 montre à quel point l'extracteur SM peut être performant par rapport au contenu textuel formel avec un jeu d'apprentissage ayant une taille décente. Le jeu de données est de 100 documents et chacun d'eux a une moyenne de 400 mots. Les systèmes comparés dans le tableau 6.23 (sauf FOX) ne sont pas capables de gérer ces types ou bases de connaissances, mais GERBIL possède une fonction d'alignement qui permet de faire correspondre les liens DBpedia vers leur équivalent Musicbrainz et la même chose pour les types DBpedia vers les types Music Ontology. À cette fin, il est possible d'évaluer ces systèmes en fonction de ce jeu de données.

[30]https://musicbrainz.org/work/25effd3c-aada-44d1-bcbf-ede30ef34cc0

Cette fois, l'extracteur SM apporte les meilleures performances, et les résultats sont encore meilleurs lorsqu'il est combiné au dictionnaire. De plus, notre formule de désambiguïsation qui promeut les entités populaires fonctionne très bien pour la base de connaissances Musicbrainz, et surpasse largement toute autre méthode utilisée dans les systèmes comparés, et alors ADEL devient le meilleur système pour ce jeu de données.

8.5.6 NEEL2014

Contrairement aux jeux de données précédents, NEEL2014 est composé de tweets et non de contenu encyclopédique ou de documents d'actualité ce qui augmente la difficulté d'extraction d'entités, de reconnaissance et de désambiguïsation. De plus, ce jeu de données est particulièrement difficile car la tâche demande aux systèmes d'extraire et de désambiguïser (mais pas de typer) toutes les entités (y compris les hashtags et les mentions des utilisateurs) qui ont une entrée dans DBpedia, peu importe à quoi elles correspondent (animaux, dates, nombres, livres, etc) mais pas les nouvelles entités. Selon la tâche, une entité peut aussi être un mélange de mots habituels avec un hashtag ou une mention d'utilisateur, par exemple *saison 2 de #Sherlock* doit être pris comme une seule entité. Comme il n'y a pas de dactylographie, il ne nous est pas possible de former un modèle de reconnaissance d'entité nommée avec le jeu de données, ce qui fait de l'extracteur de partie du discours un extracteur important comme nous pouvons le voir dans le tableau 6.24. Nous avons utilisé un extracteur de nombres (NUM) et un extracteur de date (DT) via un modèle de reconnaissance d'entité nommée formé avec le jeu de données MUC-7 fourni par défaut dans Stanford CoreNLP, et cette approche n'apporte rien en raison de la difficulté pour l'extracteur de reconnaissance d'entité nommée à analyser les tweets, notamment l'extracteur de combinaison qui a des difficultés à bien fonctionner seul pour les mêmes raisons. Au cours de cette évaluation, nous avons remarqué un nouveau bogue dans GERBIL qui avait des difficultés à gérer correctement les tweets contenant des émoticônes, ce qui pourrait biaiser les résultats d'ADEL et des autres systèmes, et ils travaillent sur une note de correction[31]. A cette fin, tous les résultats sur le jeu de données de tweets sont faits avec ce bogue. Malgré ce commentaire, le meilleur réglage implique la combinaison de modèles et les extracteurs de dictionnaire comme pour les autres jeux de données, mais cette fois avec l'ajout de l'extracteur de partie du discours. Une autre chose à noter est que le meilleur réglage ADEL pour l'extraction + désambiguïsation n'est pas le même que l'extraction parce que l'extracteur de combinaison de modèles apporte trop de faux liens positifs. Une fois de plus, comme le montre le tableau 6.25, ADEL a le meilleur processus d'extraction, tandis que le système qui a la meilleure extraction et désambiguïsation est le gagnant officiel de la compétition. Enfin, une

[31] https://github.com/dice-group/gerbil/issues/227

fois de plus, PBOH obtient le meilleur score de désambiguïsation.

8.5.7 NEEL2015

Le tableau 6.26 révèle, comme le jeu de données OKE2016, que le meilleur réglage peut être différent selon le niveau auquel on veut obtenir les meilleurs résultats, mais dans les deux cas on peut voir la combinaison des extracteurs SM et MC. Pour la meilleure extraction et reconnaissance, c'est le réglage MC+SM+POS, tandis que pour la meilleure extraction et désambiguïsation, c'est MC+SM+DIC. Bien que l'extracteur SM fasse partie des deux meilleurs réglages, on constate que ses performances sont beaucoup plus faibles sur les tweets que sur les textes formels, ce qui signifie que le réseau de neuronnes a des difficultés à gérer les tweets. Le jeu de données de NEEL2015 contient plus de 3000 tweets, ce qui est un nombre décent d'exemples pour former un tel réseau de neuronnes mais apparemment, après analyse des résultats, il est difficile pour le réseau d'identifier correctement les entités qui sont des hashtags ou des mentions utilisateur. Néanmoins, dans le tableau 6.27, la reconnaissance entre parenthèses est du au bogue identifié dans GERBIL qui influence le score de reconnaissance, ce qui signifie que nous pourrions aussi faire mieux que le gagnant.

8.5.8 NEEL2016

Les jeux de données NEEL2015 et NEEL2016 partagent la même tâche. La principale caractéristique du jeu de données de test NEEL2016 est qu'il contient une majorité de hashtags et de mentions utilisateur annotées comme entités. Ceci explique donc pourquoi les extracteurs SM et MC fonctionnent si mal et pourquoi l'extracteur de partie de discours fonctionne si bien comme le montre le tableau 6.28. Une fois de plus, ADEL obtient les meilleurs résultats au niveau de l'extraction et de la reconnaissance par rapport à tout autre système proposé dans le tableau 6.29.

8.5.9 AIDA

Nous avons eu une difficulté matérielle avec le jeu de données AIDA, la taille des documents (une moyenne de 600 mots) a parfois soulevé un problème de mémoire dans ADEL et nous n'avons pas pu étendre la mémoire qui lui est dédiée car nous avons atteint le maximum de notre machine. Par conséquent, lorsqu'une telle erreur se produit dans ADEL, elle retourne les annotations déjà calculées sans continuer les autres, ce qui apporte un jeu incomplet de résultats. Comme le montre le tableau 6.30, l'extracteur DIC fonctionne très mal, et nous n'avons pas encore été en mesure d'expliquer pourquoi. Un autre fait est que, tout comme NEEL2014, le jeu de données AIDA ne contient pas d'entités NIL, alors qu'ADEL les annote et apporte ensuite beaucoup de faux positifs, ce qui explique la faible précision des scores par

rapport au rappel. L'extracteur SM apporte les meilleurs résultats. Cette fois, nous ne sommes pas le meilleur système au niveau de l'extraction et AIDA est le meilleur système pour l'extraction et l'extraction et désambiguïsation, et pour la désambiguïsation PBOH a de nouveau le meilleur score.

