Graph Neural Network for Named Entity Recognition and Nested Named Entity Classification

ANAS OUARDINI, Univ Lyon, UCBL, CNRS, INSA Lyon, LIRIS, UMR 5205, France

Named entity recognition and classification is an essential task in information retrieval. It aims to locate and classify named entities within a document into predefined categories such as locations, persons, and more. Accurately identifying these phrases plays a significant role in simplifying information access and enhancing the understanding of text. However, this task remains challenging due to the various forms of named entities and their contextual dependencies. In this study, we experiment with Graph Neural Networks for two tasks: (1) node classification for named entity recognition and classification, and (2) graph classification for nested named entity classification. We trained and evaluated our models using a dataset of French articles from Diderot and d'Alembert's Encyclopedia (1751-1772). For the node classification task, we utilized a dataset of manually annotated articles. In this task, each node represents a token (i.e., a word) assigned a label indicating if the token is part of an entity and its class. The edges represent the connections between each pair of words in the entire corpus. This approach allows us to capture rich textual context and dependencies, which are crucial for accurate named entity recognition. Regarding the graph classification task, the dataset consists of a set of manually corrected nested named entities identified and pre_labeled with the Perdido Geoparser [12]. We use graph-based machine learning techniques to effectively capture the relationships between nodes and classify the graphs accordingly. Our experiments and evaluations demonstrate the effectiveness and robustness of our models in handling both node and graph classification tasks. The combination of Graph Neural Networks and deep learning approaches allows our models to efficiently process and interpret textual information, achieving high accuracy in named entity recognition and classification. By applying our models to French articles, we showcase their language adaptability, which makes them suitable for processing data in various languages. Moreover, the comparison with Perdido Geoparser provides valuable insights into the strengths of our model and its potential for enhancing named entity recognition tasks.

CCS Concepts: • **Graph Neural Network**; • **Natural Language Processing**; • **Named Entity Recognition**;

Additional Key Words and Phrases: Relation Extraction Name Entity Recognition(NER), Semi-structured Documents, Alembert's Encyclopédie, Graph Neural Networks

ACM Reference Format:
Anas Ouardini. 2023. Graph Neural Network for Named Entity Recognition and Nested Named Entity Classification. 1, 1 (August 2023), 21 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

In this big article, we explore the details of training Graph Neural Networks (GNNs) to handle the challenging task of Named Entity Recognition and Classification (NERC). NERC is a fundamental aspect of natural language processing (NLP), involving the identification and categorization of named entities within textual data. Named entities, which typically include proper nouns representing unique and identifiable entities like persons, organizations, locations, dates, and more, carry essential semantic information that plays a crucial role in various NLP applications, including information extraction, question answering, text summarization, and sentiment analysis.

Author's address: Anas Ouardini, anas.ouardini@etu.univ-lyon1.fr, Univ Lyon, UCBL, CNRS, INSA Lyon, LIRIS, UMR 5205, F-69621, France.

XXXX-XXXX/2023/8-ART $15.00
https://doi.org/XXXXXXX.XXXXXXX

By accurately recognizing and classifying named entities, NLP systems can gain a deeper understanding of the meaning and context embedded in the text, thereby enhancing their overall performance.

In this work, we focus on Nested Named Entities (NNEs), also known as Extended Named Entities [7]. NNEs are a fascinating concept wherein proper names coexist with common names (expansions), resulting in overlapping named entities. Put simply, NNEs are constructed by combining a proper name with a descriptive expansion, which may alter the implicit type (e.g., Place, Person, etc.) of the original proper name. For instance, consider the phrase *ville de Lyon* (city of Lyon) , which forms a NNE based on the named entity *City* is commune noun and the proper noun is *France*. The ability to identify and handle NNEs is crucial as it allows NLP systems to capture crucial contextual information that may be associated with named entities.

However, Graph-based approaches in NLP have gained significant attention due to their ability to capture complex relationships and structures within texts. One of the main challenges in NLP tasks like Named Entity Recognition (NER) and entity linking is handling Nested Named Entities (NNEs), where traditional models struggle to accurately identify and classify entities that overlap or form hierarchical relationships. By representing NNEs as graphs, with words as nodes and their co-occurrence relations as edges, we aim to address these challenges and create more sophisticated and accurate NERC models.

Graphs in NLP can be split into two types: syntactic graphs and semantic graphs. Syntactic graphs capture syntactic dependencies between words in a sentence, which can be useful for tasks like syntactic parsing and syntax-based NER. Semantic graphs, on the other hand, represent the meaning and semantic relationships between words and entities, enabling better handling of complex structures like NNEs. In other words: how words are put together and what words mean. How words are put together is called the 'way words work together,' and what words mean is called 'the meaning behind words.'. We're interested in using the meaning behind words to help us make better models for finding special words and putting them in groups.

To process these graphs efficiently and effectively, we employ Graph Neural Networks (GNNs). GNNs are a class of deep learning models designed specifically for processing graph-structured data. They can iteratively aggregate information from neighboring nodes, enabling them to learn complex patterns and dependencies within the graph, which makes them good at learning complex patterns and how things relate. This is helpful for language tasks where words and special words are connected.

The advantage of using GNNs for our graph-based approach is their ability to handle nested structures and capture long-range dependencies between words and entities. Traditional NER models often struggle to account for such complex relationships, leading to suboptimal performance in scenarios involving NNEs. GNNs, by exploiting the graph structure and context, can naturally account for overlaps and hierarchical relationships in NNEs, leading to more accurate and sophisticated NERC models.

In our experimental investigations, we thoroughly evaluate the performance of GNNs for node classification in named entity recognition and classification. This involves training the GNNs on the graph representations of the texts and fine-tuning the models to optimize their performance on NER tasks. Additionally, we also explore the use of graph classification techniques for nested named entity classification, where the goal is to classify entire graphs representing complex nested entities.

Our plan for using GNNs is to show they work well in different situations. We test the models on many different texts that talk about different special words and complicated language structures. This way, we know if our plan works in real-life situations and if it can manage the tough parts of special words.

Overall, our plan to use GNNs for finding and sorting special words is really helpful for dealing with tough parts like NNEs and putting things together. By understanding how words are connected and using this understanding, we hope to make better computer models for finding and sorting special words. This is useful in many places, especially when working with old language texts.

Moreover, our efforts are part of the larger framework of the GEODE project[1], where our ultimate aim is to enhance the recognition and classification of named entities in the context of French historical texts, specifically the renowned Encyclopedia of Diderot and d'Alembert, which was published between 1751 and 1772 [13]. To achieve this ambitious goal, we need to do a lot of tests and look carefully at our plan approach using a special set of writings from this articles from this historical French encyclopedia.

In conclusion, this article presents the way that harnesses the power of drawing neural networks to revolutionize the recognition and classification of named entities, with a focus on addressing the complexities of Nested Named Entities. Our efforts contribute to advancing the field of NLP and hold the promise of significantly improving the recognition and classification of named entities, particularly in historical French texts, thereby making invaluable strides in information extraction and text understanding across diverse domains.

The reste of this report is structured as follows. Section 2 discusses the related work about NERC and GNN for NER. Section 3 and Section 3 describe our methodology and experiments on node classification for NERC and graph classification for NNE classification. Finally, Section 5 provides conclusions and what we might do next.

2 RELATED WORKS

2.1 Named Entity Recognition and Classification

The growing internet has led to a lot of written information, necessitating the development of advanced techniques and tools to extract valuable knowledge. Among the many tasks to extract information, there is the identification and classification of named entities, it considering among the one of the most crucial method. Its importance derives from the fact that named entities often carry significant meaning and context in various applications. The concept of named entities was first proposed in 1996, and since its creation, a lot of techniques have been developed by numerous researchers through conferences and annotation campaigns, Pursuit to extract multiple types of entities from diverse languages and Text types.

NER (Named Entity Recognition) plays a pivotal role in various natural language applications, including Automatic Text Summarization [9], Machine Translation [8], Information Retrieval [2], Question Answering [11], and more. Depending on the domain of interest, NER is customized to identify specific entities. For instance, in the domain of Biomedicine, entities of interest may include genes and gene products, while in other research contexts, locations and individuals' names might take center stage as crucial entities.

NER systems typically involve four main steps [6] :

(1) Preprocessing step: We start by dividing the text into smaller pieces, we start by the text is analyzed to identify sentences and separate it into individual units (tokens), which helps us precisely identify the entities present in the text.
(2) *Feature processing step*: Given the complexity of natural language, each token is enriched with distinctive patterns or attributes, such as word features. Notably, libraries like Spacy serve as exemplary tools for this crucial processing step.
   Additionally, we utilize BERT(preprocess_item) for several tasks, including Normalization, Cleaning, Data Transformation, Handling Missing Values, and feature enhancement for both nodes and edges. The selected features to enhance the nodes and edges include: ['edge_index', 'y', 'num_nodes', 'feat_node', 'input_nodes', 'attn_bias', 'attn_edge_type', 'spatial_pos', 'in_degree', 'out_degree', 'input_edges', 'labels'].
(3) *Name recognition step*: At the heart of the NER process lies the recognition of entities within the text, followed by their assignment to specific classes or entity types.
(4) *Evaluation measures of NER*: The effectiveness of the NER system is meticulously assessed by comparing its predictions with annotations made by human annotators. Evaluation metrics, including precision, recall, and F-measure, offer valuable insights into the system's performance.

