MeMAD Deliverable

D3.1 TV programme annotation model

Grant Agreement number	780069
Action Acronym	MeMAD
Action Title	Methods for Managing Audiovisual Data: Combining Automatic Efficiency with Human Accuracy
Funding Scheme	H2020-ICT-2016-2017/H2020-ICT-2017-1
Version date of the Annex I against which the assessment will be made	3.10.2017
Start date of the project	1.1.2018
Due date of the deliverable	31.12.2018
Actual date of submission	09.01.2019
Lead beneficiary for the deliverable	EURECOM
Dissemination level of the deliverable	Public

Action coordinator's scientific representative
Prof. Mikko Kurimo
AALTO –KORKEAKOULUSÄÄTIÖ, Aalto University School of Electrical Engineering, Department of Signal Processing and Acoustics
mikko.kurimo@aalto.fi

MeMAD project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 780069. This document has been produced by theMeMAD project. The content in this document represents the views of the authors, and the European Commission has no liability in respect of the content.

Authors in alphabetical order		
Name	Beneficiary	e-mail
Ismaïl Harrando	EURECOM	ismail.harrando@eurecom.fr
Benoit Huet	EURECOM	benoit.huet@eurecom.fr
Raphaël Troncy	EURECOM	raphael.troncy@eurecom.fr
Jean Carrive	INA	jcarrive@ina.fr
Steffen Lalande	INA	slalande@ina.fr
Michael Stormbom	Lingsoft	michael.stormbom@lingsoft.fi
Tiina Lindh-Knuutila	LLS	tiina.lindh-knuutila@lingsoft.fi
Lauri Saarikoski	YLE	lauri.saarikoski@yle.fi
Kim Viljanen	YLE	kim.viljanen@yle.fi

Abstract

This deliverable primarily describes the MeMAD ontology which builds on top of the EBU Core data model. A number of additional classes and properties are defined in order to cope with the original set of metadata delivered by the MeMAD data providers (INA and Yle). Furthermore, this deliverable describes two tools that enable to convert the legacy metadata coming from both INA and Yle into RDF, the W3C standard for representing knowledge graph on the web, following the MeMAD ontology. During this process, metadata is harmonized and enriched semantically enabling to perform queries across sources.

This deliverable describes also a number of tools that perform named entity recognition and disambiguation on both automatic transcription and true subtitles of TV programs. These tools existed prior to the beginning of MeMAD but they have been further improved and developed during the first year of the project (e.g. development of API, specific training to MeMAD audiovisual material).

The converter tools are developed in open source and are publicly available from the MeMAD GitHub account. A scientific publication is already appended to this report which describes a novel ensemble methods to extract and disambiguate entities from different kinds of text including timed text (subtitles of TV programs) and which has been investigated during the first 6 months of the project. Several improvements and formal evaluations of those tools are already foreseen to happen during the second year of the project.

Table of Content

1. Introduction

Multimedia systems typically contain digital documents of mixed media types, which are indexed on the basis of strongly divergent metadata standards. This severely hampers the inter-operation of such systems. Therefore, machine understanding of metadata coming from different applications is a basic requirement for the inter-operation of distributed multimedia systems. Furthermore, the content will be processed by automatic multimedia analysis tools which have their own formats for exchanging their results. One of the main goals of MeMAD is to enrich seed video content with additional content that come from diverse sources including broadcast archives, web media, news and photo stock agencies or social networks.

The general methodology that we follow consists in: *i)* semantifying the legacy metadata coming with audiovisual content (program metadata coming from the producer, the broadcaster and/or the archive) and *ii)* automatically extracting concepts and entities from the true subtitles or the text generated by automatic speech recognition on the audiovisual content. The resulting knowledge graph can then be used to infer additional information in order to enrich and hyperlink key video content moments.

In this deliverable, we first study the diversity of metadata models by proposing a comprehensive overview of numerous multimedia metadata formats and standards that have been proposed by various communities: broadcast industry, multimedia analysis industry, news and photo industry, web community (Section 2). Based on this survey, we have selected the EBU Core data model which we have extended to propose the MeMAD ontology (Section 3). We describe the converter tools we have developed for both INA (Section 4) and Yle (Section 5) which are the MeMAD data providers. Finally, we describe in Section 6 several tools that perform named entity recognition and disambiguation on transcriptions and subtitles, initially for some common types (person, organization, location, etc.) and some languages (English, French, Finnish, Swedish) and that will be further extended during the second year.

2. State of the art : ontologies for TV/Media content

The broadcast industry has developed several metadata formats for representing TV programs, their broadcast information or targeted audience and their content in order to generate Electronic Program Guides. In this section, we review those different standards. First, we describe the XML-based formats such as DVB, BMF developed by the German broadcaster ARD and TV Anytime. Second, we present more recent models that are largely inspired by the Semantic Web technologies such as the BBC Programmes ontology and the EBU standard (together with its application in EU Screen and Europeana).

2.1 DVB metadata model

The Digital Video Broadcasting Project (DVB[1]) is an industry-led consortium of around 250 broadcasters, manufacturers, network operators, software developers, regulatory bodies and others in over 35 countries committed to designing open technical standards for the global delivery of digital television and data services.

The DVB metadata model is composed of various XML Schemas:

- DVB Classification Scheme schema: http://www.dvb.org/metadata/schema/dvbCSschema.xsd
- Content Item Information which uses mostly MPEG7 and TV Anytime content types: http://www.dvb.org/metadata/schema/ContentItemInformation.xsd
- File Content Item Information with duration and geolocation information: http://www.dvb.org/metadata/schema/FileContentItemDescription.xsd

The DVB transport stream includes metadata called Service Information (DVB-SI). This metadata delivers information about transport stream as well as a description for service / network provider and programme data to generate an EPG and further programme information. The Service Information information tables which are of interest for MeMAD are the EIT (Event Information Table) and the SDT (Service Description Table).

The EIT contains additional sub tables with information about the present and following events by each service. This includes:

- Start time (Start time of the event)
- Duration (Duration of the event)
- Short event descriptor (Name and a short description of the current event)
- Extended event descriptor (Detailed long text description of the event)
- Content descriptor (Classification of the event)

The SDT delivers particular information about the service of the current transport stream such as the Service name and the Service identification. The content descriptor from the EIT table defines a classification schema for a programme event. It provides various genre categories using a two-level hierarchy. First it specifies a first (top) level genre which is

[1] http://www.dvb.org/metadata/index.xml

categorized more specifically in the second level. The top level branch contains about 12 genres (with several sub genres): Undefined, Movie/Drama, News/Current affairs, Show/Game show, Sports, Children's/Youth programs, Music/Ballet/Dance, Arts/Culture (without music), Social/Political issues/Economics, Education/Science/Factual topic, Leisure hobbies, Special characteristics. Each top level genre contains several sub genres describing the content of the current broadcast more specifically. The classification information is encoded in the EIT table using 4-bit fields assigned to each level within DVB transport stream.

2.2 ARD BMF

The Broadcast Metadata Exchange Format Version 2.0 (BMF 2.0[2]) has been developed by IRT (*Institut für Rundfunktechnik / Broadcast Technology Institute*) in close cooperation with German public broadcasters with focus on the harmonization of metadata and the standardized exchange thereof. The standard particularly reflects the requirements of public broadcasters. BMF contains metadata vocabulary for TV, radio and online content and defines a standardized format for computer-based metadata exchange. It facilitates the reuse of metadata implementations and increases the interoperability between both computer-based systems and different use case scenarios.

BMF enables to describe TV, radio and online content as well as production, planning, distribution and archiving of the content. Metadata in BMF are represented in XML documents while the structure for the XML metadata is formalized in an XML Schema. The latest version of the format is the version BMF 2.0 Beta[3].

2.3 TV Anytime

The TV-Anytime Forum is a global association of organizations founded in 1999 in USA focusing on developing specifications for audio-visual high volume digital storage in consumer platforms (local AV data storage). These specifications for interoperable and integrated systems should serve content creators/providers, service providers, manufacturers and consumers. The forum created a working group for developing a metadata specification, so-called TV-Anytime[4] and composed of:

- Attractors/descriptors used e.g. in Electronic Program Guides (EPG), or in web pages to describe content (information that the consumer – human or intelligent agent – can use to navigate and select content available from a variety of internal and external sources).
- User preferences, representing user consumption habits, and defining other information (e.g. demographics models) for targeting a specific audience.

[2] http://www.irt.de/en/activities/production/bmf.html
[3] http://bmf.irt.de/en
[4] http://www.tv-anytime.org/

- Describing segmented content. Segmentation Metadata is used to edit content for partial recording and non-linear viewing. In this case, metadata is used to navigate within a piece of segmented content.
- Metadata fragmentation, indexing, encoding and encapsulation (transport-agnostic).

2.4 BBC Programmes Ontology

The British Broadcasting Corporation (BBC) is one of the largest broadcasters in the world. One of the main resources used to describe programmes is the so-called Programmes ontology[5]. This ontology provides the concepts of brands, series (seasons), episodes, broadcast events, broadcast services, etc. and it is modeled in OWL/RDF. The design of this ontology is based on the Music Ontology and the FOAF Vocabulary. The programmes model is based on the PIPS database schema used previously at the BBC. It describes content in terms of: Brands, Series, Episodes and Programs.

Publishing is then described in terms of Versions of episodes and Broadcasts. Versions are temporally annotated. Publishing of content is related to medium, that is described in terms of: Broadcaster, Service-outlet and Channel. This conceptual scheme describes how brands, series, episodes, particular versions of episodes and broadcasts interact with each other. The BBC Programmes ontology also re-uses other ontologies such as FOAF to express a relationship between a programme to one of its actors (a person who plays the role of a character)

2.5 EBUCore

The *EBU* (European Broadcasting Union) is the collective organization of Europe's 75 national broadcasters claiming to be the largest association of national broadcasters in the world. EBU's technology arm is called EBU Technical. EBU represents an influential network in the media world. The EBU projects on metadata are part of the Media Information Management (MIM) Strategic Programme. MIM benefits from the expertise of the EBU Expert Community on Metadata (EC-M), for which the participation is open to all metadata experts, or users and implementers keen to learn and contribute.

The *EBUCore* (EBU Tech 3293) is the main result of this effort to date and the flagship of EBU's metadata specifications. It can be combined with the Class Conceptual Data Model of simple business objects to provide the appropriate framework for descriptive and technical metadata for use in Service Oriented Architectures. It can also be used in audiovisual ontologies for Semantic Web and Linked Data environment. EBUCore has a relatively high adoption rate around the world. It is also referenced by the UK DPP (Digital Production Partnership). All EBU metadata specifications are coherent with the EBU Class Conceptual Data Model or CCDM (EBU Tech 3351).

[5] http://purl.org/ontology/po/

EBUCore is the foundation of technical metadata in FIMS 1.0 (Framework for Interoperable Media Service)[6]. IMS is currently under development. It embodies the idea of sites like Google, Twitter, YouTube and many other web sites that offer service interfaces to remotely initiate an action, export data, import a file, query for something, etc. FIMS specifies how media services should operate and cooperate in a professional, multi-vendor, IT environment – not just through a web site interface

EBUCore has been used by several European projects such as NoTube and VisionCloud, EUSCreen (the European portal to public broadcasting archives), by Deutsche Welle in Germany, RAI in Italy, RTP in Portugal, Bloomberg, A&E, Turner, CBC in the US and Canada.

EBUCore is published under the Creative Commons license. Users and implementers have the freedom to change EBUCore to address their respective needs. They should mention that the new specification is based on EBUCore. This flexibility is also one of the reasons why this standard has been chosen as the basis of the MeMAD ontology that we further describe in the next section.

[6] https://www.ebu.ch/contents/news/2012/10/fims-10-jointly-published-by-ebu.html

3. MeMAD ontology and controlled vocabularies

The MeMAD ontology largely re-uses EBUCore as a backbone to define most first-class objects and relations. Furthermore, to model some specific metadata from the MeMAD data providers (INA and Yle), we also define 3 new classes and 10 new properties. The MeMAD ontology provides mappings between the legacy metadata models of INA and Yle with the standard EBUCore data model and could therefore be used by those industries to improve their metadata interoperability systems.

The labels of classes and properties are provided in both English and French. It is our aim to add labels in more languages such as Finnish, Swedish, etc.

3.1 Classes

memad:Record

`http://data.memad.eu/ontology#Record`

rdfs:subClassOf	ebucore:BibliographicalObject
rdfs:label	Record@en Notice@fr
rdfs:comment	Defines a bibliographical object describing any other editorial object (Programme, Part, ..)

memad:Timeslot

`http://data.memad.eu/ontology#Timeslot`

rdfs:subClassOf	ebucore:Collection
rdfs:label	Timeslot@en Tranche horaire@fr
rdfs:comment	Defines a collection of programs that are scheduled on a given period or time interval, e.g. *“Les matins de France Culture”, ”Mercredi c'est ciné”*

memad:FirstRun

`http://data.memad.eu/ontology#FirstRun`

rdfs:subClassOf	ebucore:PublicationEvent
rdfs:label	FirstRun@en Première diffusion@fr
rdfs:comment	Links a program to its first publication event (when provided).

3.2 Properties

memad:lead

`http://data.memad.eu/ontology#lead`

rdfs:subPropertyOf	ebucore:description
rdfs:label	lead@en chapeau@fr
rdfs:domain	ebucore:EditorialObject
rdfs:range	String
rdfs:comment	A short summary of the programme

memad:titleNote

`http://data.memad.eu/ontology#titleNote`

rdfs:subPropertyOf	ebucore:description
rdfs:label	title note@en note de titre@fr
rdfs:domain	ebucore:EditorialObject
rdfs:range	String
rdfs:comment	A note to further describe the title of the programme

memad:producerSummary

`http://data.memad.eu/ontology#producerSummary`

rdfs:subPropertyOf	ebucore:description
rdfs:label	producer summary@en résumé du producteur@fr
rdfs:domain	ebucore:EditorialObject
rdfs:range	String
rdfs:comment	A short summary provided by the producer of the programme

memad:sequence

`http://data.memad.eu/ontology#sequence`

rdfs:subPropertyOf	ebucore:description
rdfs:label	sequence@en séquence@fr
rdfs:domain	ebucore:EditorialObject
rdfs:range	String
rdfs:comment	

memad:hardware

`http://data.memad.eu/ontology#hardware`

rdfs:subPropertyOf	ebucore:resourceDescription
rdfs:label	hardware@en matériel@fr
rdfs:domain	ebucore:MediaResource
rdfs:range	String
rdfs:comment	The hardware being used for storing this media resource

memad:hasRecord

`http://data.memad.eu/ontology#hasRecord`

rdfs:subPropertyOf	ebucore:references
rdfs:label	has record@en
rdfs:domain	ebucore:EditorialObject
rdfs:range	memad:Record
rdfs:comment	

memad:legalNote

`http://data.memad.eu/ontology#legalNote`

rdfs:subPropertyOf	skos:note
rdfs:label	legal note@en note juridique@fr
rdfs:domain	ebucore:EditorialObject
rdfs:range	String
rdfs:comment	A legal note attached to this editorial object

memad:hasISANIdentifier

`http://data.memad.eu/ontology#hasISANIdentifier`

rdfs:subPropertyOf	ebucore:hasIdentifier
rdfs:label	has ISANI identifier@en identifiant ISANI@fr
rdfs:domain	ebucore:EditorialObject
rdfs:range	String
rdfs:comment	The ISANI identifier for this program

memad:hasImediaIdentifier

`http://data.memad.eu/ontology#hasImediaIdentifier`

rdfs:subPropertyOf	ebucore:hasIdentifier
rdfs:label	has Imedia identifier@en identifiant Imedia@fr
rdfs:domain	ebucore:EditorialObject
rdfs:range	String
rdfs:comment	The Imedia identifier for this program

memad:hasMetroIdentifier

`http://data.memad.eu/ontology#hasMetroIdentifier`

rdfs:label	has Metro identifier@en identifiant Metro@fr
rdfs:subPropertyOf	ebucore:hasIdentifier
rdfs:range	String
rdfs:domain	ebucore:EditorialObject
rdfs:comment	The Metro identifier for this program

3.3 Controlled vocabulary

In addition to the MeMAD ontology, we also make use of a number of controlled vocabularies, from EBU Core or from the MeMAD data providers.

From EBUCore, we can use the following classification schemes:

- Genres: https://www.ebu.ch/metadata/cs/web/ebu_ContentGenreCS_p.xml.html
- Roles: https://www.ebu.ch/metadata/cs/web/ebu_RoleCodeCS_p.xml.htm
- Picture formats: https://www.ebu.ch/metadata/cs/web/tva_PictureFormatCS_p.xml.htm
- Languages: https://www.ebu.ch/metadata/cs/web/ebu_Iso639_1LanguageCodeCS_p.xml.htm
- Technical (codecs, file formats, aspect ratio..)

Additional vocabularies will be defined in the future for the following classes:

- Keywords (“*war*”, “*elections*”, ..)
- Themes (“*literature*”, “*politics*”, ..)

3.4 URI Design Policy

To identify the objects instantiating the classes defined in the MeMAD ontology, we define the following guidelines. The base namespace for all resources identified by MeMAD is: http://data.memad.eu/

3.4.1 String cleaning (slug generation)

We define a "string cleaning" (named also "slugify") process to transform any text string (e.g. a program title, a collection name) into a valid resource name following a number of character replacement rules.

1. Transform all accented characters into their ASCII counterpart (e.g. 'é' to 'e'), typically using unicodedata.normalize[7] in Python;
2. Replace all special characters appearing in the text (*'\', '/', '"', ',', '.', '"', ':', ';', '[', ']', '(', ')', '!', '?', ' ', '#', '=', '&', '$', '%', '@', '{', '«', '°', '»', '¿', '=', '>', '+', '**'*) with a *dash* '-';
3. Lowercase all characters in the string;
4. Remove successive duplicate dashes;
5. Remove the dashes positioned at the beginning and at the end of the string, if any.

Example:
"Qu'est-ce qu'on a fait au Bon Dieu ?" : succès d'un film, succès du multiculturalisme ?
Result:
qu-est-ce-qu-on-a-fait-au-bon-dieu-succes-d-un-film-succes-du-multiculturalisme

3.4.2 Naming schemes

3.4.2.1 Channels

Scheme: http://data.memad.eu/channel/[channel_code]

With channel_code:

- For INA, we use the 3 characters code[8] in lowercase identifying each channel
- For Yle, we use lowercase channel names, without spaces, i.e. : 'tvfinland', 'yle24', 'yleareena', 'yletv1', 'yletv2', 'yleteema', 'ylefem', 'yleteemafem'

Examples:
http://data.memad.eu/channel/fr2
http://data.memad.eu/channel/yle24

[7] https://docs.python.org/2/library/unicodedata.html#unicodedata.normalize
[8] https://docs.google.com/spreadsheets/d/1hzvJbLgz_PadKwwRaObsDSQnYtUzLxPmOotFZ6SgbME

3.4.2.2 Collections, Series, Timeslots

Scheme: http://data.memad.eu/[source]/[resource_title]
With:

- source is either *'yle'* for data coming from Yle, or the channel codename for INA.
- resource_title is the name or the title of the collection / series / timeslot once slugified

Examples:
http://data.memad.eu/fr2/les-chemins-de-la-foi
http://data.memad.eu/yle/stromso

3.4.2.3 Editorial Objects

Editorial objects represent media resources (i.e. TVProgramme, RadioProgramme, Episode, Part).
Scheme: http://data.memad.eu/[source]/[parent]/[UUID]
With:

- source is either *'yle'* for data coming from Yle, or the channel codename for INA.
- parent is the name of the series, collection or timeslot that this segment belongs to (in this order). If the resource does not have a parent collection, we use *orphan*.
- UUID is a hashed version of the resource's internal identifier (GUID for Yle, 'Identifiant de la notice' for INA's professional archive and 'Identifiant' for INA's legal deposit, respectively). We use the SHA-1 algorithm for hashing the internal identifier.

Examples:
http://data.memad.eu/yle/stromso/aceaea52f14631bfbedc478fc04c04be8f89c598
http://data.memad.eu/fr2/7h00-le-journal/fb9cd99182887aee143940765509a9c21bbcacf3

We use this URI as a basis to identify other resources attached to the Editorial Object:

- Subtitles: http://data.memad.eu/[source]/[parent]/[UUID]/subtitling/[n]
- Audio tracks: http://data.memad.eu/[source]/[parent]/[UUID]/audio/[n]
- Publication event: http://data.memad.eu/[source]/[parent]/[UUID]/publication/[n]
- Records: http://data.memad.eu/[source]/[parent]/[UUID]/record

With *n* as a unique sequence number for the resource.

