ConnectionLens: Entity and Relationship Extraction from French textual data sources

Catarina Conceição
Instituto Superior Técnico
Universidade de Lisboa
Lisboa, Portugal
catarina.conceicao@tecnico.ulisboa.pt

***Abstract*—As a result of the large amounts of data digitally available nowadays, journalists are turning their attention to data processing and visualization, a task called *data journalism*. In investigative journalism, the available data is used to find connections between entities and analyze their nature. CONNECTIONLENS is a software prototype that addresses the investigative journalism's issues of having data from different sources and different formats, while allowing keyword-based queries to find connections. To obtain the entities and connections in textual data sources it is necessary to perform Named-Entity Recognition (NER) and Relationship Extraction (RE). We propose to develop a solution for NER and RE for French news texts that can be incorporated in CONNECTIONLENS. Our goal is to adapt and make use of tools, more specifically, third-party libraries, for both NER and RE, to create machine learning models capable of extracting named-entities and relationships, respectively, from French texts. In addition, to provide a comprehensive evaluation of these models using precision, recall and $F1$-score for NER, and precision-recall curves, area under the curve (AUC), micro-$F1$ and Precision@N for RE. Finally, to select the best performing model for each task, to be integrated in CONNECTIONLENS. The best performing model for NER achieved an overall $F1$-score of 73.31%. And, the best performing model for RE achieved an AUC and a micro-$F1$ of 97.10% and 91.78%, respectively.**

***Index Terms*—Information Extraction, Natural Language Processing, Named-Entity Recognition, Relationship Extraction, Deep Learning, Distant Supervision**

I. INTRODUCTION

Traditionally, journalists focused on gathering data and being the first to deliver news. Nowadays, with the large amount of data digitally available, journalists are shifting their focus to extracting, transforming and visualizing data with the goal of having a good story to tell. These data processing and visualization tasks are typically known under the name *data journalism* [1]. Furthermore, investigative journalism uses available data and tries to find connections between entities and analyze their nature. There is typically a network of interconnections between those entities that is not visible. It is the intention of investigative journalism to bring to light these interconnections, through the analysis of combined data from different sources. The set of data sources used in investigative journalism may be heterogeneous and independently produced. Moreover, it is necessary to deal with the changing nature of the data sources because journalists are always collecting more data with different structure and format. Once the data is integrated, journalists need to query it to find connections. They do not know the exact structure of the data, so queries are typically keyword-based.

CONNECTIONLENS [2] is a software prototype that was developed to address the investigative journalism problem described above. This prototype supports keyword search across a set of heterogeneous and independently produced data sources, to find connections. It deals with different types of data sources, namely JSON documents, text files, RDF graphs and relational tables. CONNECTIONLENS was developed in the context of the ContentCheck ANR project[1] with the French newspaper Le Monde, so it focuses on French data. This project was a collaborative project between the Inria CEDAR team[2] and AIST Japan[3].

In CONNECTIONLENS all data sources are mapped into a single virtual graph. Most nodes and edges of the virtual graph come from the data sources. Except for the text type, data sources have inherently defined entities and connections, so, to define nodes and edges it is only necessary to map them to the virtual graph. Edges in the virtual graph can also be links between two nodes (of the same or from different data sources) whose data is considered to be similar. These are called *sameAs* links e.g., "Philippe Varin" and "P. Varin".

To obtain the entities and connections present in a natural language text it is necessary to perform Information Extraction (IE) [3], in particular, Named-Entity Recognition (NER) and Relationship Extraction (RE). NER is a typical task of IE that focuses on identifying names of entities and then classifying them into a pre-defined set of classes, like people, locations, organizations and others. RE is a task of IE that aims at extracting relationships, usually from a pre-defined set, between the previously identified entities.

The goal of this work is to develop a solution for NER and RE for French news texts that can be incorporated in CONNECTIONLENS.

A. Existing NER and RE Solutions

Approaches for performing both NER and RE may consist in using a software tool or implementing a technique from

[1]https://team.inria.fr/cedar/contentcheck/
[2]https://team.inria.fr/cedar/
[3]https://www.aist.go.jp/waterfront/

scratch. Software tools implement techniques and are able to perform the given task *off-the-shelf* and/or facilitate its implementation.

The software tools that are available for NER can be: (i) black-box or (ii) third-party libraries. Black-box tools are usually offered as a web service made available by an API, to which one makes a request sending a text as input and it returns the named-entities it was able to extract. We typically have no knowledge of what is happening internally in the system and these solutions usually support a maximum number of accesses per time period. The constraints they present are not desired for integrating CONNECTIONLENS. On the other hand, third-party libraries implement a machine learning (ML) technique and only require train data to be given as input to obtain a ML model. They also make pre-trained models available, which are models already trained and parameterized. We found several third-party libraries for NER that implement state-of-the-art supervised neural-based techniques. In addition there are freely available datasets, in French, annotated with named-entities that can be given as input to those tools. Some of the third-party libraries we found, also make available French pre-trained models, in particular, Flair[4] and SpaCy[5].

In what concerns RE, there are no datasets annotated with entities and relationships for French. Therefore, we are limited to using software tools that can perform French RE *off-the-shelf* or that facilitate the implementation of RE, by using techniques that do not require manually labeled data. There are three types of software tools capable of performing RE: (i) black-box, (ii) Open IE systems and (iii) third-party libraries.

We only found two tools that can perform RE for French *off-the-shelf* : IBM Watson NLU[6] and the French version of ReVerb[7]. IBM Watson NLU is a black-box web service and thus entails the typical limitations and constraints, that are not desired for CONNECTIONLENS. Moreover, the French version of ReVerb is an Open IE system, which means we have no control over which relationships are extracted, which is equally not desired for extracting relationships in CONNECTIONLENS.

Regarding third-party libraries, we only found one that allowed training non supervised models, which is OpenNRE[8]. It integrates training of bag-level RE models, a widely applied method for distantly supervised RE. Moreover, distantly supervised RE proposes a procedure that automatically creates annotated data for training ML models, using relationship instances from a knowledge base (KB), and sentences expressing the relationships, from a text corpus.

