Automatic information extraction in business documents

Santiago Andres Moreno Acevedo

Tesis de maestría presentada para optar al título de Magíster en Ingeniería de Telecomunicaciones

Director
Juan Rafael Orozco Arroyave, Doctor (PhD) en Procesamiento de Señales.
Asesor
Juan Camilo Vásquez Correa, Doctor (PhD) en Ciencias de la computación.

Universidad de Antioquia
Facultad de Ingeniería
Maestría en Ingeniería de Telecomunicaciones
Medellín, Antioquia, Colombia
2023

Cita	(Moreno-Acevedo, 2018)
Referencia **Estilo APA 7 (2020)**	Moreno-Acevedo, S. A. (2023). *Automatic information extraction in business document.* [Tesis de maestría]. Universidad de Antioquia, Medellín, Colombia.

Maestría en Ingeniería de Telecomunicaciones, Cohorte Seleccione cohorte posgrado.

Grupo de Investigación Telecomunicaciones Aplicadas (GITA).

Centro de Investigación Ambientales y de Ingeniería (CIA).

Automatic information extraction in business documents

UNIVERSIDAD DE ANTIOQUIA

1 8 0 3

Master's Thesis in Telecommunication Engineering

Santiago Andres Moreno-Acevedo

Director: Prof. Dr.-Ing. Juan Rafael Orozco Arroyave

Advisor: Dr.-Ing. Juan Camilo Vásquez-Correa

Faculty of Engineering

Department of Electronics and Telecommunications

University of Antioquia

Acknoledgements

First of all, I would like to thank my mother, Isabel Acevedo, and my father, Andres Moreno; they made all this possible with their invaluable support, unconditional love, and understanding. Also, I'm infinitely grateful to my grandparents, Octavio and Maria, and my aunt Elizabeth; they were part of my whole process bringing me support and facilities to allow me to focus just on my Masters. I would also thank my family, who have trusted me for the culmination of my research work; everybody in my family has shown me the value and power of patience and perseverance. Moreover, I thank my friends Santiago, Sebastian, and Esteban for their personal support during this process.

Now, I want to thank my colleagues Daniel Escobar, Cristian Ríos, and Nestor Calvo for supporting me in this process, giving me their friendship, their academic support, and endless discussions that bring many ideas. I also want to thank the colleagues of GITA Lab at the University of Antioquia because they made all the working hours more enjoyable. I want to express also my gratitude to my director Prof. Dr.-Ing. Juan Rafael Orozco, and my advisor Dr.-Ing. Juan Camilo Vásquez, for the knowledge transfer that I could obtain through their experiences, for their support, for always being ready to answer any questions, for the large amount of advice, and for their patience during this long research process.

Finally, I want to thank the University of Antioquia for its financial support during the development of this master's thesis through the master's scholarship and the CODI project by grant Number PI2019-24110 Entitled: "Desarrollo de nuevas tecnologías para el análisis de voz y texto con el fin de calificar automáticamente la calidad del servicio brindado por personal (humano ó virtual) de atención al cliente." Also I would thank Pratec Group S.A.S. for providing the raw information used in this work, and for founding this research.

Abstract

Information Extraction (IE) is a topic of Natural Language Processing that has gained interest in the research community for its applications in real-world areas, such as law environments where the analysis of documents is very important. So far, IE has been extensively studied in general contexts with ideal data with many samples per class. However, real-world contexts do not have either large amounts of data or balance among classes. Therefore, it is necessary to develop models that can handle real-world data problems. This master's thesis aims to investigate techniques and methods for handling limited and unbalanced data in Natural Language Processing (NLP) contexts. The goal is to implement these techniques and methods into a software tool that can automatically extract information from documents. With this aim, two NLP approaches were studied: Named Entity Recognition (NER) and Relation Classification (RC). Different methods were analyzed, including both architectural and data-related approaches. To address the class imbalance, several loss functions were explored to create a model that prioritizes samples that are hard to classify. Additionally, data augmentation strategies were employed to face the limited data problem. A methodology for NER was developed, integrating data augmentation strategies and the focal loss function into a benchmark model. For RC, we identified a state-of-the art architecture that uses the focal loss function and performs well with limited data. The outcomes for NER and RC were satisfactory at the end of the work. Finally, both the NER methodology and the RC architecture wwew integrated into a software tool that enables automatic NER and RC tasks for any given document. This work is the first stage in creating an automatic document analysis tool.

Contents

1 Introduction **6**
- 1.1 Motivation . . . 6
- 1.2 Research Problem . . . 8
- 1.3 State of the art . . . 9
  - 1.3.1 Named Entity Recognition . . . 9
  - 1.3.2 Relation Classification . . . 13
- 1.4 Objectives . . . 14
  - 1.4.1 General . . . 14
  - 1.4.2 Specific . . . 14
- 1.5 Contribution of this study . . . 15
- 1.6 Outline . . . 15

2 Theoretical Background **16**
- 2.1 Information extraction (IE) . . . 16
  - 2.1.1 Named Entity Recognition (NER) . . . 16
  - 2.1.2 Relation Classification (RC) . . . 17
- 2.2 Methods . . . 18
  - 2.2.1 Conditional Random Fields (CRF) . . . 18
  - 2.2.2 Fully Connected Neural Networks (FCNN) . . . 19
  - 2.2.3 Convolutional Neural Networks (CNN): . . . 20
  - 2.2.4 Recurrent Neural Networks (RNN) . . . 22
  - 2.2.5 Transformers . . . 25
  - 2.2.6 Loss functions . . . 27
  - 2.2.7 Data augmentation . . . 29
- 2.3 Software . . . 30
  - 2.3.1 Libraries . . . 31
  - 2.3.2 Formats . . . 31

3 Data **36**
3.1 CoNLL-02 [24] . . . 36
3.2 Legal Entity Recognition (LER) [13] . . . 36
3.3 Colombian Chamber of Commerce (CCC) . . . 37
3.4 Multilingual Anonymisation for Public Administrations (MAPA) [17] . . . 38
3.5 SemEval 2010 task 8 [72] . . . 38

4 Experiments and results **40**
4.1 Problem's identification and architecture selection . . . 40
4.1.1 Data characterization . . . 40
4.1.2 Benchmark architectures . . . 42
4.1.3 Comparison . . . 44
4.1.4 Problems of real-world data . . . 46
4.2 Methodology against problems . . . 50
4.2.1 Loss Functions . . . 50
4.2.2 Data Augmentation . . . 52
4.3 Methodology evaluation . . . 53
4.3.1 Test in different scenarios . . . 53
4.3.2 Knowledge Generalization . . . 55
4.3.3 Semantic Similarity . . . 57
4.4 Relation classification . . . 57
4.4.1 Sentence Representation . . . 58
4.4.2 Architecture . . . 58
4.4.3 Test in benchmark database . . . 59

5 Software **62**
5.1 NER . . . 63
5.1.1 Training . . . 63
5.1.2 Tagging . . . 63
5.1.3 General View . . . 64
5.2 RC . . . 66
5.2.1 Training . . . 66
5.2.2 Tagging . . . 66
5.2.3 General View . . . 68

6 Conclusions and future work **70**

List of Figures **74**

List of Tables **75**

Bibliography **82**

Chapter 1

Introduction

1.1 Motivation

Documents are one of the most important sources of information in business processes. There are a lot of data that can be extracted from them. This process is manually performed for many documents, making it repetitive, tedious, error-prone, and expensive. Automatic document analysis is an important goal of computational intelligence applied to the industrial sector. It could be divided into the following tasks: character recognition and Information Extraction (IE). IE is an important field of Natural Language Processing (NLP), which has a set of different techniques that could be used to solve different problems, including document classification [1], [2], automatic summarizing [3], Relation Classification (RC) [4]–[6], and Named Entity Recognition (NER) [7]. These last two require to identify words associated with entities that represent the most important information in the text.

An entity in a text is a predefined category commonly associated to persons, places, organizations, dates, and others. NER is a task that consists in recognizing which words or tokens (groups of n words) correspond to mentions of pre-defined entities [7]. For instance, in the sentence "Jorge Robledo went to Robledo", the segment "Jorge Robledo" is a mention of an entity of person, and "Robledo" is a mention of an entity of location, respectively. NER has several uses in the field, such as in Complaints and Claims systems to identify username, date, ticket number, etc. Also could be used in recruiting processes to identify relevant information in a curriculum vitae.

Previous works of NER show its performance in general applications [8], with balanced and big databases [9]–[11]. However, these works do not

have real-world applications because their data are ideal, since it has general databases with few entity types and many samples per class. In contrast, for specific domains the data presents different problems, there are not enough amount of samples, and it is usually unbalance [12], [13]. In the last years, NER has been used in specific environments such as legal domain [14]–[16], and very few works include real applications [17], [18]. There are very few related works of NER for business environments, and these kinds of fields usually have data with the problems mentioned.

Getting a large amount of data in real-world applications takes time and effort. Generally, context-specific data presents the problems mentioned above. Therefore, implementing models used for NER with general approaches in context-specific fields such as business does not perform satisfactorily. Even using techniques such as transfer learning, taking the general knowledge to generate specific knowledge is not enough because the models cannot face the data problems. Due to this reason, the usefulness of the NER for information extraction is wasted, and its application in real-world problems such as document analysis seems to be far away. Tasks that could be automatized with NER are performed manually for many documents, making this process expensive and slow.

NER is also the first step previous to perform Relation Classification (RC). The entities recognized by NER provide information about the categories found in the text, but those entities could have a relation among them. Therefore, NER needs to be accompanied by other types of algorithms to classify the relation among entities in the text. A relation is a semantic association that could be done between named entities; for example, in the sentence "Bill Gates is the CEO of Microsoft" there are two entities, "Bill Gates" as a person entity and "Microsoft" as organization entity. Between these two entities, there is a relation of "works for". The methods of RC focuses on identifying semantic relations that exist between two or more entities that are in a sentence or in a document. RC, has a broad range of applications, for example, support in clinical decision making [4], drug discovery [5], and economic management [6]. The RC is considered fundamental to support other NLP areas, such as the construction of knowledge graphs, natural dialogue systems, and the understanding of natural language.

NLP used for IE tasks is promising in the business environment since it could automatized document analysis process that so far are done manually. This allows to reduce costs and time for the enterprises. In this project we are

focused in the Spanish context using two databases, one for the legal domain in Spain, and other for the commercial domain in colombian. The Colombian database is a first approach of IE tools in the commercial context in Colombia allowing to explore more of IE techniques to develop a tool that automatize those document process in Colombia.

This proposal aims to work on modeling methodologies for automatic text analysis, considering different methods from NLP, including NER and RC, focusing the models in environments with high semantic similarity and few and unbalanced data typical of the real-world. We propose to address the data problems in two steps: (1) Modify a model to improve its performance against the data problems. (2) Then, pre-process the data with different techniques to augment the samples. At the end of this process, we expect to create a software that automatically analyses different business documents and extracts relevant information of them. Automating document analysis to reduce costs and improve time in repetitive processes typical of business environments.

1.2 Research Problem

The processing of document information has been completely manual for many years. With the advent of the computer and data storage, the process changed and instead of doing the whole process manually, the important information is extracted and stored on a drive to be later used by a human. In recent years, different data acquisition processes were automatized facilitating it's storage, and allowing different information extraction processes to be automatized.

Although the information processing began to be automatized, the acquisition of this information is mostly repetitive, expensive, tedious, and manual. Some tools such as MAXQDA [19] and TalkWalker [20] have been developed for text analysis. Nevertheless, these tools aim to perform a general analysis of the text but do not cover industrial environments.

With the advances in computational resources, automatic information extraction in business documents is achievable by using NLP techniques. With the development of a dedicated tool to perform this information extraction, different processes in the industry will be faster and cheaper. This achievement optimizes the companies' budget and reduces the time of bureaucratic tasks. NLP techniques such as NER and RC are useful for information extraction.

However, the models of NER and RC are developed with big and general datasets [8]–[11] which does not fit into specifics fields such the industrial field.

Some studies for specific fields have been performed [13], [21], [22] but so far only at a theoretical level. Is well known that Deep Learning (DL) is the best approach for NLP tasks, but this approach requires large amount of data; also, it is not focused on few data with great imbalance. Therefore, we pose the following research question: It is possible to develop an IE tool using NER and RC models based on DL that can perform well in real-world scenarios with limited data?

During the progress of this work, we will evaluate different DL methods for NER and RC tasks, and pre-processing techniques for few and imbalanced data. The performance of the methods and techniques in different scenarios will be compared to select the better one. We hypothesize that in specific contexts with similar semantic fields where there is few data and a great imbalance of classes, DL approaches with pre-processing techniques will be efficient for NER and RC. The final product will be a software application incorporating the methods developed to automate the NER and RC processes in real-world environments.

In this work, we focus on the textual information of the document, although it is possible to analyze other data, for example, images, signatures, and others. This work is the first step in the process of creating a robust system for automatic document analysis.

1.3 State of the art

1.3.1 Named Entity Recognition

The concept of NER has been developed since the "Message Understanding Conference" (MUC-06) [23] where it was proposed for the first time. The most common datasets used for the NER were created at the "Conference on Computational Natural Language Learning" (CoNLL). These datasets are the CoNLL-02 [24], which contain 380,923 and 333,582 tokens for Spanish and Dutch, respectively. And the CoNLL-03 [25] with 301,418 tokens for English and 310,318 for German.

The NER was approached in the first instance using rule-based methodologies. In 2005 it was developed ProMiner: a rule-based tool for NER [26] which works with a rule-based methodology to identify proteins and genes in medical texts. Later, with the aim of making adaptative models, in [27] was proposed a method to recognize mentions on tweets using Conditional Random Fields

(CRF). The authors achieve an F1-score of 53% considering 10 entities.

With the increase in computational capacity, DL techniques became popular in the NER task. In [9] different models based on LSTM and CRF were proposed. Using the CoNLL-03 English dataset the authors achieve a 90.1% of F1-score. The features used are combinations of context features extracted by Part Of Speech-tagging, spelling features, and word embedding features. In [8] was implemented a DL model with a Bidirectional Long-Short Term Memory (BiLSTM) architecture, and a CRF layer. The text representation for each word is a hybrid model of character-level representations concatenated with word-level representations. Using this model the authors got an F1-score of 90.94% in the English corpus of the CoNLL-03 and 85.8% in the CoNLL-02 Spanish corpus. In [10] a new DL approach for NER was proposed. As in [8] a hybrid model of word embeddings and character-level embeddings was used. The authors implemented a technique called Dynamic Transfer Learning that consists of an encoder-decoder architecture using LSTM for both. This model achieved an F1-score of 82.12% using 10% of the training data of the CoNNL-03 dataset.

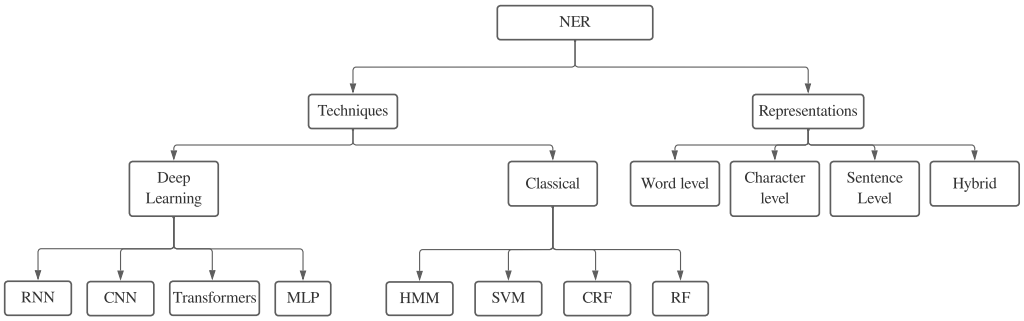

Figure 1.1. Taxonomy of the methods for NER

Most of the methods in the literature are tested considering databases with a small number of entity targets, many samples, and balanced classes as those of CoNLL. There are few works focused on specific data. For instance, in [12] the authors proposed a method to achieve a high score using unbalanced data. They used a model with two architectures for data representation, one for the entities with a large number of samples, and the other for the entities with few samples. The authors reported an F1-score of 99.1% for the entities with a big number of samples and 53.1% for the entities with few samples.

Recently, in NER field has been studied the fine-grained NER. In this approach, there are several entities with similar semantic fields which makes it difficult to recognize. In [21] the authors use different DL techniques for different purposes. To represent the text they used word embeddings from Embeddings Language Models (ELMo) [28] and Bidirectional Encoder Representations from Transformers (BERT) [29]. For the labeling process, they used an LSTM-CRF architecture, and the czech dataset CNEC 1.0 which has 10 big entities and each entity has between 3 and 10 specific sub-entities. The authors achieve an F1-score 86%. Besides, another important field in the NER has captured the interest of the research community, this is called nested NER. The idea of nested NER, is to recognize first a primary entity of a general field and with this primary recognition make another recognition with more specific entities of the field. This approach was applied to classify if a text has an advertising technique and what kind of advertising technique it has [22].

The aforementioned works are focused on general applications. Recently the interest in NER for specifics domains has been increasing especially in the legal domain [15], [16]. In [14] the authors focused on the NER task in Romanian legal documents, they achieve 90% of F1-score using a combination of different models. In [13] addressed the problem not only in the legal domain but the semantic similarity among entities, they achieve 95% using a large dataset of German documents. In the work [17] the NER was focused on the legal domain but with a real application for the Spain government. However, there are no works focused on business environments where the data is few, unbalanced, and semantically similar. The Figure 1.1 shows the summary of the main techniques used in NER and the different approaches to represent the text. In the Table 1.1 are summarized some studies about NER.