3.4.2.4 Media Resources

Media resources represent the material instances of Editorial objects.
Scheme: http://data.memad.eu/media/[UUID]
With:

- UUID is a hashed version of the media's internal identifier ('METRO_PROGRAMME_ID' for Yle, 'Identifiant Matériels' for INA's professional

archive, ‘Identifiant de la notice’ for INA’s legal deposit). We use the SHA-1 algorithm for hashing the internal identifier.

Examples:
http://data.memad.eu/media/e8659ead515a671866e58d334cdc79720e84e3db

3.4.2.5 Agents

For all the agents credited in a programme or in a segment as a contributor.

Scheme: http://data.medad.eu/agent/[clean-agent-name]
With

- clean-agent-name is the name of the agent as mentioned in the credit, once slugified.

This strategy may generate duplicates that will be removed later.

4. INA conversion

The datasets provided by INA come from two sources: the *legal deposit* and the *professional archive*. Each source has a specific metadata format that is converted in RDF using the MeMAD ontology.

4.1. Legal deposit

The dataset from the INA legal deposit covers one month of programming (May 2014) from 88 French channels (13 radio channels and 75 TV channels).

The metadata is provided as CSV files and is separated into two types:

- Programs metadata (*"Emission"*) which describe the entire programs with fields such as *title*, *broadcasting date*, *broadcasting channel*, etc.
- Segments metadata (*"Sujets"*) which further detail the content of some parts of the programs in term of audiovisual analysis, keywords, description and participants.

Each entry in the *Emission* dump corresponds to either an ebucore:TVProgramme or an ebucore:RadioProgram (both subclasses of ebucore:Program).
Each entry in the *Sujet* dump corresponds to a ebucore:Part, and are subsequently linked to their parent program with ebucore:isPartOf.
Each program has a ebucore:PublicationEvent which links it to a ebucore:PublicationChannel. Every program is instantiated by a ebucore:MediaResource and is generally part of a ebucore:Collection and/or a memad:Timeslot.

We provide below an excerpt of a TV program metadata from the legal deposit (this actual row contains 21 columns of description fields):

Identifiant	Chaine	startDate	endDate	Genres	Duree Seconds	Titre Emission	Titre Collection	...
5249098_001	ARTE	2014-05-01 05:00:02	2014-05-01 05:05:04	Animation| Création audiovisuelle| Série	302	Téléchat : [rediffusion]	Téléchat	..

The following section explains how each field is mapped to a corresponding MeMAD class or property.

4.1.1 Mapping table

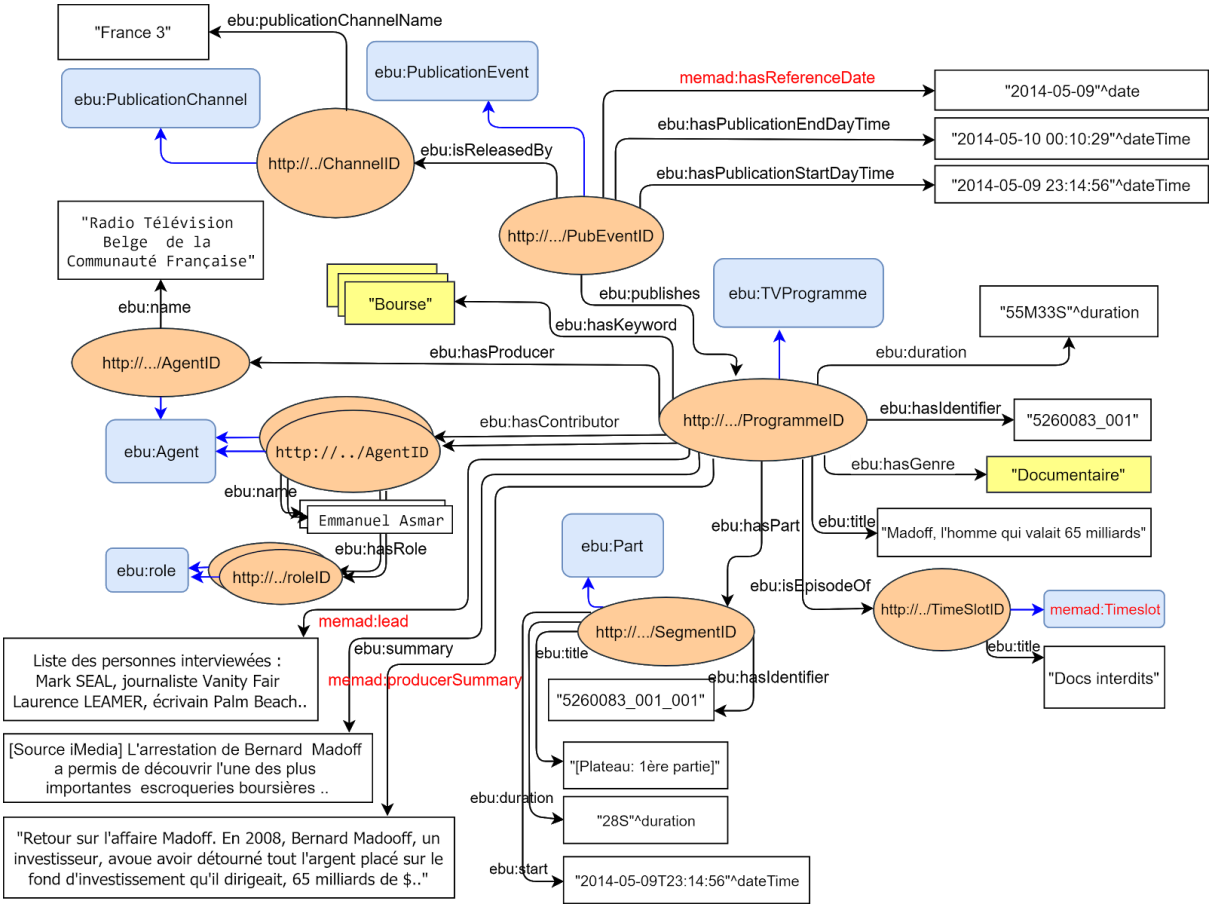
For programs:

Field	Class	Property
Identifiant	ebucore:Programme	ebucore:hasIdentifier
Chaine	ebucore:PublicationChannel	ebucore:publicationChannelName
startDate	ebucore:PublicationEvent	ebucore:publicationStartDateTime
endDate	ebucore:PublicationEvent	ebucore:publicationEndDateTime
DureeSecondes	ebucore:PublicationEvent	ebucore:duration
TitreEmission	ebucore:Programme	ebucore:title
TitreCollection	ebucore:Collection	ebucore:title
TitreTrancheHoraire	memad:Timeslot	ebucore:title
Resume	ebucore:Programme	ebucore:summary
Producteurs	ebucore:Programme	ebucore:hasProducer
Descripteurs	ebucore:Programme	ebucore:hasKeyword
Generiques	ebucore:Programme	ebucore:hasContributor ebucore:Agent ebucore:hasRole
Genres	ebucore:Programme	ebucore:hasGenre
Thematique	ebucore:Programme	ebucore:hasTheme
Dispositif	ebucore:Programme	ebucore:description
referenceDate	ebucore:PublicationEvent	memad:hasReferenceDate
Chapeau	ebucore:Programme	memad:head
ResumeProducteur	ebucore:Programme	memad:producerSummary

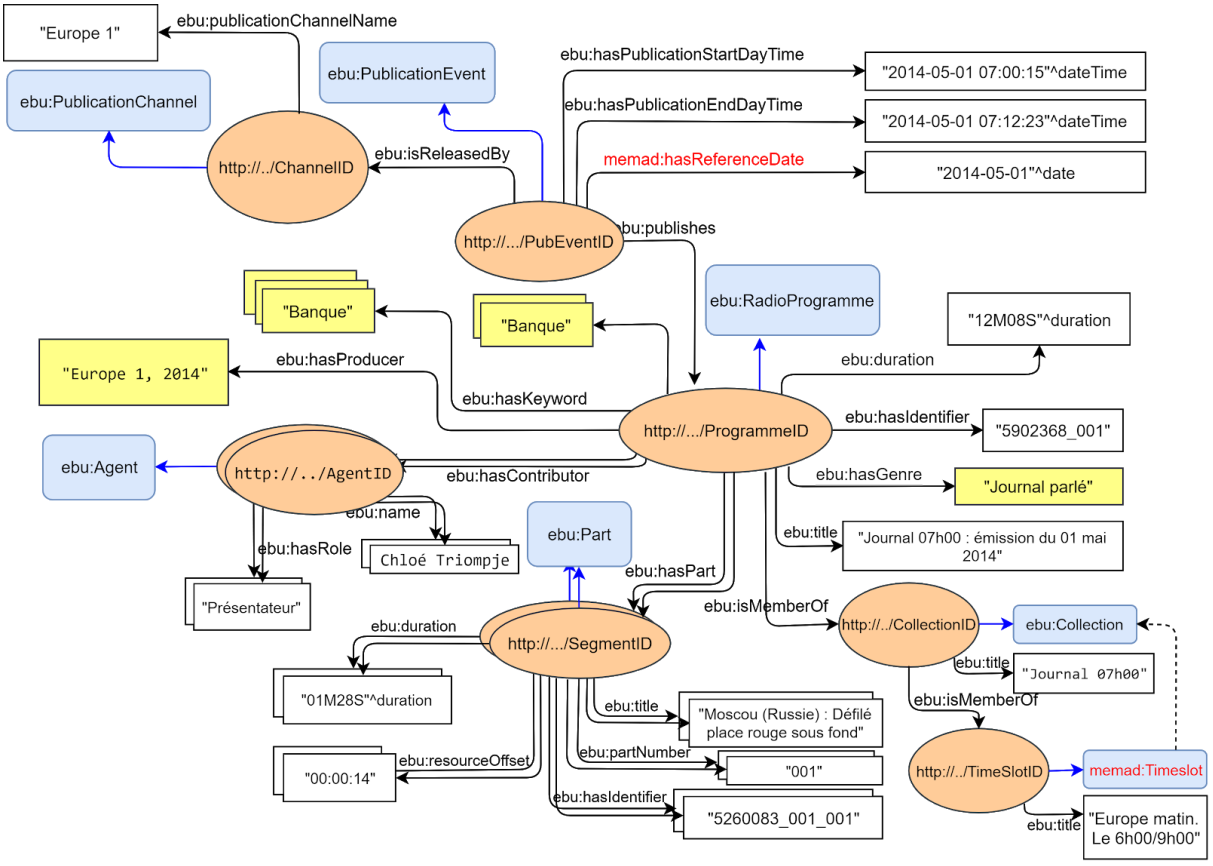
For Segments:

Field	Class	Property
Identifiant	ebucore:Part	ebucore:hasIdentifier
startDate	ebucore:Part	ebucore:start
DureeSecondes	ebucore:Part	ebucore:duration
Descripteurs	ebucore:Part	ebucore:hasKeyword
Generique	ebucore:Part	ebucore:hasContributor ebucore:Agent ebucore:hasRole

4.1.2 Examples



Example of a RDF graph representing a documentary TV program broadcasted on the France 3 TV channel on 2014-05-09



Example of a RDF graph representing a "spoken news" radio program broadcasted on the Europe 1 radio channel on 2014-05-01

4.2. Professional Archive

The professional archive dataset covers one week of programming within the month of May 2014, from 3 French channels (2 radio, 1 TV). There is therefore some overlap with the legal deposit dataset, but the description of the programs often go in much more details. The metadata is again provided as CSV files, without making any distinction between *programs* and *segments* metadata. However, unlike the legal deposit, there are some differences in metadata for TV and Radio programs.

Once again, each entry in the *Emission* data correspond to either an ebucore:TVProgramme, an ebucore:RadioProgram or a ebucore:Part.
Each program has a ebucore:PublicationEvent which links it to a ebucore:PublicationChannel, a ebucore:MediaResource and is part of a ebucore:Collection and/or a memad:Timeslot. We also have some metadata regarding the metadata records themselves (such as *type*, *creation date* and *last update date*).

We provide below an excerpt of the dataset (the actual row contains 95 columns):

Identifiant	Canal de diffusion	Générique (Aff. Lig.)	Date de diffusion	Date de modification	Descripteurs (Aff. Lig.)	...
5266008_001	2eme chaîne	REA Miramon, Philippe\n PRE Davant, Sophie\n PRE Moreau, Danielle	21/05/2014	19/05/2014	DET: recette de cuisine ; DET: mode ;	..

The fields are mapped to EBUCore / MeMAD classes and properties as described in the next section.

4.2.1 Mapping table

Field	Class	Property
Identifiant de la notice	ebucore:Programme	ebucore:hasIdentifier
Canal de diffusion	ebucore:PublicationChannel	ebucore:publicationChannelName
Date de création	memad:Record	ebucore:dateCreated
Date de diffusion	ebucore:PublicationEvent	ebucore:publicationStartDateTime
Date de modification	memad:Record	ebucore:dateModified
Durée	ebucore:Programme	ebucore:duration
Extension géographique	ebucore:PublicationEvent	ebucore:hasPublicationRegion
Titre propre	ebucore:Programme	ebucore:title
Titre collection	ebucore:Collection	ebucore:title
Titre tranche horaire	memad:Timeslot	ebucore:title
Langue de la notice	memad:Record	ebucore:language
Producteurs	ebucore:Programme	ebucore:hasProducer
Descripteurs	ebucore:Programme	ebucore:hasKeyword
Générique	ebucore:Programme	ebucore:hasContributor ebucore:Agent ebucore:hasRole
Genre	ebucore:Programme	ebucore:hasGenre
Séquences	ebucore:Programme	memad:sequence
Thématique	ebucore:Programme	ebucore:hasTheme
Résumé	ebucore:Programme	ebucore:summary
Notes	ebucore:Programme	skos:note
Notes du titre	ebucore:Programme	memad:titleNote
Notes juridiques	ebucore:Programme	memad:legalNote
Type de notice	memad:Record	ebucore:type

Identifiant Matériels	ebucore:MediaResource	ebucore:hasIdentifier
Chapeau	ebucore:RadioProgramme	memad:head

Mapping for the fields that are specific to radio programs only:

Field	Class	Property
Résumé producteur	ebucore:RadioProgramme	memad:producerSummary
Heure de diffusion	ebucore:PublicationEvent	ebucore:publicationStartDateTime

Mapping for the fields that are specific to TV programs only:

Field	Class	Property
Numéro ISAN	ebucore:TVProgramme	memad:hasISANIdentifier
Matériels	ebucore:MediaResource	memad:hardware
Matériels dispo (Détail)	ebucore:MediaResource	memad:hardware

4.2.2 Unmapped fields

The following fields were not mapped into EBUCore or MeMAD classes or relationships because: they were not valued (in the dataset provided), redundant (usually for display purposes), inconsistent, or too specific for INA's internal usage.

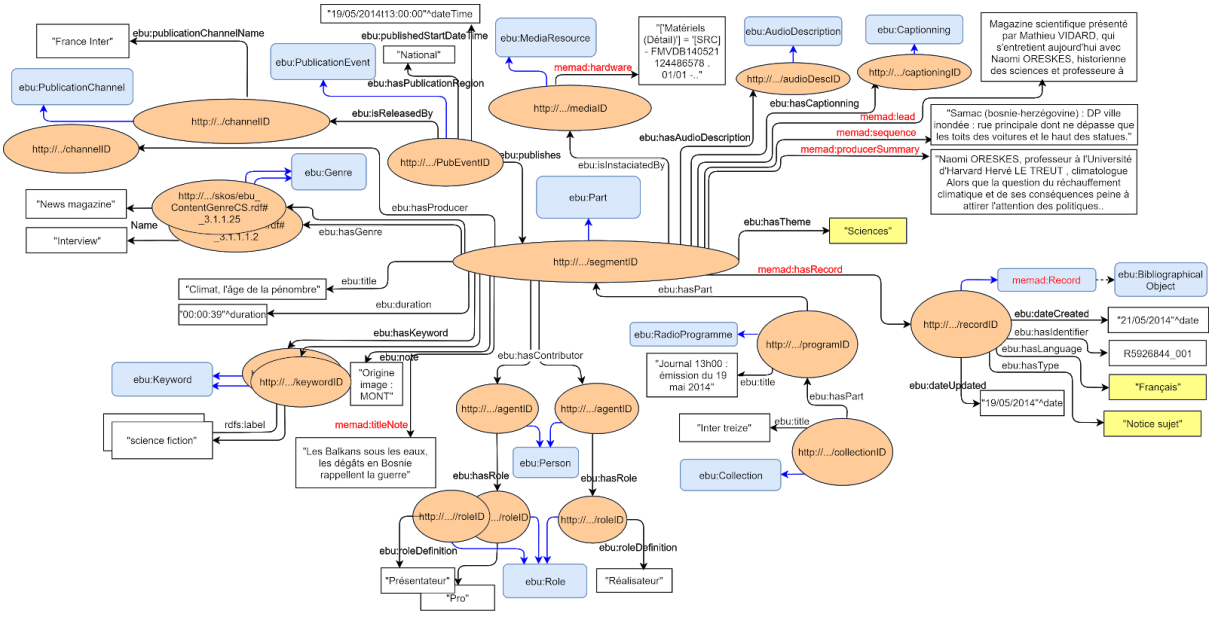
The list is as follows (for both TV and Radio programs):

Catalogage, Classe de niveau, Corpus, Date de niveau de catalogage, Date de niveau d'indexation, Diffusion, Document dévolu INA, Domaine, Ind. notice verrouillée, Indexation, Inventaire, Langue VO / VE, Lien, Lien de rediffusion, Mandat de l'émission, Mode de diffusion, N° Ordre dans collection, N° Ordre du vidéogramme, N° Série dans collection, N° Série dans sous-collection, Niveau d'indexation atteint, Oeuvres, Origine du fonds, Présence public, Public destinataire, Rediffusion, Société de programmes, Sous-titrage / doublage, Statut de numérisation, Statut Théma, Témoin niv. de catalog. validé, Témoin niv. d'indexation validé, Titre sous-collection, Type de fonds, Usage, Version courte / longue, Version originale / étrangère, Ancien lien, Corpus anglais, Date d'enregistrement, Gestion de matériel, Heure de diffusion, Langue sous-titrage / doublage, Lieu d'enregistrement, Matériel ori.(zone Mastock), Matériel type M, Matériel type MP, Matériel type P, Matériels Lien, Matériels Lien/Mastock (Détail), Matériels Mastock, Matref, Nom fichier, Séquences sonores, Source du fonds, Titre phonogramme, Anciens Supports, Corpus Anglais, Document fonds TF1, Dossier de production, Fichiers (Aff. abrégé), Fichiers), Gestion de documents, Identifiant

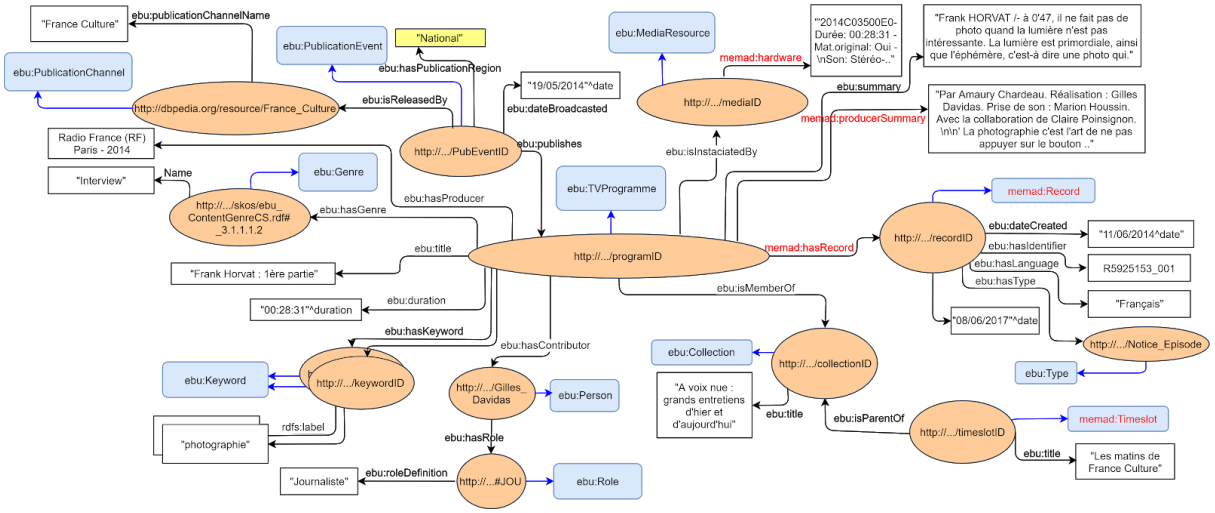
Matériels (info.), Lieu de rediffusion, Matériel de rediffusion, Multidiffusion, Nom fichier segmenté, Origine du fonds, Titre de collection de rediffusion, Titre vidéogramme.