B. Objectives

Our goal is to adapt and make use of tools, more specifically, third-party libraries, for both NER and RE, to create ML models capable of extracting named-entities and relationships, respectively, from French texts. In addition, we aim at performing a comprehensive evaluation of these models using precision, recall and $F1$-score for NER, and precision-recall curves, area under the curve, micro-$F1$ and Precision@N for RE. Finally, we will choose the best performing models for each task to be integrated in CONNECTIONLENS.

C. Contributions

The main contributions of this work are as follows:

1) Pre-processing and data exploration of three different French NER datasets with the goal of uniformizing and combining them to have as much information as possible for training.
2) The creation of multiple French NER models using Flair and SpaCy. The best performing model out of all was selected.
3) A comprehensive evaluation of the selected NER model against pre-trained models, and the model previously used in CONNECTIONLENS. To impartially evaluate the models we used the FTBNER dataset. According to the results, we trained a final model that outperformed all other models evaluated.
4) The creation of a French RE dataset using distant supervision.
5) The creation of a French RE model using OpenNRE that implements bag-level training, a widely applied method for distantly supervised RE.
6) Evaluation and experiments with dataset variants for training the French RE model, evaluated using held-out evaluation. The best performing model was selected.

The description of the NER solution, i.e., contributions 1, 2 and 3, is included in the paper [4], accepted to the 36th Conference "Gestion de Données - Principes, Technologies et Applications" (BDA 2020).

II. RELATED WORK

In this section related work for NER and RE is presented, in particular existing techniques and software tools.

A. Named-Entity Recognition

NER can be seen as a sequence labeling task, where given as input a sequence of words, e.g. a sentence, a sequence of labels, i.e., named-entity types, is returned as output. The labels will capture not only the type of the named-entities, but also their boundary. To label the tokens, an encoding scheme that encodes the boundaries and the labels of each named-entity in the original text is necessary. Approaches typically use the IOB [5] (Inside, Outside and Beginning) notation.

The different techniques that exist for NER can be grouped in the following standard methods [6]: (i) supervised, that can be further divided in feature-based and neural-based, and (ii) rule-based. Supervised methods consist in algorithms that create a model by learning from annotated training examples. In the following sections we will give more details regarding supervised techniques, giving more focus to neural-based methods, that better fit the scope of this work.

[4]https://github.com/flairNLP/flair
[5]https://spacy.io/
[6]https://www.ibm.com/watson/services/natural-language-understanding
[7]https://github.com/rali-udem/reverb-french
[8]https://github.com/thunlp/OpenNRE

1) Feature-based Methods: In feature-based techniques, each word is described by a feature vector, that contains several attributes of the word i.e., features. An example is the part-of-speech (POS) tag of the word, which is a clue for the prediction, because named-entities usually correspond to proper nouns.

One solution for the NER sequence labeling problem would be to independently classify each token of the input sequence using a conventional classifier, e .g., Support Vector Machines [7]. However it is a sub-optimal solution, because context is not taken into account. In other words, the optimal label of a certain token of the sequence should depend on the labels of the neighboring tokens, because labels have conditional dependencies. That is where sequence models, also called sequence classifiers, are useful, because when they are classifying a token, they take into consideration the labels previously assigned to the preceding tokens.

Different algorithms have been used over the years for performing NER. Considerable work was done using sequence models: Hidden Markov Models (HMM) [8], Maximum Entropy Markov Models (MEMM) [9], and Conditional Random Fields (CRF) [10].

2) Neural-based Methods: Deep learning [11] is a family of ML methods, that are composed of many processing layers, typically artificial neural networks. These attempt to simulate the human brain and contain a series of interconnected artificial neurons (or units) arranged in layers, which have associated *weights*. The *input layer* receives information of a certain format, processes and learns about it in *hidden layer*, and, an *output layer* that makes a decision or prediction about the input. Deep learning methods use deep artificial neural networks, which are networks composed of more that one hidden layer. They are not only able to learn to make predictions but also able to transform the input data to its most suitable representation for prediction i.e., learn features directly from the data. This is achieved by feeding the data into the network that will then successively transform it until the output is predicted in a final transformation. Then, the errors are propagated back through the network, adjusting the network's *weights*.

Deep learning methods are independent from hand-crafted features, that require some engineering.

Recent methods for NER are deep learning based or neural based, and achieve state-of-the-art results. Furthermore, we can say that the general architecture of a neural-learning based NER model is composed of an *embedding layer* (input layer), a *context encoding layer* and a *sequence labeling layer* (output layer).

An *embedding layer* (or *lookup layer*) maps the input sequence of words to a sequence of vectors which are distributed representations of the words i.e., *word embeddings*.

A *context encoding layer* captures the context dependencies from the input representations and produces context-dependent representations. Usually a Recurrent Neural Network (RNN) is used to achieve this, which is capable of analyzing sequential data. Furthermore, it earns the name "recurrent" because it makes the same computation for every element in the input sequence which is dependent on the previous computations. Long Short-Term Memory (LSTM) networks are a RNN variant that incorporate a memory-cell and, thus, are capable of capturing long range dependencies, as opposed to RNN. With a bidirectional LSTM (bi-LSTM) network, both past features and future features can be taken into account for prediction. A bidirectional LSTM consists on using two LSTMs, one that reads the input sequence in a left-to-right manner (*forward* LSTM), capturing the left context of a word, and another that reads the sequence in reverse order, that captures the right context of the word (*backward* LSTM).

The *sequence labeling layer* predicts the labels of the words in the original sequence. CRFs are widely used in this context. The use of this layer will also allow using past and future labels to predict the current label.

The standard neural model for NER is to combine a bi-LSTM network with a CRF to form a bi-LSTM-CRF model.

One of the first works that uses a bi-LSTM-CRF architecture for NER is [12], where word embeddings are combined with hand-crafted features. These are passed directly to the CRF layer, instead of passing through the bi-LSTM layer. This accelerates the training while having similar accuracy. A similar model was applied by [13], where no hand-crafted features were used, instead, character embeddings learned during training concatenated with word embeddings were used.