Table 1.1. Experiments for NER

Title	Technique	F1-score %	Database	Year
ProMiner: rule-based protein and gene entity recognition [26]	Rule-based	78.80	FlyBase	2005
Named Entity Recognition in tweets: an experimental study [27]	CRF	51.01	Twitter	2011
Bidirectional LSTM-CRF Models for Sequence Tagging [9]	BiLSTM-CRF	84.46	CONLL-03	2015
End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF [30]	BiLSTM-CRF	91.21	CONLL-03	2016
Neural Architectures for Named Entity Recognition [8]	LSTM-CRF	90.94	CONLL-03	2016
Deep Active Lerning for named entity recognition [31]	CNN-CNN-LSTM	86.52	OntoNotes-5.0	2017
Joint Extraction of Multiple Relations and Entities by using a Hybrid Neural Network [32]	BiLSTM-CNN-LSTM -CRF	63.20	CONLL-04	2017
End-to-end Recurrent Neural Network Models for Vietnamese Named Entity Recognition: Word-level vs. Character-level [33]	Skip_gram-CNN-BiLSTM-CRF	88.38	VSLP	2017
Named Entity Recognition with stack residual LSTM and trainable bias decoding [34]	BiRNN-CRF	91.66	CONLL-03	2017
Hybrid semi-Markov CRF for Neural Sequence Labeling [35]	LM-BiLSTM-CRF-HSCRF	91.38	CONLL-03	2018
Neural Architectures for Nested NER through Linearization [21]	BERT-Flair-BiLSTM-LSTM	84.40	ACE-2004	2019
A Multi-task Approach for Named Entity Recognition in Social Media Data [36]	LM-BiLSTM-CRF-HSCRF	41.86	WUNUT 2017	2019
FLERT: Document-Level Features for Named Entity Recognition [11]	RoBERTaXML+Linear	94.9	CONLL-03	2020
Named entity recognition in the Romanian legal domain [14]	Embeddings Combinations-BiLSTM-CRF	90.36	LegalNERo	2021
Spanish Datasets for Sensitive Entity Detection in the Legal Domain Entity Recognition [17]	RoBERTa-b+Linear	92	CPP and EUR-Lex	2022
FiNER: Financial Numeric Entity Recognition for XBRL Tagging [18]	BERT+Linear	85.3	FiNER	2022

Over time it has been used different techniques to achieve a good score in the NER task. The first approaches were using rule based techniques [26], but these kinds of models were specific for its application; hence, they began to use adaptive models based on probabilistic methods [27]. Later, with the advantage of the computational resources, the DL becomes the main approach, first using Recurrent Neural Networks (RNN). Still, these kinds of neural networks only use the past context of the words. Therefore, the Bidirectional RNN began to be used [9] to work with the past and the future context. Due to the vanishing gradient problem and the loss in the representation capacity of the RNN, the community began to work with transformers taking advantage of its Attention mechanism [11]. All these works are evaluated with the F1-

score, which is the most common metric used in NER and RC since it contains information about the accuracy, precision, and recall of the considered model.

1.3.2 Relation Classification

Additionally, NER is the base for RC,which tries to find relation between given entities. The first studies about RC appeared in the early 2000s in the conference "Conferencia Iberoamericana en Sistemas, Cibernética e Informática" (CISCI-2003) [37]. In [38] was developed a model for RC using conditional probability between the entities already identified, they got an F1-score of 72.8 % to 37 relations.

In [39], a method is proposed to implement lexicon-level and sentence-level features to identify the relations. The sentence-level features are obtained using a Convolutional Neural Network (CNN) with word-level and positional features. These features are concatenated with the lexicon-level features to feed a fully connected network with softmax activation. The proposed method achieves an F1-score of 82.7% with a database of 10717 samples and nine relations. Another approach is proposed in [40] where a cascade system is used to RC with previously recognized entities. This paper has used a combination of word-embeddings, POS, and NER for the text representation. Then, a CNN is used to classify the relation. The author reported an F1-score of up to 55.6% using a database of 726 papers of Chinese literature with nine relations.

In [41] was considered the problem of long texts (more than one sentence to extract the relations) in biology. The authors propose a transformers based model called "Bi-affine Relation Attention Networks (BRAN)", performing the NER and then the RC. This model uses the "Multi-head Attention" mechanism from transformers architecture, which is combined with a CNN for NER. The Attention-CNN cell gives a vector per token passed by two Multi-Layer Perceptron (MLP) called head and tail. Each MLP returns a vector that is called head-entity and tail-entity, according to the MLP used. Then, with the output of these two MLP, they applied the LogSumExp function to obtain the score of the head and tail entities for each relation. Using that model they achieve an F1-score of 47.3 % for 14 relations. In [42], they take the BRAN model of [41] and call the MLP layers as first order relations. Then, they add in parallel to the MLPs another 2 sequential layers of MLP, which are called second order relations. The probability of the entities for each relation is ob-

tained by adding up the first order relations and the second order relations multiplied by a hiper-parameter called alpha, which gives the weight to the second order relations.

In [43] is proposed a model that makes the NER and the RC tasks based on a pre-trained model. Specifically, the authors use the trained BERT model to obtain the word-embeddings of the tokens. These embeddings feed a Fully Connected Neural Network (FCNN) and then pass for a softmax layer, in this point the output of the model is the entity for token. Thus, to obtain the relations between the entities, the softmax output is concatenated with the original word-embeddings and it passes for an FCNN, this is made for a head-entity and for a tail-entity, similar to the model proposed in [41]. The output of these layers is used in a bi-affine classifier with a softmax layer at the end. The model achieves an F1-score of 66.83% in the CoNLL-04 database that has 5 types of relations.

Other approaches combined RNNs and CNNs to model temporal and spatial information [44]. In [45], the authors used Word2Vec model to obtain a numerical representation of the words in the text. These representations feed a Bi-LSTM followed by a CNN. F1-scores of up to 60% were achieved in the KBP37 database. The authors improved this architecture in [46] by incorporating a fusion strategy based on attention mechanisms.

1.4 Objectives

1.4.1 General

To design and implement a system that extracts relevant information from digital business documents, using named entity recognition and relation classification models, such that an automatic document analysis framework will be developed.

1.4.2 Specific

1. To select models and methods for information extraction based on NLP techniques focused on imbalanced and few data to evaluate their performance in different scenarios.

2. To implement and evaluate models and methods for information extraction based on NER and RC techniques, focusing on their performance

on imbalanced and few data from business fields.

3. To develop a software application that integrates the models of named entity recognition and relation classification to automatize the process of information acquisition in digital business documents.

1.5 Contribution of this study

According to the state of the art, since 7 years now, the scientific community has started to explore different DL methods for IE based on NLP. Therefore, this research mainly focuses on using NER and RC techniques for the Automatic IE of real-world data. In order to contribute to this aim, the following are the main outcomes of this research work.

Real-world data usually is highly unbalanced and have a limited amount of samples. Therefore, two approaches were addressed: Architecture modification and data treatment. This work explores different loss functions to adapt a model for data with high-class unbalance. Furthermore, different Data augmentation techniques focused on NER were explored for the data treatment approach. We develop a novel methodology considering architecture modification and data treatment for NER. Additionally, a state-of-the-art architecture was implemented for RC, which performs well when working with limited data using the same loss function as NER. Finally, the resulting methodology and architectures are developed into a software that automates the NER and RC tasks. The software involves the different techniques used in this study to perform NER and RC in real-world environments.

1.6 Outline

This research work is divided into five chapters. Chapter 1 contains the context, the research problem, the objectives and the contribution of the study. Chapter 2 includes the methods used in this work. Chapter 3 contains detailed information on the different databases. Chapter 4 presents details of the experiments and their results. Chapter 5 involves the software description for the NER and RC tasks. Finally, Chapter 6 includes the main conclusions, discussion and future work.

Chapter 2

Theoretical Background

2.1 Information extraction (IE)

IE with NLP are tasks that entail automatically locating and extracting structured information from unstructured text material. Unstructured text data, such as free-form text in documents, online pages, or social media posts, is text that doesn't have a pre-established format.

The objective of information extraction is to automatically recognize and extract pertinent information from unstructured text data, such as entities (such as names of people, companies, or locations), relationships between entities (such as who works for whom), and events (such as acquisitions, mergers), and then transform that information into a structured format for further analysis or database storage.

This study aims to use NER and RC to extract relevant information from unstructured data. We focus on working with real-world data, which usually has low samples and unbalanced classes.

2.1.1 Named Entity Recognition (NER)

One of the most challenging aspects of NLP is the identification of entities to which the words in each sentence refer. Entities refer to predefined categories such as persons, places, organizations, dates, and others. This recognition allows for more detailed text analysis by identifying the relationships among entities or recognizing more specific entities from a previously identified super-entity. The NER task algorithms operate with tokens, which are groups of words of varying sizes known as n-grams, where n represents the number of

words in each group; for example, 1-gram refers to a single word token. The term "entity mention" refers to the appearance of an entity in a text, which can vary in length depending on the number of words used to refer to it. Therefore, an entity mention (or segment) can consist of one or more words. The NER is a sequence classification problem, this is, a problem where the input is a set of elements into a sequence $\mathbf{Y_t} = \{ \mathbf{y}_{t0}, \mathbf{y}_{t1}, ..., \mathbf{y}_{tl}\}$ of length L, and the prediction is a label for each element into the sequence.

NER is a technique that focuses on identifying entities such as locations, persons, organizations, and others mentioned in a given text [7]. NER is an important field of study in NLP, with various applications such as text comprehension [47], information retrieval [48], and document representation [49]. The current state-of-the-art models for NER can be classified into four groups: (1) Rule-based approaches [50], (2) unsupervised algorithms [51], (3) supervised learning based on features [52], and (4) deep learning-based algorithms [8]. Generally, these models take the text as input and predict the entity corresponding to each token or n-gram.

2.1.2 Relation Classification (RC)

NER methods have traditionally been used to identify entities in text. However, recognizing entities alone is insufficient to fully comprehend the semantic information contained within a text since the relationships between entities are not known. Relation classification between entities provides a deeper level of understanding of the text. The objective of relation classification is to automatically identify the type of relationship that exists between pairs of entities in a sentence or document. For example, in the sentence "John works at Microsoft," the relationship between the entities "John" (Person) and "Microsoft" (Organization) is classified as "works at." Similarly, in the sentence "Disney acquired Pixar," the relationship between the entities "Disney" (Organization) and "Pixar" (Organization) is classified as "acquired." As NER, RC is a sequence classification problem, this is, a problem where the input is a set of elements into a sequence $\mathbf{Y_t} = \{ \mathbf{y}_{t0}, \mathbf{y}_{t1}, ..., \mathbf{y}_{tl}\}$ of length L, but the prediction is only one relation per senquence.

Since Relation classification involves working with Natural Language there are several applications. In the biomedical field, to detect drug-to-drug [53] or protein-to-protein [54] relations. Also is used in legal field [55], in construction contract documents [56], in crime data [57], among others. According to the

abstraction level and the input type of neural network, the relation extraction systems can be divided into several categories, including identifying relations at the sentence level, document level, or even corpus level.

A relation extraction system can generally be constructed from a supervised classification problem where a classifier is trained using text representations extracted from recognized entity groups [58]. DL-based systems are currently the most popular to solve problems of relation extraction. Generally architectures such as CNN [59], RNN [60], transformers [61], and recently Graph Neural Networks (GNN) [62] have been adapted to relation extraction between entities. Models of relation extraction can be divided into two types: (1) models based on sequence modeling, those that operate at the token level [39], [63]; and (2) Dependency-based models, which incorporate the concept of dependency trees in the models [64].

2.2 Methods

Along this work we use different DL techniques to perform the Information extraction approach. The models implemented use techniques as CRF, classical FCNN, CNN, RNN, and Tranformers. These techniques are described as follows.

2.2.1 Conditional Random Fields (CRF)

CRF is a stochastic method that tries to predict the label of a given input based on the conditional probability of the labels given the inputs. Unlike Hidden Markov Models, the CRF use the conditional probability of the whole set of labels given the set of inputs as it is shown in Equation 2.1. Where

$$p(\mathbf{y}|\mathbf{X}) = p(y_1, ..., y_k|\mathbf{x}_1, ..., \mathbf{x}_k) \tag{2.1}$$

These models are trained with k samples where each one contains a set of observations and its labels $(\boldsymbol{X}^i, \boldsymbol{Y}^i)_k$. The model extracts a set of features $\mathbf{f}(i, \boldsymbol{Y}_i, \boldsymbol{Y}_{i-1})$ which aims to express some features based in the previous and current labels. Moreover, $\mathbf{g}(i, \boldsymbol{Y}_i, \boldsymbol{X})$ that represents the existent dependencies between the sequence of observations. Both functions represent the existent dependencies between different states and sequences. Unlike Hidden Markov

Models, each state depends on several observations simultaneously. The training process assigns a weight for each feature according to its relevance. A scheme of this for a specific sequence is shown in Figure 2.1

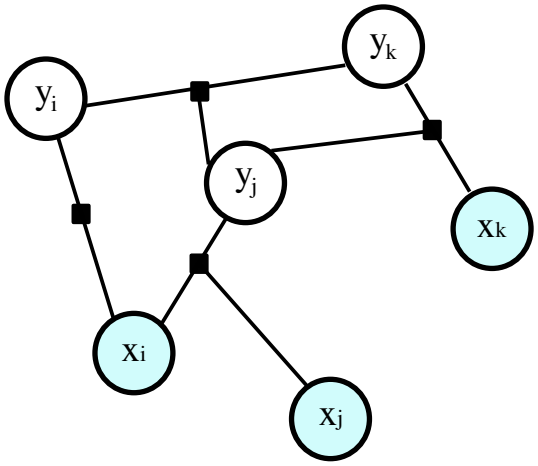

Figure 2.1. Conditional Random Field scheme

2.2.2 Fully Connected Neural Networks (FCNN)

Fully connected neural networks, are a fundamental type of neural network used for a wide range of machine learning tasks. In a fully connected network, every neuron in one layer is connected to every neuron in the next layer, forming a dense matrix of weights that map the inputs to the outputs. The main feature of the FCNN is the combination of linear and non-linear functions. The non-linear functions $\phi(x)$ (also called activation functions) include *sigmoid, tanh, softmax,* among others. The linear functions are made up of the product of the weights with the input value plus the bias: $\mathrm{f}(\mathbf{x}) = \boldsymbol{wx} + \boldsymbol{b}$. Since the FCNN can have multiple layers, the system can "learn" increasingly complex operations or functions because to the non-linear activations [65]. The output of a FCNN is given by the Equation where k is the number of layers of the FCNN.

$$y = \phi_0(f_0(\phi_1(f_1(...\phi_k(f_k(\mathbf{x})))))) \tag{2.2}$$

The FCNN is widely used in multi-class problems, where the output layer is composed of as many neurons as classes, and the score of each neuron is the score for each class. Along this study we characterized the text-information in

one numeric vector, this vector is the input of the FCNN and the output are the vector's scores of each class as is shown in Figure 2.2

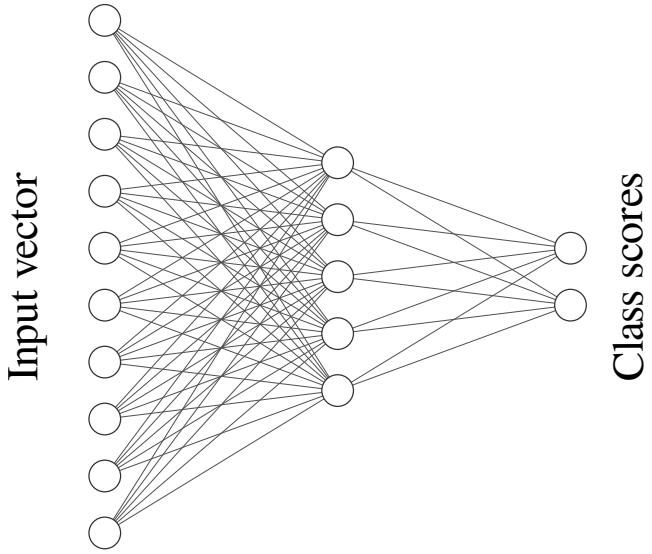

Figure 2.2. Fully Connected Neural Network

During training, the weights of the fully connected network are learned by minimizing a loss function that measures the discrepancy between the predicted outputs and the ground-truth labels. This is typically achieved using a technique called backpropagation, which computes the gradients of the loss with respect to the network's weights and updates them in the opposite direction of the gradient to improve the model's predictions.

2.2.3 Convolutional Neural Networks (CNN):

Contrary to the FCN, these networks work with a matrix as an input. Although this architecture was mainly proposed for image analysis, it has been successfully used in NLP applications, including NER [30]–[32]. In text analysis, the input matrix is built from the word-embeddings, which concatenated represents a sentence as is shown in Figure 2.3.

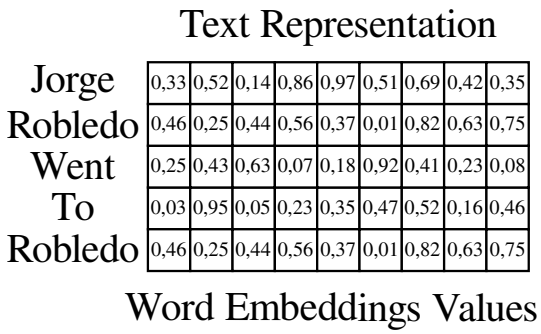

Figure 2.3. Matrix representation for a sentence

The CNN has three main procedures convolution, pooling, and flattening. The convolution consists of the application of a kernel to the matrix, the kernel multiplies different sections of the matrix by sliding through it. The values of the kernel are trained to look for a different way to analyze the information. The kernel slides through the matrix The pooling layer is a procedure that reduces the dimension of the data. This reduction can be made in different ways; some are taking the max or the average of a matrix of the layer dimension. Lastly, the final data is transformed for a $n \times m \times p$ dimension to a 1-dimensional vector to represent the input text.