Field	Sample values
Nature de production	Production propre, Mise à disposition de temps d'antenne, Coproduction, Mixte, Achat de droits commande, Achat de droits de diffusion
Dernier intervenant	DL, NON, MMO, DUB, DCD, CHC, MHW, AFA, SLD, JCU, SGT, AIN, DJL, JDE, NDU, FUN, JGN, SSC, ERR, PKA, MDS, LIF, RIE, VIC, MKA, PGG, MAD, JLG, SBN, GMI, DAH, SBE, CGI, HEV, AOA, UMI, JEA, VEI, PAU, ARN
Type de date	Diffusé, Multidiffusé, Non diffusé, Rediffusé
Correspondant de chaine	YB, FR2, JCC, mme, PGG, MAD, bar, lpi
Documentaliste	ugo, FR2, , bud, DUB, ajz, vay, BAT, aut, rf, cm1, pro, lei, rfm, bat, mdo, zan
Référence extérieure	GIB0044840891, GIB0044840911, ..
Thèque	CP (Vidéothèque production), CA (Vidéothèque actualités), PH (Phono)

4.2.3 Examples



A description of a sequence of a radio program from the professional archive



A description of an entire radio program from the professional archive

4.3 Statistics

We computed some statistics of the two INA datasets once converted in RDF.

Resource	Legal Deposit	Professional Archive
Temporal Coverage	2014/0501 to 2014/05/ 31	2014/05/19 to 2014/05/ 26
Records	190576	2118
TV Programs	89338	181
Radio Programs	18891	852
Segments	182347	1085
Collections	3602	305
Timeslots	438	21
Channels	87	3
Keywords	10999	1631
Genres	53	40
Agents	16015	1936
Roles	22	15
Producers	734	23

4.4 Conversion script

We use Python scripts to process the metadata files, depending on what fields they contain:

- For the legal deposit:
  - INA_LD_Emission2RDF.py for programs metadata
  - INA_LD_Sujet2RDF.py for segments metadata
- For the professional archive:
  - INA_PA_TV2RDF.py for TV metadata
  - INA_PA_Radio2RDF.py for Radio metadata

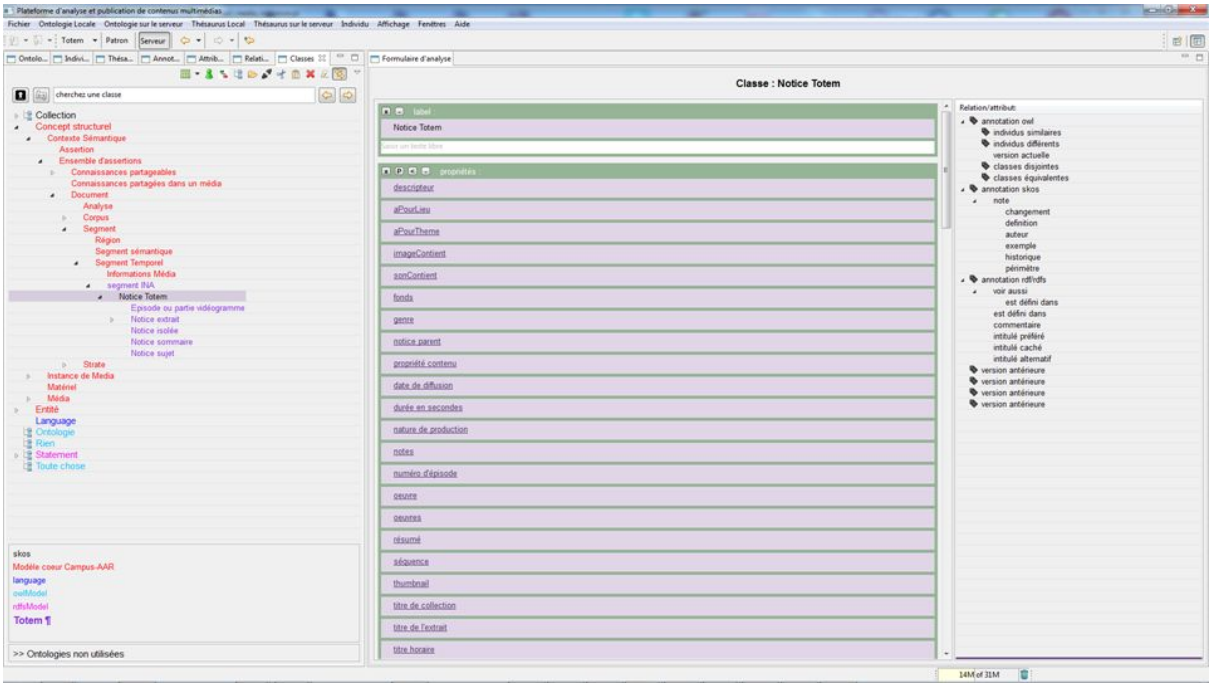
The scripts take as input the path to the CSV file containing the metadata, and output an RDF graph (serialized in Turtle[9]) in the same location. To process the input files, we use Pandas (https://pandas.pydata.org/) to read and manipulate the CSV tables, and RDFLib (https://github.com/RDFLib/rdflib) to generate the RDF graph and serialize them into Turtle files. A bash script is also provided to batch-process the entire dataset. All scripts are available in the Github repository at https://github.com/MeMAD-project/

4.5 INA Semantic Platform

In parallel to this conversion process, INA has further developed its semantic platform.

First, INA is developing the OKAPI ontology that conceptualizes in OWL/RDF the set of metadata fields used in metadata.

[9] https://en.wikipedia.org/wiki/Turtle_(syntax)

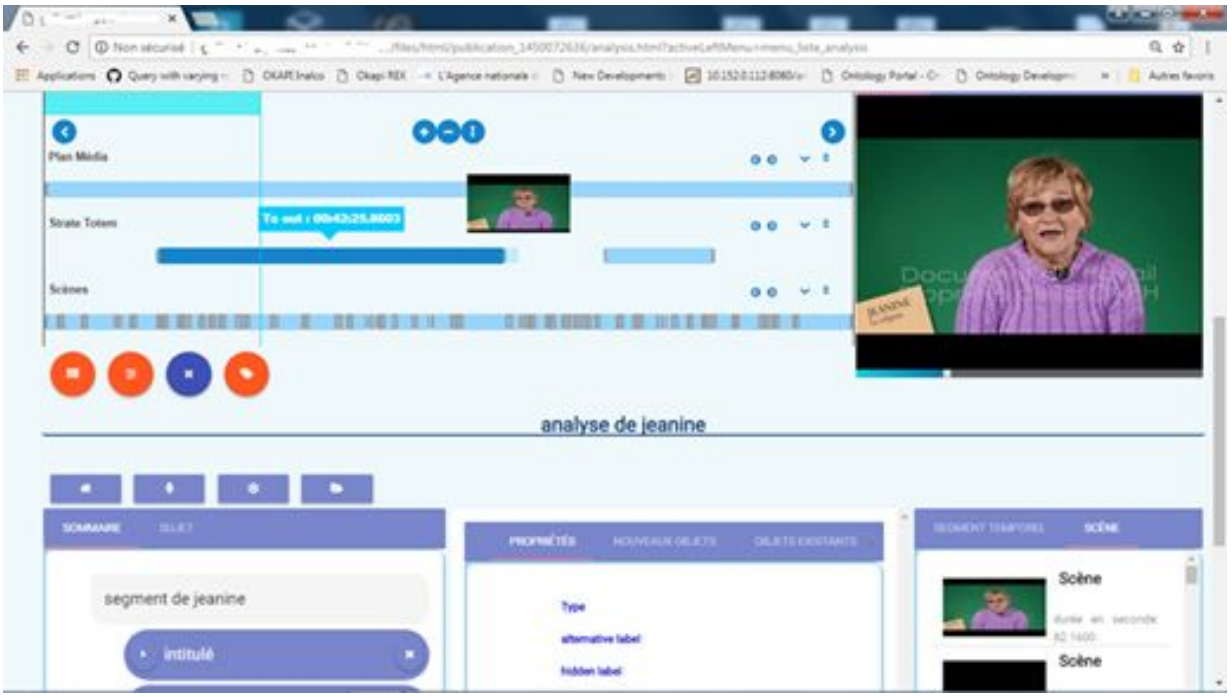


Local view for "http://www.ina.fr/resource/segment_451810_033"

Predicate	Value (sorted: default)
rdfs:label	"64' Le Monde en français, 2ème partie : [émission du 20 mai 2014]"
rdf:type	ina:Notice Totem
ina:aPourChapeau	"Deuxième partie de la tranche horaire présentée par Mohamed KACI constituée d'un sujet "GRAND ANGLE" et d'une ru
ina:aPourDuree	"1121"^^xsd:int
ina:aPourIdentifiantImedia	"77555176"
ina:aPourIdentifiantMediametrie	"0"
ina:aPourProducteurs	"TV5MONDE, 2014 (Producteur)"
ina:aPourResumeProducteur	"[Source iMedia] Sommaire : - Grand angle : Belgique, deux scrutins le 25 mai"
ina:aPourTheme	cinéma film élection européenne campagne électorale festival élection régionale Cannes Belgique Chypre Parlement européen Union européenne exposition Polo, Marco
ina:aPourTitreTrancheHoraire	"64' Le Monde en français"
ina:date de diffusion	"2014-05-20T18:30:39"^^xsd:date
ina:genre	ina:Magazine
ina:nature de production	"Production propre"
ina:PAR	Sojcher, Frédéric Delwit, Pascal
ina:PRE	Martin, Estelle Kaci, Mohamed
ina:résumé	"GRAND ANGLE "Belgique, la tentation séparatiste ?" Débat sur les élections fédérales, régionales et européennes en Bel Bruxelles, Pascal DELWIT, politologue. Slimane ZEGHIDOUR évoque l'existence d'une liste turco-grecque pour les électi

During the second year of the MeMAD project, a mapping between the OKAPI and the MeMAD (based on EBU Core) ontologies will be performed in order to increase metadata interoperability.

Second, INA has further developed its semantic annotation tool, that provides a video player and the ability to view and edit timed text annotations, i.e the set of annotations that are temporally aligned to sequences of the program.



5. Yle Conversion

Yle has provided, so far, 9 datasets summing up to nearly 235 hours of content. Some datasets correspond to a set of episodes belonging to one series (*Strömsö, Spotlight*) during a given time period, while other datasets contain metadata from different sources and different channels, all produced by Yle (*English, Retro*).

Each dataset contains media files as well as metadata stored in XML files. The root element of each file contains either media objects or elements describing the media objects. One media object can be of different types. These objects have child element <GUID> whose content is used to link data to the right TV programme. For each media object, the related metadata is in elements where the attribute name defines the field.

On some datasets, the notion of *Episodes* and *Series* appear, and we opt for an explicit display of this notion by using ebucore:Episode and ebucore:Series instead of the more generic ebucore:Programme and ebucore:Collection. When a program is not part of a series, however, we model it simply as a ebucore:TVProgramme.

We provide below an example of a file describing an episode of the series Strömsö:

```
<?xml version='1.0' encoding='utf-8'?>
<AXFRoot>
 <MAObject type="default" mdclass="PROGRAMME">
   <GUID dmname="">20161118..024140000005996B00000D0F029615</GUID>
   <Meta name="FIRSTRUN_TIME" format="string">172500</Meta>
   <Meta name="EPISODE_NUMBER" format="string">1</Meta>
   <Meta name="CLASSIFICATION_COMB_A" format="string">Asiaohjelma</Meta>
   <Meta name="DURATION" format="string">1707000</Meta>
   <Meta name="SERIES_ID" format="string">656546508527</Meta>
   <Meta name="THIRD_TITLE" format="string"/>
   <Meta name="END_OF_MSG" format="string">37707000</Meta>
   <Meta name="METRO_PROGRAMME_ID" format="string">PROG_2016_00704200</Meta>
   <Meta name="SUBJECT" format="string">Vapaa-ajan ohjelma jossa käsitellään m.m. ruokaa,
puutarhanhoitoa, askartelua ja puutöitä.</Meta>
   <Meta name="ACTORS" format="string"/>
   <Meta name="AUDIO_TYPE" format="string">2</Meta>
   <Meta name="DESCRIPTION_SHORT" format="string">Strömsö toivottaa kevätkauden
tervetulleeksi rakkauden ja ystävyyden merkeissä. Tänään aiheina ovat leikkokukat,
portviinidrinkit, neulahuovutus, persoonalliset ystäväkirjat ja grillauspaikka.
svenska.yle.fi/stromso</Meta>
   <Meta name="MEDIA_ID" format="string">MEDIA_2017_01221354</Meta>
   <Meta name="COLOUR" format="string">0</Meta>
   <Meta name="WEB_DESCRIPTION" format="string"/>
```

```
    ...
  <MVAttribute type="PUBLICATIONS" index="1" attribute="PUBLICATIONS"
mdclass="PROGRAMME"
objectid="20161118113822317202704801800505690241400000005996B00000D0F029615">
    <Meta name="PUB_ID" format="string">663525811469</Meta>
    <Meta name="PUB_TYPE" format="string"/>
    <Meta name="PUB_DATETIME" format="string">20170205080000</Meta>
    <Meta name="PUB_CHANNEL" format="string">Yle Areena</Meta>
    ...
  </MVAttribute>

  <MVAttribute type="CONTRIBUTORS" index="11" attribute="CONTRIBUTORS"
mdclass="PROGRAMME" objectid="2016111811382231...96B00000D0F029615">
    <Meta name="CONT_PERSON_NAME" format="string">Pasi Keho-Valkama</Meta>
    <Meta name="CONT_PERSON_ROLE" format="string"/>
  </MVAttribute>
```

5.1 Mapping table

Similar to INA datasets, we first develop mapping tables between the XML elements and attributes and the MeMAD classes and properties.

Path to metadata	Class	Property
./MAObject[1]/GUID	ebucore:Episode	ebucore:hasIdentifier
./MAObject[1]/Meta/[@name='EPISODE_NUMBER']	ebucore:Episode	ebucore:episodeNumber
./MAObject[1]/Meta/[@name='FIRSTRUN_TIME']	ebucore:PublicationEvent	ebucore:publicationStartDateTime
./MAObject[1]/Meta/[@name='FIRSTRUN_DATE']	ebucore:PublicationEvent	ebucore:publicationStartDateTime
./MAObject[1]/Meta/[@name='ARCHIVE_DATE']	ebucore:Episode	ebucore:archivingDate
./MAObject[1]/Meta/[@name='ASPECT_RATIO ']	ebucore:MediaResource	ebucore:aspectRatio
./MAObject[1]/Meta/[@name='DESCRIPTION_SHORT']	ebucore:Episode	ebucore:description
./MAObject[1]/Meta/[@name='DURATION']	ebucore:Episode	ebucore:duration
./MAObject[1]/Meta/[@name='FI_TITLE']	ebucore:Episode	ebucore:title
./MAObject[1]/Meta/[@name='KEYWORDS']	ebucore:Episode	ebucore:hasKeywords
./MAObject[1]/Meta/[@name='LANGUAGE']	ebucore:Episode	ebucore:language
./MAObject[1]/Meta/[@name='MAINTITLE']	ebucore:Episode	ebucore:mainTitle
./MAObject[1]/Meta/[@name='MEDIA_ID']	ebucore:MediaResource	ebucore:hasIdentifier

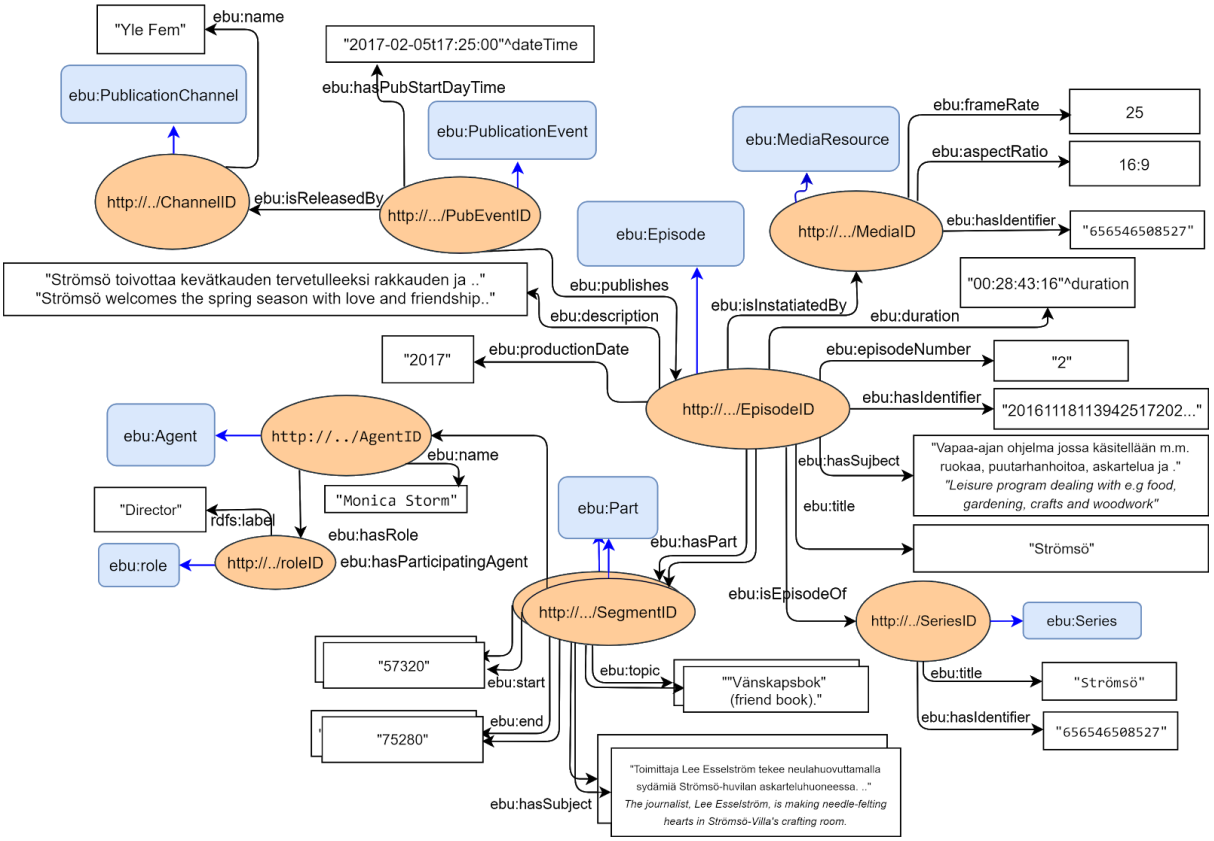
./MAObject[1]/Meta/[@name='METRO_Episode_ID']	ebucore:Episode	ebucore:hasIdentifier
./MAObject[1]/Meta/[@name='SERIES_ID']	ebucore:Series	ebucore:hasIdentifier
./MAObject[1]/Meta/[@name='SERIES_NAME']	ebucore:Series	ebucore:title
./MAObject[1]/Meta/[@name='SE_TITLE']	ebucore:Episode	ebucore:title
./MAObject[1]/Meta/[@name='SUBJECT']	ebucore:Episode	ebucore:description
./MAObject[1]/Meta/[@name='SYSTEM_DURATION_TC']	ebucore:Episode	ebucore:duration
./MAObject[1]/Meta/[@name='SYSTEM_FRAMERATE_FPS]	ebucore:MediaResource	ebucore:frameRate
./MAObject[1]/Meta/[@name='SECOND_TITLE']	ebucore:Episode	ebucore:alternativeTitle
./MAObject[1]/Meta/[@name='THIRD_TITLE']	ebucore:Episode	ebucore:alternativeTitle
./MAObject[1]/Meta/[@name='VERSION_NAME']	ebucore:Episode	ebucore:version
./MAObject[1]/Meta/[@name='VIDEO_FORMAT']	ebucore:MediaResource	ebucore:hasVideoEncodingFormat
./MAObject[1]/Meta/[@name='WORKING_TITLE']	ebucore:Episode	ebucore:workingTitle
./MVAttribute[@type='SUBTITLES']/ Meta[@name='ST_FILENAME']	ebucore:Subtitling	ebucore:filename
./MVAttribute[@type='SUBTITLES']/ Meta[@name='ST_LANGUAGE_CODES']	ebucore:Subtitling	ebucore:language
./MVAttribute[@type='SUBTITLES']/ Meta[@name='ST_DURATION']	ebucore:Subtitling	ebucore:duration
./MVAttribute[@type='SUBTITLES']/ Meta[@name='ST_TITLE']	ebucore:Subtitling	ebucore:title
./MVAttribute[@type='SUBTITLES']/ Meta[@name='ST_FILE_FORMAT']	ebucore:Subtitling	ebucore:hasFileFormat
./MVAttribute[@type='SUBTITLES']/ Meta[@name='ST_INGEST_DATE']	ebucore:Subtitling	ebucore:dateIngested
./MVAttribute[@type='AUDIO']/ Meta[@name='PMA_LANGUAGE']	ebucore:AudioTrack	ebucore:language
./MVAttribute[@type='AUDIO']/ Meta[@name='PMA_CODEC']	ebucore:AudioTrack	ebucore:hasAudioCodec
./MVAttribute[@type='AUDIO']/ Meta[@name='PMA_SAMPLE_RATE']	ebucore:AudioTrack	ebucore:hasSampleRate
./MVAttribute[@type='PUBLICATIONS']/ Meta[@name='PUB_ID']	ebucore:PublicationEvent	ebucore:publicationEventId
./MVAttribute[@type='PUBLICATIONS']/ Meta[@name='PUB_DATETIME']	ebucore:PublicationEvent	ebucore:hasPublicationStartDateTime
./MVAttribute[@type='PUBLICATIONS']/ Meta[@name='PUB_CHANNEL']	ebucore:PublicationChannel	ebucore:publicationChannelName
./MVAttribute[@type='PUBLICATIONS']/ Meta[@name='PUB_DURATION']	ebucore:PublicationEvent	ebucore:duration
./MVAttribute[@type='PUBLICATIONS']/	ebucore:PublicationEvent	ebucore:hasPublicationEndDateTime