More recent state-of-the-art works use deep learning for NER with new word embedding techniques. Traditional word embeddings (e.g., fastText [14]) are static, meaning that a word's representation is the same no matter its context. New word embedding techniques are dynamic, in the sense that the word's representation is dependent on its context. BERT (Bidirectional Encoder Representations from Transformers) [15] is a new distributed representation approach that achieves state-of-the-art results for NER. It pre-trains bidirectional representations of words' contexts i.e., conditions on both left and right context. Another state-of-the-art approach for NER uses [16] *flair embeddings* that are character-level word embeddings dependent on context.

	BB	Third-party library	French
Stanford NER		x	
SpaCy		x	x
Apache OpenNLP		x	
NeuroNER		x	
Flair		x	x
NLTK		x	
IBM Watson NLU	x		x
Open Calais	x		x

TABLE I
NER TOOLS

3) Tools: Table I summarizes the most relevant NER tools. We found multiple black-boxes (BB) capable of performing NER. We can state that most of the black-boxes support multiple languages. Some of those include IBM Watson NLU and Open Calais, that are capable of, among other languages, of dealing with French texts.

We found several third-party libraries: Stanford NER, SpaCy, Apache OpenNLP, NeuroNER and Flair. Only Flair and SpaCy have available pre-trained NER models for French.

Moreover, Stanford NER and Apache OpenNLP allow training models that implement a feature-based technique. These types of techniques require feature engineering. Additionally, NLTK supports training Stanford NER models.

SpaCy, NeuroNER and Flair allow training recent neural-based state-of-the-art techniques. NeuroNER uses a combination of character embeddings with pre-trained word embeddings to train a LSTM with or without a CRF in the output layer. Flair allows combining various word embeddings that are passed to a bi-LSTM-CRF. The architecture used by NeuroNER can be achieved by using Flair, except for the fact that NeuroNER uses a LSTM instead of a bi-LSTM, whose bi-directionality has proven to be better for understanding context. Out of the third-party libraries presented, we consider that Flair and SpaCy have the potential to train better performing models when compared to the other libraries' techniques.

B. Relationship Extraction

The focus of this work will be on binary relationships, which are relationships that occur between two entities. In binary RE, relationships are extracted from natural language text, typically from a sentence, in the form of triples. The triples can be defined as relationship instances of the form *(ent_1, ent_2, rel)*, where ent_1 and ent_2 are named-entities and rel is the relationship between them, e.g., *(Barack Obama, Honolulu, birthPlace)* extracted from the sentence *"Barack Hussein Obama II, né le 4 août 1961 à Honolulu"*.

There are various types of techniques than can used to perform RE [6]: (i) rule-based, (ii) supervised (iii) semi-supervised, (iv) distantly supervised and (v) unsupervised. In the following sections we will be focusing on the RE techniques that do not require manually labeled data or a high amount of linguist knowledge, giving more focus to distantly supervised methods, that better fit the scope of this work.

1) Semi-supervised Methods: Semi-supervised RE, through the use of an iterative bootstrapping process, assumes that we have a few seed triple instances of the target relationship(s) and a large unlabeled text corpus. A bootstrapping procedure [6] takes the seed triple instances as input and looks for sentences, in the unlabeled text corpus, where the entities occur together. Then, it takes the context of the sentences and generates patterns that are then used to extract new triple instances. The process repeats until reaching a stopping criteria. It is also possible to start the process by using seed patterns. A phenomena called semantic drift can commonly occur in these type of approaches: when a wrong pattern extracts wrong instances which will lead to the generation of wrong patterns, making the extracted instances "drift".

2) Distantly Supervised Methods: Distant supervision [17] was proposed as an alternative paradigm to RE. It uses the idea of bootstrapping, by similarly using seed relationship instances. However, instead of using a small amount of seed instances or patterns to start the process, it uses a large database that contains relationships, i.e., knowledge base (KB), e.g., DBPedia[9]. The assumption (known as the *distant supervision assumption*) is that, *if a pair of named-entities in the KB hold a certain relationship, then any sentence that contains both entities is probably expressing that relationship*. Following the assumption, for every triple *(ent_1, ent_2, rel)* in the KB, sentences containing the entity pair are collected from a large unlabeled text corpus. Furthermore, features are extracted from all the sentences that contain the entities, and used to create training instances to train a classifier. Additionally, the process requires NER to, first, identify the named-entities in every sentence of the corpus. Negative training instances i.e., instances expressing the nonexistence of a relationship between two entities, are also necessary. These are created by randomly taking entity pairs from the KB that do not occur together in a relationship, and applying the same procedure.

The approach proposed by [17] used Freebase as the KB, and collected sentences from Wikipedia articles. They used a multi-class logistic regression classifier, that receives as input a pair of entities and a respective feature vector. The features extracted from all the sentences, for a given entity pair, were aggregated in a single feature vector.

Following the distant supervision assumption generates noisy instances because: (i) not all sentences containing both entities express the relationship in the KB, which leads to the creation of *false positive* instances, (ii) if the KB is incomplete, i.e., does not contain all entity combinations for a given relationship, when creating negative instances, false negative instances will be generated and (iii) two entities may hold more than one relationship between them, meaning that a sentence containing both entities may not express the relationship from the current triple being processed.

The work of [18] states that the noise generated by the distant supervision assumption is even more evident when using text not directly related to the KB, a harder and more real scenario. To fix the noise problem, they propose to relax the distant supervision assumption, creating the *expressed-at-least-once assumption*, that says: *if a pair of named-entities is related in the KB, at least one sentence containing both entities might express the relationship*. The distant supervision problem becomes a multi-instance learning (MIL) problem, where the sentences are grouped in bags, for each entity pair, that, according to the assumption, will contain at least one positive example for their relationship. Furthermore, this means that the RE stops being on sentence-level, where a relationship is predicted for each input sentence, to start being on a bag-level, where a relationship is predicted for each entity pair or bag. They train a graphical model on data created by applying their proposals to the New York Times (NYT) corpus, using Freebase as the KB.