[1]https://geode-project.github.io

However, the domain of NER is not Without challenges [10]. Numerous factors can impact the performance of the NER task, such as the specific language in which the NER is to be executed, requiring adaptation of the system to take linguistic nuances into account. Moreover, considering the genres or textual domains becomes critical to effectively handle the data, whether it pertains to medicine, history, or other fields. Additionally, the number of entities to be extracted is a crucial consideration, and it is imperative to extract only those entities that align with the domain of interest (e.g., extracting drug entities from historical data may not be appropriate). Moreover, some languages or domains may face a lack of available resources, making the NER task more challenging. The presence of ambiguities, such as determining whether "APPLE" refers to an organization or a fruit, introduces further more complexities. Not to mention the impact of metaphors and popular expressions, which demand careful handling to ensure robust NER systems.

As research progresses, various types of entities have been successfully recognized in different languages and domains, utilizing diverse approaches. NER methods can be generally categorized into two main groups: (1) symbolic methods and (2) statistical methods. Symbolic methods depend on dictionary-based and rules-based algorithms, often utilizing lexicons and linguistic patterns created by experts. These methods are typically corpus-based and domain-specific. In contrast, statistical methods adopt data-driven approaches and machine learning techniques. The dataset is split into training and test sets, where the training data is used to train the classifier, and the test set is employed to evaluate the classifier's performance. Both sets are labeled for supervised learning.

The integration of geographical context is an exciting direction for NERC research. The Perdido library's focus on geoparsing and geocoding highlights the relevance of location-based information within named entities. By incorporating geospatial information, NERC systems can achieve more precise entity identification, especially in applications that require location awareness. The Perdido library's capabilities can be extended to enrich NERC systems, ensuring that entities' geographic context is accurately recognized and utilized.

In our article, we employed the Perdido library as a foundational framework, and we applied graph-based and machine learning approaches to construct a comparable framework. Perdido is a Python library designed for geoparsing and geocoding of French texts. It provides a toolset for processing geographic information from text, such as identifying named entities related to places and resolving toponyms (place names) to their geographic coordinates. The library is structured into three layers: back-office, API, and Python library. It enables users to manage, visualize, and export the results of geoparsing and geocoding. The tool is being developed to handle various types of texts and includes features for working with geographic annotations and spatial data.

2.1.1 Deep Learning and NLP in NER. [5] Recent years have seen remarkable growth in the field of Named Entity Recognition and Classification (NERC), driven by the surge in available text data and advancements in machine learning techniques. The rise of deep learning, particularly models like BERT, has dramatically elevated the accuracy of NERC systems. These models can understand the subtle relationships between words and grasp context in ways that traditional methods couldn't achieve. In particular, models pre-trained on large-scale language data, such as BERT, RoBERTa, have displayed extraordinary prowess in capturing the nuances of named entities. Their unsupervised learning methods allow them to develop a deep understanding of the complex structures and meanings present in textual data, leading to improvements in NERC across various languages and domains. Deep learning techniques have changed the way we do many things, and they're also making a big impact on improving how we recognize named entities. One important thing that happened is the creation of a special model called BERT (which stands for Bidirectional Encoder Representations from Transformers). BERT can understand language in both directions, which helps it learn the details of how words work together. This has led to big improvements in how well we can find named entities in text. Scientists use BERT's smart learning abilities to do even better by training it to work specifically with named entities. This helps BERT do a great job even when dealing with languages that don't have many resources.

BERT's success has made people really excited about using other smart models that were also trained on lots of text. Models like GPT-3 and RoBERTa are good examples. They are great at understanding how different things are connected in text, like names and the words around them. These models make finding named entities feel easier and more advanced than before.

Because of deep learning and the help of computers, we're also getting better at understanding when different words mean different things and when they're related to each other. We can figure out more about entities, like what they're connected to and how they're related to other things. We use special techniques like attention and special ways of organizing information to do this better.

But there's one thing we have to remember: for deep learning to work well in finding named entities, we need lots of examples to teach the computer. Sometimes, we don't have enough examples for certain areas or languages, and that can make things harder.

Transformers [1], which are like the heart behind models such as BERT, have really changed how we do NER. These special architectures, with their self-attention tricks, help models understand the tricky connections between words and the meaning around them. Because of this, NER systems can not only find important things in text but also understand the little details around them, making their results better and smarter. As NER keeps getting better, transformers will stay super important for making sure the models are accurate and strong.

But, there are some problems that come with these cool ideas. We need a lot of computer power and big sets of marked data to teach these models. Still, transformers have done something amazing by helping NER find hidden stuff in messy text in ways we couldn't do before. The mix of transformers, deep learning, and understanding language has made NER much better, and there's a lot of exciting stuff to look forward to in the future!

In the end, the way we find named entities has become much better thanks to deep learning. Models like BERT have changed how we do things, and we're learning to use them even better. As deep learning keeps improving, we'll likely see even more advanced ways to find named entities, which will be helpful for different areas and languages.

In conclusion, the extraction and classification of named entities are crucial elements of information extraction from textual data. The continuous advancements in NER techniques and tools show potential for further enhancing natural language applications. Successfully navigating the challenges posed by linguistic diversity, domain-specificity, and resource limitations will be crucial in establishing robust NER systems that empower various NLP applications to access the extensive knowledge within textual information.

The symbiotic relationship between Named Entity Recognition and deep learning has led to remarkable progress, albeit not without challenges. An ongoing challenge is the need for large annotated datasets to train deep learning models effectively. Annotating data can be labor-intensive and resource-demanding, particularly when specialized entities are involved. Furthermore, the interpretability of deep learning models poses questions about how decisions are made within the network. Addressing these challenges is crucial for building trust in NERC systems and ensuring they align with ethical considerations.

Moreover, interdisciplinary collaboration between NERC and other fields such as Text Summarization, geospatial analysis can unlock novel insights. By incorporating NERC into broader research frameworks, we can better understand the interplay between language, culture, and information extraction. This collaborative approach not only enhances NERC's accuracy but also enriches our understanding of how named entities shape narratives and history

When using an ML-based solution for Named Entity Recognition (NER), it generally involves two primary phases. In the first phase, we train the ML model using annotated documents, which serve as examples for the model to learn from. Once the model is trained in this way, it can move to the next phase. In the second phase, the trained model is applied to annotate the raw, untreated documents, identifying and labeling named entities within the text.[3] (see Fig 1).

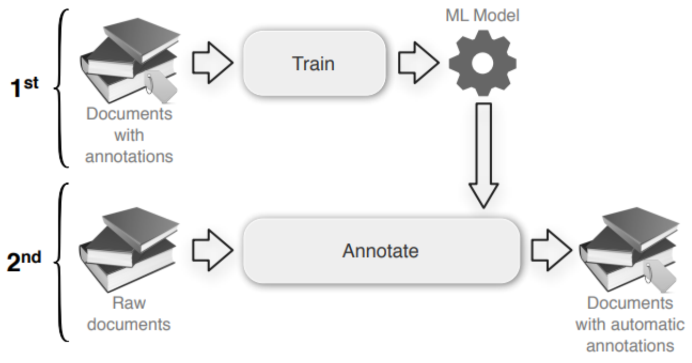

1st
Documents with annotations
Train
ML Model
2nd
Raw documents
Annotate
Documents with automatic annotations

Fig. 1. ML in NER

Recent developments in Named Entity Recognition (NER) also involve deep learning and language models. For example, researchers use advanced structures like the transformer architecture and models similar to BERT [14] to carry out NER tasks. These techniques show interesting results, especially for languages with limited resources, making use of innovative approaches such as minimal or zero prior training.

However, each method has its strengths and limitations. Rule-based systems rely on experts who possess extensive knowledge of the data, but defining rules can be challenging and time-consuming. On the other hand, machine learning techniques often require a large volume of annotated data to effectively train models.

2.2 Graph Neural Networks for NER

The advent of Graph Neural Networks (GNNs) has brought about a powerful and adaptable tool, especially for the field of semi-supervised learning. Leveraging their ability to effectively capture textual context within and across words, GNNs are good at understanding words and their connections, making them indispensable for a wide range of applications. This is very useful for many different things. The potential applications of GNNs extend far beyond the scope of this article, they are powerful and have a big impact. In this work, we have focused on employing the Graph Convolutional Network (GCN) model for node and graph label prediction, This shows just a small part of what GNNs can do. In a bigger sense, GCN and other GNN variants find relevance whenever there is a need to mix the power of graph representations with the expressiveness of deep learning. Notably, GCN has demonstrated its utility in dynamic network, node, and edge predictions when combined with other powerful architectures such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks [4].