Pour le programme AIDA, nous avons testé nos autres approches de désambiguïsation, comme nous pouvons le voir dans le tableau 6.32. L'approche de régularisation de graphes est la plus encourageante. Dans cette évaluation, nous avons utilisé M&W [108] comme mesure de similarité d'entité et Doc2Vec [89] comme mesure de similarité de document dans notre approche de normalisation de graphe. Nous pensons qu'avec une meilleure mesure de similarité d'entité telle que celle proposée dans le chapitre 4, Section 4.2.5 nous pouvons atteindre de meilleurs résultats.

8.5.10 Lemonde

Dans le tableau 6.33, il apparaît que l'hétérogénéité du réseau lexical joue un rôle important pour la désambiguïsation d'entités. En effet, les connaissances sont inégalement réparties dans le réseau et si certains domaines sont bien fournis, d'autres ne le sont pas. La couverture des entités est très satisfaisante (77.8% de rappel pour le score *strong_link_match*) mais certaines entités ont très peu d'arêtes incidentes rendant un chemin difficile à trouver. Par exemple, c'est le cas pour *Boston Dynamics* et *BigDog*, et symétriquement, les noeuds humains (*homme*, *femme*, *humain*...) sont très bien fournis car ils sont de haut degré, et il y a plus de chemins favorisant une mauvaise classification à la classe *PER*. Cela explique pourquoi certaines entités ont été mal tapées. Le score *entity_ceaf* montre que l'approche peut réussir à donner le même lien pour les entités qui ont la même signification, y compris celles qui ont été liées au même *NIL* dans les documents. Néanmoins, la précision est un peu faible à cause d'une faiblesse de notre méthode de regroupement *NIL* qui pourrait lier des entités non apparentées si elles partagent un même symbole dans leur mention. Le score indique que lorsque nous lions une entité à *NIL*, c'est une bonne supposition. Cependant, notre système propose encore quelques candidats pour plusieurs entités qui devraient être liées à *NIL*. Notre mesure word2vec pose également un problème lors du traitement d'un mot qui n'appartient pas au vocabulaire car aucune similitude n'est calculée.

8.6 Conclusions et Perspectives

8.6.1 Conclusions

Dans cette thèse, nous visions à améliorer les méthodes d'extraction des connaissances utilisables sur différents médias Web en étudiant trois défis importants liés à la reconnaissance des entités, à l'indexation des sources de données et à la désam-

biguïsation des entités. Nous proposons une méthode et un cadre d'extraction et de désambiguïsation des entités adaptables à la langue utilisée dans les documents, à la nature des documents, au type d'identités à reconnaître et à désambiguïser et à la base de connaissances ciblée. Pour atteindre ces objectifs, nous avons contribué avec une série de méthodes qui s'appuient sur le Web sémantique, le traitement du langage naturel et les approches d'apprentissage automatique. Nous avons également montré qu'une partie de notre travail peut avoir un impact sur d'autres tâches de traitement du langage naturel comme le découpage ou le marquage de la partie du discours. De nombreuses expériences ont été menées sur des documents formels (journaux ou documents encyclopédiques) en anglais avec DBpedia comme base de connaissances ciblée. Cependant, bon nombre des résultats proposés dans cette thèse ont également été testés dans d'autres langues (e.g. le français), types de documents (e.g. des tweets) ou bases de connaissances (e.g. Musicbrainz) et peuvent être facilement appliqués et adaptés à des contextes non testés comme les tweets chinois ou les sous-titres vidéo en espagnol avec IMDB[32]. Nos constatations peuvent se résumer comme suit :

- Pour la reconnaissance d'entités, la combinaison de différents extracteurs est plus efficace que l'utilisation d'un seul. L'exécution d'une combinaison entre différents extracteurs améliore la reconnaissance des entités sur différents types de documents, pour différents types d'entités et dans différentes langues. La combinaison de plusieurs extracteurs peut entraîner des chevauchements entre les entités reconnues. Pour résoudre ce problème, nous avons proposé une méthode qui résout avec succès ces chevauchements afin de continuer à accroître l'efficacité du processus de reconnaissance et d'obtenir les meilleures notes de reconnaissance pour plusieurs ensembles de données de référence selon nos évaluations.

- Nous avons amélioré l'état de l'art de la résolution de coréférence de base en combinant avec succès différentes approches provenant de différents domaines. Les réseaux neuronaux profonds sont connus pour être très efficaces pour effectuer des tâches de traitement du langage naturel, et la résolution de coréférence en est une. Les approches sémantiques sont reconnues pour être très efficaces pour modéliser et représenter l'information du monde réel, et aident à améliorer plusieurs tâches de traitement du langage naturel, y compris la résolution de coréférence. Par conséquent, afin d'améliorer la performance d'une résolution de coréférence, nous avons trouvé une approche qui rend les réseaux neuronaux profonds capables de comprendre la connaissance du monde réel représentée par une ontologie. D'après notre évaluation, cette approche donne de meilleurs résultats que l'état de l'art actuel.

[32] http://www.imdb.com/

- La reconnaissance d'entité nommées, le marquage de partie du discours et le tronçonnage sont trois tâches difficiles du traitement du langage naturel, et nous avons réussi à améliorer l'état actuel de chacune de ces tâches sur plusieurs langues avec une seule approche. Notre étiqueteuse de séquence profonde est un réseau de neuronnes qui réussit à être agnostique au type de document à traiter et au langage en mélangeant un réseau de neuronnes convolutionnel pour représenter les mots, un réseau bidirectionnel de mémoire à court terme pour représenter les phrases et un champ aléatoire conditionnel pour classer chaque mot.

- Il existe de multiples bases de connaissances sur lesquelles nous pouvons relier des entités (par exemple DBpedia, Musicbrainz ou Freebase), et ces sources ont leurs propres caractéristiques telles que modèle, contenu ou stockage. Nous avons développé avec succès une approche qui peut être adaptée à chacune de ces caractéristiques et transformer leur contenu en un indice homogène et optimisé afin d'avoir un processus puissant de génération de candidats.