The previous model is multi-instance single-label, thus does not capture that the same entity pair may have more than one relationship between them. Moreover, [19] present MultiR, a probabilistic graphical model of MIL that allows a pair of

[9]https://wiki.dbpedia.org/

entities to have multiple labels, i.e., more than one relationship, outdoing the previous model.

Most features of these models are derived from NLP tools, that introduce errors and noise into the models. [20] introduced deep learning with MIL to distant supervision using Piecewise Convolutional Neural Networks (PCNN) to automatically learn the representation of instances. A Piecewise CNN, (PCNN) is a variant of a CNN with an added *piecewise max pooling layer*, that has the goal of capturing structural information between the two entities expressing a relationship in the input sentence. Furthermore, they showed that using a PCNN is more beneficial that using CNNs.

Moreover, [21] propose a sentence-level attention-based model. A CNN or PCNN is used to construct a sentence representation, and, then, to alleviate the noise caused by the noisy instances, sentence-level *attention* is used to select, from all sentence instances of a given entity pair, the ones that actually express the relationship. They show that PCNN with selective attention over instances (PCNN-ATT) performs better than CNN-ATT.

Some of the most recent state-of-the-art work, like HRERE [22], propose to unify the learning of RE and KB embeddings (KBE) to improve RE itself. KBE is task whose goal is to represent the KB entities and relationships in a vector space.

3) Unsupervised Methods: Unsupervised methods, also denoted as Open Information Extraction (Open IE) [23] methods, do not require any annotated data nor a defined set of target relationships. Instead, an Open IE technique, automatically extracts all relationship types it is capable of finding in a text. A generic way of expressing relationships between two entities, commonly referred to as arguments in Open IE, is needed; [23] proposes the identification of relationship phrases which are phrases that typically correspond to relationships.

	BB	OIE system	Third-party library	French
Stanford OpenIE		x		
TextRazor	x			
Open Calais	x			
IBM Watson NLU	x			x
ReVerb		x		x
OLLIE		x		
OpenNRE			x	

TABLE II
RE TOOLS

4) Tools: Table II summarizes the most relevant RE tools. We only found two tools capable of performing French RE *off-the-shelf*: IBM Watson NLU and the French adaptation of ReVerb. The first is a black-box (BB) and the latter an Open IE system. Moreover, TextRazor and Open Calais are other black-box tools that are only capable of dealing with English texts. Stanford OpenIE, ReVerb, OLLIE are all Open IE systems. Except for ReVerb which has an adaption for French, the systems are only capable of dealing with English texts.

OpenNRE was the only third-party library that we found, that besides allowing to train supervised ML models, i.e., sentence-level training, also implement bag-level training, a widely used setting for distant supervision. The techniques it implements for RE are neural-based.

III. Creating a French NER model

In this section we present the approach that we took for developing a NER solution to integrate ConnectionLens.

Since we found several French datasets annotated with named-entities, we decided to train our own French NER supervised ML model using two third-party libraries, i.e., SpaCy and Flair, to see if we can improve upon their pre-trained models. We created a distinct model for each tool and selected the one that performs better.

A. Pipeline

The pipeline that enables the creation of our NER model is described below. The pipeline begins by subjecting each dataset, i.e, *Quaero Old Press* (Quaero), *KB Europeana Newspapers* (Europeana) and *WikiNER*, to a *pre-processing step*. Then, the datasets are combined in a *combined dataset*, with the goal of providing the models with as much information for learning. The *combined dataset* is divided randomly, in terms of sentences, in a train, development and test set.

The train and development sets are used in the *model selection step*. Model selection is performed for each tool, where the train is used to train different models and the one that yields the best results on the development set is selected.

Each tool's selected model is evaluated using the test set, in the *model evaluation step*. The goal of model evaluation is to estimate the model's generalization performance i.e., how it performs on unseen data, using the test set. According to the results, we choose the best performing model out of the two, and create our NER model by training, again, on the complete dataset, in the *final model creation step*.

Afterwards, our NER model will be evaluated against the pre-trained models that we found and against the current model in ConnectionLens. The best performing model will be integrated in ConnectionLens.

B. Datasets Pre-processing

The goal of the pre-processing step is to uniformize all datasets to have the same format, encoding scheme and named-entity types.

Regarding the named-entity types, it is necessary to select a common set to all the datasets, which is: *Person* ("PER"), *Location* ("LOC") and *Organization* ("ORG"). Each have an associated prefix, resulting from the encoding scheme being used, e.g., "B-PER".

An IOB encoding scheme, more specifically, IOB-1, was chosen as it is widely used, and the improvement obtained from using more expressive encoding schemes. Furthermore, with this encoding scheme we will benefit from using the popular "conlleval" evaluation script made available by several CoNLL shared tasks (e.g., CoNLL-2002[10]).

A CoNLL style format was chosen, where here is a token per line and empty lines identifying sentence boundaries. Moreover, each line has the token itself separated by a whitespace from its NER label. Besides, obviously, benefiting

[10]https://www.clips.uantwerpen.be/conll2002/ner/

from the use of the "conlleval" evaluation script by using a CoNLL style format, this format is also accepted, or easily convertible to be acceptable, by both tools used to train the models, i.e., Flair and SpaCy.

We applied the necessary pre-processing procedure to each dataset. After this, instead of combining the three datasets as originally planned, we decided to only use the pre-processed Quaero dataset, due to the errors in the other datasets.

The Quaero dataset was divided in train, development and test sets. To achieve this, we took all sentences in the dataset and randomly divided them using the rule of thumb of 60% for the train set, 20% for the development set and 20% for the test set.

C. Evaluation Methodology

When evaluating the extraction of a NER model or system, we care about how it predicts named-entities, and not each token, since that is the goal of this task. Additionally, we consider an exact-match evaluation, this means, both the boundaries and the named-entity type predicted need to match the true annotation in the dataset to be considered *correct*. This is the adopted evaluation procedure in multiple CoNLL shared tasks. Moreover, we use the "conlleval" evaluation script to measure the performance of each model.