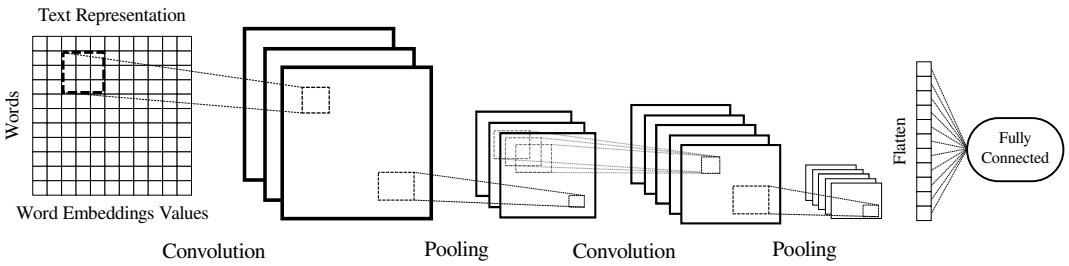

Figure 2.4. Convolutional Neural Network

As is shown in Figure 2.4 the CNN takes the word-embeddings of the text and by its process makes a vector representation for the sentence. To do that it apply the convolution of the initial text $\boldsymbol{X} \in \mathrm{R}^{m \times n \times 1}$ by a kernel $\boldsymbol{W} \in \mathrm{R}^{v \times v \times 1}$ dimensions, resulting in a different representation of the information $\boldsymbol{H} \in \mathrm{R}^{m-v+1 \times n-h+1 \times 1}$, as is shown in Equation 2.3.

$$\boldsymbol{H}(i,j,d) = conv(\boldsymbol{X}, \boldsymbol{W}_d)(i,j) \sum_{f=1}^{v} \sum_{g=1}^{v} \boldsymbol{X}(i+f, j+g)\, \boldsymbol{W}_d(f,g) \tag{2.3}$$

2.2.4 Recurrent Neural Networks (RNN)

The RNN is a neural network that works with sequential data since the output depends on the previous and present states. This kind of neural network has a parameter called hidden state that has information about the previous inputs of the sequence. The network's output is calculated whit the current input and the previous hidden state, allowing the model to work with the information of all the data entered to that point. In Figure 2.5, the representation of an RNN is shown, the upper image represents the hidden state as a loop, and the bottom image is presented the sequential interpretation of the network.

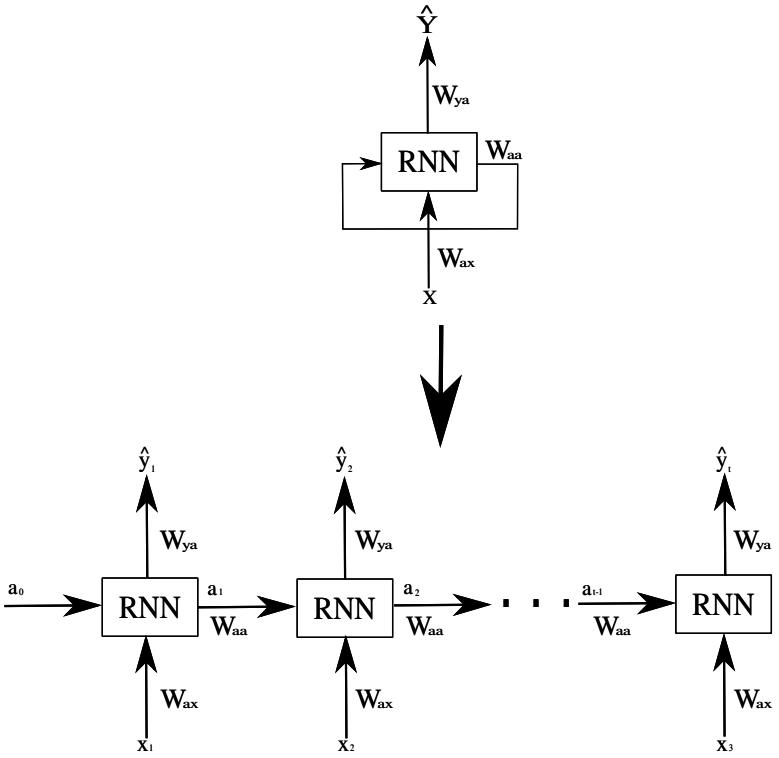

Figure 2.5. Recurrent Neural Network

The parameters of the networks are computed in the following way.

The first hidden state $\boldsymbol{a}$ is 0, and the next is calculate with the input and the previous hidden state according to Equation 2.4.

$$\boldsymbol{a}_t = g(\boldsymbol{W}_{aa}\boldsymbol{a}_{t-1} + \boldsymbol{W}_{ax}\boldsymbol{x}_t + \boldsymbol{b}_a) \tag{2.4}$$

Where g is the activation function, $\boldsymbol{a}_t$ is the currently hidden state, $\boldsymbol{x}_t$ is the currently input, $\boldsymbol{b}_a$ is the bias of the network and $\boldsymbol{W}$ are the matrix of weights for each parameter.

And the output $\boldsymbol{y}$ of the network is:

$$\boldsymbol{y}_t = g(\boldsymbol{W}_{ya}\boldsymbol{a}_t + \boldsymbol{b}_y) \tag{2.5}$$

Due to the way to save the information of the previous inputs with the hidden state, a problem appears called vanishing gradient. The problem describes the loss of information with each step of the network; the information for the early inputs vanishes with each new input. Therefore, the networks do not work well with long sequences. In order to mitigate this vanishing gradient problem, it was developed a recurrent neural network cell called LSTM.

Long-Short Term Memory (LSTM): The LSTM cell tries to remember information for a long time. To do that, the LSTM includes three new concepts. The input gate $\boldsymbol{I}$ aims to determine what new information should be added to the network status state. The forget gate $\boldsymbol{F}$ decides what information to keep for the long term and what information to forget from the status state. Finally, the status gate $\boldsymbol{O}$ decides the new hidden state as a combination of the previous hidden state, the new input, and the status state. All those gates regulate the hidden state $\boldsymbol{C}$, to keep the information longer than the traditional RNNs. The architecture of an LSTM cell is shown in the Figure 2.6

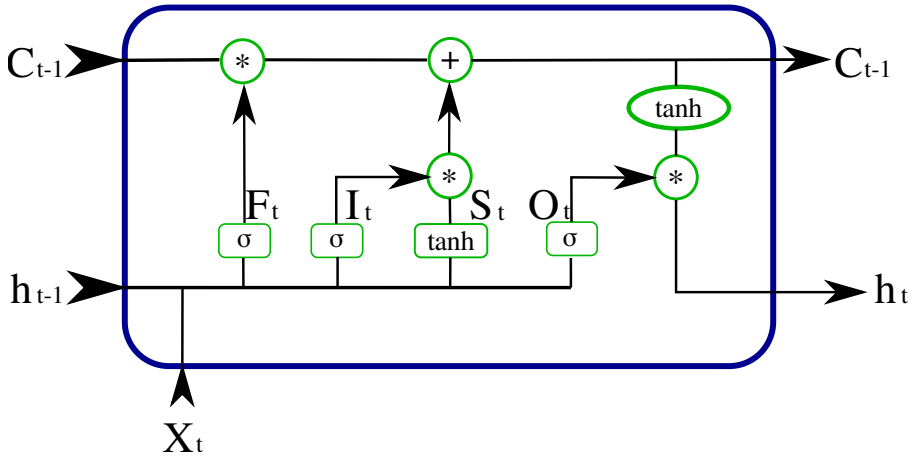

Figure 2.6. Long-Short Term Memory cell

The gates $\boldsymbol{I}$, $\boldsymbol{F}$, $\boldsymbol{S}$, $\boldsymbol{O}$, the hidden state $\boldsymbol{C}$ and the output $\boldsymbol{h}$ in the moment t, are calculated in the following way.

$$\boldsymbol{I}_t = \sigma(\boldsymbol{W}_{ih}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{ix}\boldsymbol{x}_t + \boldsymbol{b}_i) \tag{2.6}$$

$$\boldsymbol{F}_t = \sigma(\boldsymbol{W}_{fh}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{fx}\boldsymbol{x}_t + \boldsymbol{b}_f) \tag{2.7}$$

$$\boldsymbol{O}_t = \sigma(\boldsymbol{W}_{oh}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{ox}\boldsymbol{x}_t + \boldsymbol{b}_o) \tag{2.8}$$

$$\boldsymbol{S}_t = tanh(\boldsymbol{W}_{sh}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{sx}\boldsymbol{x}_t + \boldsymbol{b}_s) \tag{2.9}$$

$$\boldsymbol{C}_t = \boldsymbol{I}_t * \boldsymbol{S}_t + \boldsymbol{F}_t * \boldsymbol{C}_t \tag{2.10}$$

$$\boldsymbol{h}_t = \boldsymbol{O}_t * tanh(\boldsymbol{C}_t) \tag{2.11}$$

Where $\boldsymbol{W}_{jx}$ and $\boldsymbol{W}_{jh}$ are the weights for each gate in the LSTM cell, these weights control the connection between the input $\boldsymbol{x}$ and the previous state $\boldsymbol{h}$, respectively. Also, there are biases for each gate in the cell; they are called $\boldsymbol{b}_j$. Since the biases and the weights are present in each gate, $j = i, f, o, s$, where i, f, o and s are the input, forget, output, and status gates, respectively.

Since this cell can work with information for a longer time, it allows working with larger data sequences. This feature is helpful for NER because it can analyze sentences with more words than the conventional RNN.

2.2.5 Transformers

Due to the vanishing gradient problem of the RNN and its difficulty in parallelizing the training process, in 2017 it was developed a new neural architecture called transformers [61]. This architecture introduces two important elements in its operation: positional encoding and attention mechanism. The main advantage of this architecture is that all words are analyzed jointly. This characteristic allows parallel processing and considers the context of the word in both directions with sequential information but not in a sequential way, solving the vanishing gradient problem.

The positional encoding is a process of the transformers that gives information about the position of the values entered into the network. The idea is to add a fixed encoding vector to the embedding vector of each word, such that the model can learn the positional information in addition to the semantic information. The positional encoding vectors are calculated based on the position of the word in the sequence, by the following equations:

$$PE_{(pos,2i)} = \sin(\frac{pos}{10000^{2i/d_{model}}}) \tag{2.12}$$

$$PE_{(pos,2i+1)} = \cos(\frac{pos}{10000^{2i/d_{model}}}) \tag{2.13}$$

Where pos is the position of the word in the sequence, i is the index of each component of the embedding vector, and d is the dimensionality of the embedding vector. The constant value 10000 and the exponent $2i/d$ are used to ensure that the values of the positional encoding are within a similar range as the embeddings. Equation 2.12 is used to the odd components of the embedding vector, and Equation 2.13 for the even components.

In this way, the positional encoding vector is added to the word vector to incorporate information about the position of the words in the sequence. This process is helpful for sequence models where the position matters to understanding the data. Finally, the attention mechanism, specifically the self-attention mechanism, aims to find the relation between a word and the other words composing the sentence.

This architecture was designed for translation tasks, where the encoder represents the word in the original language, and the decoder is used to find the translation. In other NLP applications, the encoder generates word representation that preserves contextual information. Therefore, in this type of representation, as in natural language, a word can have multiple representations or meanings depending on its context. In the Figure 2.7 it is shown the encoder architecture of a transformer.

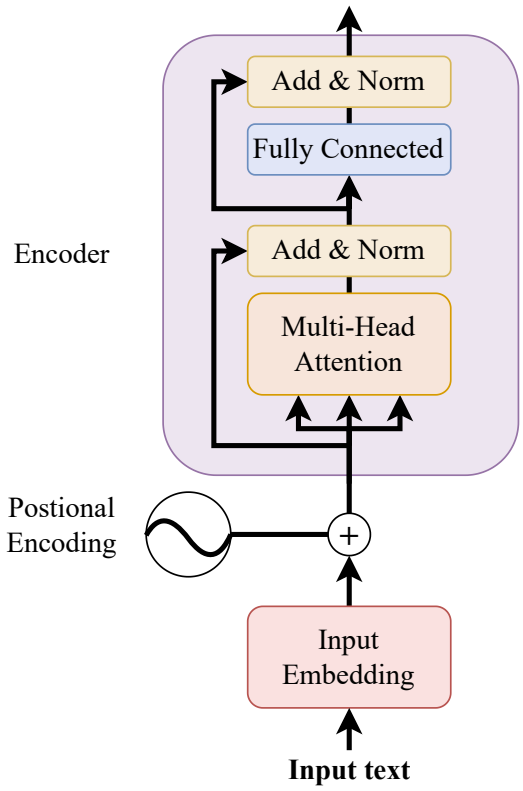

Figure 2.7. Transformer's encoder. Image adapted from [61]

Since NER is a task that depends on the sentence length for the context, thus the vanishing gradient problem was a significant problem to solve. The transformer becomes an excellent option for NER [11] thanks to the attention mechanism. One of the most popular based-transformer model is called Bidirectional Encoder Representations from Transformers (BERT) [29] and has demonstrated a great capability for NLP tasks.

RoBERTa: [66] is a pre-trained NLP model that was developed as an

improvement to the BERT model. This model includes modifications in its masking and optimization mechanisms, which have led to significant improvements in its performance. RoBERTa is composed of 24 Transformer blocks, which are made up of 16 self-attention heads and a fully connected layer containing 1024 units. The key advantage of RoBERTa is its ability to work with multiple languages. This model was pre-trained with 2.5 TB of text data from the Common Crawl corpus [67], which includes text in 100 different languages. This extensive pre-training has allowed RoBERTa to compute text representations in multiple languages, making it a valuable tool for multilingual NLP tasks. The output of RoBERTa is a 1024-dimensional vector for each word in the sentence input.

2.2.6 Loss functions

The objective of this study is to investigate the impact of cost functions in training models on challenging data. To achieve this goal, we examine various approaches for loss functions, including two proposed functions called Sigmoid Point-Wise and Sigmoid-Exponential Point-Wise, as well as the Focal Loss method introduced in [68]. We compare these methods with the commonly used Cross-Entropy loss.

For the functions addressed we define $\mathbf{Y_t} = \{ \mathbf{y}_{t0}, \mathbf{y}_{t1}, ..., \mathbf{y}_{tl}\}$ as the true labels of a sentence of length L, and a sequence of predicted labels $\mathbf{Y_p} = \{ \mathbf{y}_{p0}, \mathbf{y}_{p1}, ..., \mathbf{y}_{pl}\}$ of the same length. Where $\mathbf{y}_{tl}$ and $\mathbf{y}_{pl}$ are the true and the predicted label for the token l, respectively. The vector $\mathbf{y}_{tl}$ is one-hot encoded for N entities. Finally, $\mathbf{y}_{pl}$ is the output of the fully connected linear layer with the same shape of $\mathbf{y}_{tl}$.

The loss functions are described as follows:

Cross-Entropy Loss (CE): The Cross-Entropy loss function is widely used in Deep Learning due to its high performance in many scenarios. However, this function treats all classes equally, as shown in equation 2.14, which can result in issues when working with imbalanced datasets. When one class has significantly more samples than others, the Cross-Entropy loss gives more weight to that class, while classes with fewer samples are disregarded. This problem can be addressed by using alternative loss functions that take into account the class imbalance, such as the ones explored in this study.

$$\text{CE} = -\sum_{l=0}^{L} \log\left(y_{pl} \cdot y_{tl}\right) \tag{2.14}$$

Due to the CE loss function impartiality, we want the model to focus on classes with few samples. Therefore, we define the following two loss functions that control the importance of a class in the loss function by the parameter c.

Sigmoid Point-Wise: The purpose of this function is to adjust the penalty for errors on specific classes using the sigmoid function. The value of the parameter c_i determines the slope of the sigmoid function for a given class i, resulting in a softer or harder penalty for errors. As shown in Equation 2.15, a higher value of c_i increases the penalty for errors in class i, enabling customization of the penalty for each class. Therefore, the function offers flexibility in adjusting the penalty for errors on a per-class basis. The function is defined as follows:

$$\text{S}_1 = -\frac{1}{L}\sum_{l=0}^{L} \frac{1}{1 + e^{-c_i(y_{pl} \cdot y_{tl})}} \tag{2.15}$$

Where c_i is the c coefficient for the class of the true label i, with $i = 1, 2, ...N$.

Sigmoid-Exponential Point-Wise: Like the last one, the goal of this loss function is to modify the severity of the penalty for each class. However, unlike the prior function, it targets the entire sequence and specific tokens. The sigmoid function is then applied to the error, as indicated in equation 2.16. Equation 2.17 shows how to do calculate the error by computing the exponential average error for the entire sequence. Each token's mistake is increased as a result, and the penalty is controlled by the control parameter c_i.. As a result, this loss function focuses on both the entire sequence (during the final phase) and each token individually (during the exponential stage).

$$\text{S}_2 = -\frac{1}{1 + e^{-x}} \tag{2.16}$$

Where,

$$x = \frac{1}{Le}\sum_{l=0}^{L} e^{c_i(y_{pl} \cdot y_{tl})} \tag{2.17}$$

Focal Loss: It is a function introduced in [68], which adds a focusing parameter γ to the CE loss. The parameter γ "focus" the loss function on the

misclassified samples giving them more importance in the loss function value. The bigger γ, the more focus there is on the mistakes. With $\gamma = 0$ there is no focus, i.e., the CE Loss. The equation 2.18 shows that the penalty parameter is multiplied by the CE loss function. The larger the error, the larger the penalty.

$$\mathrm{FL} = -\sum_{l=0}^{L} \left(1 - (y_{pl} \cdot y_{tl})\right)^{\gamma} \log\left(y_{pl} \cdot y_{tl}\right) \tag{2.18}$$

2.2.7 Data augmentation

The number of tokens is a crucial factor in working with data, and therefore, techniques to enhance data quality are being studied. The objective is to explore upsampling data techniques that generate more training samples and reduce semantic similarity between entities. Four upsampling methods focused on Named Entity Recognition (NER) were implemented, as presented in [69]: Label-wise Token Replacement (LwTR), Shuffle within Segments (SiS), Synonym Replacement (SR), and Mention Replacement (MR). Additionally, a new method called Mention Back Translation (MBT) is proposed, inspired by the work of [70]. Table Table 2.1 provides an example of each method. The details of all the methods are explained below.