- Nous avons proposé cinq approches encourageantes de disambiguïsation des entités, basée sur l'adaptabilité d'un document et d'une base de connaissances donnés. La première est une approche indépendante qui considère la popularitdes entités. La seconde est une approche collaborative basée sur une représentation de graphe appelée régularisation de graphe qui utilise à la fois le contexte du document et la base de connaissances pour relier les entités. La troisième est une approche qui tire parti de la popularité des entités à partir d'une base de connaissances donnée avec la similitude locale des mentions de ces entités. La quatrième est une approche qui calcule le chemin le plus court entre deux nœuds d'un graphe en fonction des propriétés lexicales des nœuds. Enfin, notre dernière approche propose un réseau de neuronnes profond afin de calculer la similarité entre les entités d'une même base de connaissances, et utilise le contexte environnant des entités (par exemple, les types ou relations) comme caractéristique. Cette dernière approche vise à être utilisée dans l'approche de régularisation de graphes comme mesure de calcul du contexte de base de connaissances.

- Nous avons effectué un ensemble complet d'évaluations de notre travail en variant les contextes : langues, bases de connaissances, types d'entités ou de documents à analyser.

- Enfin, nous proposons ADEL, notre framework adaptable que l'on peut utiliser pour faire de la désambiguïsation d'entités ou de la reconnaissance d'entités. Ce framework met en œuvre l'architecture proposée et les travaux théoriques décrits dans cette thèse.

8.7 Perspectives

Les travaux décrits dans cette thèse laissent naturellement place à de nombreuses nouvelles pistes de recherche. Cette section propose de rassembler les axes de recherche futurs pour poursuivre ce travail, en les regroupant par durée de production, de la plus courte à la plus longue. Nous estimons que *court terme* est d'une durée maximale d'un an, *moyen terme* est d'une durée maximale de deux ans et enfin, *long terme* est d'une durée maximale de trois ans et peut même être considéré comme un nouveau sujet de thèse :

- court terme:
  - Intégrer l' approche de désambiguïsation manquante dans ADEL : régularisation de graphe.
  - Évaluer ADEL par rapport à cette approche manquante avec tous les autres paramètres possibles d'ADEL dans lesquels elle pourrait être intégrée.
  - Mettre en place un extracteur de partie du discours qui utilise notre approche deep-sequence-tagger.
- moyen terme:
  - Créer un jeu de données d'entraînement conséquent afin de former correctement notre approche de similarité profonde et d'évaluer le modèle sur le jeu de données Ceccarelli [24].
  - Intégrer dans ADEL notre modèle de similarité profonde dans l'approche de désambiguïsation de régularisation de graphe.
  - Evaluer tous les réglages possibles d'ADEL avec la régularisation de graphe à l'aide du modèle de similarité profonde.
  - Le projet NLP2RDF[33] propose plusieurs jeux de données dans différentes langues pour la reconnaissance d'entités et la liaison d'entités. L'évaluation d'ADEL par rapport à ces jeux de données aiderait à renforcer nos affirmations sur l'adaptabilité d'ADEL.
  - De la même manière que nous combinons les extracteurs pour améliorer la reconnaissance des entités, nous pouvons également étendre cette logique aux approches de désambiguïsation. Il serait intéressant de voir dans quelle mesure la combinaison d'approches de désambiguïsation peut améliorer l'état de l'art actuel.
- long terme:

[33] http://wiki-link.nlp2rdf.org/abstracts/

– La désambiguïsation d'entités est utile pour de nombreuses applications, mais il est souvent nécessaire de relier les mentions d'un document à leur référent dans une base de connaissances. Et si nous élargissions cette vision à une tâche connexe d'appariement d'instances dans les bases de connaissances ? Dans ADEL, nous proposons une approche pour indexer plusieurs bases de connaissances différentes, mais nous n'en utilisons qu'une seule à la fois comme référente. Par conséquent, si nous générons des candidats non pas à partir d'une seule base de connaissances, mais à partir de plusieurs, et que nous utilisons le résultat de la désambiguïsation pour déclarer que les liens finaux trouvés dans différentes bases de connaissances sont des doublons potentiels. Par exemple, étant donné la phrase suivante : *Eminem est un rappeur, producteur de disques et acteur américain.*. La mention à désambiguïser étant *Eminem*, nous pouvons imaginer générer des candidats à la fois à partir de DBpedia et de Musicbrainz, et sélectionner le meilleur candidat final à partir de chacune de ces deux bases de connaissances, à savoir *db:Eminem* pour DBpedia et *mb:b95ce3ff-3d05-4e87-9e01-c97b66af13d4* pour Musicbrainz. Enfin, on pourrait en déduire que le triplet *db:Eminem owl:sameAs mb:b95ce3ff-3d05-4e87-9e01-c97b66af13d4* a une probabilité élevée et représente la similarité de ces deux entités décrites dans deux bases de connaissances différentes.

– 3cixty[34] est une application qui vise à aider les touristes à visiter une ville en leur donnant des recommandations sur les événements, hôtels ou restaurants. Derrière cette application, une base de connaissances a été créée à partir de sources multiples telles que Facebook, Evensi[35], Eventful[36] ou des sources de données locales fournies par la ville elle-même. Une amélioration intéressante serait d'ajouter plus d'intelligence en rendant ADEL capable de remplir automatiquement la base de connaissances 3cixty à partir de documents touristiques, de commentaires d'utilisateurs ou de médias sociaux. Pour répondre pleinement à cette vision, ADEL doit être étendu avec un processus d'extraction de relations entre entités. En effet, avec une extraction de relations entre entités et un processus de désambiguïsation d'entités dans le même framework, ADEL sera réellement capable de réaliser une population de base de connaissances pour augmenter son contenu.

[34] https://www.3cixty.com/
[35] https://www.evensi.fr/
[36] http://www.eventful.com

Bibliography

[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] Amazon. Alexa webpage. https://developer.amazon.com/alexa/smart-home.

[3] Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. Globally normalized transition-based neural networks. In *ACL*, 2016.

[4] Apple. Siri webpage. https://www.apple.com/uk/ios/siri/.

[5] Masayuki Asahara and Yuji Matsumoto. Japanese named entity extraction with redundant morphological analysis. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, 2003.

[6] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The 6th International Semantic Web Conference Proceedings*, 2007.

[7] Amit Bagga and Breck Baldwin. Algorithms for scoring coreference chains. In *The first international conference on language resources and evaluation workshop on linguistics coreference*, 1998.

[8] Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 2009.

[9] 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*, 2014.

[10] Yoshua Bengio. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2009.