Meta[@name='PUB_DATETIME_END']		
./MVAttribute[@type='CONTRIBUTORS']/ Meta[@name='CONT_PERSON_NAME']	ebucore:Agent	ebucore:agentName
./MVAttribute[@type='CONTRIBUTORS']/ Meta[@name='CONT_PERSON_ROLE']	ebucore:Agent	ebucore:hasRole
./MAObject[@mdclass='S_CONTENT_DESCRIPTION']/Meta[@name='GUID']	ebucore:Part	ebucore:hasIdentifier
./MAObject[@mdclass='S_CONTENT_DESCRIPTION']/Meta[@name='SEGMENT_DESCRIPTION']	ebucore:Episode	ebucore:description

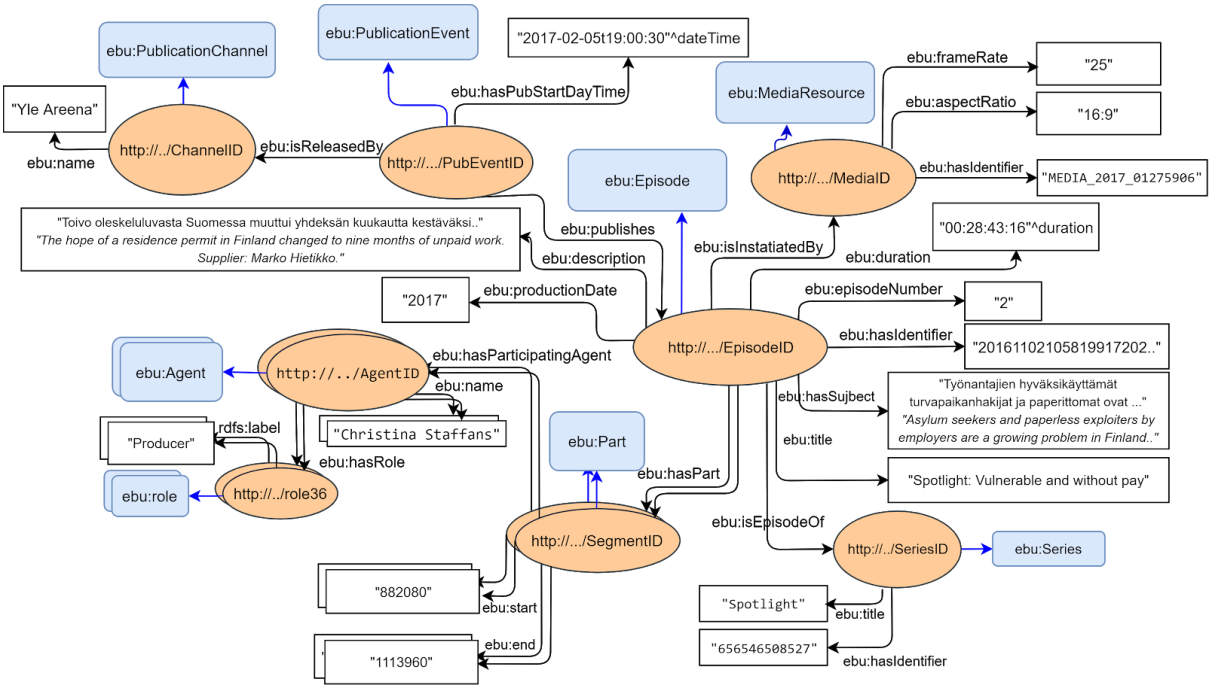
Again, a number of fields are not mapped since they were generally empty (not valued) in the datasets provided.

```
CLASSIFICATION_CONTENT, CLASSIFICATION_MAIN_CLASS, CLASSIFICATION_COMB_A,
CLASSIFICATION_SUB_CLASS, COLLECTION, COLOUR, START/END_OF_MSG,
LC_LOUDNESS_ADJUSTMENT_DATETIME,LC_LOUDNESS_ADJUSTMENT_PERFORMED,
LC_LOUDNESS_MEASUREMENT_PERFORMED, MODIFICATION_DATETIME, ORIGIN,
PART_NAME_FI, PART_NAME_SE, PRODUCTION_SEASON, PRODUCTION_YEAR,
REGISTRATION_DATETIME, SERIES_PART_SUM, SYSTEM_DURATION,
SYSTEM_FRAMERATE_DENOMINATOR, SYSTEM_FRAMERATE_DROPFRAME,
SYSTEM_FRAMERATE_NAME, SYSTEM_FRAMERATE_NUMERATOR, SYSTEM_MEDIA_TYPE,
SYSTEM_SAMPLERATE, SYSTEM_SAMPLERATE_NAME, SYSTEM_SOM, VIDEO_MD5, VIDEO_TYPE,
WEB_DESCRIPTION, WEB_DESCRIPTION_SWE, ST_NUMBER_OF_CAPTIONS,
ST_TRANSLATORS,ST_EOM/SOM, ST_TITLE_ORG, ST_PUB_NETWORK, ST_PUB_DATE,
ST_DATE, ST_ADDITIONAL_INFORMATION, ST_EXPORT_DATE,ST_EXPORT_FLAG
ST_INGEST_USER, ST_PROD_CODE, ST_VIDEO_ID, ST_PROG_DURATION,
ST_EPISODE_NUMBER, PMA_TRACK, PMA_SOM, PMA_EOM,PMA_TYPE_MIX, PMA_RESOLUTION,
PMA_CHANNELNUMBER, PMA_CHANNELORDER, PMA_TEST_TONE_LEVEL, PMA_NOTES,
PMA_LOUDNESS, PMA_DBTP, PMA_LRA, PUB_TYPE, PUB_MODE, PUB_STATUS
```

5.2 Examples



A description of a sequence of a program from the Strömsö dataset



A description of a sequence of a program from the Spotlight dataset

5.3 Statistics

Dataset	Programs	Series	Segments	Agents/ roles	Channels	Period
1-Strömsö	35	1	344	732 / 21	4	2017/02/05 - 2017/12/27
3-Spotlight	15	1	29	167 / 11	4	2017/02/06 - 2017/12/03
5-May 2014	3576	504	11734	3565 / 42	10	2014/05/01 - 2014/06/02
7-ObsDebatt	37	1	193	357 / 16	4	2016/01/21 - 2017/12/14
8-MayJune 2004	137	18	842	230 / 42	7	2004/01/19 - 2004/06/30
10-Retro 1970-1989	79	6	610	442 / 24	7	1966/11/15 - 2016/01/19
12-English Spoken	48	15	874	385 / 26	4	2002/02/06 - 2018/09/25
Total	**3927**	**537**	**11733**	**3005 / 42**	**11**	

5.4 Conversion script

Since all metadata provided in this dataset have the same structure, we use one Python script to process all these files. A bash script is also provided to batch-process the entire dataset.

The script takes as input the path to the XML file containing the metadata, and outputs an RDF graph (serialized in Turtle) in the same location. To process the input files, we use the native Python XML library to process the input file, and RDFLib (https://github.com/RDFLib/rdflib) to generate the RDF graph triples and serialize them into the Turtle file.

All scripts are available in the github repository at https://github.com/MeMAD-project/

6. SPARQL Queries

In the following, we provide some representative queries for accessing the data available inside the MeMAD knowledge graph.

6.1 Query 1: Get the list of programs from a particular channel

```
PREFIX ebucore: <http://www.ebu.ch/metadata/ontologies/ebucore/ebucore#>

SELECT ?title
WHERE {
  ?channel a ebucore:PublicationChannel;
            ebucore:hasPublicationChannelName "France 2" .
  ?pubevent a ebucore:PublicationEvent;
              ebucore:hasPublicationChannnel ?channel .
  ?pubevent ebucore:publishes ?program .
  ?program ebucore:title ?title .
}
```

6.2 Query 2: Get the list of programs featuring a certain keyword

```
PREFIX ebucore: <http://www.ebu.ch/metadata/ontologies/ebucore/ebucore#>

SELECT ?program
WHERE {
   ?program ebucore:hasKeyword ?keyword .
   FILTER (?keyword IN (“économie”, “immigration”, “France”) )
}
```

6.3 Query 3: Get the list of programs broadcasted during a given time period

```
PREFIX ebucore: <http://www.ebu.ch/metadata/ontologies/ebucore/ebucore#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?program
WHERE {
  ?pubevent ebucore:publishes ?uri.
  ?pubevent ebucore:hasPublicationStartDateTime ?date.
  FILTER ( ?date > "2014-05-22T10:20:12"^^xsd:dateTime && ?date <
"2014-05-25T10:20:12"^^xsd:dateTime)
}
```

6.4 Query 4: Get the list of all collections and their types

```
PREFIX ebucore: <http://www.ebu.ch/metadata/ontologies/ebucore/ebucore#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?title ?type
WHERE {
  ?collection a ?type.
  ?type rdfs:subClassOf* ebucore:Collection.
  ?collection ebucore:title ?title.
}
```

6.5 Query 5: Get the list of segments in which a person appears

```
PREFIX ebucore: <http://www.ebu.ch/metadata/ontologies/ebucore/ebucore#>

SELECT ?segment
WHERE {
  ?segment a ebucore:Part.
  ?segment ebucore:hasContributor <http://data.memad.eu/agents/fauvelle-marc>.
}
```

7. NER / NEL

In this section, we describe three Named Entity Recognition (NER) / Named Entity Linking (NEL) tools that we have further developed during the first year of the MeMAD project. The first one is ADEL (ADaptable Entity Linking), a generic framework that enables to perform named entity recognition and disambiguation for various kinds of documents, in different languages and adapted to various entity types or knowledge bases (Section 7.1). The second one is named Ensemble NERD, and proposes a promising ensemble approach over multiple NER tools (Section 7.2). The third one is the so-called Lingsoft NER API, a REST API over the rule-based NER system own by Lingsoft that has been developed during the first year of the project (Section 7.3).

7.1 ADEL

Four main challenges can cause numerous difficulties when developing an entity linking system: *i*) the kind of textual documents to annotate (such as social media posts, video subtitles or news articles); *ii*) the number of types used to categorise an entity (such as PERSON, LOCATION, ORGANIZATION, DATE or ROLE); *iii*) the knowledge base used to disambiguate the extracted mentions (such as DBpedia, Wikidata or Musicbrainz); iv) the language used in the documents. Among these four challenges, being agnostic to the knowledge base and in particular to its coverage, whether it is encyclopedic like DBpedia or domain-specific like Musicbrainz, is arguably one of the most challenging one.

ADEL is a system that performs entity recognition and linking using linguistic, information retrieval, and semantics-based methods. ADEL is a modular framework that is independent to the kind of text to be processed and to the knowledge base used as referent for disambiguating entities. In order to be knowledge base agnostic, we propose a method that enables to index the data independently of the schema and vocabulary being used. More precisely, we design our index such that each entity has at least two information: a label and a popularity score such as a prior probability or a PageRank score. We thoroughly evaluate the framework on six benchmark datasets: OKE2015, OKE2016, NEEL2014, NEEL2015, NEEL2016 and AIDA. Our evaluation shows that ADEL outperforms state-of-the-art systems in terms of extraction and entity typing. It also shows that our indexing approach allows to generate an accurate set of candidates from any knowledge base that makes use of linked data, respecting the required information for each entity, in a minimum of time and with a minimal size.

The ADEL framework is available on github at https://github.com/jplu/ADEL. A REST API is also available at http://adel.eurecom.fr/api/ which has been integrated in the Limecraft platform. ADEL is a framework that existed prior to MeMAD. In 2018, we have strengthen the general architecture of the system and we have integrated it in the Limecraft platform.

The details of the ADEL framework are presented in the Annex A, in a paper submitted to the Journal of Web Semantics which is currently under review.

7.2 Ensemble NERD

Named entity recognition (NER) and disambiguation (NED) are subtasks of information extraction that aim to recognize named entities mentioned in text, to assign them pre-defined types, and to link them with their matching entities in a knowledge base. Many approaches, often exposed as web APIs, have been proposed to solve these tasks during the last years. These APIs classify entities using different taxonomies and disambiguate them with different knowledge bases.

In 2018, we have researched Ensemble Nerd, a framework that collects numerous extractors responses, normalizes them and combines them in order to produce a final entity list according to the pattern (surface form, type, link). The presented approach is based on representing the extractors responses as real-value vectors and on using them as input samples for two Deep Learning networks: ENNTR (Ensemble Neural Network for Type Recognition) and ENND (Ensemble Neural Network for Disambiguation). We train these networks using specific gold standards. We show that the models produced outperform each single extractor responses in terms of micro and macro F1 measures computed by the GERBIL framework.

The Ensemble NERD system is available on github at https://github.com/D2KLab/ensemble-nerd. The details of the Ensemble NERD system are presented in the Annex B, in a paper published in the 17th International Semantic Web Conference (ISWC) held on 8-12 October 2018, Monterey, CA, USA.

7.3 Lingsoft NER

Lingsoft provides a demo analysis service for recognizing named entities with Lingsoft's analyser through an API for the MeMAD consortium. The NER analysis is based on the Lingsoft Linguistic Analyser core technologies and is currently available in Finnish and Swedish, with a possibility of adding English during the second year of the project.

At the moment, the Lingsoft NER recognizes entities in the categories listed in the Table below, but the categories will be iteratively expanded to cover new comparable named entity types for MeMAD needs in 2019 in collaboration with the MeMAD partners and interest groups.

Entity Type	**Description**
Date	Dates in different Finnish formats

Year	Year numbers
Phone numbers	Phone numbers in Finnish format
Registration plates	Vehicle registration plate numbers
URLs	Internet addresses
Email	E-mail addresses
Street Address	Street addresses in different formats
Location	Different place names
Person ID	Finnish Person Id format
Nationality	Descriptions of nationality
Person names	Names of persons in different formats
Names of companies	Heuristic with ending Oy/Ab/Oyj/Abp
Unclassified names	Words that seem to be names based on the linguistic context

The Lingsoft NER can also be extended with semantic lexicons to link the recognized entities to existing knowledge bases such as Wikidata. Explorative work is ongoing with Yle and EURECOM and will continue in 2019.

8. Annexes

8.1 Annex A: ADEL

The ADEL framework existed prior to the beginning of the MeMAD project. In 2018, we have improved the general architecture of ADEL, and we have trained new models to specifically deal with subtitles of French TV programs. Furthermore, we have integrated the ADEL API in the Limecraft platform.

We expect to further develop ADEL in order to deal with additional languages (e.g. Finnish and Swedish) and to improve both its named entity recognition and named entity disambiguation modules.

Semantic Web 1 (2017) 1–5
IOS Press

ADEL: ADaptable Entity Linking

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

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

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

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

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

1. Introduction

The age of the modern artificial intelligence as started in the middle of the 1940s. In 1950, Alan Turing as stated the earliest artificial intelligence problem that was natural language processing oriented, called the Turing test [60]. The goal of this test, as stated by Turing, can be seen as a game where a human is talking to two different interlocutors through a computer and s/he has to determine who is human and who 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 [63], being also the first natural language processing application developed to try to pass the Turing test. ELIZA was supposed to act like a psychotherapist, and was working with language pattern recognition manually written in a script. From 1978, people have started to talk about structuring knowledge in order to make machines smarter. From 1991, we see the need to automatically extract important facts from textual content by focusing on recognizing named entities [46]. Once we have started to have usable knowledge bases, we see that people have focused their at-

tention on linking these named entities, and the first approach was to disambiguate medical entities [10]. 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. One can see that the more we 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 a good example [59], making the knowledge bases a crucial resource for multiple high level Natural Language Processing tasks such as question answering, chatbots or personal assistants.

As real examples, we are working on two different projects that need entity linking: NexGenTV and ASRAEL. Within the NexGenTV project, we are developing authoring tools that enable to develop second screen applications and facilitate social TV. In particular, there is a need for near real-time automatic analysis to easily identify clips of interest, describe their content, and facilitate their enrichment and sharing [1]. In this context, we are analyzing the TV program subtitles in French for extracting and disambiguating named entities and topics of interests [5]. Within the ASRAEL project, we are analyzing large volume of English and French newswire content in order to induce fine grained schema that describe events being reported in the news. More precisely, we extract and disambiguate named entities that are head words to extract attribute values that best describe an event in a completely unsupervised manner [39].

1.1. Task Description

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

In this paper, we retake the definition [29] of several NLP notions. We denote a *mention* as the textual surface form extracted from a text. An *entity* as an annotation that varies depending of the task: *i)* when only doing the entity recognition task, an *entity* is the pair *(mention, class)*; *ii)* when only doing the entity linking task, an *entity* is the pair *(mention, link)*; *iii)* when doing both the entity recognition and linking task, an *entity* is the triplet *(mention, class, link)*. A *candidate entity* is one possible entity that we generate in order to disambiguate the extracted mention. *Novel entities* are entities that have not yet appeared in the knowledge base being used. This phenomenon happens mainly in tweets and sometimes in news when, typically, a person just become popular but does not have yet an article in Wikipedia because of a lack of notability.

Many knowledge bases can be used for doing entity linking: DBpedia[1], Wikidata[2], YAGO[3] to name a few. Those knowledge bases are known for being broad in terms of coverage, while vertical knowledge bases also exist in specific domains, such as Geonames[4] for geography, Musicbrainz[5] for music, or LinkedMDB[6] for movies.

The two main problems when processing natural language text are ambiguity and synonymy [29]. An entity may have more than one mention (synonymy) and a mention could denote more than one entity (ambiguity). For example, the mentions *HP* and *Hewlett-Packard* may refer to the same entity (synonymy), but the mention *Potter* can refer to many entities[7] (ambiguity) such as places, person, band, movie or even a boar. This problem can be extended to any language. Therefore, entity linking is also meant to solve the problems of synonymy and ambiguity intrinsic in natural language.