Label-wise Token replacement (LwTR): The sentence to be upsampled is traversed token by token. For each token, we decide if the method is applied using a Bernoulli distribution with probability p. If yes, we randomly select another token in the dataset with the same label, replace the token, and create a new sentence. Since the replacement is token-wise, the label vector is not modified.

Shuffle within segments (SiS): This approach involves dividing the input sentence into separate segments or mentions based on their entity types. Similar to the previous technique, a Bernoulli distribution is employed to determine the application of the method with probability p. If the method is applied, the tokens within the segment are shuffled, while the label vector remains unchanged.

Synonym Replacement (SR): This method is similar to LwTR, except that the replacement of the token is not from another token but is made with a synonym obtained from "wordreference" [1]. Moreover, for the same reason

[1] https://www.wordreference.com

that LwTR, the label vector is not modified.

Mention Replacement (MR): As SiS, the sentence is broken up into sections. We decide whether to replace each segment using a Bernoulli distribution with probability p. If so, a new sentence is created by replacing the segment with a different mention from the dataset that has the same label. Naturally, the new mention might have a different amount of tokens, in which case the label vector might need to be adjusted in accordance with the new mention's tokens.

Mention Back Translation (MBT): For this method, each sentence is also divided into segments, and the selection process is similar to the previous methods using a Bernoulli distribution with probability p to determine if a segment should be modified. However, instead of using a simple replacement technique, a mention is translated into a randomly selected language from a set of languages including English, Swedish, French, Japanese, Korean, Afrikaans, Albanian, Czech, German (Spanish for German database), Greek, and Gaelic. The translated mention is then back-translated to the original language. The translated version may not have the same number of tokens, so the label vector is modified accordingly to match the new mention.

Table 2.1. Example of data augmentation methods. In blue the tokens changed according to the method

Method	Sentence
Original	Company SODA GLACIER S.A.S, MAIN ACTIVITY manufacture and distribution of sweetened beverages
LwTR	Company PARK GLACIER S.A.S, MAIN ACTIVITY manufacture and food of sweetened beverages
SiS	Company GLACIER SODA S.A.S, MAIN ACTIVITY distribution beverages and manufacture sweetened of
SR	Company SODA GLACIER S.A.S, PRINCIPAL ACTIVITY manufacture and diffusion of sweetened beverages
MR	Company PARK FOOD LTDA., MAIN ACTIVITY manufacture and distribution of sweetened beverages
MBT	Company SODA GLACIER S.A.S, MAIN ACTIVITY production and distribution of sweetened beverages

2.3 Software

To implement the software tool we use the Programming Language Python. In this study different libraries and documents formats were used. All of them are described as follows:

2.3.1 Libraries

Flair [71]: Flair [2] is a framework for NLP developed by Humboldt University of Berlin. The framework allows to implement different NLP models for task as NER, Question Answering, Part of Speech Tagging, Sentiment Analysis among others. Its structure gives several objects and methods that facilitates the use of different NLP models. In this work we use this framework for the NER task.

Pytorch: PyTorch [3] is an open-source machine learning library for Python that allows developers to easily create deep learning models and build sophisticated machine learning applications. It provides two high-level features: tensor computation with strong GPU acceleration support and deep neural networks built. PyTorch has a dynamic computational graph, which means that the graph is created on the fly as the code is executed. This makes it very flexible and easy to use, allowing for dynamic, on-the-fly adjustments during training. PyTorch is widely used in research and industry for a variety of tasks such as computer vision, natural language processing, and deep reinforcement learning. We use Pytorch in this study to implement models for RC.

Gradio: Gradio [4] is a Python library that enables the creation of customizable and interactive UI components for machine learning models. With Gradio, developers and data scientists can build a simple user interface to showcase their models in a web-based application, allowing users to interact with the model by inputting data and viewing the output. Gradio is designed to be easy to use and flexible, with support for a wide range of machine learning frameworks and models. It can also be used to create multi-model and ensemble model applications, as well as to build interactive dashboards and data exploration tools.

2.3.2 Formats

PRATEC: The PRATEC format consists of a JSON file that contains a document and its entities in a given context. The format structure is shown in Figure 2.8 and described as follows: It has 3 objects, namely: text, sentences, and mentions.

[2] https://github.com/flairNLP/flair
[3] https://pytorch.org
[4] https://gradio.app

The ***text*** object contains the plain text of the whole document. The ***sentences*** object has a list of blocks as its value. Each block in the list refers to a sentence in the document and is composed of the following 3 objects:

- *text*: contains the plain text of the sentence.
- *id*: contains an identifier for the position of the sentence within the document, in the form of s#.
- *tokens*: has a list of blocks as its value. Each block in the list refers to a token in the sentence and is composed of: (1) an object named "text" that contains the plain text of the token, (2) an object named "begin" that contains the character-level position of the first character of the token, and (3) an object named "end" that contains the character-level position of the last character of the token.

The ***mentions*** object has a list of blocks as its value. Each block refers to each mention of each entity found in the document, and is composed of the following objects:

- *id*: contains an identifier for the sentence number and the mention number to which it refers, in the form of s#-m#.
- *type*: contains the name of the entity.
- *word*: contains the text of the token.
- *begin*: contains the character-level position of the first character of the token.
- *end*: contains the character-level position of the last character of the token.

```
{
  text: "Frase ejemplo Frase ejemplo",
  sentences: [
    0: {
      text: "Frase ejemplo",
      id: "s0",
      tokens: [
        0: {
          text: "Frase",
          begin: 0,
          end: 5
        } ,
        1: {
          text: "ejemplo",
          begin: 6,
          end: 13
        }
      ]
    } ,
    1: {
      text: "Frase ejemplo",
      id: "s1",
      tokens: [
        0: {
          text: "Frase",
          begin: 14,
          end: 19
        } ,
        1: {
          text: "ejemplo",
          begin: 20,
          end: 27
        }
      ]
    }
  ],
  mentions: [
    0: {
      id: "s0-m0",
      type: "Entity_type",
      begin: 0,
      end: 5
    } ,
    1: {
      id: "s1-m0",
      type: "Entity_type",
      begin: 14,
      end: 19
    }
  ]
}
```


Figure 2.8. PRATEC format.

CONLL03: The CONLL03 format consists of a text document where several sentences are found. Each sentence is composed of multiple tokens. For the format, each line of the document corresponds to a token and its respective entity separated by a space. The separation between sentences is given by a line break. This is illustrated in Figure 2.9

```
Word Entity_Type
Jorge PER
Washington PER
went O
to O
Washington LOC

Word Entity_Type
...
```

Figure 2.9. CONLL03 format.

CONLL04: The CONLL04 format consists of a text document with several sentences for RC. Each sentence is composed of multiple tokens. The composition is similar to the CONLL03 format, but at the top of a sentence is added the relation of the sentence. This is illustrated in Figure 2.10

```
Word Entity_Type Relation
The O Cause-Effect
diseases Effect -
are O -
caused O -
by O -
gene Cause -
mutations Cause -
on O -
the O -
X O -
chromosome O -

Word Entity_Type Relation
...
```

Figure 2.10. CONLL04 format.

Chapter 3

Data

As part of the framework of this study, we utilize various types of datasets to simulate different scenarios, including ideal data, real-world data, and extremely poor real-world data. Thus, we introduce the following datasets:

3.1 CoNLL-02 [24]

In this study, we focused on the training set of the CoNLL-2002 Spanish corpus. The Spanish data is a collection of news wire articles made available by the Spanish EFE News Agency. The articles are from May 2000. The annotation was carried out by the TALP Research Center of the Technical University of Catalonia (UPC) and the Center of Language and Computation (CLiC3) of the University of Barcelona (UB). The corpus includes over 210,000 tokens of general topics, grouped into four primary entity types: Organization (ORG), Person (PER), Location (LOC), and Miscellaneous (MISC). Among these entity types, Miscellaneous has the lowest number of tokens, with a total of 5,385. To enable the model's architecture, we limited the maximum sentence length to 125 tokens, which corresponds to the maximum length of 99% of the sentences.

3.2 Legal Entity Recognition (LER) [13]

This corpus comprises 66,723 sentences with 2,157,048 tokens categorized into 19 entity types. The data was taken from 750 German court decisions selected from the Federal Ministry of Justice and Consumer Protection. The documents originate from seven federal courts: Federal Labour Court (BAG),

Federal Fiscal Court (BFH), Federal Court of Justice (BGH), Federal Patent Court (BPatG), Federal Social Court (BSG), Federal Constitutional Court (BVerfG) and Federal Administrative Court (BVerwG). The sizes of the seven court-specific datasets varies between 5,858 and 12,791 sentences, and 177,835 to 404,041 tokens. The corpus includes the entities: Person (PER), Judge(RR), Lawyer (AN), Country (LD), City (ST), Street (STR), Landscape (LDS), Organization (ORG), Company (UN), Institution (INN), Court(GRT), Brand (MRK), Law (GS), Ordinance (VO), EU legal norm (EUN), Regulation (VS), Contract (VT), Court decision (RS), an Legal literature (LIT). The entity type with the lower number of tokens is Law and has 160 tokens of the entity type, which is lower than the CoNLL-02 database. As the CoNLL-02 the corpus was modified because some sentences were composed by a large number of tokens (approximately 1000), therefore all the sentences were limited to 120 tokens that corresponds to the maximum length of the 99% of the sentences. This corpus was provided by PRATEC group SAS.

3.3 Colombian Chamber of Commerce (CCC)

This corpus is a collection of registration documents from the Colombian Chamber of Commerce. The certificates contain entity details including the name of the company, its main business, the date of registration, and other pertinent information. The data set is composed by 126,198 tokens make up the corpus and includes the following entity types: Activity (ACT), Date (FCH), Person (PER), Organization (ORG), Limitations (LIM), Type of document of identy (TDID), Title registration date (TFMT), Identity document (DID), Title principal activity (TACTP), Title name or business name (TNRS), Legal representative position (CRL), Title secondary activity (TACT), Title other activity (TACTO), Organization type (TORG), Substitute position (CAS), Title state registration (TEMT), Old organization type (TORGA), Title effective date (TFV), Title limitations (TLIM), Title organization type (TTORG), State of registration (EMT). Four tokens are the minimum required for each entity type, which is fewer than the number found in the LER database. In addition, when compared to the LER database, the corpus exhibits a greater degree of imbalance and fewer tokens.

3.4 Multilingual Anonymisation for Public Administrations (MAPA) [17]

The corpus used in this study comprises of 12 court decision documents and legal dispositions in Spanish language. The corpus has been annotated manually by the MAPA project and contains the entity types: Address (ADD) , Age (AGE), Amount (AMO), Date (FCH), Ethnic category (ETC), Marital status (MRS), Organization (ORG) , person (PER), Type (TYPE). It has a total of 89,414 tokens, and the entity type with the lowest number of tokens contains only one token. Compared to the other databases used in this study, this corpus has the lowest number of tokens per entity type and the highest imbalance.

3.5 SemEval 2010 task 8 [72]

The SemEval 2010 task 8 dataset is a benchmark dataset used for evaluating relation extraction methods. It was first made available as a shared job for SemEval 2010's multi-way classification of semantic relations between pairs of textual elements. The dataset consists of a collection of sentences taken from news articles. Its aim is to determine the semantic link between each sentence's two elements. Numerous relation types are included in the dataset, Table 3.1 summarizes the relationships present in the dataset and provides some examples where the entities of interest are underlined. The dataset is composed by sentences with two entity mentions that have a semantic relation between them. The SemEval 2010 Task 8 dataset is a useful tool for developing and testing relation extraction models, and it is frequently utilized by researchers to progress the NLP discipline.

Table 3.1. Relations of the SemEval 2010 task 8 dataset

Relation	**Definition**	**Example**
Product-Producer	A producer causes a product to exist.	The *sugar factory* in Ipswich was built in 1925.
Cause-Effect	An event or object leads to an effect.	The *burst* has been caused by water hammer *pressure*.
Content-Container	An object is physically stored in a delineated area of space.	The *key* was in a *chest*.
Component-Whole	An object is a component of a larger whole.	The *introduction* in the *book* is a summary of what is in the text.
Entity-Destination	An entity is moving towards a destination.	*People* have been moving back into *downtown*.
Instrument-Agency	An agent uses an instrument.	A *telescope* assists the *eye* chiefly in two ways.
Entity-Origin	An entity is coming or is derived from an origin.	The *staff* was removed from his *position*.
Message-Topic	A message, written or spoken, is about a topic.	The *chapters* in this book investigate *issues* and communities.
Member-Collection	A member forms a nonfunctional part of a collection.	*Fish* swim in a *shoal*, but they fall one by one.

Chapter 4

Experiments and results

4.1 Problem's identification and architecture selection

In this study, we conducted an extensive investigation of the real-world data (CCC dataset) and ideal data (LER dataset) with the goal of improving the performance of Named Entity Recognition (NER) models for the real scenarios. We evaluated various benchmark architectures on these datasets and compared the results of the real-world data with those obtained on ideal data (German dataset). Through different analysis, we identified three key challenges that impact NER performance in real-world scenarios: token quantity, average length, and semantic similarity of entities. These challenges present formidable obstacles to achieving optimal performance, and thus efficient solutions need to be developed to enhance the performance of NER models in practical applications. Our study provides valuable insights into these challenges and underscores the need for further research in this area.

4.1.1 Data characterization

The LER dataset was consider as ideal data because it has a lot of samples for each class. Also, the CCC dataset is used as real-world data due to the lower number of samples and great class unbalance. A statistical detailed metrics are obtained for both datasets, LER is resumed in Table 4.1 and CCC in Table 4.2. Moreover, Figure 4.1 shows that the length of the sentences for this corpus is shorter and more unbalanced than the ones in the German corpus.

Table 4.1. Details of the data in the German (LER) dataset.

Entity type	Acronym	Tokens	Mentions	Tokens per mention
Court decision	RS	194601	12846	15.15 ± 6.19
Law	GS	116934	18536	6.31 ± 4.23
Legal literature	LIT	38119	3085	12.36 ± 5.88
Contract	VT	15227	2864	5.32 ± 4.41
European legal norm	EUN	12520	1501	8.34 ± 9.71
Institution	INN	6872	2196	3.13 ± 2.63
Court	GRT	5981	3212	1.86 ± 1.58
Ordinance	VO	5238	796	6.58 ± 7.68
Regulation	VS	4702	610	7.71 ± 7.90
Organization	ORG	2771	1164	2.38 ± 1.56
Company	UN	2384	1058	2.25 ± 1.42
Person	PER	2262	1746	1.29 ± 0.65
Country	LD	1752	1429	1.23 ± 0.49
Judge	RR	1625	1519	1.07 ± 0.25
City	ST	772	704	1.10 ± 0.35
Brand	MRK	666	283	2.35 ± 1.38
Street	STR	265	136	1.95 ± 0.91
Landscape	LDS	231	198	1.17 ± 0.54
Lawyer	AN	160	111	1.44 ± 0.53

Table 4.2. Distribution of the classes in the Spanish (CCC) dataset.

Entity type	Acronym	Tokens	Mentions	Tokens per mention
Activity	ACT	4520	417	10.84 ± 6.20
Date	FCH	3704	1050	3.52 ± 1.85
Person	PER	1295	353	3.67 ± 1.17
Organization	ORG	846	261	3.24 ± 1.21
Limitations	LIM	615	11	55.91 ± 27.76
Type of document of identy	TDID	374	301	1.24 ± 0.64
Title registration date	TFMT	371	718	2.10 ± 1.11
Identity document	DID	326	288	1.13 ± 0.41
Title principal activity	TACTP	317	159	1.99 ± 0.21
Title name or business name	TNRS	259	103	2.51 ± 0.86
Legal representative position	CRL	254	123	2.06 ± 0.40
Title secondary activity	TACTS	171	86	2 ± 0.11
Title other activity	TACTO	166	83	2 ± 0
Organization type	TORG	91	23	3.96 ± 0.46
Substitute position	CAS	67	39	1.72 ± 0.85
Title state registration	TEMT	13	7	1.86 ± 0.64
Old organization type	TORGA	11	4	2.75 ± 1.30
Title effective date	TFV	9	6	1.50 ± 1.12
Title limitations	TLIM	5	4	1.25 ± 0.43
Title organization type	TTORG	4	4	1 ± 0
State of registration	EMT	4	2	2 ± 0

4.1.2 Benchmark architectures

The initial stage of implementing NER involves evaluating benchmark models that have demonstrated promising results in the task. Therefore, we consider different methodologies for NER, based on two main paradigms: (1) Transformers and (2) Conditional Random Fields (CRF) combined with different DL models for feature extraction. These approaches have shown high

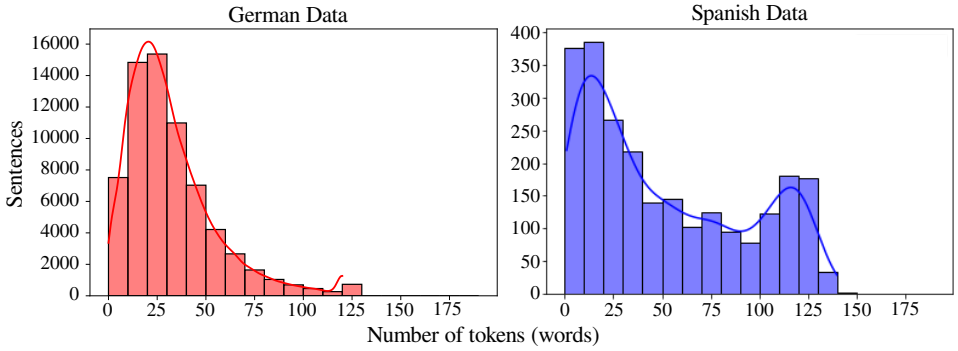

Figure 4.1. Number of tokens (words) per sentence for German (left) and Spanish (right) corpus. Taken from [73].

performance for NER in different benchmark databases [8], [11], [13]. These approaches have become popular because of the development of frameworks such as FLAIR [71] that allows a fast implementation of models based on Transformers, and UKPLab-BiLSTM [74] that simplifies the building of architectures using BiLSTM and CRF techniques.