The metrics used to evaluate each model's performance are computed for each named-entity type and they are: *precision*, *recall* and $F1$-*score*. Moreover, we aim to maximize the $F1$-score. Since this is a multi-class problem, precision, recall and $F1$-score are computed for each named-entity type, as well as their micro-average to obtain the overall metrics.

D. Model Selection

1) Flair: The Flair framework allows training sequence labeling models using a bi-LSTM-CRF architecture and facilitates the integration with different word embeddings. Additionally, it allows combining or stacking different word embeddings by concatenating their vectors.

We decided to compare different word embeddings by training different models (a model for each word embedding) and selecting the best performing model out of all.

We looked into all word embeddings that can work for French and trained a model using the train and development set using each word embedding or a combination of them:

- FastText embeddings (classical static word-level embeddings): *standard*
- Stacked forward and backward Flair embeddings (contextual string embeddings): *stacked-flair*
- Stacked FastText and forward and backward Flair embeddings: *stacked-standard-flair*
- Stacked FastText and character embeddings: *stacked-standard-char*
- Byte Pair embeddings (word embeddings precomputed on the subword-level): *bytepair-fr* (French) and *bytepair-multi* (multilingual)
- CamemBERT embeddings (a Tasty French Language Model): *camembert*
- XLM-RoBERTa embeddings (multilingual language model): *xlm-roberta-base*

We consider the model trained with FastText embeddings, "standard" embeddings, as the baseline to which we will compare the results of the other models. We limited the number of maximum epochs to 30 during the training of the different word embedding models.

Regarding the evaluation results of each model on the development set. For all models, organizations always have lower scores when compared to persons and locations, which might be a result of the dataset having a lower representation of organizations. The models trained using CamemBERT and stacked FastText and Flair embeddings are the two best performing models. Although the CamemBERT model is slightly better overall than the other model, with an $F1 - score$ of 82.75% it is much slower making predictions and it required changing the source code of the Flair library in order to make it work. For this reason, we selected the stacked FastText and Flair embeddings model, that has an $F1$-score of 82.35%.

2) SpaCy: In what concerns SpaCy models, we experimented with pre-training SpaCy's "token to vector" layer by providing it with raw French data, instead of it being initialized with random weights. Additionally, we updated SpaCy's pre-trained French NER model with our train set. Besides that, a standard model was also trained, i.e., a default model with no add-ons, that we consider as the baseline to which we compare the results of the other SpaCy models. As with Flair, we limited the number of maximum epochs to 30.

To train the "token to vector" layer it is necessary to provide it with raw text for it to learn to predict a word's vector, taking into account its surrounding words. The raw text we used was obtained from a dataset of crawled French news articles, made available by Webhose.io[11].

We chose to experiment with SpaCy's capability of online learning on their models to update one of their medium French NER model, by training it with our train set, and seeing if there is a performance improvement.

Regarding the evaluation results of each model on the development set. All models have a higher recall than precision, except for the updated model where the precision is slightly higher for organizations. Moreover, organizations have low scores, which might be a result of its lower representation in the dataset. The updated model has the highest overall precision (79.61%) and $F1$-score (80.44%) between all models. The updated model has the highest overall scores for all metrics except for the recall, when compared to the model with pre-trained weights. For this reasons, we selected the updated model.

E. Model Evaluation

The selected best performing models of each tool were evaluated on the test set, to get an estimate of their performance on unseen data. Although the SpaCy model has a better score for the organizations recall, the Flair model is superior in all other

[11] https://webhose.io/

metrics, overall and for all named-entity types. For this reason, the Flair model, which has an overall $F1$-score of 82.44% is chosen to advance to the next step in the pipeline.

F. Final Model Creation

The final step of the model creation pipeline is to train the chosen model on the complete dataset. However, it is necessary to keep a portion of the dataset to be used as the development set during the final training of the model. For this reason, we randomly divided the dataset again, but in two sets: train and development set, keeping 80% of the sentences for the train set and 20% of the sentences for the development set. We also increased the number of maximum epochs to 150. The model created will be referred to as *flair-ssf-quaero*.

IV. Distantly Supervised French RE

In this section we present the approach that we took for developing a RE solution to integrate ConnectionLens.

We use OpenNRE to train a RE model. To train this model we need to use distant supervision to automatically build a dataset using a combination of relationship instances, from a knowledge base (KB), and sentences expressing the relationships, from a text corpus. We decided to use French DBpedia and Wikipedia articles, respectively.

1) DBpedia and Wikipedia: The fact that DBpedia and Wikipedia are available in French and the fact that mappings that relate Wikipedia to DBpedia exist, motivate our choice of using both to build a distantly supervised French RE dataset, together with the particularity that Wikipedia sentences typically state facts (unlike other text domains). Additionally, there is an increased probability that the DBpedia relationships will appear in the sentences in Wikipedia articles since DBpedia is built from Wikpedia.

2) Procedure: Following the distant supervision *expressed-at-least-once assumption*, given a triple, *(ent_1, ent_2, rel)*, in the KB, it is expected that at least one sentence across both entities, ent_1 and ent_2, Wikipedia articles expresses the relationship rel. The procedure we took to build the distantly supervised RE model, based on other works including [24], consists in:

1) Get all relationships between named-entities of the type *Person*, *Location* and *Organization* from DBPedia, and filter and group the relationships. In addition, generate negative relationship triples, which are triples where the two entities involved are not related. This is detailed in Section IV-A.
2) For each entity involved in a relationship, process the text in its Wikipedia article and keep as candidate sentences the ones that contain at least two named-entities. Additionally, obtain and create a set of surface forms (alternative names an entity can be mentioned as, in the text) for the named-entity. This is further described in Section IV-B.
3) For every relationship triple, access the candidate sentences of each named-entity involved, ent_1 and ent_2. Afterwards, using the named-entities' surface forms, select the sentences that match both named-entities, ent_1 and ent_2. This step is further described in Section IV-C.
4) Use the selected sentences to train a RE model using OpenNRE, as described in Section IV-D.

A. Relationships

We are only interested in extracting relationships between entities, namely of the *Person*, *Location* and *Organization* types, which are the named-entity types the NER solution in ConnectionLens is capable of extracting. Therefore, we query DBpedia to obtain all relationships that exist between named-entities of those types, and obtain around $1.25M$ relationship triple instances in total.