We illustrate the problems of ambiguity and synonymy in an example depicted in Figure 1: the mention *Noah* may correspond to at least two entities *Yannick Noah* and *Joakim Noah*. The need to have a knowledge base with Linked Data is crucial in order to properly disambiguate this example: *Yannick Noah* is a tennis player who has played for the Chicago ATP and US Open (in New York) tournaments, the Chicago tourna-

[1] http://wiki.dbpedia.org
[2] https://www.wikidata.org
[3] http://yago-knowledge.org/
[4] http://www.geonames.org
[5] https://musicbrainz.org
[6] http://www.linkedmdb.org
[7] https://en.wikipedia.org/wiki/Potter

ment happening before the US Open one; *Joakim Noah* is a basketball player who has played for the Chicago Bulls before being enrolled by the New York Knicks team. Therefore, a useful clue in this example is the year 2007 since *Yannick Noah*'s tennis activity happened well before 2007. The proper entities for this example are *Joakim Noah*, *New York Nicks* and *Chicago Bulls*.

1.2. Challenges

Focusing on textual content, we can list four main challenges [29] that the NLP community is addressing for performing such an intelligent processing and that entity recognition and entity linking systems are facing. These challenges primarily affect the strategy used to understand the text, for extracting meaningful information units and linking those to external referents.

1. the nature of the text, referring to the fact that one can broadly consider two different categories of text: *i)* formal texts, usually well-written content provided by newspaper, magazine, or encyclopedia and respecting the principles of journalism writing[8]; *ii)* informal texts that do not entirely respects the principles of journalism writing, and are generally coming from social media platforms or search queries. Each category of textual content has its own peculiarities. For example, tweets are often written without following any natural language rules (grammar-free, slangs, etc.) and the text is mixed with Web links and hashtags.[9] This is why one does not process a tweet like a Wikipedia article;
2. the language used: textual content on the Web is available in multiple languages and these languages have some particularities that make them more or less difficult to process (for instance, Latin languages versus Asian languages);
3. the entity types: they may exist multiple classes (types) in which an entity can be classified and where each type has a definition. The definition of a type may vary depending on the information extraction task. For example, in the text *Meet you at Starbucks on the 42nd street*, one may recognize *Starbucks* as an *ORGANIZATION* while others may want to consider that *Starbucks* is a *PLACE* where the local branch of a coffee shop is making business. The two annotations may sound correct according to the setting but with two different definitions.
4. the knowledge base used: we can easily imagine that the results of an entity linking system highly depend on the knowledge base being used. First, the *coverage*: if a text is about a movie and one only uses a knowledge base containing descriptions of point of interests and places (such as Geonames), the number of disambiguated entities is likely to be small contrarily if a general purpose or cinema specific knowledge base is being used. Second, the *data model*: knowledge bases may use different vocabularies and even models which prevent to query in a uniform way (e.g. Wikidata vs DBpedia). They may also use different data modeling technology (e.g. relational database vs linked data). Third, *freshness*: if we use a release of DBpedia dated five years ago, it will not be possible to find the entity *Star Wars: The Force Awakens* and this will make the disambiguation of occurrences of this entity much harder.

1.3. Contributions

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

1. We propose an entity recognition process that can be independent of the genre of the textual content (i.e. from Twitter or Wikipedia) and language. This process can also be adapted to the different definitions that may exist for extracting a mention and classifying an entity (Section 4.1).
2. We handle the different type of linked data models that may exist to design a knowledge base by providing a generic method to index its content and to improve the recall in terms of entity candidate generations (Section 4.2).
3. We propose a modular architecture that can be used to design an adaptable entity linking system (Section 5).
4. We thoroughly evaluate ADEL across different evaluation campaigns in terms of entity recognition, entity candidate generation, and entity linking (Section 6).

[8] https://www.theguardian.com/books/2008/sep/25/writing.journalism.news

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

Fig. 1. Figure representing an entity linking task.

1.4. Paper Structure

The rest of the paper is structured as follows. In Section 2, we give some background definitions used all along the paper. Section 3 presents related work on entity recognition and entity linking. Sections 4 and 5 detail our approach. Section 6 reports on numerous evaluations of our approach on standard benchmarks. Finally, conclusions and future work are provided in Section 7.

2. Background

In this section, we list and detail the essential inputs needed for performing entity linking namely input text, knowledge base, and provenance of both input text and knowledge base.

2.1. External Entries Used for Entity Linking

We identify two external entries for an entity linking system: the text to process and the knowledge base to use for disambiguating the extracted mentions. According to [48], an external entry for an entity linking system is composed of a text to annotate, a knowledge base and a set of entities. The authors classify the entity itself as a third component because there is currently no agreed upon definition of what an entity is. We identify two cases: *i)* named entities, as defined in [23] 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 [47,40]. *ii)* named entities are a set of resources defined in a knowledge base. This definition allows to consider many more entity types but to 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 tend to adopt only one definition, making this as a requirement (an external entry) and not a feature to select. In ADEL, we have decided to integrate the two definitions in order to be able to extract, type and link entities belonging to each definition or the two at the same time.

2.1.1. Textual Content

In [48], the authors classify a textual content in two categories: short and long text. We propose a different orthogonal categorization where textual content is divided between formal text and informal text. Formal texts are well-written texts that one can find in a newspaper, magazine, or encyclopedia. These texts are often long texts and provide easier ways to detect the context in which the mentions are used. This context facilitates the way the algorithms used in entity linking are working. People who are writing these texts often use a proper and common vocabulary in order to be understood by the largest set of people and contain none (or a low amount) of misspellings. Nevertheless, formal texts can also be short texts, for example, the title of an article or the caption of a picture.

It is then harder to extract and disambiguate entities in 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 [19].

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

This categorization is far from being exclusive and video subtitles is another kind of textual content that we aim to process. Subtitles are generally well-written, but they can also come from an automatic speech recognition (ASR) system[10] 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[11], it is likely that the subtitles are informal texts.

2.1.2. Knowledge Bases

Knowledge bases are a fundamental resource for doing entity linking. They often use *linked data* to provide information about entities, their semantic categories and their mutual relationships. Nevertheless, knowledge bases can be stored in different models ranging from graph to relational databases such as Wikipedia. In [48], the authors define three characteristics of a knowledge base: *1)* domain-specific versus encyclopedic knowledge bases; *2)* relational database versus linked data; and *3)* updated versus outdated knowledge bases in terms of data freshness. We will complement this by *i)* introducing some existing knowledge bases that have been widely exploited in entity linking, and *ii)* add a fourth characteristic: the different ontologies (schemas) used to describe the data into a knowledge base. For example, Wikidata is not modeled in the same way than DBpedia [18]. We can list the following knowledge bases:

- Wikipedia[12] 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 more than 5.3 million pages. Wikipedia has a large coverage of entities and contains comprehensive knowledge about notable entities. Besides, the structure of Wikipedia provides a set of useful features for entity linking such as a unique label for entities, categories, redirect pages, disambiguation pages and links across Wikipedia pages.
- DBpedia [31] 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 also exists in multiple languages. The 2016-04 English version describes more than 4.6 million entities and has more than 583 million relations. A large ontology is used to model the data and the number of entities grows similarly to Wikipedia at each release.
- Freebase [4] is a knowledge base owned by Google that aims to create a knowledge base of the world by merging a high scalability with a collaborative process. It means that anybody can update the knowledge base and anybody can access to it with a special language, MQL[13] (Metaweb Query Language) being a query language such as SPARQL but based on a JSON syntax. It contains 1.9 billion entities. Since March 2015, Google has decided to transfer the content of Freebase to Wikidata and has stopped to maintain Freebase.
- Wikidata [17] 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 impor-

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

tantly, 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 [58] 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 [37] is a multilingual knowledge base that merges Wikipedia, Wordnet, Open Multilingual Wordnet, OmegaWiki, Wiktionary and Wikidata. The goal is to provide a multilingual lexical and semantic knowledge base that is mainly based on semantic relations between concepts and named entities. It contains over 7.7 million entities.
- Musicbrainz[14] 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 nameed LinkedBrainz[15] is also regularly generated.
- 3cixty KB [50] is a collection of city-specific knowledge base that contains descriptions of events, places, transportation facilities and social activities, collected from numerous static, near- and real-time local and global data providers. The entities in the knowledge base are deduplicated, interlinked and enriched using semantic technologies.

Besides Wikipedia, all the other cited knowledge bases are available as linked data and are modelled using different ontologies. *DBpedia* uses the DBpedia Ontology[16]; Freebase uses its own data model[17] that has been mapped into RDF by keeping the same property names; YAGO uses its own data model [58]; Babelnet implements the lemon vocabulary[18]; Wikidata has developed its own ontology [17]. 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.

3. Related Work

Regardless of the different entity linking components that intervene in typical workflows, there are different ways to use these components [48]:

1. systems composed of two independent stages: mention extraction and entity linking. For the mention extraction stage, this generally consists of 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 use all these steps at the same time called *joint recognition-linking*.

Since a few years, most of the current entity linking research endeavours are only focusing on linking process as they assume that the mention extraction is a solved problem. While the current state-of-the-art methods in mention extraction work very well for well-defined types on newswire content [52], it is far to be perfect for tweets and subtitles [22,51] or for fine-grained entity types. More recently, the TAC KBP 2018 entity linking evaluation campaign puts again emphasis on the difficulty of managing numerous (7300+) entity types. Current state-of-the-art systems, often, do 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 [43,19,34,53,6]. 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.

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

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 1 details the extraction or recognition techniques adopted, and the Table 2 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 [25], Babelfy [36], DBpedia Spotlight [34], Dexter [6], Entityclassifier.eu [14], FOX [61,55], FRED [11], FREME[19], KEA [57], TagMe 2 [19], WAT [43], X-LiSA [65], AGDISTIS [61], DoSeR [66], NERFGUN [24] and PBOH [20].

The two tables share two columns: *Recognition* and *Candidate Generation*. Both tell 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 1 for the column *Candidate Generation* it means that the candidates are generated during the entity extraction process and not during the linking.

The systems in the tables are all ordered by chronological order, from the older to the newer. In Table 1, 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, taking into account that labeled data for many (under-resources) languages are rare in order to properly train supervised approach for doing part-of-speech tagging or named entity recognition tagging, and for those languages, using a dictionary is useful. In Table 2, we can see that the trend is more oriented to a collective approach with an equal distribution between graph-based and unsupervised approaches. Independent approaches are equally distributed among supervised and unsupervised. Also, doing *NIL clustering* is not often handled by these systems including the most recent ones. The work described in this document proposes collective and independent approaches for linking entities, including NIL entities with a *NIL clustering* method.

The Table 3 gives details on the possibility to address the four challenges mentioned in 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 propose a way to tackle these challenges, as they address at most two challenges and sometimes none. Systems without a symbol in a column represent the fact that they do not do entity extraction or recognition. The work described in this document propose an adaptive approach to tackle each of these challenges at the same time.

Since recently, few methods are doing what we call *joint recognition-linking*. The goal of these methods is to recognize and link the entities at the same time [38, 32,15,54]. They are mostly based on an approach using 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 [33], and they build an entity repository and name-entity dictionary using YAGO2 to detect the potential mentions. We introduce these approaches mostly to let the readers know that they exist, but we do not focus on them because they cannot handle more than one of the four challenges mentioned in Section 1.2, and do not propose competitive results compared to the other state-of-the-art approaches.

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

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

Table 1
Analysis of Named Entity Extraction and Recognition systems.

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

Table 2

Analysis of Entity Linking systems.

4. Approach

The goal of an entity linking approach is to recognize and to link all mentions occurring in a text to existing linked data knowledge base entries and to identify new entities not yet included in the knowledge base. ADEL comes with an adaptable architecture (Figure 2) compared to the state-of-the-art ones. As seen in Table 3, those architectures are typically static and show little flexibility for extracting and linking entities according to the challenges proposed in Section 1.2. Little flexibility because they generally cannot be extended without making important changes that would require to spend a lot of time in terms of integration. For example, for the extraction, it is not possible to add a dictionary extraction engine to AIDA [25] or a NER extraction to TagME [19] without changing a part of their architecture and then directly the source code. Next, the linking process is also static as, for example, we cannot add a method based on a linear formula to Babelfy [36] which uses a graph-based approach. Finally, the knowledge base being used, often, cannot be changed as well: it is difficult to make Babelfy [36] switch from Babelnet [37] to another knowledge base that belongs to the Linked Open Data cloud.

ADEL has been designed to enable all those changes. The ADEL architecture is modular where modules fall within three main categories. The first part, (*Entity Recognition*), contains the modules *Extractors* and *Overlap Resolution*. The second part, (*Index*), contains the module *Indexing*. Finally, the third part, (*Entity Linking*), contains the modules *Candidate Generation*, *NIL Clustering* and *Linkers*. The architecture works with what we call *modules* defined as a piece of the architecture configurable through a configuration file and where each component of a module (in red color on the schema) can be activated or deactivated depending on the pipeline one wants to use. Each module is further detailed in Section 4.1, 4.2 and 4.3. A general pipeline can also be automatically configured for some modules.

4.1. Entity Recognition

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

Extractors Module. Currently, we make use of six different extractors: *1)* Gazetteer Tagger, *2)* POS Tagger, *3)* NER Tagger, *4)* Date Tagger, *5)* Number Tagger and *6)* Co-reference Tagger. If two or more of these extractors are activated, they run in parallel. The recognition process is based on external NLP systems such as Stanford CoreNLP [33], GATE, NLTK or OpenNLP. To be compliant with any external NLP system, we have based our recognition process on a Web API interface that uses NIF as data exchange format [21]. Therefore, by using this module, it is possible to switch from one NLP system to another one without changing anything in the code or to combine different systems. An example is available with Stanford CoreNLP[20].

1. The Gazetteer Tagger relies on the integrated handling proposed in NLP systems such as *RegexNER*[21] of Stanford CoreNLP, *DictionaryNameFinder*[22] of OpenNLP or the *Dictionary Setup*[23] of GATE. We also propose an automated way to generate a gazetteer by issuing SPARQL queries to a linked data knowledge base. While using a gazetteer as extractor, it gives the possibility to be very flexible in terms of entities to extract and their corresponding type, and allows to handle multiple languages.
2. The POS Tagger extractor is configured to extract singular and plural proper nouns and to attach the generic type *THING*. In order to handle tweets, we use the model proposed in [13].
3. The NER Tagger extractor aims to extract named entities that are classified through the taxonomies used by Stanford CoreNLP, OpenNLP, GATE or others NLP systems. In order to handle tweets, we train a model using the data from the NEEL Challenge [48].
4. The Date Tagger aims to recognize all surface forms that represents temporal expression such as *Today*, *December 18, 1997* or *1997/12/18* and

[20] https://github.com/jplu/stanfordNLPRESTAPI
[21] http://stanfordnlp.github.io/CoreNLP/regexner.html
[22] http://opennlp.apache.org/documentation/apidocs/opennlp-tools/opennlp/tools/namefind/DictionaryNameFinder.html
[23] https://gate.ac.uk/sale/tao/splitch13.html#x18-34700013.9.2

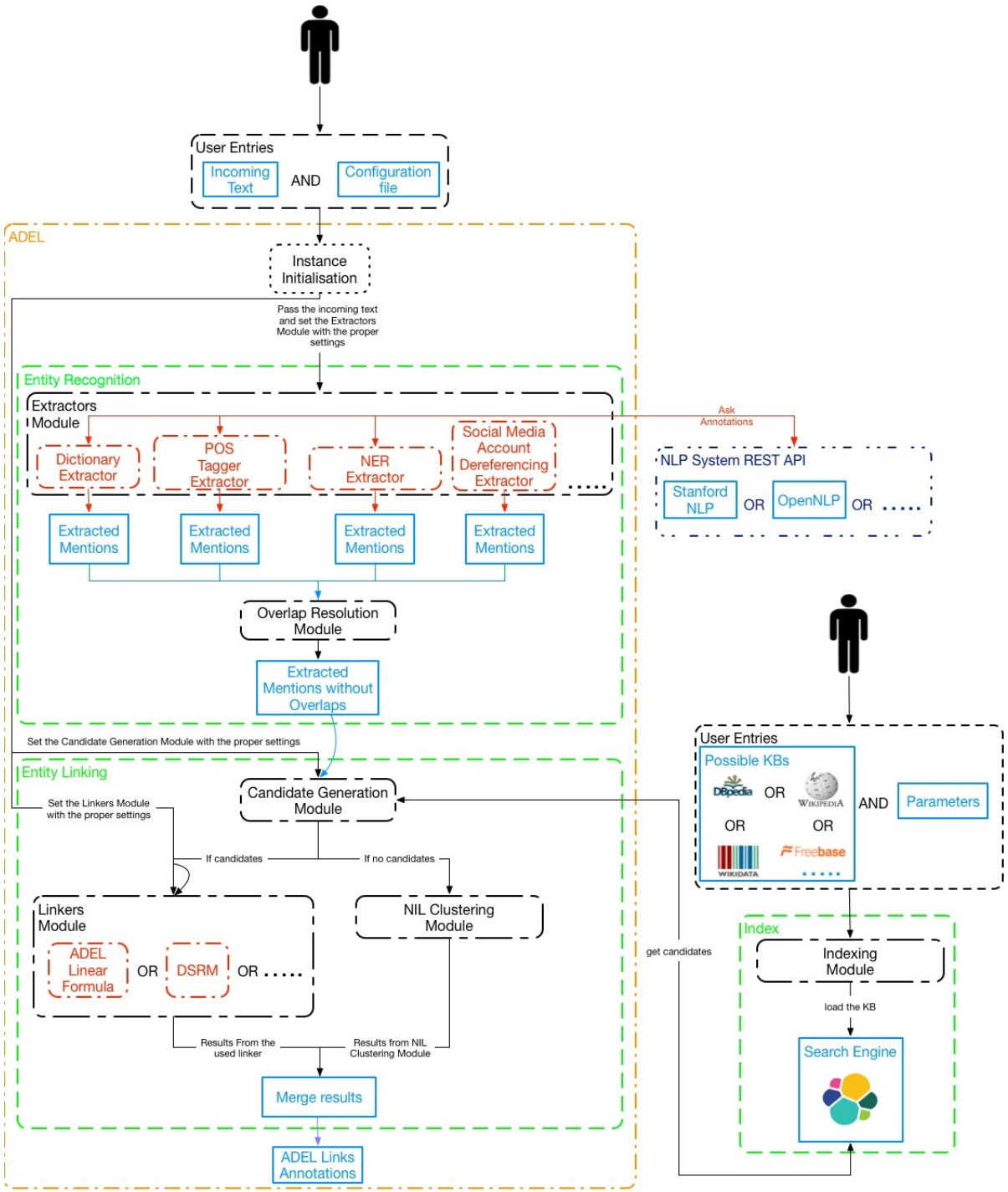


Fig. 2. ADEL architecture. There are two user entries, the text and the index (based on a knowledge base). A configuration file instantiates the launch of the framework. The text from the input goes to each extractor (relying on external NLP systems) and the output of each extractor goes to the overlap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

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 3
Availability of the systems for the four challenges tackle in this thesis.

relies on current temporal systems such as *SUTime*[24], *ManTIME*[25] or *HeidelTime*[26].

5. The Number Tagger aims to recognize the digit numbers (e.g. 15, 1, 35) or their textual representation (e.g. one, thirty), and can be done by either a NER Tagger (with Stanford NER), a POS Tagger (with the CD[27] POS tag) or regular expressions.
6. The Co-reference Tagger aims to extract co-references used within the same document but not across documents. The annotators provided by Stanford CoreNLP, OpenNLP, GATE or others NLP systems can be used.
7. The Social Media Account Dereference Tagger extractor aims to retrieve the real name of a social media account. For example, when the mention *@YouLoveJenny* is detected in a text, this extractor resolves it as *Jennifer Shelton* by querying the Twitter API.

We have the possibility to combine all these extractors, but also to combine the various NER models into one NER Tagger extractor. More precisely, we use a model combination method that aims to jointly make use of different CRF models in Stanford NER as described in the Algorithm 1. This algorithm shows that the order in which the models are applied is important. In Stanford NER, it is called *NER Classifier Com-*

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

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

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

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

biner. This logic can be extended to any other NER tagger. We explain the logic of this NER model combination using the following example: *William Bradley Pitt (born December 18, 1963) is an American actor and producer.*. The details for the models being used are available in the Stanford NER documention[28]. If we only apply the default 4 classes model (from Stanford CoreNLP), we get the following result: *William Bradley Pitt* as *PERSON*, and *American* as *MISC*. If we only apply the 7 classes model (from Stanford CoreNLP), we get the following result: *William Bradley Pitt* as *PERSON* and *December 18, 1963* as *DATE*. If we apply both models at the same time using the model combination logic, wet get the following result: *William Bradley Pitt* as *PERSON*, *December 18, 1963* as *DATE* and *American* as *MISC* corresponding here to the sets union.