FLERT: Different models based on Transformers have been proposed for NER. **FLERT** is one of them and it was introduced in [11], which has demonstrated a high capability to recognize named entities [11], [75], [76]. The model computes word representations using a multilingual RoBERTa architecture. The model was pre-trained with 2.5 TB of data from the cleaned Common Crawl corpus [67], which includes text data in 100 different languages and allows for computing text representations in multiple languages. Embeddings from RoBERTa feed a FCNN with an input and output shape of 1024 and the number of classes, recpectively. The FCNN classifies the entities using a linear activation function. In this work, we freeze the RoBERTa model and finetuned the FCNN modified the output layer according to the considered corpus (German and Spanish).

CRF: CRFs are stochastic models used to label and segment sequential data and also to extract information from documents [77]. We implemented and compared four methods based on CRF, considering different feature extraction approaches: (I) CRF with features (**CRF-F**), where words are represented according to the following questions: (1) Are all characters uppercase?, (2) Are all characters lowercase?, (3) Is the first character uppercase and the rest lowercase?, (4) Are all characters digits?, (5) Are all characters letters?, (6) Are all characters numbers? and (7) Is the word a title? or all the characters

are lowercase/uppercase?. Each question represents a feature to be classified by the CRF. In addition, the feature extraction process considers four context words, two on each side, i.e., a context window of five words. (II) The second approach is called **BiLSTM-CRF** because it uses word embeddings extracted from a word2vec model to feed BiLSTM layer before the CRF classifier. The BiLSTM-CRF model allows finding semantic features from the embeddings by itself. (III) The third model extracts a character embedding via an LSTM layer and concatenates the word embeddings from word2vec with the character embeddings before the BiLSTM layer from the previous model, forming a model known as **LSTM-BiLSTM-CRF**. (IV) Finally, we considered a **CNN-BiLSTM-CRF** model, which is similar to the LSTM-BiLSTM-CRF one but replaces the LSTM layer with a CNN.

For models that utilize deep layers, namely models II, III, and IV, pre-trained word2vec models based on the skip-gram architecture ere used to represent words. The pre-trained model for the German data was introduced in [78], which was trained on 116 million sentences and 648,462 tokens from various German corpus. Meanwhile, the pre-trained model from [79] was employed for the Spanish corpus. This model was pre-trained with 1.4 billion words extracted from the Spanish Billion Word Corpus, and contains 1,000,653 tokens.

In the experiments, we maintained the original setup of the methods. The FLERT-based Transformer model was fine-tuned with 20 epochs, a learning rate of 5×10^{-6}, and a context window of 128 words, in accordance with the configuration specified in [11]. For the BiLSTM-CRF, LSTM-BiLSTM-CRF, and CNN-BiLSTM-CRF models, we used two BiLSTM layers with a dropout rate of 0.25 and 100 units for the hidden layer, as described in [80]. Furthermore, we employed an early-stopping strategy to halt the training when the F1-score failed to improve for five consecutive epochs.The validation process consisted of a cross-validation strategy for both corpora. The number of folds was made based on the number of available samples in each corpus, 10 folds for the German dataset and 5 folds for the Spanish dataset.

4.1.3 Comparison

Figure 4.2 shows the F1-scores obtained for both German and Spanish corpus. The left radar chart shows the ideal data (German dataset); high F1-scores are achieved for the 19 entities addressed in this dataset. In addition, Table Ta-

ble 4.3 presents the results of each method, where F1-scores of up to 0.94 are obtained using the FLERT model.

In contrast, details of the average results obtained when working with real-world data (Spanish dataset) are reported in Table Table 4.4. Notice that all models achieved lower F1-scores than those obtained with the German dataset. The right side of Figure 4.2 shows the F1-scores obtained in each entity type for the CCC dataset. Note that entity types such as ACT, FCH, PER, ORG, and others achieve F1-scores between 0.75 and 0.8; however, other entity types like LIM, TORGA, TFV, TLIM, and others yield results close to 0. We observed a similar behavior regardless of the model.

Despite its high computational complexity, the model with higher performance in both datasets was, as expected, the model based on transformers, FLERT. Results shown in Figure Figure 4.2 confirm the fact that a handcrafted dataset allows finding better and consistent results (i.e., the case of the German corpus). Unfortunately, this is not the case when data captured in real environments are considered. In the Spanish corpus, approximately 52% of entity types have an F1-score above 0.75, 14% have an F1-score between 0.4 and 0.5, and 34% have an F1-score close to 0. These results reflect the unbalanced and complex distribution of the entities when real-world data are used.

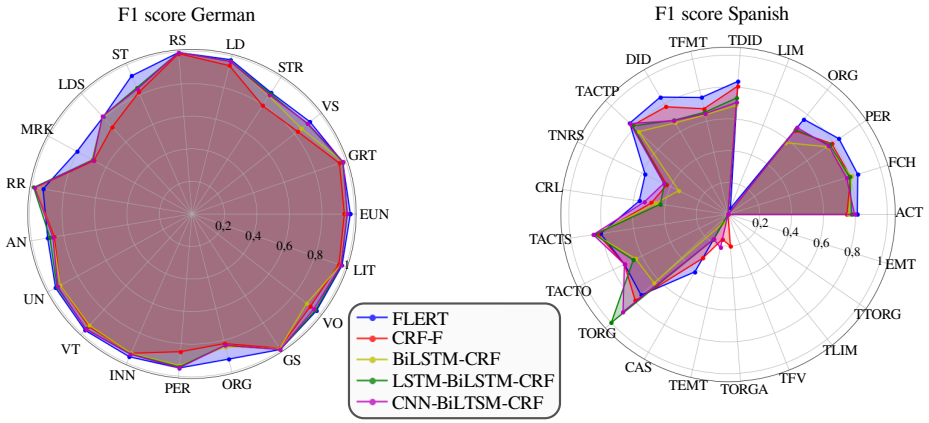

Figure 4.2. F1-scores obtained per entity. (Right) German dataset and (Left) Spanish dataset. Taken from [73].

Table 4.3. Results for German dataset.

Model	Precision	Recall	F1-score
FLERT	**0.94 ± 0.01**	**0.94 ± 00.1**	**0.94 ± 0.01**
CRF	**0.94 ± 0.01**	0.85 ± 0.02	0.89 ± 0.01
BiLSTM-CRF	0.91 ± 0.01	0.91 ± 0.01	0.91 ± 0.01
LSTM-BiLSTM-CRF	0.92 ± 0.01	0.92 ± 0.01	0.92 ± 0.01
CNN-BiLSTM-CRF	0.91 ± 0.01	0.92 ± 0.01	0.92 ± 0

Table 4.4. Results for Spanish dataset.

Model	Precision	Recall	F1-score
FLERT	0.47 ± 0.04	**0.54 ± 0.02**	**0.50 ± 0.02**
CRF	**0.52 ± 0.05**	0.45 ± 0.04	0.48 ± 0.04
BiLSTM-CRF	0.41 ± 0.02	0.42 ± 0.03	0.41 ± 0.03
LSTM-BiLSTM-CRF	0.44 ± 0.01	0.48 ± 0.01	0.45 ± 0
CNN-BiLSTM-CRF	0.45 ± 0.03	0.50 ± 0.03	0.46 ± 0.03

4.1.4 Problems of real-world data

The different models tested in this study yielded similar results in the German dataset according to the average F1-scores reported in Table 4.3. However, the results were not satisfactory for the Spanish corpus, although they were consistent across the models, as shown in Figure 4.2. Despite the low average results obtained in Spanish, we suggest that Transformer-based models are a promising option, given that the FLERT model exhibited superior performance in both datasets.

The results observed with the Spanish dataset are not systematic. Some of the entity types yielded relatively high F1-scores while some others dropped down to almost zero. This behavior was model-independent and could be due to several factors including the reduced number of tokens available to train the model, the average length of the segments, or the semantic similarity between

the tokens in each entity type. Further details about the analysis of these factors are shown below. For the evaluation, we picked the FLERT model along with the Spanish dataset (real-world data)

Number of available tokens: First, we analyze the correlation between the number of tokens and the performance for each entity. Figure 4.3 shows the F1-score for the 10 entity types with the lowest number of available tokens, the remaining entity types are not showed for display purposes. The Spearman's correlation coefficient was computed for all the entity types and the results indicate a strong correlation between the number of available tokens and the F1-score. According to these results, entities with more than 90 tokens can be accurately recognized by the FLERT model, conversely, entities with a small number of tokens are not accurately modeled which limits the recognition of these entities.

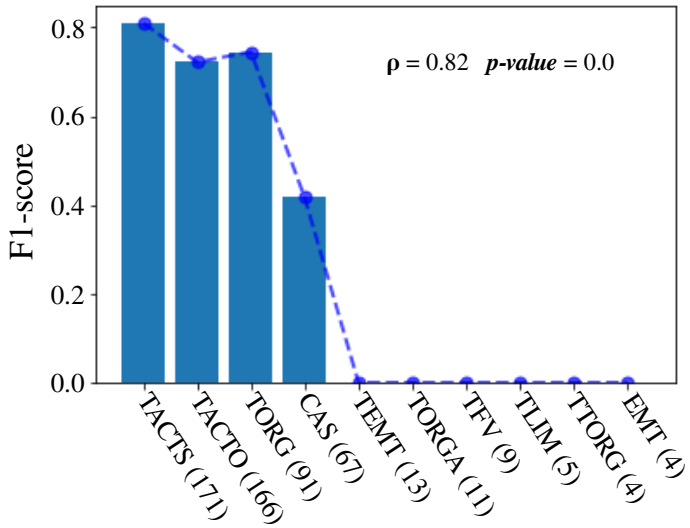

Figure 4.3. F1-score for the 10 entities with the lower number of tokens for Spanish corpus. Taken from [73].

Average length of the segments:

Upon observation, we found that the entity LIM had poor performance despite having a large number of tokens. The average length of segments for a given entity seems to be a factor affecting its performance, where the metric represents the average number of tokens per segment, i.e., how many tokens are on average for each mention of the entity LIM. The top part of Figure 4.4 displays the performance of the model for all entities, along with the average

segment length in terms of token count (bottom). Despite having a high number of tokens, the LIM entity exhibited an average length of 56 tokens and a low F1-score. It appears that a high number of tokens in a segment for a given entity leads to poor model performance due to the failure of existing models to capture long-range dependencies requiring extensive context.

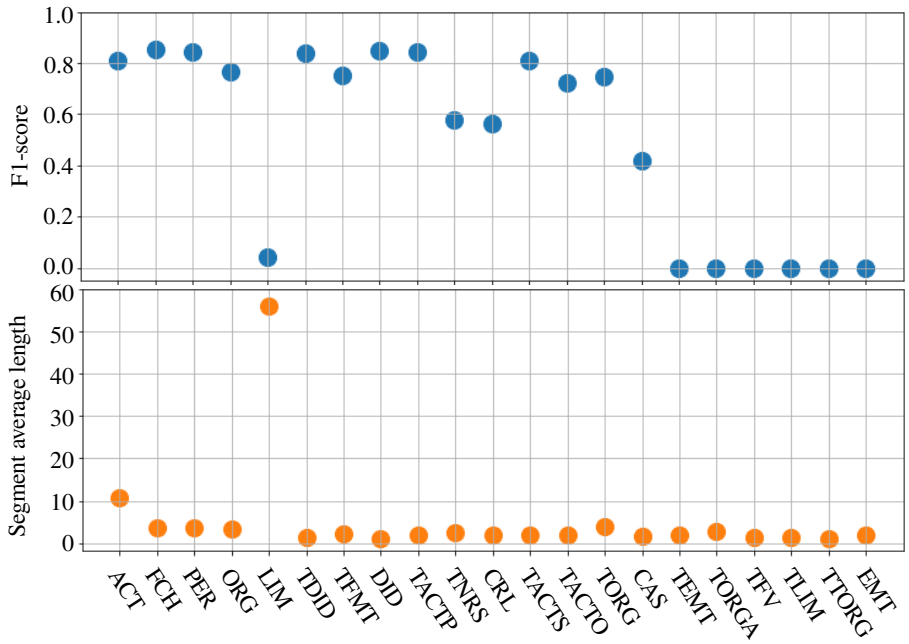

Figure 4.4. Comparison of F1-score (up) and the average length of the segment (bottom) for each entity. The entities are sorted according to the number of tokens. Taken from [73].

Semantic similarity: This factor has to do with the uniqueness of the terms used to name the entities in the corpus. Unfortunately, existing documents like those produced by companies, the court, chambers of commerce, and others, were not thought to generate consistent representations in terms of named entities. Given this fact, what language scientists have done is to associate unknown tokens to a kind of "trash" entity called OTHER. Semantic similarity is defined as the ratio between the number of unique tokens that appear in a given entity and also appear in the entity OTHER with respect to the total number of unique tokens in the given entity. High semantic similarity between an interest entity and the "OTHER" entity is observed when a significant number of words from the interest entity are also present in the "OTHER" entity. This phenomenon is often a result of labeling difficulties,

where there is insufficient information to generate additional entities that are coherent across the corpus documents

Regarding this study, it's worth noting that the entity labeled as "OTHER" comprises a total of 126,198 tokens, while the number of tokens for the remaining entities varies between 4 and 4,520. To conduct a more thorough examination of this problem, we computed the Spearman's correlation between the semantic similarity of each entity and its F1-score (using FLERT). Figure Figure 4.5 shows the analysis results. Notice that entities with higher similarity are the ones with lower F1-score, which suggests that entities in which there were a lot of tokens that also appeared in the OTHER entity, generates the performance of the model to decrease. This finding also suggest that the labeling process of document corpus should be performed paying special attention to the semantic similarity measure. Similar experiments were performed with the German database (not reported here) and we confirmed that all of the entities had semantic similarities of below 0.8 and F1-scores between 0.8 and 1.0, which is in agreement with the results presented in Figure Figure 4.2 (left).

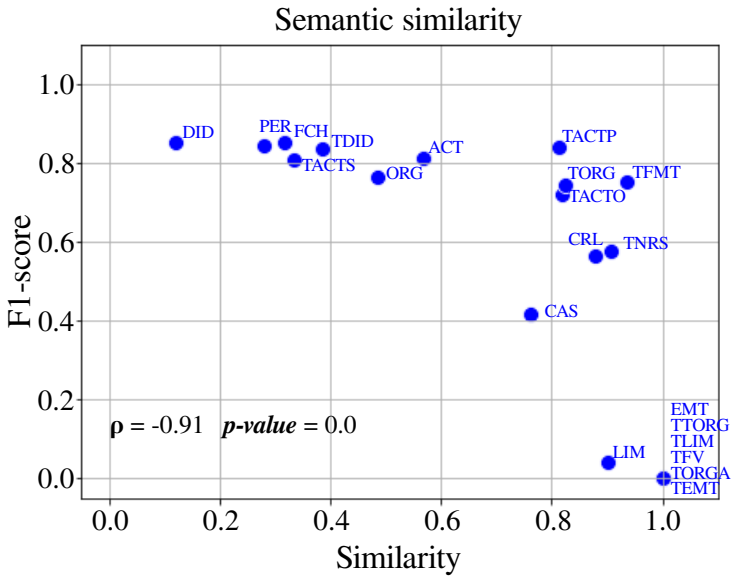

Figure 4.5. Semantic similarity analysis. Taken from [73].

4.2 Methodology against problems

Once the problems are identified, we aim to develop a methodology that faces two of those problems: the low number of tokens and the semantic similarity. Therefore, to attack those problems, a new methodology was developed addressing two approaches, one for the model FLERT, and other for the data. For FLERT we tried different loss functions with the aim of focusing on the entities with the problems. Additionally, since the data were low, we implement a data augmentation step before the training. Then we test the methodology in three different scenarios: one with many and balanced data, other with few and highly unbalanced data, and other with less data and extremely unbalanced classes. Also, we analyze the knowledge generalization capability of the model with the new methodology and the semantic similarity changes.

4.2.1 Loss Functions

We train the FLERT model using the CCC dataset and following a document-independent 5-fold stratify-cross-validation strategy. This experiment was performed using the four loss functions presented in Section 2: Cross-Entropy (CE) as the baseline, Sigmoid Point-Wise (S_1), Sigmoid-Exponential Point-Wise (S_2), and Focal Loss (FL). To implement the S_1 and the S_2 loss functions is mandatory to assign a value of c for each class. Therefore, we divided the classes of the CCC dataset into two sets: The **good set** is formed by the entity types with more than 50 tokens (ACT, FCH, PER, ORG, LIM, TDID, TFMT, DID, TACTP, TNRS, CRL, TACTS, TACTO, TORG), and the **poor set** with the remaining entity types (CAS, TEMT, TORGA, TFV, TLIM, TTORG, EMT). Figure 4.6 summarizes the performance for both good and poor set into a radar chart.