[11] Giacomo Berardi, Diego Ceccarelli, Andrea Esuli, and Diego Marcheggiani. On the impact of entity linking in microblog real-time filtering. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, 2015.

[12] Raffaella Bernardi, Ruket Çakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli Ikizler-Cinbis, Frank Keller, Adrian Muscat, and Barbara Plank. Automatic description generation from images: A survey of models, datasets, and evaluation measures. *CoRR*, 2016.

[13] Romaric BesanÃğon, GaÃńl de Chalendar, Olivier Ferret, Faiza Gara, Olivier Mesnard, Meriama LaÃŕb, and Nasredine Semmar. Lima : A multilingual framework for linguistic analysis and linguistic resources development and evaluation. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, 2010.

[14] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 2003.

[15] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. arXiv:1607.04606, 2016.

[16] 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.

[17] Andrew Borthwick, John Sterling, Eugene Agichtein, and Ralph Grishman. Nyu: Description of the mene named entity system as used in muc-7. In *In Proceedings of the Seventh Message Understanding Conference (MUC-7*, 1998.

[18] Volha Bryl, Claudio Giuliano, Luciano Serafini, and Kateryna Tymoshenko. Using background knowledge to support coreference resolution. In *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, 2010.

[19] Razvan Bunescu and Marius Pasca. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06), Trento, Italy*, 2006.

[20] Amev Burman, Arun Jayapal, Sathish Kannan, Madhu Kavilikatta, Ayman Alhelbawy, Leon Derczynski, and Robert J. Gaizauskas. Usfd at kbp 2011:

Entity linking, slot filling and temporal bounding. In *Proceedings of the Text Analytics Conference Workshop*, 2011.

[21] Liu Cao, He Shizhu, Yang Hang, Liu Kang, and Zhao Jun. Unsupervised joint entity linking over question answering pair with global knowledge. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, 2017.

[22] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[23] Taylor Cassidy, Zheng Chen, Javier Artiles, Heng Ji, Hongbo Deng, Lev-Arie Ratinov, Jiawei Han, Dan Roth, and Jing Zheng. Cuny-uiuc-sri tac-kbp2011 entity linking system description. In *Proceedings of the Text Analytics Conference Workshop*, 2011.

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

[25] Angel X. Chang and Christopher D. Manning. Sutime: A library for recognizing and normalizing time expressions. In *LREC*, 2012.

[26] Kai-Wei Chang, He He, Hal Daumé III, and John Langford. Learning to search for dependencies. *CoRR*, 2015.

[27] Zheng Chen and Heng Ji. Collaborative ranking: A case study on entity linking. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2011.

[28] Chung-Cheng Chiu, Tara N. Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J. Weiss, Kanishka Rao, Ekaterina Gonina, Navdeep Jaitly, Bo Li, Jan Chorowski, and Michiel Bacchiani. State-of-the-art speech recognition with sequence-to-sequence models. *CoRR*, 2017.

[29] Kevin Clark and Christopher D. Manning. Entity-centric coreference resolution with model stacking. In *Association for Computational Linguistics (ACL)*, 2015.

[30] Kevin Clark and Christopher D. Manning. Deep reinforcement learning for mention-ranking coreference models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, (EMNLP)*, 2016.

[31] Kevin Clark and Christopher D. Manning. Improving coreference resolution by learning entity-level distributed representations. In *ACL*, 2016.

[32] Aaron M. Cohen. Unsupervised gene/protein named entity normalization using automatically extracted dictionaries. In *Proceedings of the ACL-ISMB Workshop on Linking Biological Literature, Ontologies and Databases: Mining Biological Semantics*, 2005.

[33] Sergio Consoli and Diego Reforgiato Recupero. Using fred for named entity resolution, linking and typing for knowledge base population. In *SemWebEval@ESWC*, 2015.

[34] Silviu Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of EMNLP-CoNLL*, 2007.

[35] James R. Curran and Stephen Clark. Language independent ner using a maximum entropy tagger. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, 2003.

[36] Gianluca Demartini, Djellel Eddine Difallah, and Philippe Cudré-Mauroux. Zencrowd: Leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *Proceedings of the 21st International Conference on World Wide Web*, 2012.

[37] Lingjia Deng and Janyce Wiebe. How can NLP tasks mutually benefit sentiment analysis? A holistic approach to sentiment analysis. In *WASSA@NAACL-HLT*, 2016.

[38] Omkar Deshpande, Digvijay S. Lamba, Michel Tourn, Sanjib Das, Sri Subramaniam, Anand Rajaraman, Venky Harinarayan, and AnHai Doan. Building, maintaining, and using knowledge bases: A report from the trenches. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, 2013.

[39] Cambridge Dictionary. Formal / informal texts. https://dictionary.cambridge.org/fr/grammaire/grammaire-britannique/types-of-english-formal-informal-etc/formal-and-informal-language.

[40] Dennis Diefenbach, Kuldeep Singh, Andreas Both, Didier Cherix, Christoph Lange, and Sören Auer. Integrating and benchmarking entity linking for question answering with qanary. empty note, 2016.

[41] Milan Dojchinovski and Tomáš Kliegr. Entityclassifier.eu: Real-time classification of entities in text with wikipedia. In *Machine Learning and Knowledge Discovery in Databases: European Conference*, 2013.

[42] Xin Luna Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *The 20th International Conference on Knowledge Discovery and Data Mining Proceedings*, 2014.

[43] Mark Dredze, Paul McNamee, Delip Rao, Adam Gerber, and Tim Finin. Entity disambiguation for knowledge base population. In *Proceedings of the 23rd International Conference on Computational Linguistics*, 2010.

[44] Greg Durrett and Dan Klein. A Joint Model for Entity Analysis: Coreference, Typing, and Linking. *Transactions of the Association for Computational Linguistics*, 2014.

[45] Asif Ekbal and Sivaji Bandyopadhyay. A hidden markov model based named entity recognition system: Bengali and hindi as case studies. In *Proceedings of the 2^{nd} International Conference on Pattern Recognition and Machine Intelligence*, 2007.

[46] Elastic. Elasticsearch. https://www.elastic.co/products/elasticsearch.

[47] Marieke Van Erp, Giuseppe Rizzo, and Raphaël Troncy. Learning with the web: Spotting named entities on the intersection of nerd and machine learning. In *WWW 2013, 3^{rd} International Workshop on Making Sense of Microposts (#MSM'13), Concept Extraction Challenge, May 13, 2013, Rio de Janeiro, Brazil*, 2013.