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

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

A first case is when two labels represent the same category, for example *LOCATION* from the Stanford 3-class model and *dul:Place* from a model trained with the OKE2015 dataset[29]. In order to solve this ambiguity, we have developed a manual mapping represented in SKOS between the types from multiple sources where the sources are: the labels given by the three default models of Stanford NER, the DUL ontology[30], the Schema.org ontology[31], the DBpedia ontology[32], the Music ontology [45], the NERD ontology [49] and the NEEL taxonomy [48]. An excerpt for the mapping of the type *PERSON* is provided in the listing 1.

```
dbo:Person
        a                       skos:Concept ;
        skos:prefLabel          "Person"^^xsd:string ;
        itsrdf:taSource         "DBpedia"^^xsd:string ;
        skos:exactMatch         schema:Person, stanford:Person,
                                neel:Person, dul:Person,
                                nerd:Person, mo:SoloMusicArtist ;
        skos:broadMatch         mo:MusicArtist .
```

Listing 1: Mapping for the type PERSON from the DBpedia ontology.

The full definition of this mapping for the type *PERSON* is provided at `https://gist.github.com/jplu/74843d4c09e72845487ae8f9f201c797` and the same logic is applied for the other types. With this mapping, it is then possible to switch from one source to another with a SPARQL query. We are also using the notion of *broad* and *narrow* matches from SKOS in order to introduce a hierarchy among the types allowing the possibility to get a parent or subcategory if an equivalent one does not exist.

This recognition process allows us to handle a large set of languages and document types by *i)* cleverly combining different annotators from multiple external systems, and *ii)* merging their results by resolving their overlaps and aligning their types. Once we succeed to

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

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

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

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

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

recognize the entities, we generate entity candidates retrieved from the knowledge base. In the next section, we describe in detail the process of indexing a knowledge base as an essential task for the entity retrieval.

4.2. Indexing Linked Data

In order to generate the entity candidates we have to query an index, and properly querying an index is not that easy because the query used to generate these candidates might change from one case to another. For example, in DBpedia, it exists a large amount of properties that contain useful information. Hence, sometimes the proper candidate will be found by querying the property `rdfs:label` but sometimes it is better to query the property `dbo:birthName`. In this section, we propose an indexing module in order to answer the question: how to optimally select which property should be used to retrieve relevant entity candidates?

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

Indexing. The first step consists in extracting all entities that will be indexed using a SPARQL query. This query defines as many constraints as necessary. The minimum requirements for an entity to be indexed is to have an ID, a label, and a score. This score can correspond to the PageRank of the entity, or to any other way to score the entities in a linked data knowledge base. For example, with DBpedia, the corresponding required dumps[33] 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[34] 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 2 for DBpedia or Query 4 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 3 and the Query 5 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 2: SPARQL query that filters the entities we would like to index.

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

Listing 3: 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 .
```

[33] `http://wiki.dbpedia.org/downloads-2016-04`
[34] `http://jung.sourceforge.net/`

```
    ?s dbo:wikiPageRank ?pr .
    {
        ?s rdfs:label ?label .
    } UNION {
        ?s foaf:name ?label .
    } UNION {
        ?s dc:title ?label .
    }
}
```

Listing 4: 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#>
PREFIX mba: <http://musicbrainz.org/artist/>
SELECT DISTINCT ?p
(GROUP_CONCAT(DISTINCT ?o;separator="------") AS ?vals)
FROM <http://musicbrainz.org> WHERE {
    {
        mba:0002cb05-044d-46b8-98e2-8115ba9d24cb#_ ?p ?o .
        FILTER(DATATYPE(?o) = xsd:string ||
            LANG(?o) = "en") .
    } UNION {
        VALUES ?p {dbo:wikiPageRank mo:musicbrainz_guid} .
        mba:0002cb05-044d-46b8-98e2-8115ba9d24cb#_ ?p ?o .
    }
}
```

Listing 5: SPARQL query 2 for Musicbrainz to retrieve interesting content for the entity *http://musicbrainz.org/artist/0002cb05-044d-46b8-98e2-8115ba9d24cb#_*. This query is extended to each entity retrieved from the first Musicbrainz query.

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

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

The DBpedia index has 4726950 rows (entities) and 281 columns (datatype properties). Given some benchmark datasets such as OKE2015, OKE2016, NEEL2014, NEEL2015 and NEEL2016, we parse their content in order to extract a list of distinct pairs (mention, link). Next, for every pair, we query the index against every single columns (in the case of DBpedia, this represents 281 queries for each pair), and for each query, we check whether the proper link of the pair is among the results or not. If yes, we put the property in a white list, and if not, the property is ignored as not being helpful to retrieve the good candidate link. At the end, we end up with a file that looks like the excerpt depicted in the Listing 6.

```
{
  "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 6: Excerpt of the result file for the optimization process.

This file indicates the columns that must be queried to get the proper link for each pair. We notice that most of the pairs share similar columns. Therefore, we make a union of all these columns to obtain a list of unique columns to use to query the index. For the excerpt depicted in Listing 6, the distinct union yields the following list of 9 properties:

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

In the case of DBpedia, this reduces the number from 281 to 72 columns to query but this list is still

too large. If we check closely this excerpt, we notice that the column *dbo_wikiPageWikiLinkText* belongs to each list which means that with 1 single column (instead of 9) we can retrieve all pairs except the pair *AnotherYou—-http://dbpedia.org/resource/Another_You*. The logic behind is that we have to maximize the number of pairs we retrieve for each column, and the goal is then to minimize the number of columns. At the end, we finish with a minimum list of columns that maximize the coverage of the pairs. This optimization can be done with the Algorithm 2. The source code is also available[35].

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
```

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

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

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

This 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.

4.3. Entity Linking

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

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

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

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

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 (ti-

tle) 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.

5. Implementation

The ADEL framework is implemented in Java and is publicly accessible via a REST API[36] or via Github[37]. ADEL addresses the aforementioned four challenges being adaptable to the language and the kind of text to process, the types of entity to extract and the knowledge base to use for providing identifiers to entities.

ADEL needs a configuration file expressed in YAML that we call *profile* (Listing 7) in order to adapt its workflow. The configuration is composed of three distinct parts: extract, index and link. In the reminder of this section, we will detail how each part works.

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

Listing 7: An example of an ADEL profile.

Extract. In Listing 7, the object *extract* configures the entity recognition component. It is composed of one object for each extractor used (NER, POS, COREF, dic, date and number), the value of these objects being a list of instances. For example, in Listing 7, there are two extractors: *ner* and *pos*, where each extractor generates one instance. An instance is composed of four mandatory properties: *address*, *name*, *profile*, *className*, and an optional one: *tags*. The property *address* is the Web API HTTP address used to query the extractor. The property *name* is a unique name given to the instance of the extractor. The property *profile* is the profile that the extractor has to adopt[38]. The property *className* is the full name of the Java class (package + class) that has to be used internally to run the extractor. This property allows anyone to manage the extractor behavior via the reflection of Java[39]. The single optional property, *tags*, represents the list of tags that have to be extracted (all if empty or not present). It is also composed of two other mandatory properties that are *mapping* and *reference*. The former is the location of the SKOS mapping file for the types, and the latter is the source that will be used for typing the entities.

Index. In Listing 7, the object *index* configures the index that is composed of four mandatory properties: *type*, *address*, *strict* and *name*. The property *address* is the Web API HTTP or the folder address used to locate the index. The property *type* defines the index type to be used. Currently, we only handle Elasticsearch and Lucene but our indexing process can be extended to any other indexing system. As Elasticsearch and Lucene require different aspect of configuration, we had to define some properties that are specific to Elasticsearch or Lucene. 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 8. The property *strict* can have two values: *true* if we want a strict search, or *false* if we want a fuzzy search.

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

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

[37] https://github.com/jplu/adel

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

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

```
size: 1000
```

Listing 8: Lucene example for an index object

Link. In Listing 7, the object *link* configures the linkers module. This property contains the full name of the Java class (package + class) that has to be used internally to run the corresponding linking method.

6. Evaluation

In this section, we present a thorough evaluation of ADEL over different benchmark datasets namely OKE2015 [40], OKE2016 [41], NEEL2014 [2], NEEL2015 [47], NEEL2016 [51] and AIDA [25]. Each of these datasets have its own characteristics detailed in Table 4. The scores are computed with GERBIL [62]. Depending on the guideline of a given 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.

We propose to evaluate several configurations of ADEL in order to show 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 for a specific dataset. 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 (3-classes, 4-classes and 7-classes) 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 Stanford CoreNLP.
- *POS* (part-of-speech): Use Stanford CoreNLP 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 Stanford CoreNLP deep-coref.
- *DIC* (dictionary): Use a dictionary specifically built for a benchmark dataset with DBpedia.

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

6.1. Experimental Setup

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

	NEEL2014		
	Precision	Recall	F1
MC	74.61	29.38	42.16
MC+POS	**67.79**	**52.47**	**59.15**
POS	66.67	49.04	56.51
MC+NUM+DT	51.02	35.96	42.19
MC+POS+NUM+DT	54.40	59.32	56.75
POS+NUM+DT	53.90	57.26	55.53

Table 5

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

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

Table 4
Characteristics for each benchmark dataset

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

Table 6
Results over the OKE2015 and OKE2016 datasets at extraction level for different ADEL Entity Recognition module configurations. Scores in bold represent the best ADEL configuration

The NEEL2014 and AIDA dataset are not evaluated at recognition level because the guidelines do not require such evaluation. We also remove the ADEL configurations that use the POS Tagger because the POS Tagger cannot type an entity. The Table 13 has no specific configuration because, for now, we do have only one linking method to evaluate.

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

Table 7
Results over the NEEL2015 and NEEL2016 datasets at extraction level for different ADEL Entity Recognition module configurations. Scores in bold represent the best ADEL configuration

	AIDA		
	Precision	Recall	F1
MC	95.82	91.45	93.58
SM	**96.59**	**94.24**	**95.4**
MC+SM	95.82	91.45	93.58
MC+POS	81	88.21	84.45
SM+POS	81.94	89.83	85.7
SM+MC+POS	81	88.21	84.45
POS	76.76	75.66	76.21

Table 8
Results over the AIDA dataset at extraction level for different ADEL Entity Recognition module configurations. Scores in bold represent the best ADEL configuration

6.2. Results Analysis

OKE2015 and OKE2016. Regarding the OKE datasets, it is interesting to notice that the models trained with the corresponding training sets is less performing in comparison to a general purpose model learned on news, probably due to the amount of data, the datasets being too small, while having a dictionary can significantly improve the results (+13% in aver-

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

Table 17

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

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

Table 9

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

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

Table 10

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

age). By analysing the results, we have seen that the coreference Tagger is not that useful for extracting entities if we use the respective OKE models. Basically, these models are able to extract the coreference mentions (e.g. he, she, him, etc.) because these mentions are well represented into the training datasets. While this fact is interesting, the coreference Tagger is important as it links these mentions to their proper reference, what the NER Tagger cannot do because it is not possible for such tagger to make a relation between the extracted entities. For example, in the sentence *Barack Obama was the President of the United States. He was born in Hawaii.*, a NER Tagger might extract *Barack Obama* and *He* and type them as a PERSON, but will never make the relation that *He* refers to *Barack Obama* and then that *Barack Obama* must be used to disambiguate *He*. This is why we need a Coreference Tagger that provides this relation.

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

NEEL2015 and NEEL2016. The first configuration mainly fails to identify the hashtags and user mentions while the second configuration works relatively well. We also notice that adding a POS Tagger increases the recall but decreases the precision. The best configuration for doing entity recognition is the same than for the extraction. Contrarily to the NEEL2015 dataset, for NEEL2016, the test set has a lower amount of annotated tweets (1663 against 296). Inside this small amount, most of the entities are hashtags or Twitter user mentions, explaining why the *conf1* performs poorly. For NEEL2016, it is interesting to notice that, to only extract entities but not typing them, the *conf7* performs the best. For entity recognition, for both datasets, the best configurations are different from the extraction, which shows that it is not necessarily the best extraction process that will have the best recognition. Furthermore, for these two datasets, we can see that the best configuration is not the same, due to a more important training set for NEEL2016, the resulting model is more accurate. For analysing tweets in general, a simple POS tagger can achieve good results in terms of extraction, which is something useful

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

Table 11

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

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

Table 12

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

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

Table 13

Results at linking level for ADEL

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

Table 14

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

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

Table 15

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

	OKE2015			OKE2016		
	Precision	Recall	F1	Precision	Recall	F1
MC	76.47	48.21	59.14	82.67	39.01	53
SM	64.19	31.93	42.65	84.9	32.87	47.39
MC+SM	87.62	53.27	66.26	81.56	43.41	56.66
MC +COREF +DIC	**81.34**	**62.59**	**70.74**	**86.43**	**61.98**	**72.19**
SM +COREF +DIC	73.57	45.72	56.39	84.09	49.25	62.12
MC+SM +COREF +DIC	78.04	62.65	69.5	85.23	59.17	69.85

Table 16

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

as one can do entity linking on tweets without a NER model. While NER models trained over newswire content seem not to be appropriate for a proper entity recognition on tweets, we can still achieve fair results as long as there are not too many hashtags and Twitter user mentions.

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

Table 18

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

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

Table 19

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

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

As an overall overview of these per level 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 configuration for the NEEL2015 dataset is not the same than for the NEEL2016 dataset despite the fact that both datasets are made of tweets.

Index Optimization. Our index optimization process allows us to get a high score in terms of recall for the entity linking process. The results have been computed with a list of at most 8177 candidates. This optimization also reduces the time of the query to generate the entity candidates from around 4 seconds (without optimization) to less than one second (with optimization). Providing more candidates does not further increase the recall. We originally observe, though, a significant drop in terms of recall for the NEEL datasets which is mainly due to the presence of hashtags and Twitter user mentions (see the numbers in parenthesis for the 3 NEEL datasets in the Table 17). For example, it is hard to retrieve the proper candidate link *db:Donald_Trump_presidential_campaign,_2016* for the mention corresponding to the hashtag *#TRUMP2016*. We tackle this problem by developing a novel hashtag segmentation method inspired by [56,28]. For the previous example, this will result in *trump 2016*, those two tokens being then enough to retrieve the good disambiguation link in the candidate set. The 3 NEEL datasets, when using the hashtag segmentation method, and the 3 other datasets (OKEs and AIDA) have then a near-perfect recall if one retrieves sufficient candidate links. The few errors encountered correspond to situations where there is no match between the mention and any property values describing the entity in the index.

Comparison with Other Systems. Tables 9, 10, 11, 12, 14, 18 and 19 show that ADEL outperforms all other state-of-the-art systems in terms of extraction and recognition, except for the NEEL2015 dataset. The reason is because the system that achieves the best score makes use of a full machine learning approach for each sub-task: entity linking (mention extraction + disambiguation), type prediction for entities, NIL mention extraction and type prediction for NIL entities. It works very well but needs a large amount of data for being trained, and, therefore, it will not perform efficiently over the OKE datasets (3498 tweets for NEEL2015 and 95 sentences in OKE2015). In Table 14, we did not put another system for OKE2015 because the winner of the challenge was ADEL. The best system at linking level for OKE2016, is the challenge winner [7]. In Table 19, the winner [8] has the

best score. In Table 18, for NEEL2015, the winner has the best scores as well [64]. In Tables 9, 11, 10 and 12, ADEL is not the best system for linking, except for NEEL2016. At the linking level, xLisa-NGRAM [42] is the best for OKE2015, DoSeR [67] is the best for OKE2016 and NEE2014, AGDISTIS [61] is the best for NEEL2015, and WAT [43] is the best for AIDA. At extraction and linking level: AIDA [25] is the best for OKE2015, xLisa-NER [42] is the best for OKE2016, DBpedia Spotlight [12] is the best for NEEL2014, and AIDA [25] is the best for AIDA.

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

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

7. Conclusion and Future Work

In this paper, we presented the design and implementation of ADEL, and we demonstrate that our approach enables to be adaptable for at least three challenges:

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

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

Linking. The linking step is currently the main bottleneck in our approach. The performance drops significantly at this stage mainly due to a fully unsupervised method. Two new methods will be investigated in order to improve this step. The first one consists in using the new fastText[3] method which is an efficient learning of word representations and sentence classification. In comparison to Word2Vec [35], fastText is robust against out of vocabulary words allowing to create and compute similarities between words that do not belong to its model. The second method is to use the Deep Structured Semantic Models [27] as a relatedness score. This method can be customized to compute a relatedness score of entities in a knowledge base. Next, with this score, we can build a graph regularization as detailed in [26] in order to properly disambiguate the entities. We are also investigating how to use the French lexical network Rezo [30] in order to link entities in French texts. Finally, other general knowledge bases such as Freebase and Wikidata will be tested, but also specific ones like Geonames and 3cixty for different kind of text in order to broaden the evaluation domain of our approach.

Recognition. We are currently working on a coreference approach based on [9] to improve the accuracy of their approach by adding a semantic layer detailed in [44] to the deep neural network. During the overlap resolution, when we merge the results from multiple extractor, if at least two of them extract the same entity but assign a different type (e.g. one with PERSON and the other one with LOCATION), then it is difficult to select the proper type. Therefore, it can be improved by using an ensemble learning approach over each extractor such as the method proposed in [16].

Architecture. Although ADEL has a parallel architecture, we are not yet capable of handling live streams of text as the current system is not designed to be distributed. However, multiple instances of ADEL can run at the same time, and a solution could be to plug on top of multiple instances (workers) a load balancing implementation such as the one proposed in Apache Spark[40].

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

8. Acknowledgments

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

References

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

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

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

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

[5] Lorenzo Canale, Pasquale Lisena, and Raphäel Troncy. A Novel Ensemble Method for Named Entity Recognition and Disambiguation based on Neural Network. In *17th International Semantic Web Conference (ISWC)*, 2018.

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

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

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

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

[10] 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.

[11] Sergio Consoli and Diego Reforgiato Recupero. Using fred for named entity resolution, linking and typing for knowledge base population. In *SemWebEval@ESWC*, 2015.

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

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

[14] 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.

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

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

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

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

[19] 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.

[20] 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.

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

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

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

[24] 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.

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

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

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

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

[29] Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2Nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2009.

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

[31] 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.

[32] 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.

[33] 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.

[34] 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.

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

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

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

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

[39] 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.

[40] 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.

[41] 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.

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

[43] 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.

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

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

[46] Lisa F. Rau. Extracting company names from text. In *Proceedings of the Seventh Conference on Artificial Intelligence Applications CAIA-91 (Volume II: Visuals)*, 1991.

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

[48] 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.

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

[50] 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.

[51] 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.

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

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

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

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

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

[57] 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.

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

[59] 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.

[60] Alan M. Turing. Computing machinery and intelligence. *Mind*, 1950.

[61] 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.

[62] 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.

[63] Joseph Weizenbaum. Eliza-a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 1966.

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

[65] Lei Zhang and Achim Rettinger. X-lisa: Cross-lingual semantic annotation. In *Proc. VLDB Endow.*, 2014.

[66] 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.

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

8.2 Annex B: Ensemble NERD

In 2018, we have researched and developed Ensemble NERD from scratch. We wanted to confirm that an ensemble approach using appropriately trained deep learning frameworks would yield competitive results in terms of named entity extraction and disambiguation. The main weakness of this approach, so far, is its high computational cost. We will aim to further study and optimize this method in the remainder of the MeMAD project.

A Novel Ensemble Method for Named Entity Recognition and Disambiguation based on Neural Network

Lorenzo Canale[1,2], Pasquale Lisena[1], and Raphaël Troncy[1]

[1] EURECOM, Sophia Antipolis, France
{canale|lisena|troncy}@eurecom.fr
[2] Politecnico di Torino, Italy

Abstract. Named entity recognition (NER) and disambiguation (NED) are subtasks of information extraction that aim to recognize named entities mentioned in text, to assign them pre-defined types, and to link them with their matching entities in a knowledge base. Many approaches, often exposed as web APIs, have been proposed to solve these tasks during the last years. These APIs classify entities using different taxonomies and disambiguate them with different knowledge bases. In this paper, we describe Ensemble Nerd, a framework that collects numerous extractors responses, normalizes them and combines them in order to produce a final entity list according to the pattern (surface form, type, link). The presented approach is based on representing the extractors responses as real-value vectors and on using them as input samples for two Deep Learning networks: ENNTR (Ensemble Neural Network for Type Recognition) and ENND (Ensemble Neural Network for Disambiguation). We train these networks using specific gold standards. We show that the models produced outperform each single extractor responses in terms of micro and macro F1 measures computed by the GERBIL framework.