The optimal hyper-parameter values for each loss function and its corresponding F1-scores are presented in Table Table 4.5. The F1-scores obtained by the S_1 and S_2 loss functions were lower than those of the Cross-Entropy function. Nonetheless, the Focal Loss function's ability to concentrate on mislabeled samples resulted in the highest F1-score, outperforming the other loss functions and even surpassing the baseline score of the Cross-Entropy Loss.

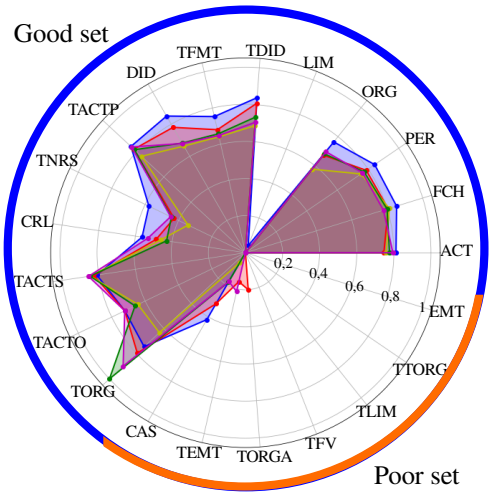

Figure 4.6. Division of the good and poor sets.

Table 4.5. F1-score (%) of FLERT with different loss functions

Loss Function	**Average F1-score**	C good set	C poor set	Gamma
CE	49	-	-	-
S_1	32	0.3	0.3	-
S_2	27	1	0.3	-
FL	**53**	-	-	1.5

4.2.2 Data Augmentation

We implement the five upsampling methods defined in Section 2 and evaluate their performance in three sets. Firstly, we applied each method separately. Secondly, we employed all methods on the dataset without re-upsampling. Lastly, we examined the nested combination of methods where the output of one method was used as input for the next method, and re-upsampling was conducted on the previous method. To minimize external information noise that some methods might introduce, we used a nested upsampling approach and sorted the methods in the following order: SiS, LwTR, MR, SR, and MBT.

Table 4.6 shows the average F1-score for both all and the poor set of entity types. We can note that each method gives an improvement over the original data. However, combining all the methods leads to a higher average F1-score than when each method is used independently. Moreover, the nested combination of all methods yields the best outcomes improving the average F1-score for all entity types. Additionally, the nested combination performs the best results for the poor-set were increments of more than 16 % are demonstrated.

Table 4.6. F1-score (%) of FLERT with different up-sampling methods applied in CCC dataset

Method	Average F1-score	F1-Score of poor set
None	53	6
LwTR	54	21
SiS	54	20
SR	54	17
MR	56	22
MBT	53	17
All	55	20
All nested	**56**	**24**

The methodology that achieved the highest score is the nested combination of methods, which we selected for further analysis. Figure 4.7 presents a comparison between the number of tokens per entity type before (blue bar) and

after (red bar) the upsampling process. The number of tokens has increased for all entity types.

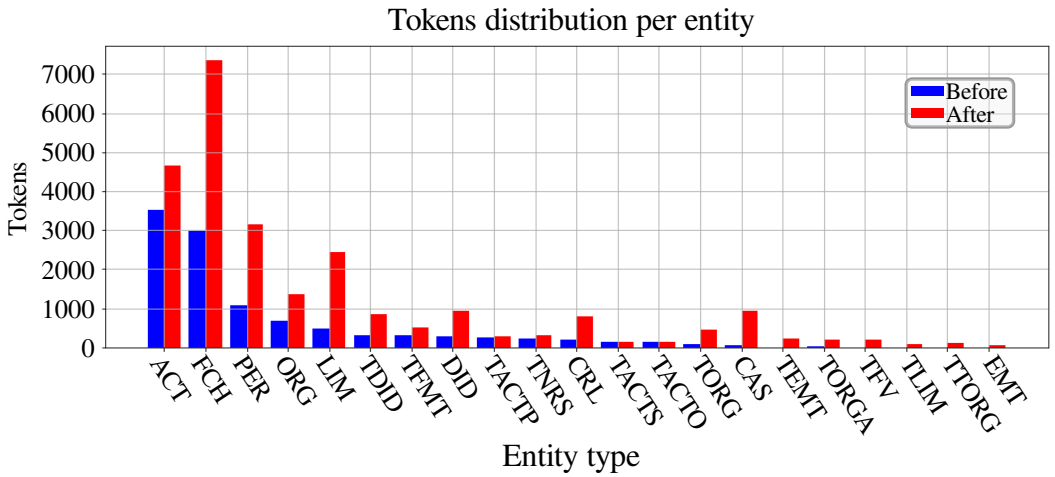

Figure 4.7. Distribution of tokens per entity before and after data augmentation.

4.3 Methodology evaluation

Up to this stage, we have utilized an approach consisting of an architectural aspect that incorporates the Focal Loss function and a data aspect involving various techniques for data augmentation. The following phase entails evaluating the efficacy of this methodology by conducting a range of experiments.

4.3.1 Test in different scenarios

To evaluate the methodology's performance in different scenarios; we perform an experiment with three different scenarios. A dataset represents each scenario; first is an ideal scenario with many samples and balanced classes provided by the LER dataset. Second, the CCC dataset represents a standard scenario with few samples and classes with medium-level imbalance. Finally, the MAPA dataset provides the last scenario and has few samples with extreme class imbalance. We applied two approaches to perform the NER task on the three datasets using FLERT: (1) before implementing the methodology and (2) after implementing the methodology.

The results are presented in Figure 4.8 as a radar graph, where each point on the circle represents an entity type, denoted by its acronym. The three scenarios were evaluated using the FLERT model before and after applying the methodology proposed in this study. In the LER dataset, which has a large amount of balanced data, the methodology did not affect the model's performance when there is already good performance. In the CCC dataset, the model with and without the methodology achieved similar results for the good set of entity types. However, for the poor set of entity types, including those of interest, the methodology significantly improved the performance, with some entity types showing an increase in F1-score of up to 35%. Finally, we tested the methodology on a few samples with an extremely imbalanced class distribution scenario (MAPA dataset), where the entity types of interest were AGE, ETC, MRS, and TYPE. As with the CCC dataset, the model maintained its performance for the entity types with a good score, but for TYPE, the F1-score improved from less than 20% to more than 60%, representing an improvement of more than 40%.

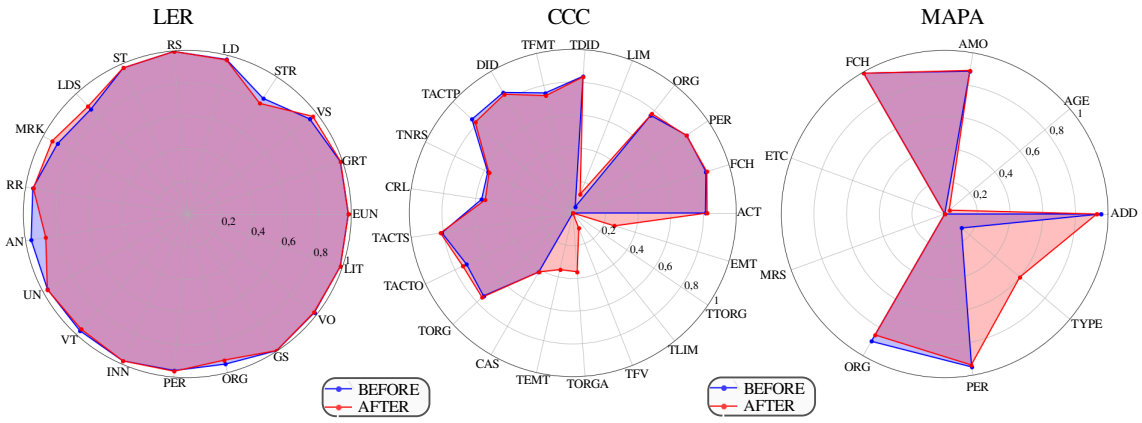

Figure 4.8. Methodology test in three scenario, LER at left, CCC in the center, and MAPA at right.

The proposed methodology does not significantly impact the performance of ideal data with balanced classes and sufficient samples. However, the methodology has proven to be effective in handling real-world data, which often suffer from class imbalances and limited data instances. As such, this methodology can be used confidently in scenarios where the data already perform well, without any adverse effects on the performance.

4.3.2 Knowledge Generalization

To analyze the performance of FLERT when training and test set contexts differ, as well as to determine the amount of data required for a model to generalize knowledge to a new context, we performed an experiment selecting training and target datasets with the same language. The model was trained with the training set and then gradually added 0, 10, 20, 30, 40, and 50 percent of information from the target dataset to the training process. Finally, we tested the model on the remaining percentage of the target dataset. As each database has its own entity types, we conducted experiments only on those entities that both the training and testing datasets had in common. The following tests were then performed:

General to specific domain: In the first two experiments, we utilized the CoNLL-02 dataset as the general domain data for the training set. We then selected the CCC and MAPA datasets as specific domain data for the testing sets. The experiment was conducted with two entity types, namely Organization and Person, which are common among CoNLL-02 and the testing sets. Following the defined process, we performed the experiments. The results are summarized in Figure 4.9, with the left part showing the outcomes when the target set is the MAPA dataset, while the right part displays the results when the test set is the CCC dataset. The Figure 4.9 indicates that if the model is trained solely with general information, it can recognize information from specific domains, achieving a score of approximately 50%. However, if we incorporate information from the target domain, the model's performance improves, reaching an F1-score of over 80% using 40% of the information for each MAPA and CCC database.

Specific to specific domain In the following set of experiments, we focus on evaluating the generalization of knowledge from a specific domain data to another specific domain data. To this end, we select the CCC and MAPA datasets, which share three entity types: Organization, Person, and Date. The experiments are designed as follows: firstly, we train the model with CCC and test it with MAPA, and secondly, we train the model with MAPA and test it with CCC. The left-hand side of Figure 4.10 illustrates the experiment when CCC is used as the training dataset and MAPA as the target dataset. Surprisingly, the performance drops to zero when no information from the target dataset exists in the training process. However, if we add only 10% of information from the target dataset, the performance improves significantly,

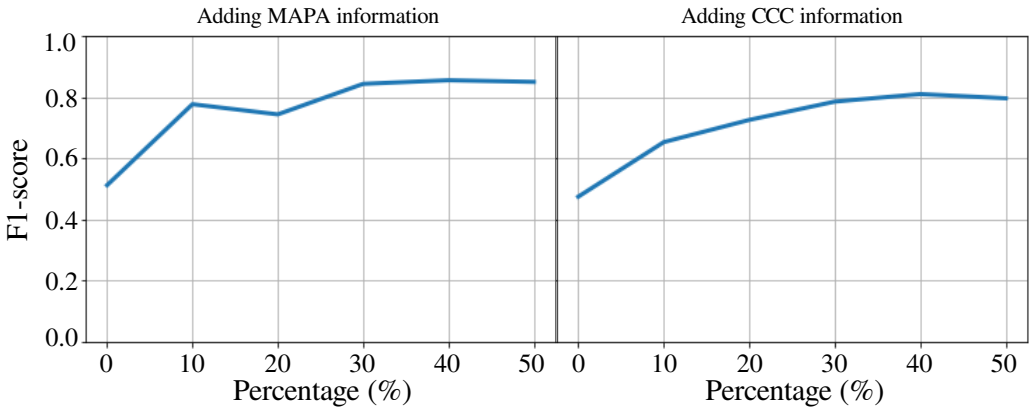

Figure 4.9. F1-score adding information from general to specific domains.

reaching over 80%. Similar results are obtained in the experiment on the right-hand side, where the model is trained with MAPA and tested with CCC.

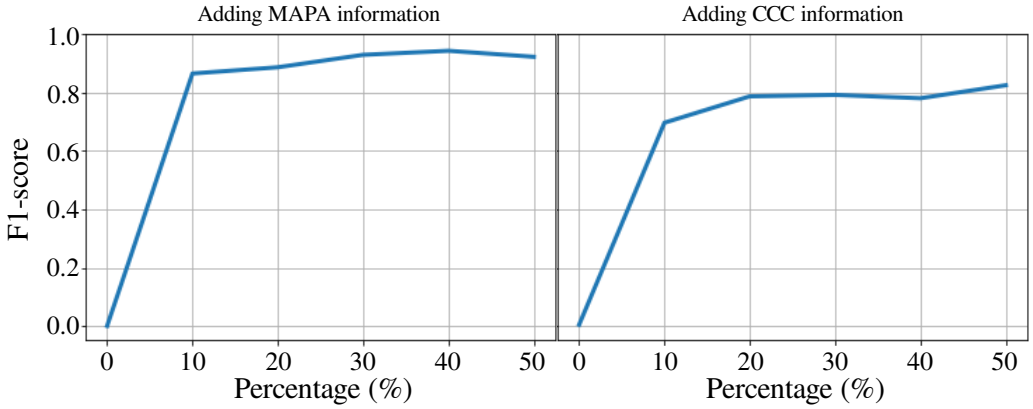

Figure 4.10. F1-score adding information from specific to specific domain.

An interesting feature of the model used with the focal loss function is its ability to generalize knowledge among domains. Training the model with data from a general domain without information from a new domain allows the model to recognize general information reaching an F1-score of approximately 50%. Nevertheless, the model trained with data from a specific domain without information of a new domain cannot recognize the information of the new domain. We hypothesized that this behavior is due to the different contexts of the databases; from general to specific domain the knowledge generalization is better than from specific to specific domain. However, in both cases the model increases its performance significantly with a piece of information (10%) from

the new data in the training process. If we deal with a field with little data, we can use this feature to make a more extensive training set and improve the results.

4.3.3 Semantic Similarity

The data augmentation methods increase the number of tokens helping those datasets that the models cannot recognize well because it does not have enough samples per class. So the data augmentation augments the number of tokens, but what happens with the semantic similarity? In the left-hand side of Figure Figure 4.11, we observe a reduction in semantic similarity and an improvement in performance for entities of interest in the CCC dataset after applying the methodology, which combines focal loss and data augmentation techniques. Hence, this methodology appears to tackle two of the three issues that were previously identified.

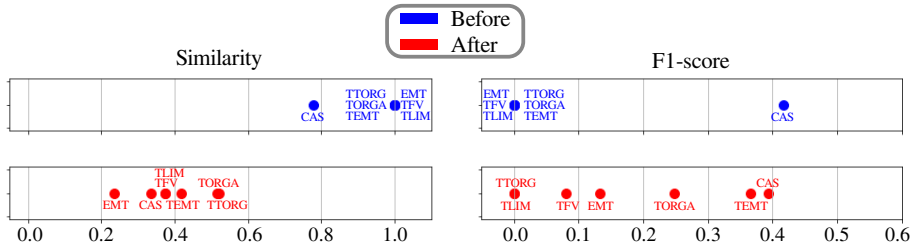

Figure 4.11. Semantic similarity (left) and F1-score (rigth) per class, before (up) and after (bottom) the methodology proposed.

In conclusion, we identify three main problems in NER with real-world data: The number of tokens, the average length, and the semantic similarity among entities. We perform a methodology that includes the focal loss function and data augmentation techniques to face two of those problems. The semantic similarity and the number of tokens were mitigated by the methodology. Also, we found the indication of knowledge generalization in the FLERT model.

4.4 Relation classification

Current state-of-the-art relation classification (RC) relies on accessing big amounts of data and are based on complex deep models. However, in contrast, a recent study [81] proposes a simpler architecture in which they addressed the challenges of limited data by using an early-fusion strategy, which involves

concatenating or merging the raw inputs to create a single input tensor that is fed into a FCNN. The approach divides a sentence into three contexts: initial, intermediate, and final, and also into two entities: head and tail.

4.4.1 Sentence Representation

Following the approach proposed, the sentences were divided into five tokens, where a token is a set of words. The first entity in a sentence is considered the head entity and is assigned to the head entity token, while the second entity is assigned to the tail entity token. The tokens that precede the head entity, come between the two entities, and follow the tail entity are referred to as the initial, intermediate, and final context tokens, respectively. On average, these tokens contain 5, 4, and 7 words. To determine the dimension of the embedding matrix for input to the CNN, we use the third quartile of the distribution of words in each context token, which is 7, 5, and 11, for the initial, intermediate, and final tokens, respectively. This approach has been previously used in other CNN-based NLP methodologies. However, since entities in the corpus usually consist of one, two, or three words, they are not analyzed using CNN [82].

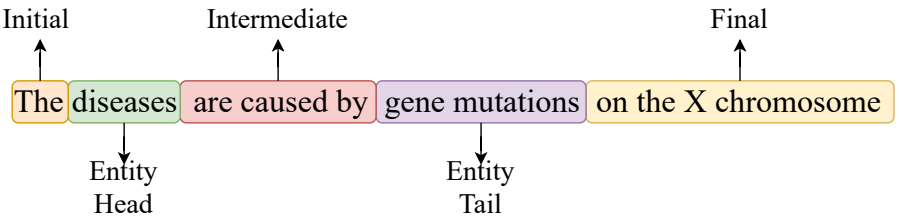

Figure 4.12. Sentence tokens division.

4.4.2 Architecture

The RoBERTa pre-trained model is used to obtain a numerical representation for each word in the sentence, resulting in a dynamic representation $M \in \mathbb{R}^{n \times d}$ for each token, where n is the number of the words that composed each token and d is the word-embedding dimension. Based on the methodology proposed in [81], we obtained a static representation for context tokens using a CNN-based approach.

The convolution is developed along the sequence of words in one dimension (i.e., vertically), and the filter's size in the convolution defines the semantic

relationships analyzed among the words. In this work, based on the third quartile of words in the context tokens, we defined the kernel size to map tri-gram semantic relationships. For entity tokens, we computed the average embedding of the dynamic representation. A *tanh* is afterward applied to create a soft representation of the averaged embeddings. The classification stage is made by a FCNN of 4 layers with 5120, 2048, 1024 and 256 units, respectively, The FCNN is fed by the concatenation of the token's representations. As loss function, we use the Focal Loss function.