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

[49] Pavlos Fafalios, Vasileios Iosifidis, Kostas Stefanidis, and Eirini Ntoutsi. Multi-aspect entity-centric analysis of big social media archives. In *TPDL*, 2017.

[50] Wei Fang, Jianwen Zhang, Dilin Wang, Zheng Chen, and Ming Li. Entity disambiguation by knowledge and text jointly embedding. In *CoNLL*, 2016.

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

[52] Dimitra Farmakiotou, Vangelis Karkaletsis, John Koutsias, George Sigletos, Constantine D. Spyropoulos, and Panagiotis Stamatopoulos. Rule-based named entity recognition for greek financial texts. In *In Proceedings of the Workshop on Computational Lexicography and Multimedia Dictionaries (COMLEX 2000*, 2000.

[53] 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.

[54] 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.

[55] Apache Fundation. Lucene. http://lucene.apache.org/.

[56] Apache Fundation. Solr. http://lucene.apache.org/solr/.

[57] Ignazio Gallo, Elisabetta Binaghi, and Moreno Carullo. Named entity recognition by neural sliding window. In *The 8th IAPR International Workshop on Document Analysis Systems*, 2008.

[58] Octavian-Eugen Ganea, Marina Ganea, Aurelien Lucchi, Carsten Eickhoff, and Thomas Hofmann. Probabilistic bag-of-hyperlinks model for entity linking. In *Proceedings of the 25th International Conference on World Wide Web*, 2016.

[59] Octavian-Eugen Ganea and Thomas Hofmann. Deep joint entity disambiguation with local neural attention. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017.

[60] Abhishek Gattani, Digvijay S. Lamba, Nikesh Garera, Mitul Tiwari, Xiaoyong Chai, Sanjib Das, Sri Subramaniam, Anand Rajaraman, Venky Harinarayan, and AnHai Doan. Entity extraction, linking, classification, and tagging for social media: A wikipedia-based approach. *Proc. VLDB Endow.*, 2013.

[61] Wael H. Gomaa and Aly A. Fahmy. A survey of text similarity approaches, 2013.

[62] Google. Google home webpage. https://store.google.com/us/product/google_home?hl=en-US.

[63] Swapna Gottipati and Jing Jiang. Linking entities to a knowledge base with query expansion. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2011.

[64] C. Green, D. Luckham, R. Balzer, T. Cheatham, and C. Rich. Readings in artificial intelligence and software engineering. In *Report on a Knowledge-based Software Assistant*, 1986.

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

[66] Stephen Guo, Ming-Wei Chang, and Emre Kiciman. To link or not to link? a study on end-to-end tweet entity linking. In *HLT-NAACL*, 2013.

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

[68] Aria Haghighi and Dan Klein. Coreference resolution in a modular, entity-centered model. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2010.

[69] Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, 2009.

[70] Sherzod Hakimov, Hendrik ter Horst, Soufian Jebbara, Matthias Hartung, and Philipp Cimiano. Combining textual and graph-based features for named entity disambiguation using undirected probabilistic graphical models. In *European Knowledge Acquisition Workshop*, 2016.

[71] James Hammerton. Named entity recognition with long short-term memory. In *7th Conference on Natural Language Learning at HLT-NAACL*, 2003.

[72] Xianpei Han and Le Sun. A generative entity-mention model for linking entities with knowledge base. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, 2011.

[73] Xianpei Han and Le Sun. An entity-topic model for entity linking. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2012.

[74] Xianpei Han, Le Sun, and Jun Zhao. Collective entity linking in web text: A graph-based method. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2011.

[75] Zhengyan He and Houfeng Wang. Collective entity linking and a simple slot filling method for TAC-KBP 2011. In *Proceedings of the Text Analytics Conference Workshop*, 2011.

[76] 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)*, 2011.

[77] Hongzhao Huang, Yunbo Cao, Xiaojiang Huang, Heng Ji, and Chin-Yew Lin. Collective tweet wikification based on semi-supervised graph regularization. In *Association for Computational Linguistics*, 2014.

[78] Hongzhao Huang, Larry Heck, and Heng Ji. Leveraging deep neural networks and knowledge graphs for entity disambiguation, 2015.

[79] 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 *22^{nd} ACM International Conference on Information & Knowledge Management (CIKM)*, 2013.

[80] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. *CoRR*, 2015.

[81] Filip Ilievski, Giuseppe Rizzo, Marieke van Erp, Julien Plu, and Raphaël Troncy. Context-enhanced adaptive entity linking. In *10^{th} edition of the Language Resources and Evaluation Conference*, 2016.

[82] Noman Islam, Zeeshan Islam, and Nazia Noor. A survey on optical character recognition system. *CoRR*, 2017.

[83] Alexey Grigorevich Ivakhnenko and Valentin Grigorevich Lapa. *Cybernetic Predicting Devices*. CCM Information Corporation, 1965.

[84] Matthias Jarke, Bernd Neumann, Yannis Vassiliou, and Wolfgang Wahlster. Kbms requirements of knowledge-based systems. In *Foundations of Knowledge Base Management: Contributions from Logic, Databases, and Artificial Intelligence Applications*, 1978.

[85] Xu Jian, Hector Liu, Qin Lu, Patty Liu, Chenchen Wang, Yan Li, Xiaoning Li, Hanying Huang, Yang Song, Cheng Chang, Liaoming Zhou, Jing Xiao, Dian Yu, Weiran Xu, Guang Chen, and Jun Guo. Polyucomp in tac 2011 entity linking and slot-filling. In *Proceedings of the Fourth Text Analysis Conference*, 2011.

[86] Ridong Jiang, Rafael E. Banchs, and Haizhou Li. Evaluating and combining named entity recognition systems. In *Proceedings of the 6^{th} Named Entity Workshop, joint with ACL2016*, 2016.

[87] Mathieu Lafourcade. Making people play for lexical acquisition with the jeux-demots prototype. In *7th ACM International Symposium on Natural Language Processing (SNLP'07)*, 2007.

[88] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *HLT-NAACL*, 2016.

[89] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, 2014.

[90] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1989.

[91] Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task. In *Conference on Natural Language Learning (CoNLL) Shared Task*, 2011.

[92] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal*, 2015.

[93] Xiaohua Liu, Yitong Li, Haocheng Wu, Ming Zhou, Furu Wei, and Yi Lu. Entity linking for tweets. In *Proceedings of the 26rd International Conference on Computational Linguistics*, 2013.

[94] Linyuan Lü, Ci-Hang Jin, and Tao Zhou. Similarity index based on local paths for link prediction of complex networks. *Phys. Rev. E*, 2009.