2.2.1 Node Classification. Node classification stands as a fundamental task in GNNs, involving the determination of labels for individual samples represented as nodes within a graph. This process heavily relies on considering both the labels of neighboring nodes and the features associated with each node. Typically addressed through semi-supervised learning, node classification tasks involve working with only a partially labeled graph during the training phase. In other words, a portion of the graph is masked during training, and the GNN model attempts to predict the labels for these masked nodes during the testing phase.

2.2.2 Graph Classification. Graph classification ventures into the domain of categorizing entire graphs into different predefined categories. This task draws parallels to image classification, with the distinction that the targets are now graphs rather than individual images. The applications of graph classification are extensive and diverse. In our specific context, we're looking at figuring out if a graph, which represents a nested named entity, pertains to a place, a person, or another category of interest.

The potential of GNNs to tackle multifaceted challenges across various domains is vast. From text processing tasks to dynamic predictions and complex graph classifications, GNNs continue to drive progress in natural language processing, network analysis, social network understanding, drug discovery, and numerous other fields. By capturing rich structural information and contextual dependencies, GNNs hold promise for addressing complex real-world problems in novel and efficient ways.

In the domain of Named Entity Recognition and Classification (NERC) and Graph Classification for nested named entity recognition, various methods have been explored to tackle the intricate challenges of information extraction from text. Traditional approaches include dictionary-based, rule-based systems, and machine learning (ML) methods [3]. While dictionary-based and rule-based systems like Perdido [12] provide valuable solutions, recent advancements in ML have been made to the development of more sophisticated models. Notably, transformer-based models such as BERT [5] have demonstrated remarkable performance in NERC, it's really good at understanding the meaning of words in sentences. Concurrently, Graph Neural Networks (GNNs) have emerged as a powerful tool for understanding complex relationships within text data. These programs are great at understanding the connections between words in text [4]. These models, coupled with graph-based machine learning libraries like DGL and PyTorch-Geometric [7], have enabled researchers to enhance the accuracy and efficiency of named entity recognition and classification tasks. Also, some researchers have made these models work with different languages, which is really important because languages can be different, Even with all these good things, there are still some problems to solve. One big problem is figuring out how to understand named entities that are inside other named entities, like a name of a place inside a name of a person [9], researchers are working on finding new ways to understand these complex names by using smaller groups of words that are connected. These smaller groups are called sub-graphs [10].

In conclusion, the emergence of Graph Neural Networks has opened new frontiers in machine learning and artificial intelligence. Their unique ability to process graph-structured data, coupled with the power of deep learning, has paved the way for advancements in various areas of research and applications. As we further explore the capabilities of GNNs, we are likely to witness their widespread adoption and continual refinement in diverse domains. The future of GNNs holds exciting prospects, and their potential to revolutionize how we understand and process data. This work aims to contribute to this evolving landscape by presenting a novel model that leverages GNNs and deep learning for robust named entity recognition and classification, while also advancing the understanding of nested named entity recognition in complex linguistic contexts.

3 METHODOLOGY

This research article is focused on two crucial tasks in Natural Language Processing (NLP): Node classification for named entity recognition and classification, and Graph classification for nested named entity classification. Named entity recognition (NER) is a fundamental aspect of information extraction from text, involving the identification and categorization of entities like persons, locations, organizations, dates, and more. The concept of named entities was introduced in 1996, and since then, researchers have developed various techniques, which have been presented in conferences and annotation campaigns, to extract different types of entities from diverse languages and text genres.

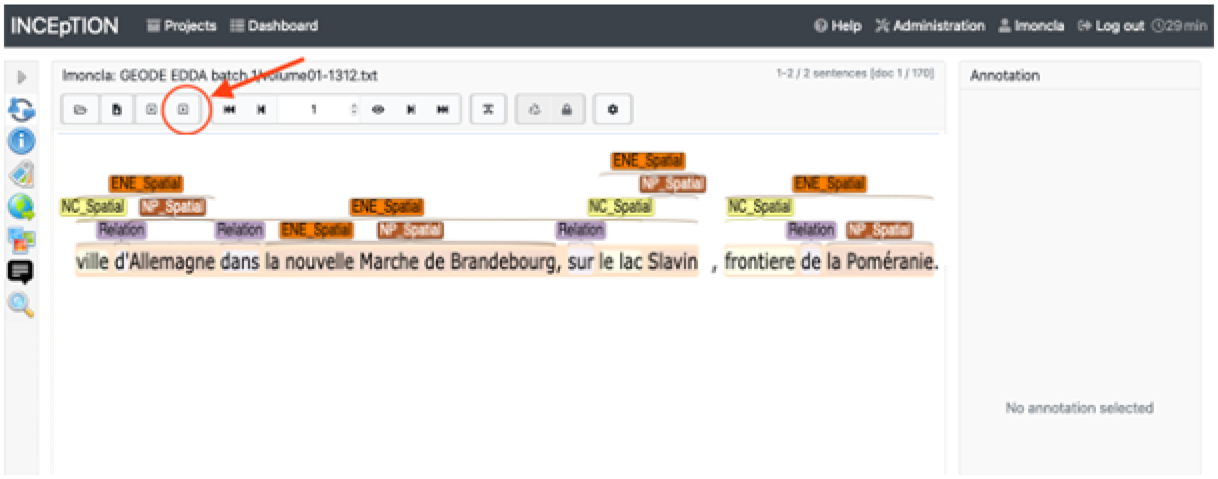
For this study, we chose the Encyclopedia Diderot and Alembert as our dataset. The semi-structured format of the data was well-suited for our research. Additionally, we utilized the English DataSet version to test our model's performance in the authors' language.

To annotate the dataset, we employed two methods: AUTOMATIC ANNOTATION using Perdido and MANUAL ANNOTATION conducted by domain experts. Both approaches resulted in XML representations of the annotations, as illustrated in Figure 4. The manually annotated data is considered more robust, but it constitutes a smaller portion compared to the automatic data, which is substantial but may contain noise and potential errors. In both approaches, we will convert the XML data to CSV format.

Perdido is a library that we can use for recognizing named entities; it is also generates XML from the text, similar to what is shown in Figure 2.
Perdido's Example (automatic annotation)

```python
from perdido.geoparser import Geoparser
geoparser = Geoparser()
doc = geoparser('ville de France en Bourgogne, Bailliage de
    Chatillon')
for entity in doc.named_entities:
    print(f'entity: {entity.text}\ttag: {entity.tag}')
```

Fig. 2. Python Code Example: Geoparsing with Perdido Library

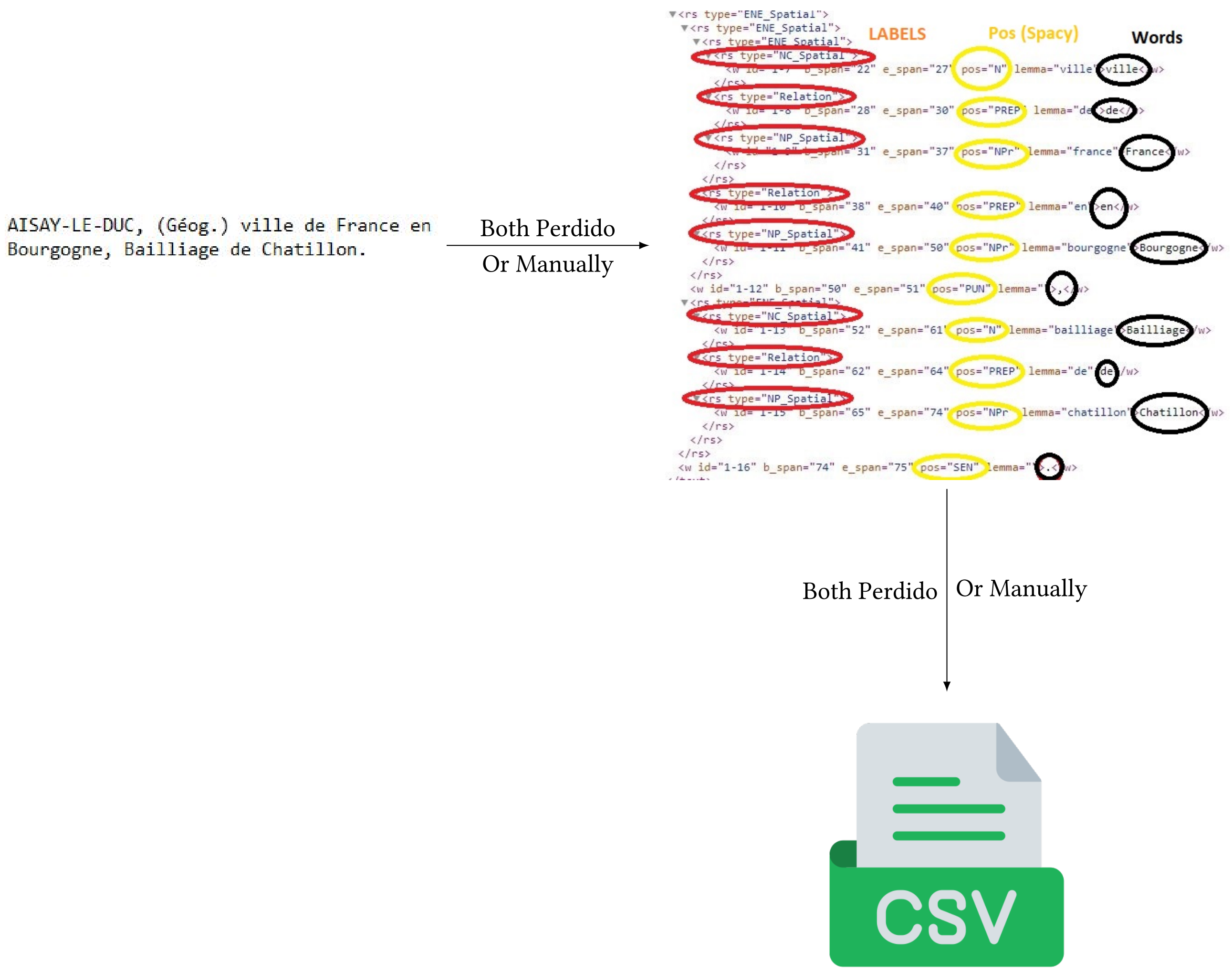


ville d'Allemagne dans la nouvelle Marche de Brandebourg, sur le lac Slavin , frontiere de la Poméranie.