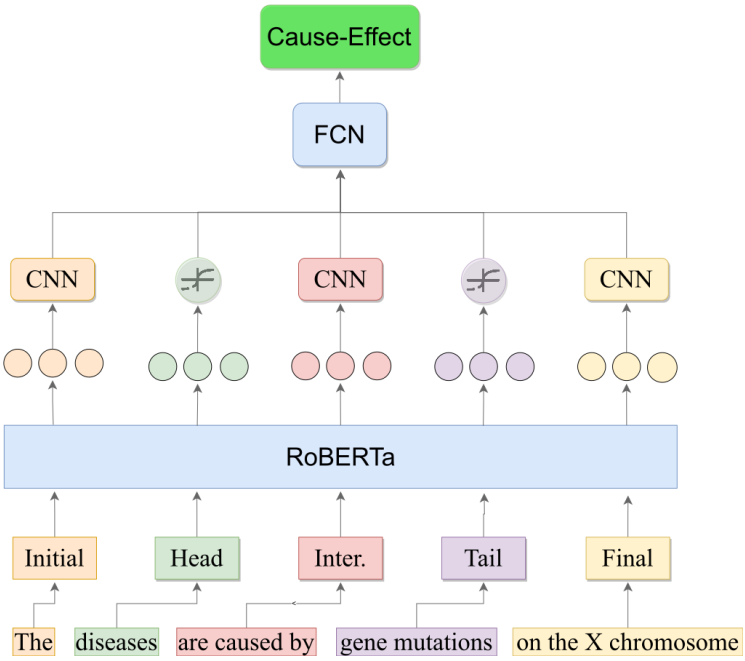

Figure 4.13. Adopted architecture.

4.4.3 Test in benchmark database

The performance of the proposed architecture was evaluated on the SemEval 2010 task 8 dataset, and we obtained an F1-score of 70% for 9 relations. The performance of each entity in the dataset is presented in Figure 4.14. Our approach achieved more than 80% F1-score for the "Cause-Effec" relation, and the lowest value was 57.64%, demonstrating that the model performs well even when working with limited data.

The focal loss function utilized in the NER problem is already incorporated in the RC model. Although we attempted to test the data augmentation

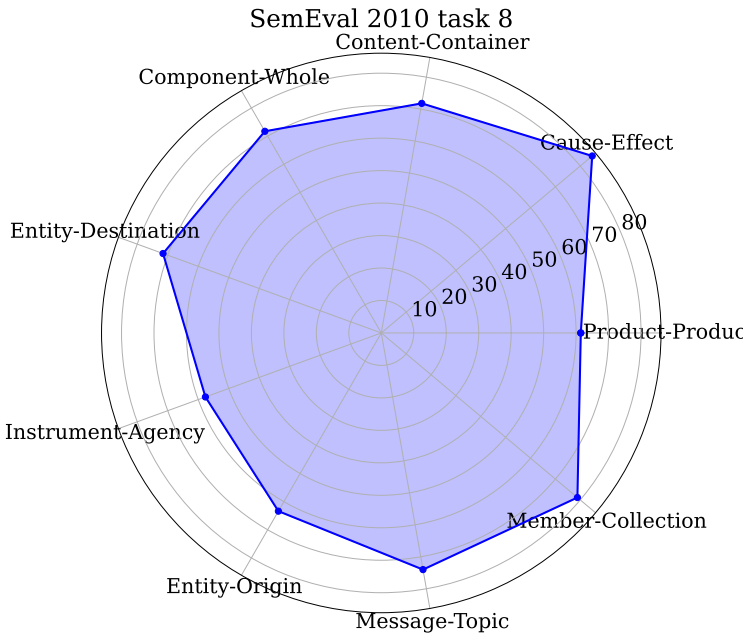

Figure 4.14. Performance per relation in the SemEval 2010 task 8 dataset.

techniques in RC, the outcomes were unsatisfactory. This resulted in a complex and resource-intensive process that does not improve the results for the RC problem. Therefore, we only implement the architecture with the focal loss function.

Chapter 5

Software

The Named Entity Recognition (NER) and Relationship Classification (RC) software is an application that implements artificial intelligence techniques to automatically extract relevant information from a document. This is why it uses cutting-edge technologies to develop a framework that allows them to be used and presented in a user-friendly way.

The software allows for the automatic finding of entities within a text and can classify the semantic relationship between two entities found in a sentence. This streamlines and automates the task of identifying People, Places, Documents, among others; and also, finding the possible relations between them. Its functionality can be applied to a specific sentence or an entire document, providing versatility for its field of use. All the information on the entities found in a document is saved in another document that can be used later for the analysis or processing of relevant information. Moreover, incorporates the training of new models for other contexts of different information, thus generating the model for later use. All the model implementation was done using the library Flair and the framework Pytorch, which allows to implement NLP models easily.

It also has a graphical interface based on Gradio that facilitates the understanding and use of the system. Additionally, it provides convenience in visualizing the information found in the input text by using the display of the entered text with the relevant information highlighted. This way, the software streamlines the implementation of natural language processing techniques for document analysis in a user-friendly manner.

5.1 NER

5.1.1 Training

The flow of the software for training a new model is comprised of the following phases:

- **Load data:** Data is taken by default from a JSON file in PRATEC format and converted to CONLL03 format. The user can also provide data in CONLL02 format directly. Once in CONLL03 format, the data is saved in the internal data directory for later use.
- **Create corpus object:** The flair library provides the corpus object from which an instance is defined that will load the train.txt and test.txt files from the data directory.
- **Create embeddings object:** The flair library provides the embeddings object from which an instance is defined that contains the configuration of the embedding layer of the model and downloads the pre-trained RoBERTa model.
- **Create model:** The flair library provides the tagger object from which an instance is defined that defines the complete model using the previously defined embeddings object instance.
- **Create trainer:** The flair library provides the trainer object from which an instance is defined that will be responsible for training the model using the tagger and corpus.
- **Train:** With the trainer object instance, the train method is called and configured with a learning rate of 5e-6, 20 epochs, and the Adam optimizer. This method will generate the model training process and save the resulting model in the models directory.

The flow of information of the train process is showed in Figure 5.1

5.1.2 Tagging

The software flow for using an existing model is comprised of the following phases:

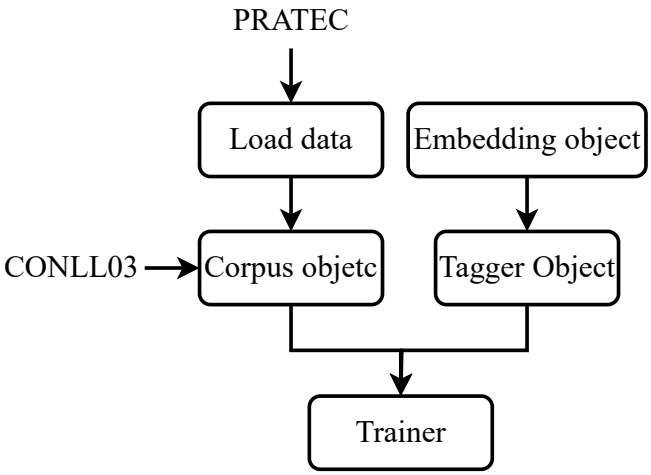

Figure 5.1. Flow of information of the NER train process.

- **Load document or sentence:** Data is taken from a JSON document in Input Document format (Document option) or directly from the textbox (Sentence option) and loaded into a system variable.
- **Load selected model:** The flair library provides the tagger object, from which an instance is defined to load the selected model.
- **Predict labels for each word in input data:** The tagger model instance has a method called "predict" which predicts the entities for each word within the previously loaded data.
- **Organize results in output format:** Once the entities are predicted, the information is organized and saved in a Python dictionary.
- **Return results:** The dictionary is saved in JSON format within the previously specified output path and then returned to the front-end for further use.

The flow of information of the tagging process is showed in Figure 5.2

5.1.3 General View

Figure 5.3 shows the general view of a NER tagged document using the software developed.

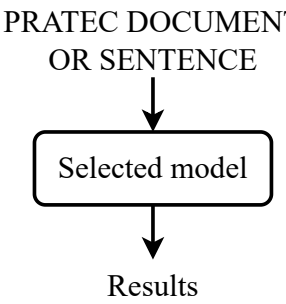

Figure 5.2. Flow of information of the NER tagging process.

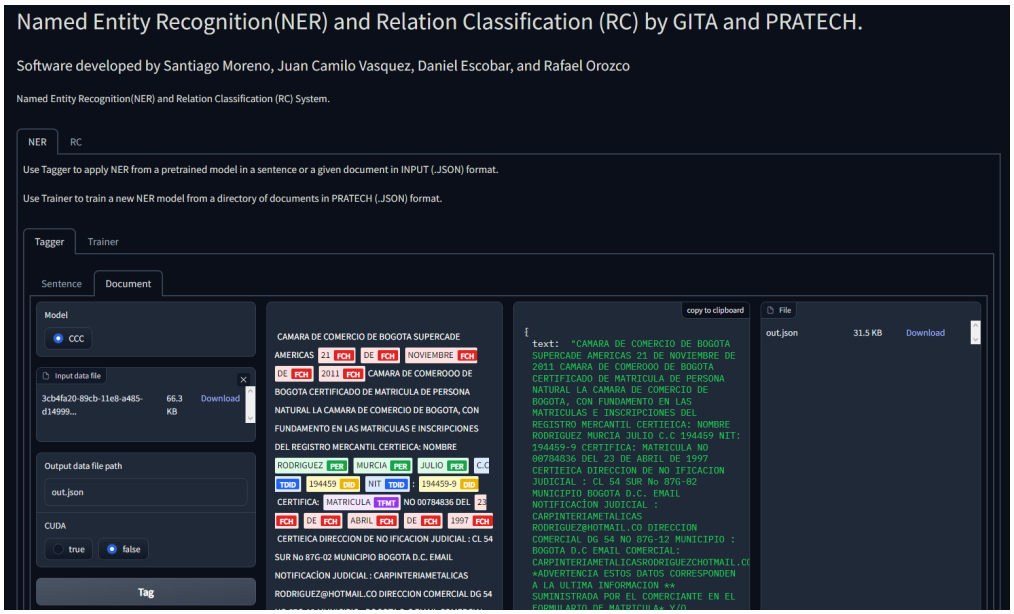

Figure 5.3. View of the Software's NER stage.

5.2 RC

5.2.1 Training

The flow of the software for training a new model is comprised of the following phases:

- **Prepare the data:** Data is taken by default from a TXT file in CONLL04 format, then the data is saved in the internal data directory for later use. The flair library provides the embeddings object from which an instance is defined that contains the configuration of the embedding layer of the model and downloads the pre-trained RoBERTa model. Then this object is used to obtain the numerical representation for each word. The Pytorch library provides the dataset and dataloader object from which an instance is defined that will load the data for the training process. Using this two objects the data is organized in the format that the model understand. Also it is applied the fixed shape for the initial, intermediate and final contexts tokens as well as the average for the entity tail and head tokens.

- **Create model:** We define a model class based on the Pytorch framework. This class contain the implementation of the whole model including the forward propagation method. Then we create an instance of this class.

- **Train:** With the model object instance we create the Adam optimizer with a learning rate of 1e-4, and 200 epochs. Then the model is trainer with the data object making the backpropagation after the forward propagation. This process will generate and save the resulting model in the models directory.

The flow of information of the train process is showed in Figure 5.4

5.2.2 Tagging

The software flow for using an existing model is comprised of the following phases:

- **Prepare the data:** *(Is the same process used for training).* Data is taken by default from a TXT file in CONLL04 format, then the data

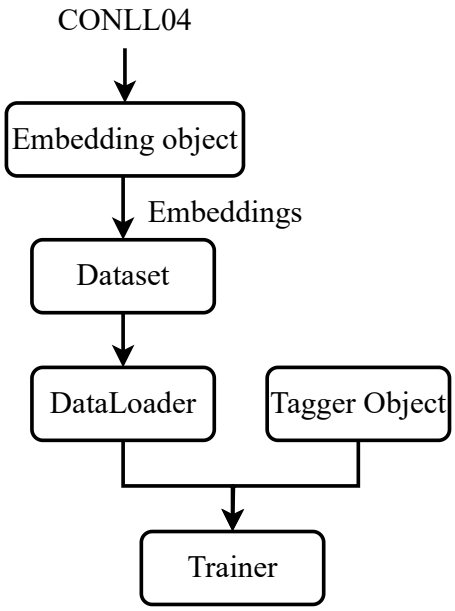

Figure 5.4. Flow of information of the RC train process.

is saved in the internal data directory for later use. The flair library provides the embeddings object from which an instance is defined that contains the configuration of the embedding layer of the model and downloads the pre-trained RoBERTa model. Then, this object is used to obtain the numerical representation for each word. The Pytorch library provides the dataset and dataloader object from which an instance is defined that will load the data for the tagging process. Using this two objects the data is organized in the format that the model understand. Also, it is applied the fixed shape for the initial, intermediate and final contexts tokens as well as the average for the entity tail and head tokens.

- **Load selected model:** We create an instance of the model and then we load the weigths of the pre-trained model selected.
- **Predict labels for each sentence in input data:** The forward propagation method of the model is called with the data object. The results are the relations predicted for each sentence.
- **Organize results in output format:** Once the relation are predicted, the information is organized in and saved in a Python dictionary.

- **Return results:** The dictionary is saved in JSON format within the previously specified output path and then returned to the front-end for further use.

The flow of information of the tagging process is showed in Figure 5.5

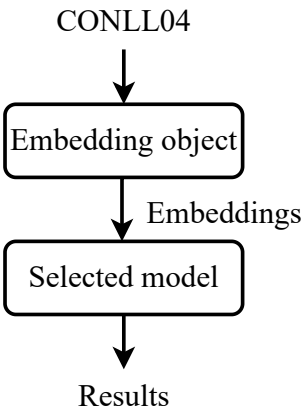

Figure 5.5. Flow of information of the RC tagging process.

5.2.3 General View

Figure 5.3 Shows the general view of a RC tagged document using the software developed.

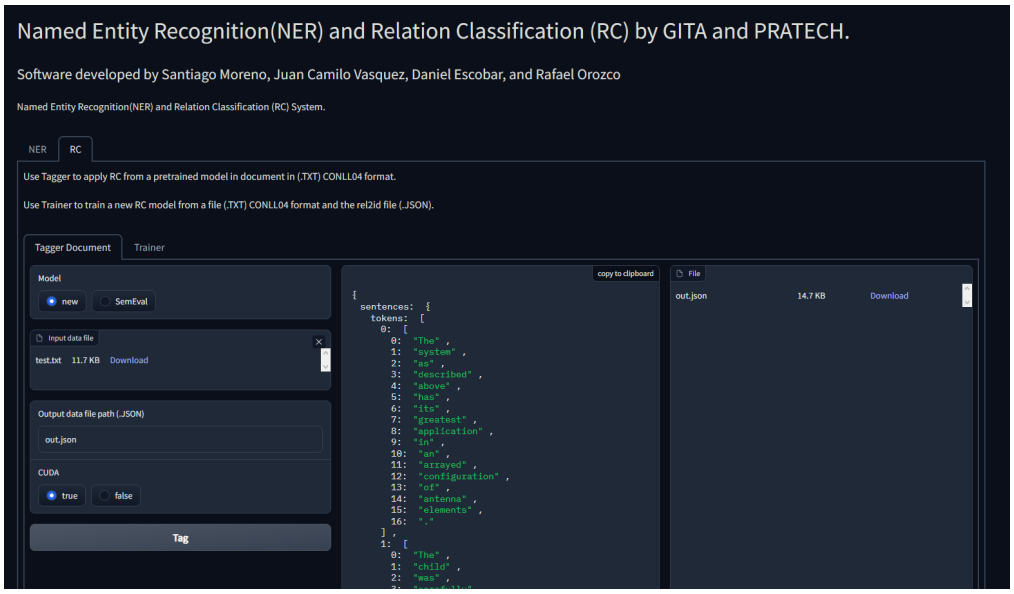

Figure 5.6. View of the Software's RC stage.

Chapter 6

Conclusions and future work

This study focuses on implementing NLP techniques for IE in real-world environments. To achieve this objective, models for NER and RC using real-world data were developed. In the NER approach, we analyze the real-world data compared to ideal data; the outcomes revealed that the performance is affected by the number of tokens and the semantic similarity among entities, which are mainly caused by the limited number of samples and class unbalance. To address these issues, we proposed a methodology that modifies the loss function to face the class unbalances and uses data augmentation to increase the number of samples. The methodology implemented shows satisfactory results in various tests, including different scenarios and knowledge generalization. For RC, we implemented a simple architecture that works effectively with limited data; then, we tested it on a benchmark dataset. The results for all relations were satisfactory. Finally, we integrated the NER and RC models into a software tool for automatic information extraction from a given document.

To perform the NER approach, we propose a methodology to address two issues in performing NER on real-world data: few samples and class unbalance. First, we compared the performance of the FLERT model on ideal data (LER dataset) and real-world data (CCC dataset). The results on the ideal dataset were highly accurate, achieving F1-scores around 90% in the classification of 19 entities. However, the results on the CCC corpus were unsatisfactory, with some entities achieving nearly zero F1-scores. Further analyses shows three possible factors that explain the situation: (I) a small number of tokens in the interest entity, (II) a high average segment length, and (II) a high semantic similarity. Therefore, we implement a methodology that includes the focal loss function in the FLERT model and data augmentation techniques to tackle

both few samples and class unbalance issues. In this study, several experiments were performed to evaluate the effectiveness of the proposed methodology for NER on real-world data. Firstly, we compared the performance of the FLERT model before and after applying the methodology on data with class imbalance, and found a noticeable improvement in the F1-score for the classes with few samples and those with scores lower than 60%. Next, the methodology was tested on three different datasets: LER, CCC, and MAPA, and was observed that it only had a significant impact on classes with few samples, while the performance of other classes was not significantly affected, even when class imbalance was present. Finally, we evaluated the model's ability to generalize knowledge from one domain to another using the CoNLL-02 dataset as general data and CCC and MAPA datasets as specific data. The results showed that the model could successfully generalize information from a specific domain using data from another field, even with a small amount (10%) of data from the target domain in the training process. This outcomes suggest that the proposed methodology works on fields with limited data by using data from other domains to augment the training set.