1 Introduction

A crucial task in knowledge extraction from textual document consists in the two complementary tasks of Named Entity Recognition (NER) and Named Entity Disambiguation (NED), achieving the goal of assigning to parts of text (tokens) respectively a type —from a pre-defined taxonomy— and a unique identifier — normally in the form of URI— that points univocally to the referred entity in a given knowledge base. The combination of these two tasks is often abbreviated with the acronym NERD [5,6]. The current state of the art offers an interesting number of NERD extractors. Some of them can be trained by a developer on his own corpus, while other ones are only accessible as black-box services exposed via web APIs offering a limited number of parameters.

In terms of NER, each service provides generally its own taxonomy of named entity types which can be recognised. While they all provide support for three major types (person, organization, location), they largely differ for more fine-grained types which makes hard their comparison and combination. In terms

of NED, each extractor can potentially disambiguate entities against specific knowledge bases (KB), but in practice, they mostly rely on popular ones, namely DBpedia, Wikidata, Freebase or YAGO. For this reason, comparing and merging the results of these extractors require some post-processing tasks that typically rely on mappings between those KBs. This task is however simpler than the type alignment, because of the large presence of `owl:sameAs` links between the different KBs.

In this paper, we present **Ensemble Nerd**, a multilingual ensemble method that combines the responses of different NERD extractors. This method relies on a real-value vectorial representation as input samples for two Deep Learning networks, ENNTR (Ensemble Neural Network for Type Recognition) and ENND (Ensemble Neural Network for Disambiguation). The networks provide models for performing type alignment and named entity linking to a knowledge base. This strategy is evaluated against some well-known gold standards, showing that the output of the ensemble outperforms the results of single extractors.

This work aims to answer the following research questions: Can we define an ensemble method that combines the extractors responses in order to create a new more powerful extractor? Is it possible to define an ensemble method that avoids a type alignment step or that computes it automatically, without any human intervention? Which ensemble method should be adopted to exploit all the collected information? Considering that extractors return list of named entities – together with the type and the disambiguation link of each of them –, how this data can be numerically represented? Can we better understand which features contribute more to improve the ensemble output response? How dependant is this feature selection of the corpora, language, entity types and what is the influence of the KB?

The remainder of this paper is organised as follows: Section 2 describes some related work. Section 3 details how we represent the extractors responses, while Section 4 presents the core of the ensemble method. An evaluation is proposed in Section 5, while conclusion and and future work are discussed in Section 6.

2 State of the Art

Ensemble methods for the NER and NED tasks have already largely been studied in the literature. The **NERD** framework [5, 6] allows to compare and evaluate some of the most popular named entity extractors. It can analyse any textual resource published on the web and to extract the named entities that are detected, typed and disambiguated by various named entity extractor APIs. For overcoming the different type taxonomies, the authors designed the *NERD ontology* which provides a set of mappings between these various classifications and consequently makes possible an evaluation of the quality of each extractor. This task was originally a one time modeling exercise: the authors manually mapped the different taxonomies to the NERD ontology.

NERD-ML, a machine learning approach developed on top of the NERD framework, combines the responses of single extractors applying alternatively

three different algorithms: Naive Bayes (NB), k-Nearest Neighbours (k-NN) and Support Vector Machines (SVM) [6, 11]. It is a more sophisticated and robust approach that uses machine learning inductive techniques for passing from the output type of single extractors to the right entity type in a normalized types set, i.e. the NERD Ontology [7]. **FOX** [9, 10] is a framework that relies on ensemble learning by integrating and merging the results of four NER tools: the **Stanford Named Entity Recognizer** [3], the **Illinois Named Entity Tagger** [4], the **Ottawa Baseline Information Extraction** (Balie) and the **Apache OpenNLP Name Finder**. FOX compares the performance of these tools for a small set of classes namely LOCATION, ORGANIZATION and PERSON. For achieving this goal, the entity types of each NER tools is mapped to these three classes. Given any input text t, FOX processes t with each of the n tools it integrates. The result of each tool T_i is a piece of annotated text t_i, in which either a specific class or zero (not belonging to the label of a named entity) is assigned to each token. The tokens in t are then represented as vectors of length n and are used for getting the final type. The author demonstrates that a Multi-Layer Perceptron (MLP) gets the best results among a pool of 15 different algorithms [9].

3 Feature Engineering for NERD

Ensemble Nerd currently integrates a set of 8 extractors shown in Table 3. An extractor can belong to the set T (extractors that perform NER task) or to the set U (extractors that perform NED task). Currently, *TextRazor* is the only one in both sets: $T \cap U = \{TextRazor\}$. All these extractors relies on Wikidata, Wikipedia or DBpedia for entity disambiguation.

Each extractor produces a list of named entities as response for a specific input text. From this output, we generate 4 different kinds of feature.

1. Surface form features. They are strictly related to the text used to extract named entity. The input text is split into tokens and a word embedding

Extractor	Type recognition	NE disambiguation
AlchemyAPI	✓	✗
DandelionAPI	✗	✓
DbSpotlight	✗	✓
TextRazor	✓	✓
Babelfy	✗	✓
MeaningCloud	✓	✗
Adel	✓	✗
OpenCalais	✓	✗

Table 1. Extractor included in Ensemble Nerd. ✓ indicates that the extractor supports the action (type recognition or named entity disambiguation)

.

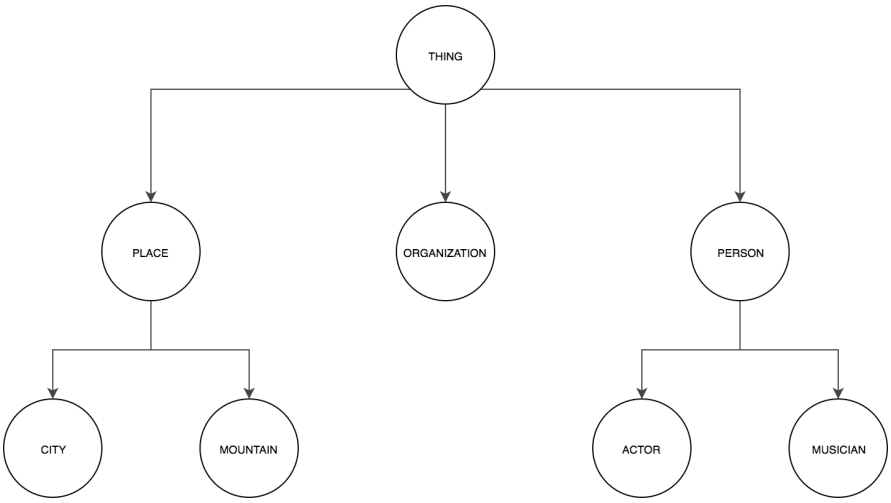


Fig. 1. Example of type taxonomy for a generic extractor.

representation is assigned to each of them. We consider also the stop words, assigning also to them a real-value vectorial representation. The word vectors are computed using *fastText* [1]. We define $\boldsymbol{s^x}$ as the real-valued vector associated to a specific token x:

$$\boldsymbol{s^x} = \left[\boldsymbol{s_p^x} | \boldsymbol{s_c^x}\right], dim(\boldsymbol{s^x}) = 400 \tag{1}$$

where | (pipe) is the concatenation operator and dim is the vector dimension.

$\boldsymbol{s_p^x}$, $dim(\boldsymbol{s_p^x}) = 300$, consists in the token embedding computed using the Wikipedia pre-trained *fastText* models released by the authors. The model changes depending on the language used in the text, since all localised Wikipedia have been used to train language specific models.

$\boldsymbol{s_c^x}$, $dim(\boldsymbol{s_c^x}) = 100$, is the token embedding computed when training *fastText* directly on a particular textual corpus – i.e. the one for which we want to perform the NERD tasks. This means that $\boldsymbol{s_c^x}$ does not vary depending on the language but on the gold standard itself.

2. Type features. Each extractor $e \in T$ has its own type taxonomy o which is a taxonomy of a maximum depth L. In the following, we consider a simple example of an taxonomy o with just a 2 levels hierarchy (Figure 1):

1. Level 1 includes three types: PLACE, ORGANIZATION and PERSON.
2. Level 2 includes four types: CITY and MOUNTAIN (subtypes of PLACE) and ACTOR and MUSICIAN (subtypes of PERSON).

We name C_i the number of different types inside the level i (e.g. $C_1 = 3$). We infer a one-hot encoding representation for each level as shown in Table 3.

For a generic type τ in the last layer (e.g. ACTOR), the features vector v_τ consists in the concatenation of the one-hot representation of each type founded

LEVEL 1		LEVEL 2	
Type	**Representation**	**Type**	**Representation**
PERSON	001	ACTOR	0001
ORGANIZATION	010	MUSICIAN	0010
PLACE	100	CITY	0100
		MOUNTAIN	1000

Table 2. Representation of types through one-hot encoding.

on the walk from the root to the leaf associate to τ. The features vector for ACTOR is therefore `0010001`, where the first three values `001` derive from PERSON and the last four values `0001` derive from ACTOR. Hence, we can state that $dim(v_\tau) = \sum_i^L C_i$. If the extractor $e \in T$ returns a type that is not the last level in the hierarchy, as PERSON, we fill the missing vector positions with `0`. The features vector v_{PERSON} associate to PERSON is thus `0010000`. This mechanism is extensible to any taxonomy. However the $dim(v_\tau)$ is different for each extractor, depending on the taxonomy that it uses.

This procedure can be extended also to extractors that do not perform NER. A generic extractor e, where $e \in U \wedge e \notin T$, returns a link for each entity. Following the interlinks between KBs, we can always obtain an entity in Wikidata. The type of the entity would be the class of this entity in Wikidata, which is the value of the property *instance of (P31)*[3]. Entities might possess multiple types and for this reason they are represented through K-hot encoding.

For a **typed named entity** w^t with the format `(surface form, type)`, the type feature vector $\boldsymbol{v_e^{w^t}}$ is computed for the extractor e where $e \in U \vee e \in T$. $dim(\boldsymbol{v_e^{w^t}})$ varies accordingly to the considered extractor. In fact, we get a real-value numerical type representation without a type alignment phase. For this reason, the number of dimensions that forms the type features vector depends on the the number of types in the extractor taxonomy.

3. Entity features. These features represent the similarity between two Wikidata entities w_1 and w_2, as a vector of 5 dimensions. The first four dimensions correspond to **semantic knowledge**:

1. the first dimension $S_{uri}(w_1, w_2)$ indicates if the compared entities share the same URI with a Boolean;
2. the second dimension provides the string similarity between the labels l_{w_1} and l_{w_2} associated to the compared entities:

$$S_{Lev}(w_1, w_2) = max(1 - d_{Lev}(l_{w_1}, l_{w_2})/\beta, 0), \beta = 8$$

where $d_{Lev}(l_{w_1}, l_{w_2})$ is the **Levenshtein distance** between the compared strings and β is a constant equals to the number of maximum differences after which the similarity is saturated to 0.

[3] `https://www.wikidata.org/wiki/Property:P31`

3. the third dimension $S_{TfIdf}(w_1, w_2)$ represents the **TF-IDF Cosine Similarity** between the abstracts associated to the compared entities. This dimension represents a textual knowledge as in [12];
4. the fourth dimension $S_{occ}(w_1, w_2)$. value indicates if the compared entities share the same *occupation (P106)*.[4] This property is specific for entities of type PERSON: this Wikidata class has no other subclasses, as opposed to the other types. For this reason this similarity dimension greatly helps in the disambiguation of people with similar names but different professions. $S_{occ}(w_1, w_2)$ is set to 1 when the two entities referred to people that have the same profession, and 0 otherwise (different profession or not a PERSON).

The fifth and last dimension of the vector represents the structural similarity as in [12]. We define a property set P, containing three properties: *subclass of (P279)*[5], *instance of (P31)*[6], and *part of (P361)*[7]. A subgraph G is extracted from Wikidata selecting all the triples in which a property in P appears. We define the distance d_{w_1,w_2} between two generic entities w_1 and w_2 as the shortest path length that links w_1 and w_2 in G. Then, we compute the maximum distance between two nodes in the graph G, defining it as d_{max}. We assess the structural similarity between w_1 and w_2 as:

$$S_{stc}(w_1, w_2) = -\frac{d_{w_1,w_2}}{d_{max}} + 1$$

The total similarity between w_1 and w_2 can be expressed as:

$$\begin{aligned} &\boldsymbol{S(w_1, w_2)} = \\ &= [S_{uri}(w_1, w_2), S_{Lev}(w_1, w_2), S_{TfIdf}(w_1, w_2), S_{occ}(w_1, w_2), S_{stc}(w_1, w_2)] \end{aligned} \tag{2}$$

The choice of representing the similarity between two entities as a real-value vectors rather than using an entity embedding is in line with our goal of representing how the extractors differ in the prediction rather than directly representing an entity. This approach avoids to compute embeddings on the whole Wikidata KB. We rely on interlinks between KBs for guaranteeing that we can always compare Wikidata entities. This causes the risk that no Wikidata entity exists for the source one, i.e. because the information is not present. However, this case is very rare (Table 3) in all the considered benchmarks in the evaluation, thanks to the reliance of all the involved extractors on Wikidata, Wikipedia or DBpedia, which containing similar information. This would become a limit when using different KBs (e.g. thematic ones), not fully interlinkable to Wikidata and for which a loss in information should be taken in account.

4. Score features. Some extractors return scores representing either the confidence or the saliency for each named entity. For each extractor $e \in K$, w^k is a **named entity score** with the format `(surface form, scores)`. We define

[4] `https://www.wikidata.org/wiki/Property:P106`
[5] `https://www.wikidata.org/wiki/Property:P279`
[6] `https://www.wikidata.org/wiki/Property:P31`
[7] `https://www.wikidata.org/wiki/Property:P361`

Extractor	Disambiguation KB	WD Coverage
Dandelion	Wikipedia	99%
DBSpotlight	DBpedia Fr	98%
TextRazor	Wikidata	100%
Babelfy	DBpedia	100%

Table 3. Coverage of matching against Wikipedia of disambiguated entity in the ground truth.

$\boldsymbol{v_e^{w^k}}$ as the features vector representing the scores for w^k and the extractor e. $dim(\boldsymbol{v_e^{w^k}})$ depends on the considered extractors, more precisely on the number of scores returned by it.

4 Ensemble NERD: ENNTR and ENND

Our experimental ensemble method relies on two Neural Networks that receive in input the features described in the previous Section. We respectively name them with the acronyms **Ensemble Neural Network for Type Recognition (ENNTR)** and **Ensemble Neural Network for Disambiguation (ENND)**. For both networks, the hyper parameter optimization was done using Grid Search.

These networks architectures come after a series of previous experiments that involved LSTM and BiLSTM, receiving a complete vector including all the features as input sample. A really slow training, the ease of network overfitting to the sample input, and huge difference in dimensionality (and so in impact to the results) between the different features were some of the reasons for which we have abandoned these approaches.

Ensemble Neural Network for Type Recognition (ENNTR). We consider a generic ground truth GT formed by N textual fragments (e.g. sentences), such that we can split each fragment in tokens. X_i is the ordered list of tokens for fragment i. Concatenating the lists X_i, we get a list X, that is the ordered list of tokens for the whole corpus. We call x a generic token in X.

GT associates a type in a taxonomy o_{Gt} to each token x. We identify the neural network target as Y_t. The number of samples in Y_t is equal to the total number of tokens: $dim(Y_t) = dim(X)$. The neural network goal is to assign the right type to each token and its architecture is represented in Figure 2.

ENNTR has an output layer O formed by $H = card(o_{GT})$ neurons, where $card(o_{GT})$ is the number of different types (or cardinality) in o_{GT}. As a consequence, each value returned by a neuron in the output layer corresponds to the probability that a token x belongs to a specific type. Hence, each target sample $\boldsymbol{y_t}$ is a vector formed by H values, where each value corresponds to a type and a neuron. In Figure 2, we are assuming that $H = 4$.

ENNTR presents many input layers. Using the same notation used in Section 3, T is the set of extractors that return type information, K is the set of extractors

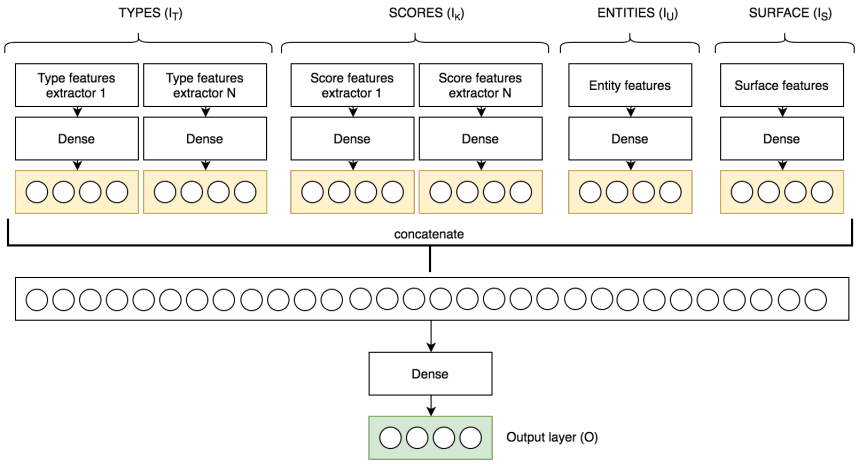


Fig. 2. ENNTR architecture

that return score information, U is the set of extractors that perform disambiguation. Defining I as the set of input layers of ENNTR, we can identify four different types of input layer depending on the kind of features being input.

$$I = I_T \cup I_K \cup I_U \cup I_S$$

$$|I| = |I_T| + |I_K| + |I_U| + |I_S| = |T \cup U| + |K| + 1 + 1$$

All the input layers works at token level, so that the features at entity level defined in Section 3 requires a transformation to token-level. The surface form of an entity w (e.g. *Barack Obama*) can be tokenised, producing the list of tokens X_w (e.g. *[Barack, Obama]*). The feature vector of token x is equal to the one of an entity w if x is a token in X_w. Otherwise it is equal to a padding vector $\boldsymbol{d}$, of the same dimension and containing only `0` values.

In particular, I_T receives in input a type features vector $\boldsymbol{t_e^x}$, computed like:

$$\boldsymbol{t_e^x} = \begin{cases} \boldsymbol{v_e^{w^t}} & if\ x \in X_{w^t} \\ \boldsymbol{d_t} & if\ x \notin X_{w^t} \end{cases} \tag{3}$$

$$\boldsymbol{d_t} = [0, ..., 0], dim(d_k) == dim(\boldsymbol{v_e^{w^t}})$$

Similarly, I_K receives in input a type features vector $\boldsymbol{k_e^x}$, computed like:

$$\boldsymbol{k_e^x} = \begin{cases} \boldsymbol{v_e^{w^k}} & if\ x \in X_{w^k} \\ \boldsymbol{d_k} & if\ x \notin X_{w^k} \end{cases} \tag{4}$$

$$\boldsymbol{d_k} = [0, ..., 0], dim(d_k) == dim(\boldsymbol{v_e^{w^k}})$$

The Wikidata entity u_e^x for the token x is:

$$u_e^x = \begin{cases} u_e^{w^u} \quad if\ x \in X_{w^u} \\ NAN\ if\ x \notin X_{w^u} \end{cases} \tag{5}$$

The layers I_U receive in input the entity features vector $\boldsymbol{u^x}$, computed for a token x as:

$$\boldsymbol{u^x} = [S(u_1^x, u_1^x), S(u_1^x, u_2^x), ..., S(u_P^x, u_P^x)]$$

Finally, the input layers I_S receive the surface features vector $\boldsymbol{s^x}$ without any further transformation.

Each input layer I_n is fully connected with a layer M_n. M_n, like O, is composed by H neurons, where H is the number of types in the ground truth. The activation of neurons in M_n is linear.

In this first part of the network, each I_n —composed by a different number of neurons depending on the related features vector— is mapped on H neurons in M_n. This avoids that the neural network privileges features vectors with higher dimension – it happens directly concatenating different features vectors. This part of the network can be considered as an **alignment block** since it automatically map the types between the extractors and the ground truth taxonomy. This is pretty similar to the *Inductive Entity Typing Alignment* work described in [7], with the difference that the alignment step is learned by a fully connected layer. Differently from previous works [9, 10], the approach does not need any preliminary alignment and recognition, because they are part of the same network.