Fig. 3. Manual's Example (manual annotation)

3.1 Node classification for named entity recognition and classification

In the Node classification task, data processing involved extracting words, labels, and Part-of-Speech (POS) tags from the XMLs, which were then stored in CSV format, as illustrated in Figure 5. We removed the stop words during data preprocessing. Unlike the next methods, where separate graphs were constructed, we took a different approach by building "ONE BIG graph" (look at Fig 6) contain all nodes" using the DGL library (look at Fig 6). In this graph, nodes represented words, and edges were determined by the word sequence within the sentences. To enrich the node features, we utilized FastText, a skip-gram model, to convert words into 300-dimensional vectors and incorporated their corresponding POS tags. The node labels, indicating entity types like "place,.." were provided by Perdido (or experts). We forego the traditional cross-validation approach and mask approach with graphs, instead opting to utilize fixed CSV files for testing. We prepare a separate set of CSV files for testing,



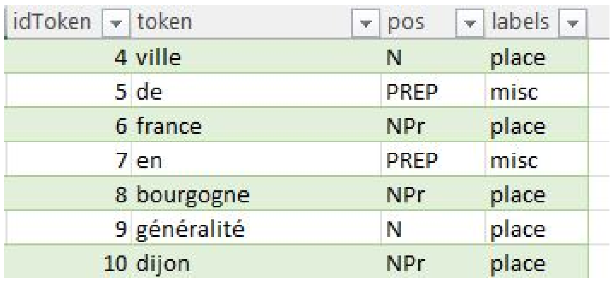
```
AISAY-LE-DUC, (Géog.) ville de France en
Bourgogne, Bailliage de Chatillon.
```

```
▼<rs type="ENE_Spatial">
 ▼<rs type="ENE_Spatial">
  ▼<rs type="ENE_Spatial">
   ▼<rs type="NC_Spatial">
      <w id="1-7" b_span="22" e_span="27" pos="N" lemma="ville">ville</w>
    </rs>
   ▼<rs type="Relation">
      <w id="1-8" b_span="28" e_span="30" pos="PREP" lemma="de">de</w>
    </rs>
   ▼<rs type="NP_Spatial">
      <w id="1-9" b_span="31" e_span="37" pos="NPr" lemma="france">France</w>
    </rs>
  </rs>
  <rs type="Relation">
    <w id="1-10" b_span="38" e_span="40" pos="PREP" lemma="en">en</w>
  </rs>
 ▼<rs type="NP_Spatial">
    <w id="1-11" b_span="41" e_span="50" pos="NPr" lemma="bourgogne">Bourgogne</w>
  </rs>
 </rs>
 <w id="1-12" b_span="50" e_span="51" pos="PUN" lemma=",">,</w>
▼<rs type="ENE_Spatial">
 ▼<rs type="NC_Spatial">
    <w id="1-13" b_span="52" e_span="61" pos="N" lemma="bailliage">Bailliage</w>
  </rs>
 ▼<rs type="Relation">
    <w id="1-14" b_span="62" e_span="64" pos="PREP" lemma="de">de</w>
  </rs>
 ▼<rs type="NP_Spatial">
    <w id="1-15" b_span="65" e_span="74" pos="NPr" lemma="chatillon">Chatillon</w>
  </rs>
 </rs>
</rs>
<w id="1-16" b_span="74" e_span="75" pos="SEN" lemma=".">.</w>
```

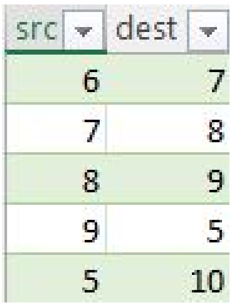
Fig. 4. TxT To XML BY Perdido Or Manually

containing 20% of the graphs from of manual annotations, ensuring a consistent and reliable evaluation of our model's performance. For node classification. In our task of sorting things into different groups, we used two very useful tools: DGL and PyTorch-Geometric. These tools helped us work with complicated networks and the data about them. We made a big graph that included all nodes we were working with using a function called "dgl.graph()". We also used another tool, "torch_geometric.data", to manage the information about the networks in a smart way.

The collaboration between DGL, specifically GCN and GIN, and PyTorch-Geometric (GCN and GIN) simplified our ability to compare various node sorting methods within the graph. We also utilized the Graph Isomorphism Network (GIN) and Graph Convolutional Networks (GCN) to enhance our analyses. We used both tools to see which one works better for our task. We evaluated the effectiveness of these methods using essential metrics such



idToken	token	pos	labels
4	ville	N	place
5	de	PREP	misc
6	france	NPr	place
7	en	PREP	misc
8	bourgogne	NPr	place
9	généralité	N	place
10	dijon	NPr	place



src	dest
6	7
7	8
8	9
9	5
5	10

Fig. 5. CSV of nodes/edges for Node classification (manual/automatic annotation)

as the accuracy, the average F1-score (both macro and micro), average precision (both macro and micro), and average recall (both macro and micro). These metrics provided a comprehensive view of the model's performance, enabling us to identify the approach that worked best for our specific task.

Besides the network stuff, we also looked at how our model understood language. We did this by comparing our model with another really smart language model called CamemBERT, which is great at understanding French. This helped us make sure that our method is really good at understanding French language data and showed that our model can keep up with the best models out there.

Algorithm 1 Convert XML to CSV (Node Classifications)

```
Require: XML data
  Preprocess the data: Remove stop words.
  Extract lemma of the text and assign an ID to each token
  Extract the label and part-of-speech (POS)
  Create edges between nodes
  Balanced data (if necessary)
  for all XML files do
    Iterate through all XML files, extracting entity elements
    Write data to CSV files
  end for
```

To save graph in CSV format, two CSVs files are required (Edges, Nodes):

Edges CSV:

- Edges: According to the sequence of the words in the sentence.
- Contains columns for Source, Destination

Nodes CSV:

- Contains columns for Token IDs with their corresponding Tokens, pos and labels.
- Nodes: words (look at Fig 5 and 6)
- Features of nodes(words): We use FastText (it's the skip-gram model) to embedding words to a vector of 300 dimensions, we add the POS with the features.

3.2 Graph classification for nested named entity classification

In the Graph classification task, the focus was on nested named entities (NNEs) rather than individual words. NNEs consisted of multiple words forming a place or person entity, such as "New York City." Each NNE corresponded to