Since there is a significant number of different relationships types and, furthermore, a significant number of similar relationship types, i.e., which coarsely have the same meaning, and where sentences expressing them are expected to be similar, we decided to select a smaller set of relationships types. This set is the result of selecting the most frequent relationships types and grouping the ones with a similar meaning, that result in 29 relationship types.

Moreover, we reduced the number of relationship triple instances for time and efficiency reasons. In total, we end up with about $116K$ relationship triples.

We generated negative relationship instances by combining entities in a relationship triple *(ent_1, ent_2, NA)*, where NA denotes a negative relationship, that are not related in DBpedia, i.e., do not have a triple containing both entities in the KB.

Moreover, we randomly split the relationship instances in the dataset, in train, development and test sets, using the rule of thumb of 60% for the train set, 20% for the development set and 20% for the test set.

B. Obtaining Candidate Sentences

We define as *candidate sentences*, for a given entity, all sentences in the entity's Wikipedia article text that contain at least two named-entities.

For each entity in the processed dataset, we access the entity's French Wikipedia article. Each article's text is segmented in sentences and tokenized. Then, NER is performed over the sentences, using the best French NER model we selected in Section V-D, to annotate *Person*, *Organization* and *Location* named-entities in the text. We keep as candidate sentences the ones that possess two or more named-entities.

We collected all alternative names that an entity can be mentioned as in the text, i.e., surface forms, using a combination of its article title (in French), the entity's DBpedia *redirect* property, and more. All the strings in the resulting surface forms set are then pre-processed to be lowercase, without punctuation, accents and without stop words.

C. Selecting Sentences

For each relationship triple *(ent_1, ent_2, rel)*, in each set, i.e., train, development and test set, we access the candidate sentences of the named-entities involved, ent_1 and ent_2. For each candidate sentence, we take the detected entities and pre-process the corresponding string like we did for surface forms.

Moreover, for each detected entity, we apply an exact matching procedure between its pre-processed string and all the surfaces forms of each involved entity, ent_1 and ent_2. If there is at least one surface form of ent_1 or ent_2 that matches the detected entity, and both are of the same named-entity type, we consider it a match. Finally, if the candidate sentence has at least one detected entity that matches ent_1 and at least one detected entity that matches ent_2, we select the sentence as an example of the relationship rel between ent_1 and ent_2.

We collected $59,625$ sentences for the train set, $14,707$ sentences for the development set and $19,322$ sentences for the test set, for a total of $93,654$ sentences.

D. Training

We use the OpenNRE to train a bag-level Piecewise CNN with selective attention over instances (PCNN-ATT) model, the same architecture as the model proposed in [21].

V. Experimental Evaluation

This section presents the experimental evaluation that we performed for NER and RE models as well as the integration of the best performing model of each task in ConnectionLens.

A. Named-Entity Recognition

1) Evaluated Models: We evaluated and compared the performance of different NER models:

- *flair-ssf-quaero* - the model selected and trained in Section III.
- *flair-pre-trained* - the French pre-trained Flair model. It is trained with the *WikiNER* dataset, and uses French character embeddings and French fastText embeddings.
- *spacy-pre-trained-md* - the medium pre-trained French SpaCy model. It is trained with the *WikiNER* dataset using the SpaCy's architecure.
- *spacy-pre-trained-sm* - the small pre-trained French SpaCy model. Trained with the same dataset and architecture as the medium model. However, unlike the medium model, it does not include word vectors.
- *stanford-quaero* - the model previously integrated in ConnectionLens [2], trained using *Stanford NER*, on the *Quaero* dataset.

B. Evaluation Dataset

For evaluating the different models, we chose the FTBNER [25] dataset. It is composed of sentences extracted from the French newspaper *Le Monde*, of different domains that span between 1989 and 1993.

Since the models being compared are all trained using different datasets, by having an evaluation dataset not used by any of the models creates a better comparison in the sense that the dataset does not influence the results.

FTBNER was pre-processed in order to produce the named-entity types, encoding scheme and format as described in Section III. After the pre-processing procedure, the dataset contains $12,351$ sentences with $364,522$ tokens and $11,507$ named-entities. Moreover, there are $2,017$ persons, $3,754$ locations and $5,736$ organizations.

C. Evaluation Methodology and Metrics

In Section III-C, we introduced the evaluation methodology and metrics that are going to be applied.

D. Results

The evaluation results for each NER model are represented in Table III. All models show, for locations and persons, a higher recall than precision. This means the models detect more named-entities at the cost of wrongly predicting some. It is also true for locations except for the *flair-pre-trained* model.

flair-pre-trained	Precision	Recall	$F_{\beta=1}$
LOC	53.26%	77.71%	63.20%
ORG	**74.57%**	**75.61%**	**75.09%**
PER	**71.76%**	84.89%	**77.78%**
Overall	**65.55%**	**77.92%**	**71.20%**

flair-ssf-quaero	Precision	Recall	$F_{\beta=1}$
LOC	**68.34%**	73.87%	**71.00%**
ORG	37.75%	25.28%	30.28%
PER	65.91%	**92.67%**	77.03%
Overall	56.76%	52.95%	54.79%

spacy-pre-trained-md	Precision	Recall	$F_{\beta=1}$
LOC	55.77%	78.00%	65.04%
ORG	72.72%	54.85%	62.53%
PER	53.09%	74.98%	62.16%
Overall	61.06%	65.93%	63.40%

spacy-pre-trained-sm	Precision	Recall	$F_{\beta=1}$
LOC	54.92%	**79.41%**	64.93%
ORG	71.92%	53.23%	61.18%
PER	57.32%	79.19%	66.50%
Overall	61.25%	66.32%	63.68%

stanford-quaero	Precision	Recall	$F_{\beta=1}$
LOC	62.17%	69.05%	65.43%
ORG	15.82%	5.39%	8.04%
PER	55.31%	88.26%	68.00%
Overall	50.12%	40.69%	44.91%

TABLE III
NER evaluation results on FTBNER

The model previously used in ConnectionLens, the *stanford-quaero* model, is, overall, outperformed by all models, having an overall $F1$-score of about 45%.

flair-ssf-quaero manages to achieve the highest precision and $F1$-score for locations, respectively 68.34% and 71%, and recall for persons, 92.67%. Despite that, it is the overall second worst model, with an overall $F1$-score of about 55%.