[95] 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.

[96] Xiaoqiang Luo. On coreference resolution performance metrics. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, 2005.

[97] Xiaoqiang Luo, Abraham Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. A mention-synchronous coreference resolution algorithm based on the bell tree. In *ACL*, 2004.

[98] Chao Ma, Janardhan Rao Doppa, John Walker Orr, Prashanth Mannem, Xiaoli Z. Fern, Thomas G. Dietterich, and Prasad Tadepalli. Prune-and-score: Learning for greedy coreference resolution. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, 2014.

[99] Xuezhe Ma and Eduard H. Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *ACL*, 2016.

[100] 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.

[101] Andrew McCallum and Wei Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, 2003.

[102] Andrew McCallum and Ben Wellner. Toward conditional models of identity uncertainty with application to proper noun coreference. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web*, 2003.

[103] 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.

[104] Microsoft. Cortana webpage. https://www.microsoft.com/en-us/windows/cortana.

[105] Rada Mihalcea and Andras Csomai. Wikify!: Linking documents to encyclopedic knowledge. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, 2007.

[106] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, 2013.

[107] George A. Miller. Wordnet: A lexical database for english. *Communication of the ACM (CACM)*, 1995.

[108] David Milne and Ian H. Witten. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy*, 2008.

[109] Ruslan Mitkov, Branimir Boguraev, and Shalom Lappin. Introduction to the special issue on computational anaphora resolution. *Computational Linguistics*, 2001.

[110] Andrea Moro, Alessandro Raganato, and Roberto Navigli. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2014.

[111] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 2007.

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

[113] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, 2001.

[114] Vincent Ng. Machine learning for entity coreference resolution: A retrospective look at two decades of research. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, 2017.

[115] Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. J-NERD: Joint Named Entity Recognition and Disambiguation with Rich Linguistic Features. *Transactions of the Association for Computational Linguistics*, 2016.

[116] 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.

[117] Feng Niu, Ce Zhang, Christopher RÃľ, and Jude W. Shavlik. Elementary: Large-scale knowledge-base construction via machine learning and statistical inference. *Int. J. Semantic Web Inf. Syst.*, 2012.

[118] 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.

[119] 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.

[120] Georgios Paliouras, Vangelis Karkaletsis, Georgios Petasis, and Constantine D. Spyropoulos. Learning decision trees for named-entity recognition and classification. In *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI 2000)*, 2000.

[121] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[122] Georgios Petasis, Frantz Vichot, Francis Wolinski, Georgios Paliouras, Vangelis Karkaletsis, and Constantine D. Spyropoulos. Using machine learning to maintain rule-based named-entity recognition and classification systems. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, 2001.

[123] 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.

[124] Anja Pilz and Gerhard Paaß. From names to entities using thematic context distance. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, 2011.

[125] Danuta Ploch, Leonhard Hennig, Ernesto William De Luca, and Sahin Albayrak. Dai approaches to the tac-kbp 2011 entity linking task. In *Proceedings of the Text Analysis Conference*, 2011.

[126] Julien Plu. Knowledge extraction in web media: At the frontier of nlp, machine learning and semantics. In *25th World Wide Web Conference, PhD Symposium*, 2016.

[127] Julien Plu, Giuseppe Rizzo, and Raphaël Troncy. An experimental study of a hybrid entity recognition and linking system. In *14th International Semantic Web Conference, Poster Demo Session*, 2015.

[128] Julien Plu, Giuseppe Rizzo, and Raphaël Troncy. A hybrid approach for entity recognition and linking. In *12th European Semantic Web Conference, Open Extraction Challenge*, 2015.

[129] Julien Plu, Giuseppe Rizzo, and Raphaël Troncy. Revealing entities from textual documents using a hybrid approach. In *3rd International Workshop on NLP & DBpedia*, 2015.

[130] Julien Plu, Giuseppe Rizzo, and Raphaël Troncy. Enhancing entity linking by combining ner models. In *13th European Semantic Web Conference, Open Extraction Challenge*, 2016.

[131] Julien Plu, Giuseppe Rizzo, and Raphaël Troncy. Adel: Adaptable entity linking. *Semantic Web Journal (SWJ), Special Issue on Linked Data for Information Extraction*, (to appear), 2019.

[132] Julien Plu, Raphaël Troncy, and Giuseppe Rizzo. Adel : une méthode adaptative de désambiguïsation d'entités nommées. In *28emes Journées francophones d'Ingénierie des Connaissances*, 2017.

[133] Julien Plu, Raphaël Troncy, and Giuseppe Rizzo. Adel@oke 2017: A generic method for indexing knowledge bases for entity linking. In *14th European Semantic Web Conference, Open Extraction Challenge*, 2017.

[134] Simone Paolo Ponzetto and Michael Strube. Exploiting semantic role labeling, wordnet and wikipedia for coreference resolution. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, 2006.

[135] Hoifung Poon and Pedro Domingos. Joint unsupervised coreference resolution with markov logic. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2008.

[136] Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, 2012.

[137] Roman Prokofyev, Alberto Tonon, Michael Luggen, Loic Vouilloz, Djellel Eddine Difallah, and Philippe Cudré-Mauroux. Sanaphor: Ontology-based coreference resolution. In *International Semantic Web Conference (1)*, 2015.

[138] Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. A multi-pass sieve for coreference resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, 2010.

[139] Altaf Rahman and Vincent Ng. Narrowing the modeling gap: A cluster-ranking approach to coreference resolution. *J. Artif. Intell. Res. (JAIR)*, 2011.

[140] Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual*

Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1, 2011.

[141] Lisa F. Rau. Extracting company names from text. In *Proceedings of the Seventh Conference on Artificial Intelligence Applications CAIA-91 (Volume II: Visuals)*, 1991.

[142] J. Reuter, J. Pereira-Martins, and J. Kalita. Segmenting twitter hashtags. *International Journal on Natural Language Computing(IJNLC)*, 2016.

[143] 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*, 2015.

[144] Giuseppe Rizzo, Oscar Corcho, Raphaël Troncy, Julien Plu, Juan Carlos Ballesteros Hermida, and Ahmad Assaf. The 3cixty knowledge base for expo milano 2015: Enabling visitors to explore the city. In *8th International Conference on Knowledge Capture*, 2015.

[145] 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.

[146] 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.

[147] Giuseppe Rizzo, Raphaël 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 Garcia. 3cixty@expo milano 2015 enabling visitors to explore a smart city. In *14th International Semantic Web Conference, Semantic Web Challenge*, 2015.