The last part of the network is the **ensemble block**. M_k layers are concatenated forming a new layer R. $|o_{GT}|$ is the number of types in the ground truth, $|I|$ the number of input layers and $|P|$ the number of neurons in R:

$$|P| = |o_{GT}| \cdot |I|$$

R is fully connected to the output layer O. The activation of the neurons in O is linear. This means that ENNTR finally consists in a linear combinations of features: the key is the way in which the features are generated and entered in the network. The values v_h of the H output neurons in O correspond to the probability that a given type is correct. We take the highest value v_{max} between them and if it is greater than a threshold θ, we set the type related to its neuron as the predicted one. The final output of the ensemble method is a list of predicted type l_p for each token x. In a final step, sequences of token which belong to the same type are merged to a single entity, similarly to [9, 10].

Ensemble Neural Network for Disambiguation (ENND) We consider a ground truth GT, similar to the one seen for ENNTR, that this time associates a Wikidata entity identifier (URI) to each token. We identify the target as Y_d.

The ENND architecture is represented in Figure 3. Differently from related work, the goal of the network would not be to directly predict the right disambiguated entity, but to determine if the predicted entity by an extractor e, where

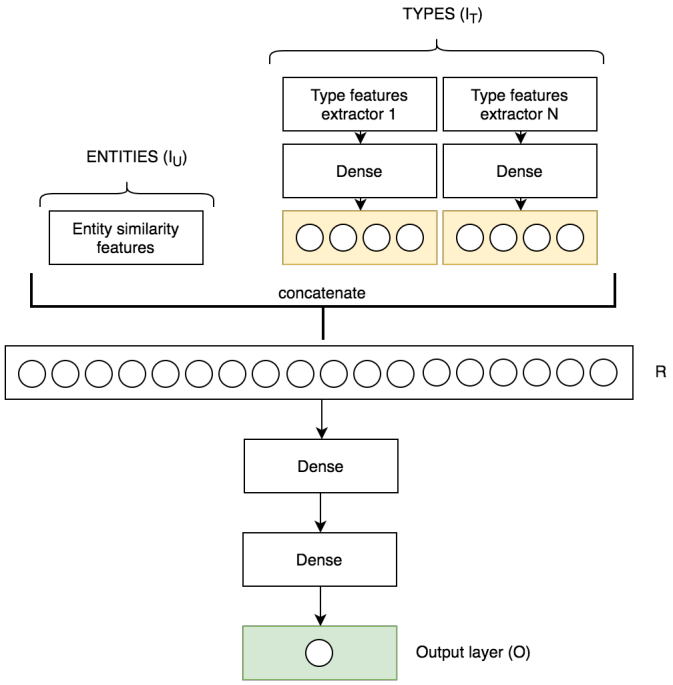


Fig. 3. ENND architecture

$e \in U$, is correct or not. For this reason, the number of samples in target Y_d is not equal to the number of tokens. For each token x, each extractor e returns a predicted entity $\boldsymbol{u_e^x}$: we call C_x the set of predicted entities for the token x, and v_x the correct entity; $|C_x| \leq |U|$ because more extractors could predict the same entity. For each candidate $c_{x,j} \in C_x$, where $0 < j \leq |C_x|$, we generate a target sample $y_d \in Y_d$:

$$y_d = \begin{cases} 1 \; if \; c_{x,j} = v_x \\ 0 \; if \; c_{x,j} \neq v_x \end{cases}$$

The output layer O contains a single neuron that should converge to y_d. The O activation is a sigmoid. Naming I the set of input layers of ENND, two different types of input can be identified depending on the kind of features.

$$I = I_U \cup I_T$$
$$|I| = |I_U| + |I_T| = 1 + |T \cup U|$$

The entity similarity features enter through I_U. We define $c_{x,j}$ as a candidate entity for the token x. For each target sample y_d, we compute a similarity features sample $\boldsymbol{u^{x,j}}$ as:

$$\boldsymbol{u_{x,j}} = [\boldsymbol{S}(c_{x,j}, u_1^x)|\boldsymbol{S}(c_{x,j}, u_2^x)|...|\boldsymbol{S}(c_{x,j}, u_R^x)] \; where \; R = card(U)$$

$$dim(\boldsymbol{u_{x,j}}) = dim(\boldsymbol{S(w_1, w_2)}) \cdot card(U)$$

The input layers I_T receive in input the the type feature vector $\boldsymbol{t_e^w}$, computed with the same method used for ENNTR. I_T layers are fully connected to the layers M_n as in ENNTR. M_n is formed by H neurons, where H is an hyper-parameter, set to 4 during our experiment. As for ENNTR, the M_n activation is linear.

After this step, the I_U layer and the M_k layers are concatenated in a new layer R. In this layer, some neurons represent the type information, some other the entity features. This combination aims to exploit the fact that some extractors better disambiguate on certain types. The number of neurons in R is equal to $dim(\boldsymbol{u_{x,j}}) + |T \cup U| \cdot H$.

The last part of the network is composed by two dense layers[8] and the output layer O discussed before. The activation functions of the dense layers cannot be a *softmax* function since the number of candidates —and so is the number of neurons in the output layer— is variable according to each specific token. We so opted for the **Scaled Exponential Linear Units (selu)**:

$$selu(x) = \lambda \begin{cases} x & if\ x > 0 \\ \alpha e^x - \alpha & if\ x \leq 0 \end{cases}$$

The loss function used to train the network is the Mean Square Error, that gives slightly better results and similar training time if compared to MSE.

The neural network goal is to determine the probability that an entity candidate is right. In fact, for each sample, we get an output value that corresponds to this probability. $o_{x,j}$ corresponds to the output value of the input sample associated to the candidate entity j for token x. We select the candidate associated with the highest value $o_{x,max}$ among all output values $\left\{o_{x,1}, o_{x,2}, ..., o_{x,card(C_x)}\right\}$. Defining a threshold τ_d, if $o_{x,max} > \tau_d$, we can select as predicted entity for token x the one related to $o_{x,max}$. Otherwise, we consider that the token x is not part of a named entity. This process of **candidate selection** returns the list z_p of predicted Wikidata entities identifiers at token level. In a final step, sequences of tokens which belong to the same Wikidata entity identifiers are merged to a single entity. A_p represents the predicted corpus of annotated fragments.

5 Experiment and Evaluation

We developed an implementation of the two neural networks using Keras.[9] In order to make our approach comparable with the state of the art, our evaluation relies on well-known corpora and metrics, which have been already applied to related work. Moreover, we evaluate our approach on a new gold standard that we provide to the community.

[8] A *dense layer* is a layer fully connected to the previous one.

[9] The source code is available at `https://github.com/D2KLab/ensemble-nerd`, together with the documentation for accessing the live demo at `http://enerd.eurecom.fr`

– **OKE2016**: annotated corpus of English textual resources, created for the 2016 OKE Challenge. The types set contains 4 different tags. [10] This ground truth disambiguates the entities using DBpedia. The ensemble technique we use for scoring is averaging, but not boosting or bagging.
– **AIDA/CoNLL**: English corpus and contains assignments of entities to the mentions of named entities, linked to DBpedia. This dataset does not infer types for NEs and can only be used for evaluating NED.
– **NexGenTV corpus**:[11] dataset composed of 77 annotated fragments of transcripts from politician television debates in French.[12] Each fragment lasts in average 2 minutes. The corpus is split in 64 training and 13 test samples. The list of types includes 13 different labels.[13] Entities are disambiguated through Wikidata.

	TOKEN BASED			ENTITY BASED		
	fsc	**pre**	**rec**	**fsc**	**pre**	**rec**
adel	0,87	0,88	0,87	0,84	0,85	0,83
alchemy	0,79	0,93	0,68	0,88	0,92	0,86
babelfy	0,66	0,88	0,7	0,74	0,79	0,7
dandelion	0,64	0,89	0,51	0,78	0,83	0,75
dbspotlight	0,59	0,75	0,49	0,6	0,77	0,52
meaning cloud	0,59	0,91	0,44	0,72	0,78	0,69
opencalais	0,56	0,97	0,39	0,69	0,71	0,68
textrazor	0,74	0,86	0,65	0,77	0,81	0,74
ensemble	**0,91**	**0,91**	**0,91**	**0,94**	**0,95**	**0,92**
ensemble ($I = I_T$)	**0,88**	**0,91**	**0,85**	**0,88**	**0,92**	**0,84**
ensemble ($I = I_S$)	**0.50**	**0,53**	**0,47**	**0.50**	**0,52**	**0,48**
ensemble ($I = I_U$)	**0.44**	**0,47**	**0,41**	**0.43**	**0,43**	**0,43**
ensemble ($I = I_K$)	**0,37**	**0,40**	**0,34**	**0,38**	**0,40**	**0,36**

Table 4. OKE2016 corpus NER Evaluation

Type recognition. For each gold standard GT, two different kinds of score are computed. The *token based* scores have been used in [9, 10]. From GT, a list of target types l_t with dimension $|X|$ is extracted. We can obtain from ENNTR the list of predicted types l_p. For each type t_{GT} in GT, we compute precision $Precision(l_t, l_p, t_{GT})$, recall $Recall(l_t, l_p, t_{GT})$ and F1 score $F1(l_t, l_p, t_{GT})$. Then,

[10] PERSON, ORGANIZATION, PLACE, ROLE.
[11] `http://enerd.eurecom.fr/data/training_data/nexgen_tv_corpus/`
[12] The debates are in the context of the 2017 French presidential election.
[13] PERSON, ORGANIZATION, GEOGRAPHICAL POINT, TIME, TIME INTERVAL, NUMBER, QUANTITY, OCCURRENCE, EVENT, INTELLECTUAL WORK, ROLE, GROUP OF HUMANS and OCCUPATION.

we compute micro averaged measures $Precision_{micro}(l_t, l_p)$, $Recall_{micro}(l_t, l_p)$ and $F1_{micro}(l_t, l_p)$. [8]

The *entity based* scores follow the definition of precision and recall coming from the **MUC-7 test scoring** [2]. Given A_t and A_p as the annotated fragment in GT, the computed measures are $Precision_{brat}(A_t, A_p)$, $Recall_{brat}(A_t, A_p)$ and $F1_{brat}(A_t, A_p)$.

The computed scores for OKE2016 and NexGenTv corpora are reported in Table 4 and 5. The tables show also the same metrics applied to single extractors, after that their output types have been mapped to the ones of GT through the alignment block of ENNTR. For both token and entity scores, the ensemble method outperforms the single extractors for all metrics.

	TOKEN BASED			ENTITY BASED		
	fsc	**pre**	**rec**	**fsc**	**pre**	**rec**
adel	0,68	0,84	0,57	0,75	0,83	0,7
alchemy	0,80	0,83	0,77	0,87	0,97	0,81
babelfy	0,55	0,83	0,41	0,65	0,74	0,59
dandelion	0,26	0.69	0,16	0,51	0,69	0,42
dbspotlight	0,48	0,75	0,34	0,5	0,61	0,45
meaning cloud	0,82	0,88	0,77	0,8	0,87	0,76
opencalais	0,58	0,81	0,45	0,81	0,9	0,76
textrazor	0,81	0,89	0,74	0,75	0,8	0,72
ensemble	**0,94**	**0.97**	**0,91**	**0,92**	**0,98**	**0,87**
ensemble ($I = I_T$)	**0,87**	**0,91**	**0,83**	**0,89**	**0,93**	**0,85**
ensemble ($I = I_S$)	**0.54**	**0,58**	**0,50**	**0.53**	**0,56**	**0.50**
ensemble ($I = I_U$)	**0.47**	**0,49**	**0,45**	**0.46**	**0,47**	**0,45**
ensemble ($I = I_K$)	**0,40**	**0,42**	**0,38**	**0,39**	**0,40**	**0,38**

Table 5. NexGenTv corpus NER Evaluation

In order to identify the most impacting features in the obtained results, ENTTR has been sequentially adapted and retrained in order to receive in input only a specific kind of features, i.e. only I_T, I_K, I_U or I_S. The tokens based scores for these new trained networks reveals that the type features I_T are the only ones that, used alone as input, continue to make ENTRR outperforming single extractors, as can be expected given the type recognition goal. The other feature kinds, while having a lower impact, are still improving the final results when combined in the ensemble.

Entity Linking. We evaluate the entity linking for both OKE2016, AIDA/CoNLL and NexGenTv corpora using the GERBIL framework[14] and in particular micro

[14] GERBIL is a general Linked Data benchmarking that offers an easy-to-use web-based platform for the agile comparison of annotators using multiple datasets and uniform measuring approaches.

and macro scores for the experiment type "Disambiguate to Knowledge Base" (D2KB). The computed scores are reported in Table 6 and 7; the ensemble method outperforms again the single extractors that it integrates for all metrics. As for type recognition, we repeated the experiment using only a specific kind of features, in order to show the feature impact. In such case, the most influential features are the entity ones I_U. However, the impact of type features I_T is still crucial because its absence reduce drastically the improvement of the ensemble method with respect to the single extractors.

Table 8 and 9 compare the NED extractors presented on GERBIL with our ensemble. For OKE2016, PBOH is the only tool which obtains a better score However this extractors reaches very low scores for AIDA/CoNLL, while our ensemble still continues to have good performances. For the NexGenTV dataset, we cannot compare the other NERD extractors because the majority of them perform NED only for the English language.

6 Conclusion and Future Work

In this paper, we presented two multilingual ensemble methods which combine the responses of web services (extractors) performing Named Entity Recognition and Disambiguation. The method relies on two Neural Networks that outperform the single extractors respectively in NER and NED tasks. Furthermore, the NER network allows to avoid the manually type alignment between the type taxonomies of each extractor and the ground truth taxonomy. We demonstrated the importance of the features generation for the success of these ensemble methods. In terms of NER, the type features play most of the work in the ensemble. For the NED task, while entity features have the greater impact, only a combination with type features really improve the effectiveness of the ensemble method with respect to single extractor predictions.

As future work, we plan to enhance the input feature set with Part of Speech tags features that would be assigned to each token. We also aim to vary the neural network architecture, and in particular, we are planning to replace the dense layer receiving the surface features with a BiLSTM, which would also take in consideration the context in which the tokens are sequentially appearing. Finally, all the neural networks models have been trained when all extractors APIs were reachable. A training that involves some samples which simulates the extractors failures and unavailability would make the network models more robust to API failures.

Acknowledgements

This work has been partially supported by the French National Research Agency (ANR) within the ASRAEL project (ANR-15-CE23-0018), the French Fonds Unique Interministériel (FUI) within the NexGen-TV project and the European Union's Horizon 2020 research and innovation programme via the project MeMAD (GA 780069).

	OKE2016			NEXGEN			AIDA		
	fsc	**pre**	**rec**	**fsc**	**pre**	**rec**	**fsc**	**pre**	**rec**
babelfy	0,54	0,64	0,47	0,51	0,51	0,51	0,66	0,70	0,62
dandelion	0,59	0,77	0,48	0,34	0,50	0,26	0,45	0,66	0,34
dbspotlight	0,39	0,53	0,30	0,38	0,29	0,54	0,47	0,65	0,36
textrazor	0,53	0,78	0,40	0,61	0,55	0,69	0,62	0.57	0.53
ensemble	**0,66**	**0,88**	**0,52**	**0,69**	**0,70**	**0,64**	**0,68**	**0,79**	**0,60**
ensemble ($I = I_U$)	**0,59**	**0,80**	**0,47**	**0,59**	**0,60**	**0,58**	**0,55**	**0,60**	**0,50**
ensemble ($I = I_T$)	**0,41**	**0,45**	**0,38**	**0,42**	**0,47**	**0,38**	**0,48**	**0,52**	**0,45**

Table 6. GERBIL Micro scores on OKE2016, NexGenTV and AIDA/CoNLL corpus

	OKE2016			NEXGEN			AIDA		
	fsc	**pre**	**rec**	**fsc**	**pre**	**rec**	**fsc**	**pre**	**rec**
babelfy	0,54	0,65	0,47	0,51	0,52	0,51	0,60	0,65	0,57
dandelion	0,59	0,76	0,49	0,35	0,50	0,27	0,43	0,52	0,37
dbspotlight	0,39	0,52	0,32	0,38	0,29	0,55	0,45	0,63	0,37
textrazor	0,54	0,77	0,42	0,61	0,54	0,71	0,57	0,78	0,45
ensemble	**0,65**	**0,86**	**0,53**	**0,67**	**0,69**	**0,64**	**0,68**	**0,76**	**0,61**
ensemble ($I = I_U$)	**0,59**	**0,77**	**0,48**	**0,59**	**0,59**	**0,59**	**0,55**	**0,59**	**0,51**
ensemble ($I = I_T$)	**0,42**	**0,44**	**0,40**	**0,41**	**0,42**	**0,40**	**0,49**	**0,51**	**0,47**

Table 7. GERBIL Macro scores on OKE2016, NexGenTV and AIDA/CoNLL corpus

	Micro scores			Macro scores		
	fsc	**pre**	**rec**	**fsc**	**pre**	**rec**
agdistis	0,50	0,50	0,50	0,52	0,52	0,52
aida	0,49	0,63	0,41	0,5	0,64	0,42
dexter	0,44	0,92	0,29	0,43	0,81	0,31
fox	0,48	0,77	0,35	0,47	0,69	0,37
freme ner	0,31	0,57	0,21	0,26	0,27	0,25
kea	0,64	0,67	0,61	0,63	0,66	0,61
pboh	0,69	0,69	0,69	0,69	0,69	0,69
ensemble	**0,66**	**0,88**	**0,52**	**0,65**	**0,86**	**0,53**

Table 8. GERBIL scores on OKE2016

	Micro scores			Macro scores		
	fsc	**pre**	**rec**	**fsc**	**pre**	**rec**
agdistis	0,58	0,58	0,58	0,59	0,59	0,59
aida	0,00	0,00	0,00	0,00	0,00	0,00
dexter	0,51	0,76	0,38	0,47	0,75	0,36
fox	0,57	0,63	0,51	0,56	0,64	0,51
freme ner	0,38	0,62	0,27	0,29	0,30	0,27
kea	0,60	0,65	0,56	0,59	0,63	0,56
pboh	0,00	0,00	0,00	0,00	0,00	0,00
ensemble	**0,68**	**0,79**	**0,60**	**0,68**	**0,76**	**0,61**

Table 9. GERBIL scores on AIDA-CoNLL

References

1. P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.
2. N. Chinchor. Appendix b: Muc-7 test scores introduction. In *Seventh Message Understanding Conference (MUC-7)*, Fairfax, Virginia, USA, 1998.
3. J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *43rd Annual Meeting on Association for Computational Linguistics (ACL)*, pages 363–370, Ann Arbor, Michigan, USA, 2005.
4. L. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. In *13th Conference on Computational Natural Language Learning (CoNLL)*, pages 147–155, Boulder, Colorado, USA, June 2009.
5. G. Rizzo and R. Troncy. Nerd: A framework for unifying named entity recognition and disambiguation extraction tools. In *13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 73–76, Avignon, France, 2012.
6. G. Rizzo, M. van Erp, and R. Troncy. Benchmarking the Extraction and Disambiguation of Named Entities on the Semantic Web. In *9th International Conference on Language Resources and Evaluation (LREC)*, Reykjavik, Iceland, 2014.
7. G. Rizzo, M. van Erp, and R. Troncy. Inductive Entity Typing Alignment. In *1st International Workshop on Linked Data for Information Extraction (LD4IE)*, Riva del Garda, Italy, 2014.
8. F. Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, 2002.
9. R. Speck and A.-C. N. Ngomo. Ensemble learning of named entity recognition algorithms using multilayer perceptron for the multilingual web of data. In *9th International Conference on Knowledge Capture (K-CAP)*, Austin, TX, USA, 2017.
10. R. Speck and A.-C. Ngonga Ngomo. Ensemble learning for named entity recognition. In *13th International Semantic Web Conference (ISWC)*, pages 519–534, Riva del Garda, Italy, 2014.
11. M. van Erp, G. Rizzo, and R. Troncy. Learning with the Web: Spotting named entities on the intersection of NERD and machine learning. In *3rd International Workshop on Making Sense of Microposts (#MSM), Concept Extraction Challenge*, Rio de Janeiro, Brazil, 2013.
12. F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma. Collaborative Knowledge Base Embedding for Recommender Systems. In *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 353–362, San Francisco, California, USA, 2016.