The SpaCy pre-trained models are overall better than the *stanford-quaero* and even *flair-ssf-quaero*.

The *flair-pre-trained* is the overall best model, not just in $F1$-score, for which it has 71.20%, but all overall metrics. Its overall $F1$-score is almost 8% higher than its preceding best model. Moreover, it has the best scores for organizations, with a difference from the other models of more than 10% in $F1$-score and 20% in recall. This model is the best NER model we evaluated.

The pre-trained models are overall better than the models that we trained, i.e., *flair-ssf-quaero* and *stanford-quaero*. Focusing on *flair-ssf-quaero*, it uses the word embedding combination that showed better results in Section III. Considering the results, we decided to train one last model on the WikiNER dataset, in its pre-processed version, using the word embedding combination of *flair-ssf-quaero*, i.e., *stacked forward and backward French Flair embeddings* with *French fastText embeddings*. Moreover, we performed the same steps as in Section III-F. The results of evaluating this model, which we call *flair-ssf-wikiner*, on the FTBNER dataset are shown in Table IV.

flair-ssf-wikiner	Precision	Recall	$F_{\beta=1}$
LOC	59.52%	79.36%	68.02
ORG	**76.56%**	74.55%	**75.54**
PER	**72.29%**	84.94%	**78.10**
Overall	**69.20%**	**77.94%**	**73.31**

TABLE IV
RESULTS OF *flair-ssf-wikiner* ON FTBNER

flair-ssf-wikiner has an overall $F1$-score of 73.31%. We consider *flair-ssf-wikiner* to be even better than *flair-pre-trained*, showing better $F1$-scores, overall and named-entity specific. The most noticeable improvement is in terms of organizations, where the model shows better scores for all metrics, when compared to *flair-pre-trained*. Moreover, it surpasses *flair-pre-trained* for all scores except the recall of organizations. This is the best performing model evaluated and it was selected to be integrated into CONNECTIONLENS.

E. Relationship Extraction

1) Dataset: We train and evaluate our model using the dataset that we created as explained in Chapter IV.

The train set contains $59,625$ sentences, $38,064$ entities and $41,536$ relationship instances. The development set contains $14,707$ sentences, $11,451$ entities and $10,269$ relationship instances. And, the test set contains $19,322$ sentences, $14,177$ entities and $13,063$ relationship instances.

2) Evaluation Methodology and Metrics: Following the evaluation of previous related works, we evaluate our model using *held-out evaluation*. A part of the relationships instances in the KB, in our case DBPedia, is "held-out" to create a test set of relationship instances. When we divided the model in terms of relationship instances we were already taking into consideration the held-out evaluation. We evaluate the performance of our model using the following metrics, based on the work of [21] and subsequent works, and based on the capability of the model ranking the relationships it predicts (with a confidence score): *precision-recall (PR) curves*, *area under curve (AUC)*, *Micro-$F1$*, *Precision@N (P@N)*.

In the test set, there are $10,242$ relationship instances that possess only one sentence, and $2,821$ relationship instances with more than one sentence. Following [21], we evaluate our model on the relationship instances with more than one sentence, on three different test settings: (i) *one*, where for each relationship instance we randomly select one sentence to use for prediction, (ii) *two*, where for each relationship instance we randomly select two sentences to use for prediction, and, (iii) *all*, where all sentences of the relationship instances are used for prediction. We compute P@100, P@200, P@300, and their means, using the three test sets that result from the different settings described. We also compute, for each test set setting, the AUC and micro-$F1$.

3) Training Details: We use OpenNRE to train a bag-level PCNN-ATT model, the same architecture as the model proposed in [21]. For training, we use a maximum of 30 epochs, using the train set. We use OpenNRE's default hyperparameters for bag-level training.

We used the freely available fastText word embeddings in French, trained on Wikipedia[12].

4) Experimenting with Dataset Variants: We call the original dataset we have previously described by *version 1* (v1). For *version 2* (v2) we separated the *locatedIn* relationship in: *partOf* for relationships where both entities are locations and *locatedIn* when ent_1 is an organization and ent_2 is a location. For *version 3* (v3) we took the original dataset, and separated the *capital* relationship from where it was inserted into, i.e., *locatedIn*, to be a separate relationship. The goal of these dataset variants is to examine if the changes in the dataset significantly affect the performance of the model. We train three different models and the results obtained are presented and analyzed in Section V-E5.

5) Results: We observed the PR curves of the different models, and there was hardly any difference between their performance. This means that the changes made to the original dataset, overall, have no effect on the precision and recall.

Moreover, Table V shows a comparison of the AUC and micro-$F1$ values. The original version, *v1*, is slightly better that the other versions, *v2* and *v3*, in terms of AUC and micro-$F1$, with 97.10% and 91.78%, respectively. The good results produced by the models are influenced by the fact that the train and the test set are chunks of the same dataset.

	AUC	micro-$F1$
v1	**97.10%**	**91.78%**
v2	97.09%	91.61%
v3	97.06%	91.65%

TABLE V
AUC AND MICRO-$F1$ OF RE MODELS

In Table VI the P@100, P@200, P@300, AUC and micro-$F1$ with one, two and all sentences for each entity pair, for all model variants are presented. For all models, increasing the number of sentences improves the results, particularly in terms of AUC and micro-$F1$. From the *one* test setting to the *all* test setting there is an improvement of around 5% for micro-$F1$, for all models. Moreover, improvements are noticeable just by increasing the number of sentences from one to two.

In conclusion, we decided to select the *v1* model, i.e., the model trained on the original dataset, although, the differences in performance between all models are not significant. With *v1* we also do not have the *locatedIn* relationship divided in other

[12]https://fasttext.cc/docs/en/pretrained-vectors.html

relationships, which may confuse the model if the relationships prove to be too similar.