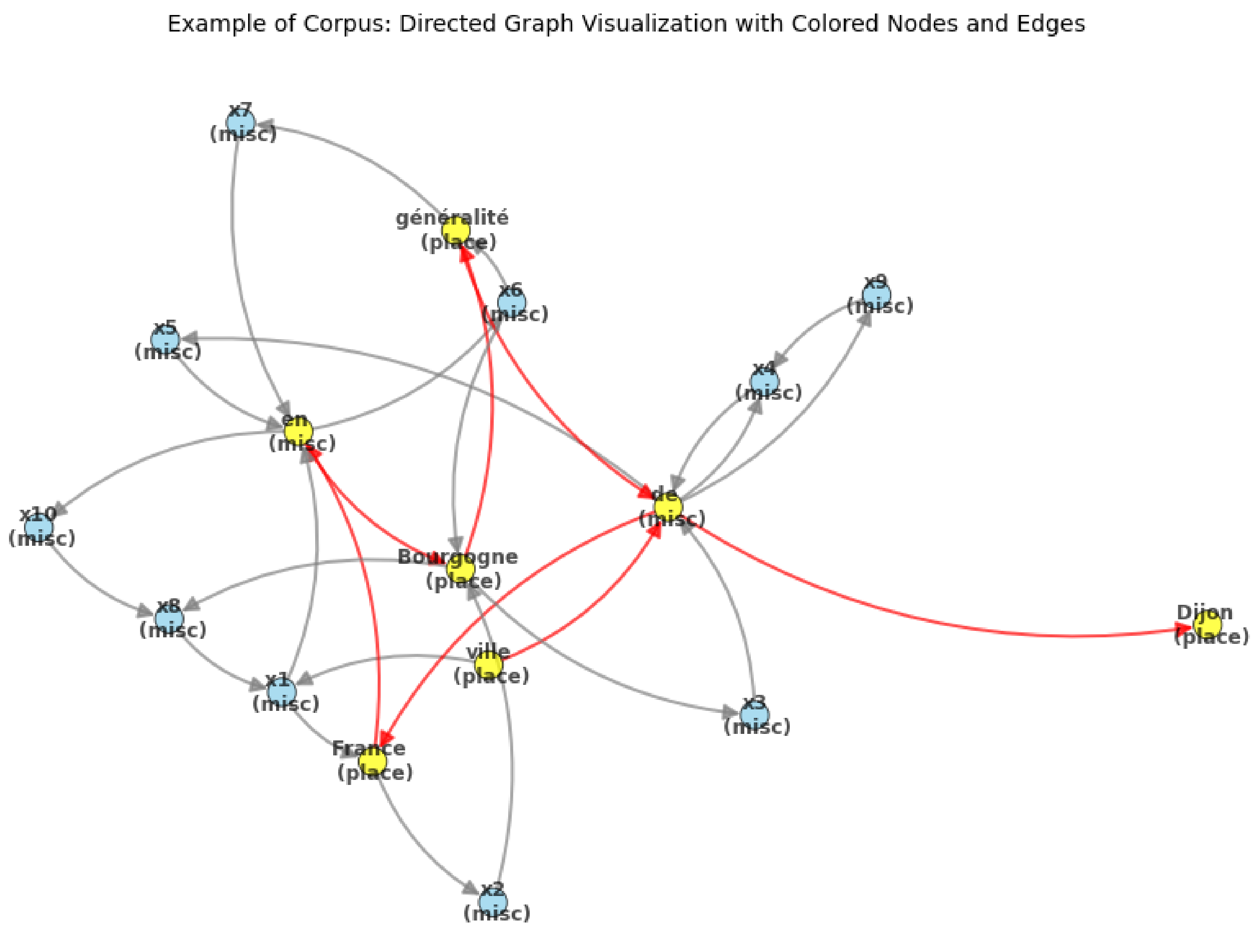


Fig. 6. Subgraph Showcase: Trio of Directed Graphs

a sub-graph, where nodes represented words, and edges were determined by the word sequence. Similar to Node classification, FastText was utilized to convert words into 300-dimensional vectors for enriching node features. Sub-graphs were labeled with entity types like "place," "Person," or "misc." Example: New York City, this NNE here have 3 words (3 nodes) and the label for this example is Place so the sub-graph is a small graph with 3 nodes connected and the label is Place. We'll stock the result in 3 CSV (Nodes, Labels, Edges, the way to store them in csv is totally different than the other method). Then we will read the csv to build a multiple Sub-Graph Fig 7): We trained and tested the model using both automatic and manual sub-graphs, utilizing PyTorch (GCN, GIN) and PyTorch-Geometric (GCN, GIN).

As mentioned earlier, there are two types of annotation generation methods. The first method involves manual annotation, as illustrated in Figure 3. The second method is automatic annotation carried out by Perdido, as depicted in Figure 2. Additionally, in the context of Graph Classifications, a third method involves manual intervention, where we utilize annotations generated by Perdido. We convert the XML data to CSV format and subsequently refine the annotations manually. This approach entails working with both the original manual data and the Perdido-generated data that has been manually corrected.

To summarize:

- Manual Annotation (Figure 3)
- Automatic Annotation by Perdido (Figure 2)
- Manual Intervention with Perdido Annotations - XML to CSV conversion and manual correction.

To save graphs in CSV format, three CSV files are required (Manual Annotation, Automatic Annotation, Manual Intervention):

Edges CSV:

- Edges: According to the sequence of the words in the sentence
- Contains columns for Source, Destination, and Graph IDs.

Nodes CSV:

- Contains columns for Token IDs and their corresponding Tokens.
- Nodes: words (look at Fig 7)
- Features of nodes: features of words: we use FastText to convert words to vectors (300 columns)

Labels(Graph Information) CSV

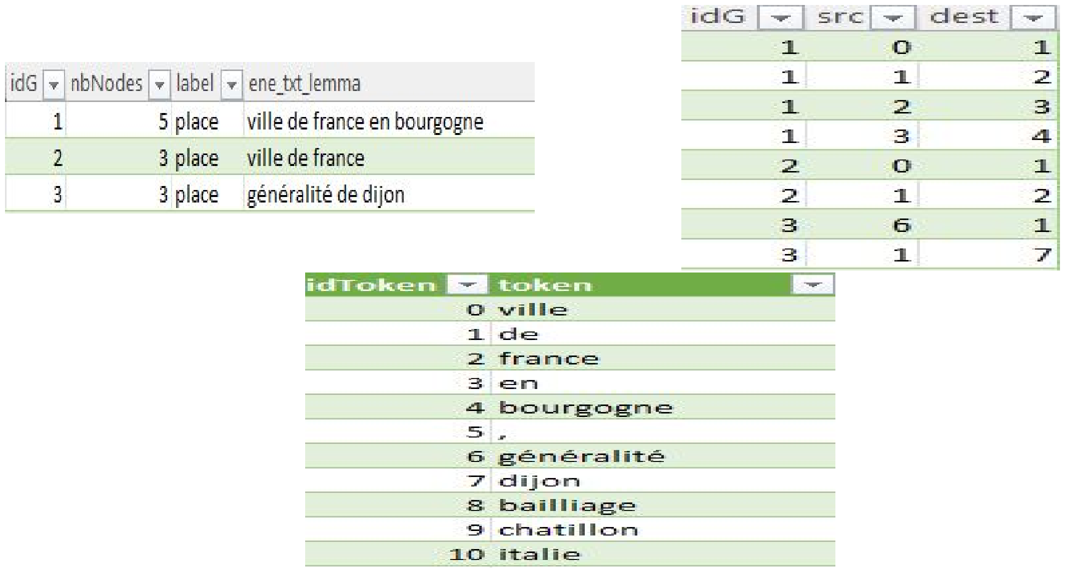
- Labels of Sub-Graph is the misc or place or person:(look at Fig 7)
- Contains columns for Graph IDs, Number of Nodes, Lemmatized Text, and Labels.

Algorithm 2 Convert XML to CSV (Graph Classifications)

```
Require: XML data
  Extract lemma of the text and assign an ID to each node
  Count the number of nodes
  Extract the label
  Create edges between nodes
  Assign an ID to each graph (IdG)
  Balanced data (if necessary)
  for all XML files do
    Iterate through all XML files, extracting entity elements
    Write data to CSV files
  end for
```

In our approach, we deviate from the traditional cross-validation approach and instead utilize fixed CSV files for testing. We prepare a separate set of CSV files for testing, containing 20% of the Sub-graphs, ensuring a consistent and reliable evaluation of our model's performance. This approach leverages the error-free and more reliable nature of the manual data, as we base the testing set on manual annotations. The training set, however, remains distinct and variable. We have the flexibility to incorporate either manual or automatic data into the training set, as long as the sub-graphs (look at Fig 8) in the training set do not overlap with the set of test sub-graphs. This fixed split methodology guarantees a standardized assessment of our model's effectiveness.

We used DGL and PyTorch-Geometric to handle complex graph-based datasets. We created large sub-graphs with "dgl.graph()" (look at Fig 8) and we do the same thing with "torch_geometric.data." This setup allowed us to compare different techniques for sorting sub-graphs. We also utilized DGL with Graph Isomorphism Network



idG	nbNodes	label	ene_txt_lemma
1	5	place	ville de france en bourgogne
2	3	place	ville de france
3	3	place	généralité de dijon

idG	src	dest
1	0	1
1	1	2
1	2	3
1	3	4
2	0	1
2	1	2
3	6	1
3	1	7

idToken	token
0	ville
1	de
2	france
3	en
4	bourgogne
5	,
6	généralité
7	dijon
8	bailliage
9	chatillon
10	italie

Fig. 7. CSV of nodes/edges/labels for graphs classification (manual/automatic annotation)

(GIN) and DGL with Graph Convolutional Networks (GCN) to enhance our analyses. Both tools were used to determine which one was more effective for our task.

Additionally, we evaluated our language model by comparing it to CamemBERT and Generative pre-trained transformers (GPT), a pre-trained French language model. This highlighted the strong language understanding capabilities of our approach, demonstrating its competitiveness alongside top models.

We tested how good our different methods were using important measurements like accuracy, average F1-score, average precision, and average recall (both macro and micro). These measurements helped us see how well the model worked and figure out which method was the best for our task.