[148] 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.

[149] Mahalakshmi G S, Betina Antony, Akshaya Kumar, and Bagawathi Roshini. Domain based named entity recognition using naive bayes classification. *Australian Journal of Basic and Applied Sciences*, 2016.

[150] Erik F. Tjong Kim Sang. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*, 2002.

[151] Erik F. Tjong Kim Sang and Sabine Buchholz. Introduction to the conll-2000 shared task: Chunking. In *2nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning - Volume 7*, 2000.

[152] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In 7^{th} *Conference on Natural Language Learning (HLT-NAACL)*, 2003.

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

[154] Satoshi Sekine. Nyu: Description of the japanese ne system used for met-2. In *Proceedings of the Seventh Message Understanding Conference (MUC-7*, 1998.

[155] Ying Shan, T. Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[156] S. P. Sharmila and P. K. Sujatha. Segmentation based representation for tweet hashtag. In *2015 Seventh International Conference on Advanced Computing (ICoAC)*, 2015.

[157] Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. Linden: Linking named entities with knowledge base via semantic knowledge. In *Proceedings of the 21st International Conference on World Wide Web*, 2012.

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

[159] Thoudam Doren Singh, Kishorjit Nongmeikapam, Asif Ekbal, and Sivaji Bandyopadhyay. Named entity recognition for manipuri using support vector machine. In *PACLIC*, 2009.

[160] Dezhao Song, Frank Schilder, Shai Hertz, Giuseppe Saltini, Charese Smiley, Phani Nivarthi, Oren Hazai, Dudi Landau, Mike Zaharkin, Tom Zielund, Hugo Molina-Salgado, Chris Brew, and Dan Bennett. Building and querying an enterprise knowledge graph. *IEEE Transactions on Services Computing*, 2016.

[161] Mauro Sozio and Aristides Gionis. The community-search problem and how to plan a successful cocktail party. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2010.

[162] René Speck and Axel-Cyrille Ngonga Ngomo. Ensemble learning for named entity recognition. In *13th International Semantic Web Conference (ISWC)*, 2014.

[163] René Speck and Axel-Cyrille Ngonga Ngomo. Ensemble learning for named entity recognition. In *The Semantic Web – ISWC 2014*, 2014.

[164] RenÃľ Speck, Michael RÃűder, Sergio Oramas, Luis Espinosa-Anke, and Axel-Cyrille Ngonga Ngomo. Open knowledge extraction challenge 2017. In *14th European Semantic Web Conference (ESWC)*, 2017.

[165] Nadine Steinmetz and Harald Sack. Semantic multimedia information retrieval based on contextual descriptions. In *Proceedings of the 10th Extended Conference of Semantic Web (ESWC)*, 2013.

[166] Veselin Stoyanov and Jason Eisner. Easy-first coreference resolution. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, 8-15 December 2012, Mumbai, India*, 2012.

[167] Michael Strube and Simone Paolo Ponzetto. Wikirelate! computing semantic relatedness using wikipedia. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, 2006.

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

[169] Gerry Tesauro, David Gondek, Jonathan Lenchner, James Fan, and John M. Prager. Analysis of watson's strategies for playing jeopardy! *Journal of Artificial Intelligence Research*, 2013.

[170] Peter Korosec Tome Eftimov, Barbara Korousic Seljak. A rule-based named-entity recognition method for knowledge extraction of evidence-based dietary recommendations. *PLoS ONE*, 2017.

[171] Quan Tran, Andrew MacKinlay, and Antonio Jimeno-Yepes. Named entity recognition with stack residual lstm and trainable bias decoding. *CoRR*, 2017.

[172] Raphaël Troncy, Giuseppe Rizzo, Anthony Jameson, Oscar Corcho, Julien Plu, Enrico Palumbo, Juan Carlos Ballesteros Hermida, Adrian Spirescu, Kai-Dominik Kuhn, Catalin Barbu, Matteo Rossi, Irene Celino, Rachit Agarwal, Christian Scanu, Massimo Valla, and Timber Haaker. 3cixty: Building comprehensive knowledge bases for city exploration. *Journal of Web Semantics*, 2015.

[173] Alan M. Turing. Computing machinery and intelligence. *Mind*, 1950.

[174] Stanford University. Deep learning. http://cs230.stanford.edu/.

[175] Olga Uryupina, Massimo Poesio, Claudio Giuliano, and Kateryna Tymoshenko. Disambiguation and filtering methods in using web knowledge for coreference resolution. In *FLAIRS Conference*, 2011.

[176] 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.

[177] 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.

[178] Kees van Deemter and Rodger Kibble. What is coreference, and what should coreference annotation be? In *Workshop on Coreference and Its Applications*, 1999.

[179] Kees van Deemter and Rodger Kibble. On coreferring: Coreference in muc and related annotation schemes. *Computational Linguistics*, 2000.

[180] Marieke van Erp, Pablo Mendes, Heiko Paulheim, Filip Ilievski, Julien Plu, Giuseppe Rizzo, and Joerg Waitelonis. Evaluating entity linking: An analysis of current benchmark datasets and a roadmap for doing a better job. In *10th edition of the Language Resources and Evaluation Conference*, 2016.

[181] Vasudeva Varma, Praveen Bysani, Kranthi Reddy, Vijay Bharath Reddy, Sudheer Kovelamudi, Srikanth Reddy Vaddepally, Radheshyam N, Kiran Kumar

N, Santhosh Gsk, and Prasad Pingali. Iiit hyderabad in guided summarization and knowledge base population. In *Proceedings of the Text Analytics Conference Workshop*, 2010.

[182] W3C. Http rfc. https://tools.ietf.org/html/rfc2616.

[183] W3C. Hyperlinks definition. https://www.w3.org/Provider/ServerWriter.html.

[184] W3C. Iri rfc. https://tools.ietf.org/html/rfc3987.

[185] W3C. Rdf. https://www.w3.org/RDF/.

[186] Peilu Wang, Yao Qian, Frank K. Soong, Lei He, and Hai Zhao. A unified tagging solution: Bidirectional lstm recurrent neural network with word embedding. *CoRR*, 2015.

[187] Gerhard Weikum and Martin Theobald. From information to knowledge: Harvesting entities and relationships from web sources. In *Proceedings of the Twenty-ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2010.

[188] Joseph Weizenbaum. Eliza-a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 1966.

[189] Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. Learning global features for coreference resolution. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California*, 2016.