	One					
	P@100	P@200	P@300	P@Mean	AUC	micro-$F1$
v1	**99%**	**99.5%**	**99%**	**99.2%**	88.64%	81.18%
v2	**99%**	98%	97.7%	98.2%	88.25%	80.87%
v3	**99%**	**99.5%**	98.7%	99.1%	**88.96%**	**81.86%**
	Two					
	P@100	P@200	P@300	P@Mean	AUC	micro-$F1$
v1	99%	99.5%	99.3%	99.3%	92.05%	84.36%
v2	**100%**	**100%**	**99.7%**	**99.9%**	92.18%	**85.18%**
v3	99%	99.5%	99%	99.2%	**92.37%**	85.01%
	All					
	P@100	P@200	P@300	P@Mean	AUC	micro-$F1$
v1	99%	**99.5%**	99.3%	99.3%	**93.28%**	85.99%
v2	**100%**	**99.5%**	**99.7%**	**99.7%**	93.16%	**86.08%**
v3	99%	**99.5%**	99.3%	99.3%	93.03%	85.96%

TABLE VI
P@N, AUC AND MICRO-$F1$ FOR DIFFERENT # OF SENTENCES IN BAGS

F. Integration in ConnectionLens

Both the NER and the RE models we developed and selected require Python packages. However, CONNECTIONLENS is implemented in Java. To integrate the selected NER and RE models, we create a micro web-service, running with the CONNECTIONLENS code, using Flask[13]. The Flask service starts by loading the models and waits for HTTP requests, that are issued by CONNECTIONLENS. When CONNECTIONLENS needs to extract named-entities and/or relationships from text an HTTP POST request is sent to the service. In what concerns RE, the integration with CONNECTIONLENS is not accomplished.

VI. CONCLUSIONS

In this work we proposed and developed a solution for NER and RE for French news text, to be integrated in CONNECTIONLENS. This entails discovering the best approach for performing NER and RE, that can be applied effectively and efficiently to French texts. We created a French NER model that outperformed the available pre-trained models and the previously present in CONNECTIONLENS. This model is trained using the Flair library on the WikiNER dataset using stacked forward and backward French Flair embeddings with French fastText embeddings. Moreover, we created, to the best of our knowledge, the first French RE model and dataset using distant supervision and the OpenNRE library.

A. Future Work

In terms of future work, the integration of the RE solution in CONNECTIONLENS is going to be accomplished in the near future.

REFERENCES

[1] J. Gray, L. Chambers, and L. Bounegru, *The Data Journalism Handbook: How Journalists Can Use Data to Improve the News*. O'Reilly Media, 2012.

[2] C. Chanial, R. Dziri, H. Galhardas, J. Leblay, M.-H. L. Nguyen, and I. Manolescu, "Connectionlens: Finding connections across heterogeneous data sources," *Proc. VLDB Endow.*, vol. 11, pp. 2030–2033, Aug. 2018.

[13]https://flask.palletsprojects.com/

[3] S. Sarawagi, "Information extraction," *Foundations and Trends in Databases*, vol. 1, no. 3, pp. 261–377, 2008.

[4] O. Balalau, C. Conceição, H. Galhardas, I. Manolescu, T. Merabti, J. You, and Y. Youssef, "Graph integration of structured, semistructured and unstructured data for data journalism," *BDA 2020 (informal publication)*.

[5] L. Ramshaw and M. Marcus, "Text chunking using transformation-based learning," in *Third Workshop on Very Large Corpora*, 1995.

[6] D. Jurafsky and J. H. Martin, *Speech and Language Processing (3rd Edition, draft)*. 2018.

[7] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[8] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, pp. 257–286, Feb 1989.

[9] A. McCallum, D. Freitag, and F. C. Pereira, "Maximum entropy markov models for information extraction and segmentation.,"

[10] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," 2001.

[11] Y. Goldberg, *Neural Network Methods for Natural Language Processing*, vol. 37 of *Synthesis Lectures on Human Language Technologies*. San Rafael, CA: Morgan & Claypool, 2017.

[12] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.

[13] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," *arXiv preprint arXiv:1603.01360*, 2016.

[14] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[16] A. Akbik, D. Blythe, and R. Vollgraf, "Contextual string embeddings for sequence labeling," in *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 1638–1649, 2018.

[17] M. Mintz, S. Bills, R. Snow, and D. Jurafsky, "Distant supervision for relation extraction without labeled data," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pp. 1003–1011, Association for Computational Linguistics, 2009.

[18] S. Riedel, L. Yao, and A. McCallum, "Modeling relations and their mentions without labeled text," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 148–163, Springer, 2010.

[19] R. Hoffmann, C. Zhang, X. Ling, L. Zettlemoyer, and D. S. Weld, "Knowledge-based weak supervision for information extraction of overlapping relations," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, (Portland, Oregon, USA), pp. 541–550, Association for Computational Linguistics, June 2011.

[20] D. Zeng, K. Liu, Y. Chen, and J. Zhao, "Distant supervision for relation extraction via piecewise convolutional neural networks," in *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 1753–1762, 2015.

[21] Y. Lin, S. Shen, Z. Liu, H. Luan, and M. Sun, "Neural relation extraction with selective attention over instances," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Berlin, Germany), pp. 2124–2133, Association for Computational Linguistics, Aug. 2016.

[22] P. Xu and D. Barbosa, "Connecting language and knowledge with heterogeneous representations for neural relation extraction," *arXiv preprint arXiv:1903.10126*, 2019.

[23] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni, "Open information extraction from the web.,"

[24] A. P. Aprosio, C. Giuliano, and A. Lavelli, "Extending the coverage of dbpedia properties using distant supervision over wikipedia.,"

[25] B. Sagot, M. Richard, and R. Stern, "Annotation référentielle du corpus arboré de paris 7 en entités nommées (referential named entity annotation of the paris 7 French TreeBank) [in French]," in *Proceedings of the Joint Conference JEP-TALN-RECITAL 2012, volume 2: TALN*, (Grenoble, France), pp. 535–542, ATALA/AFCP, June 2012.