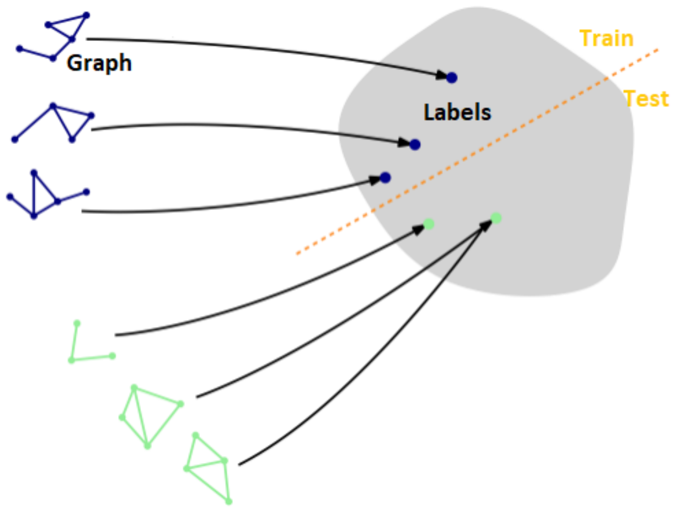
Now, as discussed earlier in the related work, there are three types of Named Entity Recognition and Classification (NERC) methods: Dictionary-based, Rule-based systems, and Machine Learning (ML) methods. Our model belongs to the ML method category, while Perdido is categorized as a dictionary-based and rule-based system.

To compare the two methods, we perform the following steps:

- We obtain labels generated manually.
- We obtain labels generated by Perdido for the same set of data, which represents a dictionary-based and rule-based approach to NERC.
- We also acquire labels generated by our ML model for the same data, representing our approach.

By comparing the three sets of labels, we gain insights into the strengths and weaknesses of each method. Understanding why the ML model outperforms Perdido in certain cases can potentially lead to improvements in the Perdido method. Conversely, we can also identify areas where our ML model may be less effective and enhance it by incorporating insights from Perdido.

Each method has its own advantages, and by leveraging the strengths of one method to improve the other, in the future, we have a great opportunity to make a much better NERC (Named Entity Recognition and Classification) model. My supervisor, who made Perdido, can help us combine these two models seamlessly. This would create a



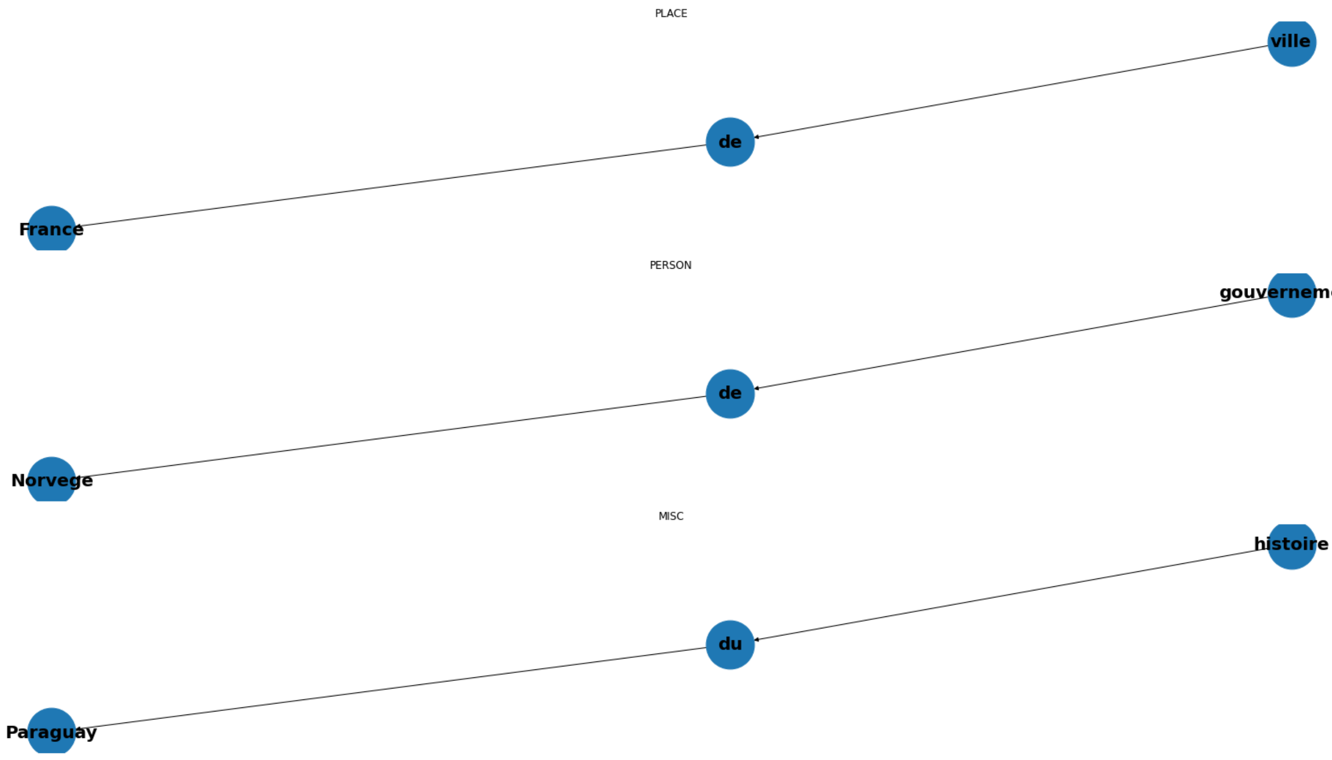


Fig. 8. Example of subgraphs

single system that can accurately and effectively identify and categorize named things, and it would work really well.

Our ultimate objective is to contribute to the advancement of NERC techniques, leading to more accurate and efficient language understanding systems. By thoroughly evaluating and comparing different approaches, we aim to develop a powerful and versatile model that can significantly enhance information extraction from textual data across various domains and languages.

4 EXPERIMENTS AND RESULTS

4.1 Node Classification for Named Entity Recognition and Classification

In this section, we present the experimental setup and results for the node classification task, which focuses on named entity recognition and classification at the token level using Graph Neural Networks (GNNs). Our objective is to evaluate the performance of GNN-based models in accurately identifying and classifying named entities within French articles from Diderot and d'Alembert's Encyclopédie (1751-1772).

4.1.1 Dataset Preparation. For the node classification task, we employed a dataset of manually annotated articles. Each article was tokenized, and each token was assigned a label indicating if it belongs to a named entity and its respective class. The token-level labels were represented as nodes in a graph, where edges were established between tokens based on their co-occurrence within the text. This graph structure enabled us to capture the rich textual context and dependencies crucial for accurate named entity recognition. We implemented two distinct forms of data for the task of node classification: one utilizing the DGL library and the other utilizing data from the `torch_geometric.data` library. The structure of the data in both forms is defined as follows:

For `torch_geometric.data`:

```
Data(x=nodes, edge_index=edge_index,...)
```

For DGL:

```
dgl.graph(edges=edges, num_nodes=n, ...)
```

Both forms of data were developed with the intention of processing the graph representations of the articles and perform node classification tasks.

4.1.2 Model Architecture. We implemented two distinct GNN models for the node classification task: one using the DGL library and the other using the PyTorch-Geometric library. Both models were designed to process the graph representations of the articles and perform node classification.

4.1.3 Experimental Procedure. The models were trained using the annotated articles, and the training process included aggregating information from neighboring nodes in an iterative manner. We utilized appropriate loss functions and optimization techniques to fine-tune the models for named entity recognition and classification. During training, we employed techniques to address class imbalances and minimize overfitting.

We conduct experiments to evaluate our Model on two distinct annotation datasets: one containing entities recognized by Perdido than corrrect manually and the other with entities identified manually. The objective is to develop a robust model capable of recognizing and classifying named entities accurately and efficiently. We compare our results against Perdido, as well as other methods for identifying named entities.

The task involves creating a model that can identify instances of various named entity classes (e.g., Place, Person, etc.) for each word in the test set. We evaluate the model using the two manuals data. By comparing the results with Perdido's output, CammemBERT and Transformer, we can assess the model's performance and effectiveness in handling different types of data.

To perform the experiments, we utilize the DGL (Deep Graph Library) and PyTorch-Geometric libraries, which enable us to leverage graph-based machine learning techniques. We implement two models: one using DGL and the other using PyTorch-Geometric. Both models are evaluated to compare their performance on the task at hand.