[190] Sam Wiseman, Alexander M. Rush, Stuart M. Shieber, and Jason Weston. Learning anaphoricity and antecedent ranking features for coreference resolution. In *53^{rd} Annual Meeting of the Association for Computational Linguistics and the 7^{th} International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, 2015.

[191] Ledell Wu, Adam Fisch, Sumit Chopra, Keith Adams, Antoine Bordes, and Jason Weston. Starspace: Embed all the things! arXiv:1709.03856, 2017.

[192] Yu-Chieh Wu, Teng-Kai Fan, Yue-Shi Lee, and Show-Jane Yen. Extracting named entities using support vector machines. In *Knowledge Discovery in Life Science Literature: PAKDD 2006 International Workshop, KDLL 2006, Singapore, April 9, 2006. Proceedings*, 2006.

[193] Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. Joint learning of the embedding of words and entities for named entity disambiguation. *CoRR*, 2016.

[194] Xiaofeng Yang, Jian Su, Jun Lang, Chew Lim Tan, Ting Liu, and Sheng Li. An entity-mention model for coreference resolution with inductive logic programming. In *ACL*, 2008.

[195] Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou, and Hinrich Schütze. Simple question answering by attentive convolutional neural network. In *COLING*, 2016.

[196] Shihong Yu, Shuanhu Bai, and Paul Wu. Description of the kent ridge digital labs system used for muc-7. In *In Proceedings of the MUC-7*, 1998.

[197] Feifei Zhai, Saloni Potdar, Bing Xiang, and Bowen Zhou. Neural models for sequence chunking. In *AAAI*, 2017.

[198] Lei Zhang and Achim Rettinger. X-lisa: Cross-lingual semantic annotation. In *Proc. VLDB Endow.*, 2014.

[199] Wei Zhang, Yan Chuan Sim, Jian Su, and Chew Lim Tan. Nus-i2r: Learning a combined system for entity linking. In *Proceedings of the Text Analytics Conference Workshop*, 2010.

[200] Wei Zhang, Yan Chuan Sim, Jian Su, and Chew Lim Tan. Entity linking with effective acronym expansion, instance selection, and topic modeling. In *International Joint Conference on Artificial Intelligence*, 2011.

[201] Wei Zhang, Jian Su, and Chew Lim Tan. A wikipedia-lda model for entity linking with batch size changing instance selection. In *International Joint Conference on Natural Language Processing*, 2011.

[202] Wei Zhang, Chew Lim Tan, Jian Su, Bin Chen, Wenting Wang, Zhiqiang Toh, Yanchuan Sim, Yunbo Cao, and Chin-Yew Lin. I2r-nus-msra at tac 2011: Entity linking. In *Proceedings of the Text Analysis Conference*, 2011.

[203] GuoDong Zhou and Jian Su. Named entity recognition using an hmm-based chunk tagger. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, 2002.

[204] Stefan Zwicklbauer, Christin Seifert, and Christin Granitzer. Doser - a knowledge-base-agnostic framework for entity disambiguation using semantic embeddings. In *Proceedings of the 13th International Conference on The Semantic Web. Latest Advances and New Domains*, 2016.

[205] Stefan Zwicklbauer, Christin Seifert, and Michael Granitzer. Robust and collective entity disambiguation through semantic embeddings. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2016.

A

List Of Publications

1. Julien Plu, Giuseppe Rizzo, and Raphaël Troncy. A hybrid approach for entity recognition and linking. In *12th European Semantic Web Conference, Open Extraction Challenge*, 2015

2. Giuseppe Rizzo, Oscar Corcho, Raphaël Troncy, Julien Plu, Juan Carlos Ballesteros Hermida, and Ahmad Assaf. The 3cixty knowledge base for expo milano 2015: Enabling visitors to explore the city. In *8th International Conference on Knowledge Capture*, 2015

3. Julien Plu, Giuseppe Rizzo, and Raphaël Troncy. Revealing entities from textual documents using a hybrid approach. In *3rd International Workshop on NLP & DBpedia*, 2015

4. Julien Plu, Giuseppe Rizzo, and Raphaël Troncy. An experimental study of a hybrid entity recognition and linking system. In *14th International Semantic Web Conference, Poster Demo Session*, 2015

5. Giuseppe Rizzo, Raphaël 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 Garcia. 3cixty@expo milano 2015 enabling visitors to explore a smart city. In *14th International Semantic Web Conference, Semantic Web Challenge*, 2015

6. Julien Plu. Knowledge extraction in web media: At the frontier of nlp, machine learning and semantics. In *25th World Wide Web Conference, PhD Symposium*, 2016

7. 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

8. Marieke van Erp, Pablo Mendes, Heiko Paulheim, Filip Ilievski, Julien Plu, Giuseppe Rizzo, and Joerg Waitelonis. Evaluating entity linking: An analysis of current benchmark datasets and a roadmap for doing a better job. In *10th edition of the Language Resources and Evaluation Conference*, 2016

9. Filip Ilievski, Giuseppe Rizzo, Marieke van Erp, Julien Plu, and Raphaël Troncy. Context-enhanced adaptive entity linking. In *10th edition of the Language Resources and Evaluation Conference*, 2016

10. Julien Plu, Giuseppe Rizzo, and Raphaël Troncy. Enhancing entity linking by combining ner models. In *13th European Semantic Web Conference, Open Extraction Challenge*, 2016

11. Julien Plu, Raphaël Troncy, and Giuseppe Rizzo. Adel@oke 2017: A generic method for indexing knowledge bases for entity linking. In *14th European Semantic Web Conference, Open Extraction Challenge*, 2017

12. Julien Plu, Raphaël Troncy, and Giuseppe Rizzo. Adel : une méthode adaptative de désambiguïsation d'entités nommées. In *28emes Journées francophones d'Ingénierie des Connaissances*, 2017

13. Raphaël Troncy, Giuseppe Rizzo, Anthony Jameson, Oscar Corcho, Julien Plu, Enrico Palumbo, Juan Carlos Ballesteros Hermida, Adrian Spirescu, Kai-Dominik Kuhn, Catalin Barbu, Matteo Rossi, Irene Celino, Rachit Agarwal, Christian Scanu, Massimo Valla, and Timber Haaker. 3cixty: Building comprehensive knowledge bases for city exploration. *Journal of Web Semantics*, 2015

14. Julien Plu, Giuseppe Rizzo, and Raphaël Troncy. Adel: Adaptable entity linking. *Semantic Web Journal (SWJ), Special Issue on Linked Data for Information Extraction*, (to appear), 2019