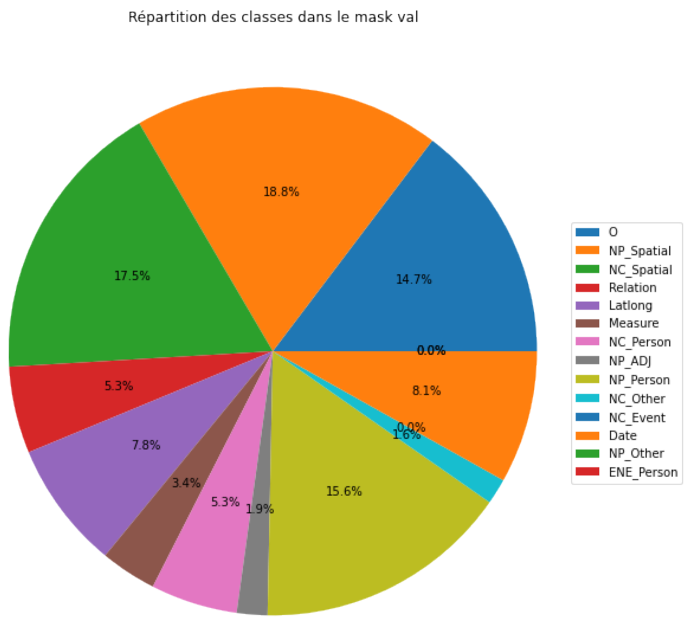


Fig. 9. Distribution with ALL LABELS

Additionally, we ensure that the data is balanced and exclude some occurrences with limited presence in the dataset, as they might not significantly contribute to the overall results.

in Fig 10 As you can observe, the matrix on the left represents our model, while the matrix on the right corresponds to Bert. A distinct convergence between them is evident, and the curve at the bottom illustrates the progression of our models, whether they are based on Bert or our own approach.

As depicted in Fig 9, the data distribution is well balanced, and the predictions made by our models show a strong correlation with the true targets (manual data). However, it's important to note that not all of these labels are employed in our models. Specifically, we focus on the categories of Place, Person, Misc and Other. For the remaining labels, including NP_person, NC_person, and ENE_person, they are categorized as Person. Additionally, labels NP_space and NC_space are categorized as Place, while the label O is categorized as Others. Any remaining labels are collectively categorized as Miscellaneous (Misc). This approach allows us to effectively classify entities into the Person, Place, Other and Miscellaneous categories, as per the defined criteria.

The trained models were evaluated using standard metrics for named entity recognition, including precision, recall, and F1-score. Additionally, we compared the models' performance against other named entity recognition methods, including Perdido, as well as other traditional machine learning approaches. We also visualized the predictions made by our models to assess their alignment with the ground truth annotations.

Our results demonstrate the effectiveness of the GNN-based models in accurately identifying and classifying named entities. The models achieved an impressive score of 71%, showcasing their proficiency in handling different types of named entities across the dataset observed in table 1. The comparison with other methods, including Perdido, highlighted the advantages of our GNN-based approach in capturing complex relationships and context within textual data show in table 1.

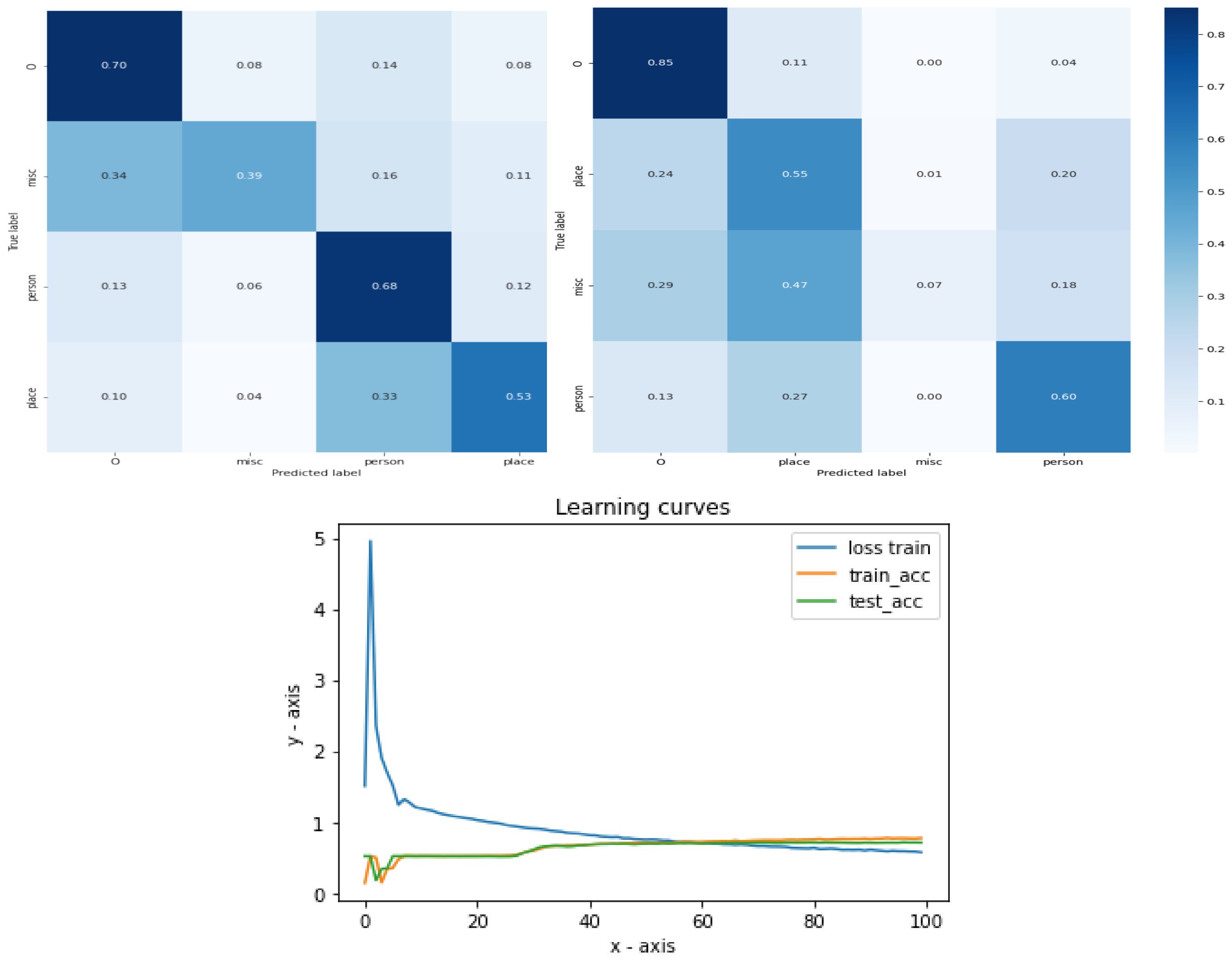


Fig. 10. Validation: Score Node prediction with 4 LABELS (Place, Person, Other, Misc)

Table 1. Performance Evaluation (Score): Node Classification

Type Of Data	Model			
	DGL	Pytorch	Bert	Perdido
Data	70%	68%	71%	70%
Graph	71%	69%		

4.2 Graph classification for nested named entity classification

Building upon the success of our GNN-based approach for node classification, we extended our experiments to address the task of graph classification for nested named entity classification.

4.2.1 Dataset Preparation. For the graph classification task, we utilized a dataset of manually corrected nested named entities pre-labeled with the Perdido Geoparser. Each nested named entity was represented as a sub-graph, where nodes corresponded to individual words and edges indicated co-occurrence relationships. The sub-graphs were categorized into three classes: Person, Place, and Misc. Much like our approach in the node classification task, we implemented two distinct forms of data (DGL and Data)

4.2.2 Model Architecture. Similar to the node classification task, we adapted our GNN models to handle the graph classification task. The models were designed to process the sub-graph representations of the nested named entities and perform graph classification.

4.2.3 Experimental Procedure. The models were trained using the sub-graphs of nested named entities, and the training process involved capturing relationships and dependencies between nodes within each sub-graph. We employed appropriate loss functions and optimization techniques for graph classification.

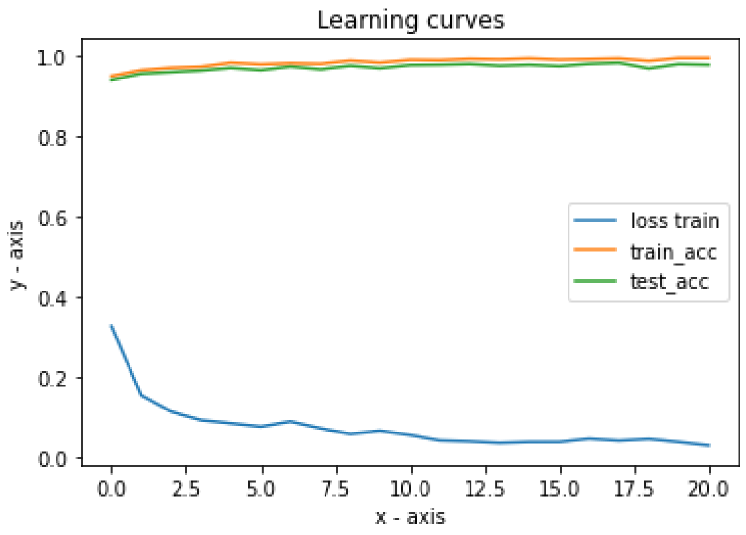
In this part of the experiments, we follow a similar approach but focus on Sub_graphs instead of individuals nodes. Each Sub-graph consists of more than two nodes and has a single target label, representing an extended named entity (NNE) or nested named entity. We restrict the target labels to three categories: Person, Place and Misc, resulting in fewer labels compared to the node classification task.

4.2.4 Evaluation and Results. As in the previous section, The trained models were evaluated using standard metrics for graph classification, including accuracy, precision, recall, and F1-score. We compared the models' performance with other graph-based classification methods, and also examined their ability to handle nested named entities with overlapping and hierarchical relationships. we compare the performance of our models on both manually created Sub-graphs and Sub-graphs generated by Perdido observed in table **??** . The correlation observed in Fig 11 reaffirms the effectiveness of our models with both types of graphs.

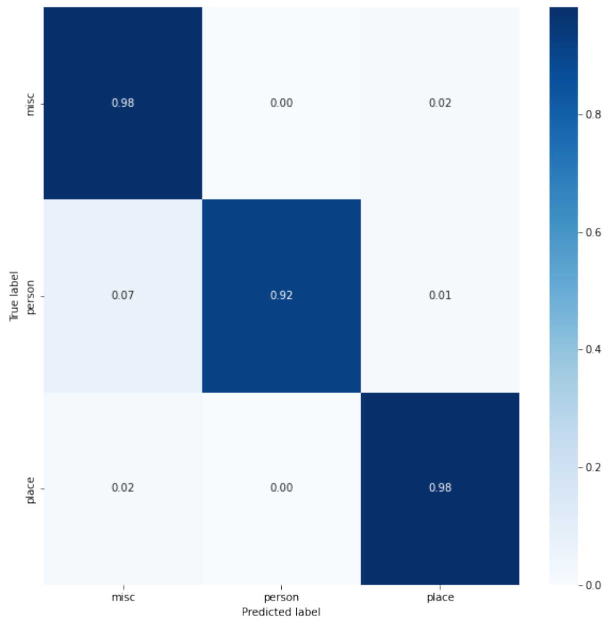
Our results observed in table 2.demonstrate that our GNN-based models perform consistently and accurately on the graph classification task as well. The models achieved a high accuracy of 98% in correctly classifying the sub-graphs of nested named entities. The comparison with other methods reinforced the efficacy of our approach in capturing complex structures and relationships within nested named entities.

4.2.5 Language Adaptability and Comparison with Perdido. In this subsection, we showcase the language adaptability of our GNN-based models by applying them to French articles. We evaluate their performance in processing data in various languages, highlighting their versatility and effectiveness.

Furthermore, by comparing our models' results with the output of the Perdido Geoparser observed in table 2, we gain valuable insights into the strengths and weaknesses of each method. The comparison aids in guiding potential improvements and enhancing the overall efficiency of our Named Entity Recognition and Classification (NERC) system. In conclusion, the experiments demonstrate that our models perform well and consistently on both types of data, whether it is in the context of individual node classification or nested named entity classification within Sub-graphs. This robust performance indicates that our models are capable of accurately capturing the textual context and relationships between words in the dataset, making them powerful tools for named entity recognition and classification tasks. The comparison with Perdido's results in table 2 allows us to gain valuable insights into the strengths and weaknesses of each method, guiding us towards potential improvements and enhancing the overall efficiency of our NERC system. What we noticed and described in the previous section is the same here, our models are good with both types of graphs (manual and automatic), the correlation in Figure Fig 11 and **??** shows that's.



(a) Progression of our Models



(b) Correlation score

Fig. 11. Validation: Score Sub-Graph Prediction with 3 LABELS (Place, Person, Misc)

Table 2. Performance Evaluation (Score): Graph Classification

Type Of Data	Models				
	DGL	Pytorch	Bert	Transformer	Perdido
Data	70%	98%	98%	71%	89%
Graph	85%	88%			

5 CONCLUSIONS

In this study, we have investigated into the domain of Named Entity Recognition and Classification (NERC) using the power of Graph Neural Networks (GNNs). Our objective was to address the challenges associated with contextual dependencies and nested structures of named entities, which traditional NER models struggle to handle. Through a combined mixture of graph-based machine learning techniques and deep learning approaches, we have effectively demonstrated the potential of our models in achieving accurate named entity recognition and classification tasks.

Our journey of experimentation has been driven by the aspiration to advance the field of NERC and to develop models capable of processing and interpreting textual information effectively. The outcomes have been notably favorable, as our model achieved an impressive accuracy score of around 98% in the prediction of nodes and sub-graphs. This score not only highlights the model's proficiency in identifying and categorizing individual words but also its ability to deal with complex nested named entities. For instance, it accurately recognizes "France" as a Place entity and adeptly identifies compound nouns like "New York City." OR "Ville de France"

One notable advantage of our model lies in its flexibility and language adaptability. While Perdido is tailored for the French language, our model showcases its prowess by achieving admirable results on English data as well. This adaptability highlights the strength of our approach, positioning it as a flexibility tool for handling diverse languages and varying text genres.

The journey of exploring the interaction between Graph Neural Networks and NERC has provided invaluable insights. The integration of GNNs with deep learning techniques empowers our models to encapsulate Complex textual context and complex relationships between words. This capability, in turn, it basically lies the model's exceptional performance in entity recognition and classification tasks. Moreover, comparative analysis with Perdido provided us with a comprehensive perspective on the model's strengths and potential areas for revision.

As we are about to complete this study, we stand on the threshold of exciting possibilities that lie ahead. Future work involves extending our model's language support beyond English and French, expanding its applicability to a diverse linguistic landscape. Additionally, we are dedicated to further improving the model's effectiveness and fine-tuning its accuracy in named entity recognition. Leveraging concepts such as Pointwise Mutual Information (PMI), Euclidean distance, or the cosine distance measure to calculate the distance between nodes within our graph-based model, we also aspire to explore strategies that enable the model to achieve effective predictions without necessitating data balancing.

In conclusion, our research provides a significant contribution to the field of Natural Language Processing through the utilization of Graph Neural Networks for Named Entity Recognition and Classification tasks. We introduce a robust approach that holds the promise of significantly improving tasks such as information retrieval, question answering, text summarization, sentiment analysis, and various other applications in NLP. As we continue our journey of exploring the collaboration between GNNs and language, We are facing exciting possibilities. Our determination to develop the horizons of precise and advanced named entity recognition and classification it pushes us towards achieving excellence.

REFERENCES

[1] Francisca Adoma Acheampong, Henry Nunoo-Mensah, and Wenyu Chen. 2021. Transformer models for text-based emotion detection: a review of BERT-based approaches. *Artificial Intelligence Review* (2021), 1–41.

[2] Partha Sarathy Banerjee, Baisakhi Chakraborty, Deepak Tripathi, Hardik Gupta, and Sourabh S Kumar. 2019. A information retrieval based on question and answering and NER for unstructured information without using SQL. *Wireless Personal Communications* 108 (2019), 1909–1931.

[3] David Campos, Sérgio Matos, and José Luís Oliveira. 2012. Biomedical named entity recognition: a survey of machine-learning tools. *Theory and Applications for Advanced Text Mining* 11 (2012), 175–195.

[4] Alberto Cetoli, Stefano Bragaglia, Andrew D O'Harney, and Marc Sloan. 2017. Graph convolutional networks for named entity recognition. *arXiv preprint arXiv:1709.10053* (2017).

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[6] Safaa Eltyeb and Naomie Salim. 2014. Chemical named entities recognition: a review on approaches and applications. *Journal of cheminformatics* 6 (2014), 1–12.

[7] Mauro Gaio and Ludovic Moncla. 2017. Extended named entity recognition using finite-state transducers: An application to place names. In *The ninth international conference on advanced geographic information systems, applications, and services (GEOProcessing 2017)*.

[8] Alankar Jain, Bhargavi Paranjape, and Zachary C Lipton. 2019. Entity projection via machine translation for cross-lingual NER. *arXiv preprint arXiv:1909.05356* (2019).

[9] Mohammad Ebrahim Khademi and Mohammad Fakhredanesh. 2020. Persian automatic text summarization based on named entity recognition. *Iranian Journal of Science and Technology, Transactions of Electrical Engineering* (2020), 1–12.

[10] Robert Leaman, Ritu Khare, and Zhiyong Lu. 2015. Challenges in clinical natural language processing for automated disorder normalization. *Journal of biomedical informatics* 57 (2015), 28–37.

[11] Diego Mollá, Menno Van Zaanen, and Daniel Smith. 2006. Named entity recognition for question answering. In *Proceedings of the Australasian language technology workshop 2006*. 51–58.

[12] Ludovic Moncla and Mauro Gaio. 2023. Perdido: Python library for geoparsing and geocoding French texts. In *First International Workshop on Geographic Information Extraction from Texts (GeoExT)*.

[13] Ludovic Moncla, Denis Vigier, Katherine Mcdonough, Alice Brenon, and Thierry Joliveau. 2021. Combinaison d'approches qualitative et quantitative pour le repérage et la classification des entités nommées dans l'Encyclopédie de Diderot et d'Alembert (1751-1772). In *Theoretical linguistics in the light of the interaction of qualitative and quantitative approaches*.

[14] Urchade Zaratiana, Pierre Holat, Nadi Tomeh, and Thierry Charnois. 2022. Hierarchical Transformer Model for Scientific Named Entity Recognition.