Then, we also addressed the task of RC on real-world data by implementing a novel architecture based on a state-of-the-art model, which performs well with limited data. The proposed architecture aimed to identify relationships between entities in a sentence by dividing a sentence into five tokens, including three context tokens (initial, intermediate, and final) and two entity tokens (head and tail). We utilized the RoBERTa model to obtain numerical representations of each token. Then, a vector-representation was obtained for each token; the context tokens were processed using a CNN, while the entity tokens were processed using the average of the words and the tanh function. Finally, the resulting representations were combined using the early-fusion strategy, and the resulting vector-representation was fed into a fully-connected layer for classification. The architecture was tested on The SemEval 2010 Task 8 dataset, and the results showed satisfactory performance for all relations.

Finally, the strategies for NER and RC developed in this work were implemented into a software tool that automated those tasks. The NER approach in the tool employed the FLAIR library, enabling us to create NER models quickly and efficiently. For the RC we use the framework Pytorch which allows the building of hand-crafted models easily. Finally, those models were used in a web platform that facilitates the usage and understanding of NER and RC models. The platform was developed under the Gradio Framework.

The work developed in this study is the first step for a tool of automatic document analysis that allows to work in real environments where the data is limited. Here, we addressed the problems of NER and RC working with real-world data. Therefore, we provided a positive answer to our research question; it was possible to develop an IE tool using NER and RC models based on DL that can perform well in real-world scenarios with limited data. To continue the progress of the tool, it is necessary to research and develop other topics around IE via NLP, and IE from images. Future works will include research about IE in NLP with specific topics such as automatic document summary, and document classification. Also, since the documents can have images such as logos recognition, signature recognition, and structured information recognition, it is mandatory to implement techniques for IE in images. Finally, to continue the development of the software tool for automatic document analysis, those improvements must be implemented via backend and frontend and then adapted to the tool developed here.

List of Figures

1.1 Taxonomy of the methods for NER 10

2.1 Conditional Random Field scheme 19
2.2 Fully Connected Neural Network 20
2.3 Matrix representation for a sentence 21
2.4 Convolutional Neural Network 21
2.5 Recurrent Neural Network 22
2.6 Long-Short Term Memory cell 24
2.7 Transformer's encoder. Image adapted from [61] 26
2.8 PRATEC format. 33
2.9 CONLL03 format. 34
2.10 CONLL04 format. 35

4.1 Number of tokens (words) per sentence for German (left) and Spanish (right) corpus. Taken from [73]. 43
4.2 F1-scores obtained per entity. (Right) German dataset and (Left) Spanish dataset. Taken from [73]. 45
4.3 F1-score for the 10 entities with the lower number of tokens for Spanish corpus. Taken from [73]. 47
4.4 Comparison of F1-score (up) and the average length of the segment (bottom) for each entity. The entities are sorted according to the number of tokens. Taken from [73]. 48
4.5 Semantic similarity analysis. Taken from [73]. 49
4.6 Division of the good and poor sets. 51
4.7 Distribution of tokens per entity before and after data augmentation. 53
4.8 Methodology test in three scenario, LER at left, CCC in the center, and MAPA at right. 54

4.9 F1-score adding information from general to specific domains. . 56
4.10 F1-score adding information from specific to specific domain. . . 56
4.11 Semantic similarity (left) and F1-score (rigth) per class, before (up) and after (bottom) the methodology proposed. 57
4.12 Sentence tokens division. 58
4.13 Adopted architecture. 59
4.14 Performance per relation in the SemEval 2010 task 8 dataset. . 60

5.1 Flow of information of the NER train process. 64
5.2 Flow of information of the NER tagging process. 65
5.3 View of the Software's NER stage. 65
5.4 Flow of information of the RC train process. 67
5.5 Flow of information of the RC tagging process. 68
5.6 View of the Software's RC stage. 69

List of Tables

1.1 Experiments for NER 12

2.1 Example of data augmentation methods. In blue the tokens changed according to the method 30

3.1 Relations of the SemEval 2010 task 8 dataset 39

4.1 Details of the data in the German (LER) dataset. 41
4.2 Distribution of the classes in the Spanish (CCC) dataset. 42
4.3 Results for German dataset. 46
4.4 Results for Spanish dataset. 46
4.5 F1-score (%) of FLERT with different loss functions 51
4.6 F1-score (%) of FLERT with different up-sampling methods applied in CCC dataset 52

Bibliography

[1] X. Li *et al.*, "Automating document classification with distant supervision to increase the efficiency of systematic reviews: A case study on identifying studies with hiv impacts on female sex workers," *PloS one*, vol. 17, no. 6, e0270034, 2022.

[2] V. Wagh *et al.*, "Comparative study of long document classification," in *Proceedings of TENCON*, 2021, pp. 732–737.

[3] W. El-Kassas *et al.*, "Automatic text summarization: A comprehensive survey," *Expert systems with applications*, vol. 165, p. 113 679, 2021.

[4] M. Agosti *et al.*, "A relation extraction approach for clinical decision support," *arXiv preprint arXiv:1905.01257*, 2019.

[5] S. Zheng *et al.*, "Text mining for drug discovery," *Bioinformatics and Drug Discovery*, pp. 231–252, 2019.

[6] A. Jabbari *et al.*, "A french corpus and annotation schema for named entity recognition and relation extraction of financial news," in *Proceedings of The 12th Language Resources and Evaluation Conference*, 2020, pp. 2293–2299.

[7] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," *Lingvisticae Investigationes*, vol. 30, no. 1, pp. 3–26, 2007.

[8] G. Lample *et al.*, "Neural architectures for named entity recognition," *arXiv preprint arXiv:1603.01360*, 2016.

[9] Z. Huang and et al, "Bidirectional lstm-crf models for sequence tagging," *arXiv preprint arXiv:1508.01991*, 2015.

[10] P. Bhatia and et al, "Dynamic transfer learning for named entity recognition," in *Proceedings of International Workshop on Health Intelligence*, Springer, 2019, pp. 69–81.

[11] S. Schweter and A. Akbik, "Flert: Document-level features for named entity recognition," *arXiv preprint arXiv:2011.06993*, 2020.

[12] T. Nguyen and et al, "Adaptive name entity recognition under highly unbalanced data," *arXiv preprint arXiv:2003.10296*, 2020.

[13] E. Leitner *et al.*, "Fine-grained Named Entity Recognition in Legal Documents," in *Proceedings of International Conference Semantic Systems*, ser. Lecture Notes in Computer Science, 10/11 September 2019, Karlsruhe, Germany: Springer, Sep. 2019, pp. 272–287.

[14] V. Pais *et al.*, "Named entity recognition in the Romanian legal domain," in *Proceedings of the Natural Legal Language Processing Workshop 2021*, Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 9–18. DOI: 10.18653/v1/2021.nllp-1.2. [Online]. Available: https://aclanthology.org/2021.nllp-1.2.

[15] C. Çetindağ *et al.*, "Named-entity recognition in turkish legal texts," *Natural Language Engineering*, pp. 1–28, 2022.

[16] H. Vardhan *et al.*, "Named-entity recognition for legal documents," in *Proceedings of International Conference on Advanced Machine Learning Technologies and Applications*, Springer, 2020, pp. 469–479.

[17] O. de Gibert Bonet *et al.*, "Spanish datasets for sensitive entity detection in the legal domain," in *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, 2022, pp. 3751–3760.

[18] L. Loukas *et al.*, "Finer: Financial numeric entity recognition for xbrl tagging," *arXiv preprint arXiv:2203.06482*, 2022.

[19] A. M. Kuckartz and U. Kuckartz, *Qualitative text analysis with maxqda*, 2002.

[20] B. Foenix-Riou, "Talkwalker: Une plateforme de veille et un service d'alertes mail," *Netsources*, no. 103, p. 10, 2013.

[21] J. Straková and et al, "Neural architectures for nested ner through linearization," *arXiv preprint arXiv:1908.06926*, 2019.

[22] P. Gupta *et al.*, "Neural architectures for fine-grained propaganda detection in news," *arXiv preprint arXiv:1909.06162*, 2019.

[23] R. Grishman and B. M. Sundheim, "Message understanding conference-6: A brief history," in *Proceedings of the 16th International Conference on Computational Linguistics*, 1996.

[24] E. F. Sang and F. De Meulder, "Introduction to the conll-2003 shared task: Language-independent named entity recognition," *arXiv preprint cs/0306050*, 2003.

[25] T. K. Sang, "Erik. f. 2002. introduction to the conll-2002 shared task: Language-independent named entity recognition," in *Proceedings of Conference on Natural Language Learning*.

[26] D. Hanisch *et al.*, "Prominer: Rule-based protein and gene entity recognition," *BMC bioinformatics*, vol. 6, no. 1, pp. 1–9, 2005.

[27] A. Ritter *et al.*, "Named entity recognition in tweets: An experimental study," in *Proceedings of the 2011 conference on empirical methods in natural language processing*, 2011, pp. 1524–1534.

[28] M. E. Peters *et al.*, "Deep contextualized word representations," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, 2018.

[29] J. Devlin *et al.*, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[30] X. Ma and et al, "End-to-end sequence labeling via bi-directional lstm-cnns-crf," *arXiv preprint arXiv:1603.01354*, 2016.

[31] Y. Shen *et al.*, "Deep active learning for named entity recognition," *arXiv preprint arXiv:1707.05928*, 2017.

[32] P. Zhou *et al.*, "Joint extraction of multiple relations and entities by using a hybrid neural network," in *Proceedings of Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, Springer, 2017, pp. 135–146.

[33] T.-H. Pham and P. Le-Hong, "End-to-end recurrent neural network models for vietnamese named entity recognition: Word-level vs. character-level," in *Proceedings of the International Conference of the Pacific Association for Computational Linguistics*, Springer, 2017, pp. 219–232.

[34] Q. Tran and et al, "Named entity recognition with stack residual lstm and trainable bias decoding," *arXiv preprint arXiv:1706.07598*, 2017.

[35] Z.-X. Ye and Z.-H. Ling, "Hybrid semi-markov crf for neural sequence labeling," *arXiv preprint arXiv:1805.03838*, 2018.

[36] G. Aguilar *et al.*, "A multi-task approach for named entity recognition in social media data," *arXiv preprint arXiv:1906.04135*, 2019.

[37] S. Sánchez-Cuadrado *et al.*, "Extracción automática de relaciones semánticas," in *Proceedings of Conferencia Iberoamericana en Sistemas, Cibernética e Informática*, 2003, pp. 265–268.

[38] C. Lee *et al.*, "Fine-grained named entity recognition and relation extraction for question answering," in *Proceedings of the conference on Research and development in information retrieval*, 2007, pp. 799–800.

[39] D. Zeng *et al.*, "Relation classification via convolutional deep neural network," in *Proceedings of International Conference on Computational Linguistics: Technical Papers*, 2014, pp. 2335–2344.

[40] J. Xu *et al.*, "A discourse-level named entity recognition and relation extraction dataset for chinese literature text," *arXiv preprint arXiv:1711.07010*, 2017.

[41] P. Verga *et al.*, "Simultaneously self-attending to all mentions for full-abstract biological relation extraction," *arXiv preprint arXiv:1802.10569*, 2018.

[42] G. Singh and P. Bhatia, "Relation extraction using explicit context conditioning," *arXiv preprint arXiv:1902.09271*, 2019.

[43] J. Giorgi *et al.*, "End-to-end named entity recognition and relation extraction using pre-trained language models," *arXiv preprint arXiv:1912.13415*, 2019.

[44] L. Wiest, "Recurrent neural networks-combination of rnn and cnn," *Published on*, vol. 7, 2017.

[45] L. Zhang and F. Xiang, "Relation classification via bilstm-cnn," in *Proceedings of DMBD*, Springer, 2018, pp. 373–382.

[46] X. Zhang *et al.*, "A combination of rnn and cnn for attention-based relation classification," *Procedia computer science*, pp. 911–917, 2018.

[47] P. Cheng and K. Erk, "Attending to entities for better text understanding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 7554–7561.

[48] J. Guo *et al.*, "Named entity recognition in query," in *Proceedings of Conference on Research and development in information retrieval*, 2009, pp. 267–274.

[49] D. Petkova and W. B. Croft, "Proximity-based document representation for named entity retrieval," in *Proceedings of Conference on information and knowledge management*, 2007, pp. 731–740.

[50] J.-H. Kim and P. C. Woodland, "A rule-based named entity recognition system for speech input," in *Proceedings of Conference on Spoken Language Processing*, 2000.

[51] S. Zhang and N. Elhadad, "Unsupervised biomedical named entity recognition: Experiments with clinical and biological texts," *Journal of biomedical informatics*, vol. 46, no. 6, pp. 1088–1098, 2013.

[52] V. Krishnan and C. D. Manning, "An effective two-stage model for exploiting non-local dependencies in named entity recognition," in *Proceedings of Conference on Computational Linguistics and Association for Computational Linguistics*, 2006, pp. 1121–1128.

[53] Z. Zhao *et al.*, "Drug drug interaction extraction from biomedical literature using syntax convolutional neural network," *Bioinformatics*, vol. 32, no. 22, pp. 3444–3453, 2016.

[54] B. Rosario and M. A. Hearst, "Multi-way relation classification: Application to protein-protein interactions," in *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, 2005, pp. 732–739.

[55] G. Boella *et al.*, "Semantic relation extraction from legislative text using generalized syntactic dependencies and support vector machines," in *Proceedings of International Symposium, RuleML 2013, Seattle, WA, USA, July 11-13, 2013. Proceedings 7*, Springer, 2013, pp. 218–225.

[56] M. Al Qady and A. Kandil, "Concept relation extraction from construction documents using natural language processing," *Journal of construction engineering and management*, vol. 136, no. 3, pp. 294–302, 2010.

[57] P. Das *et al.*, "A graph based clustering approach for relation extraction from crime data," *IEEE Access*, vol. 7, pp. 101 269–101 282, 2019.

[58] N. Bach and S. Badaskar, "A review of relation extraction," *Literature review for Language and Statistics II*, vol. 2, pp. 1–15, 2007.

[59] C. Liu *et al.*, "Convolution neural network for relation extraction," in *Proceedings of International Conference on Advanced Data Mining and Applications*, Springer, 2013, pp. 231–242.

[60] Z. Li *et al.*, "Recurrent neural networks with segment attention and entity description for relation extraction from clinical texts," *Artificial intelligence in medicine*, vol. 97, pp. 9–18, 2019.

[61] A. Vaswani *et al.*, "Attention is all you need," in *Proceedings of Advances in neural information processing systems*, 2017, pp. 5998–6008.

[62] A. Bastos *et al.*, "Recon: Relation extraction using knowledge graph context in a graph neural network," in *Proceedings of Web Conference*, 2021, pp. 1673–1685.

[63] L. Wang *et al.*, "Relation classification via multi-level attention cnns," in *Proceedings of Association for Computational Linguistics*, 2016, pp. 1298–1307.

[64] N. Peng *et al.*, "Cross-sentence n-ary relation extraction with graph lstms," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 101–115, 2017.

[65] C. M. Bishop *et al.*, *Neural networks for pattern recognition*. Oxford university press, 1995.

[66] Y. Liu *et al.*, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[67] G. Wenzek *et al.*, "CCNet: Extracting high quality monolingual datasets from web crawl data," English, in *Proceedings of Language Resources and Evaluation Conference*, Marseille, France: European Language Resources Association, May 2020, pp. 4003–4012, ISBN: 979-10-95546-34-4. [Online]. Available: https://aclanthology.org/2020.lrec-1.494.

[68] T.-Y. Lin *et al.*, "Focal loss for dense object detection," in *Proceedings of the International Conference on Computer Vision*, 2017, pp. 2980–2988.

[69] X. Dai and H. Adel, "An analysis of simple data augmentation for named entity recognition," *arXiv preprint arXiv:2010.11683*, 2020.

[70] D. Escobar Grisales, "Demographic information retrieval from text for subject characterization and market segmentation," M.S. thesis, 2022.

[71] A. Akbik *et al.*, "Flair: An easy-to-use framework for state-of-the-art nlp," in *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics*, 2019, pp. 54–59.

[72] I. Hendrickx *et al.*, "Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals," *arXiv preprint arXiv:1911.10422*, 2010.

[73] S. Moreno-Acevedo *et al.*, "Comparison of named entity recognition methods on real-world and highly imbalanced business document datasets," in *Proceedings of WEA*, Springer, 2022, pp. 41–53.

[74] N. Reimers and I. Gurevych, "Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging," in *Proceedings of Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark, Sep. 2017, pp. 338–348. [Online]. Available: http://aclweb.org/anthology/D17-1035.

[75] H. Yan *et al.*, "Tener: Adapting transformer encoder for named entity recognition," *arXiv preprint arXiv:1911.04474*, 2019.

[76] M. Arkhipov *et al.*, "Tuning multilingual transformers for language-specific named entity recognition," in *Proceedings of Workshop on Balto-Slavic Natural Language Processing*, 2019, pp. 89–93.

[77] J. R. Finkel *et al.*, "Incorporating non-local information into information extraction systems by gibbs sampling," in *Proceedings of annual meeting of the association for computational linguistics*, 2005, pp. 363–370.

[78] N. Reimers *et al.*, *Nested named entity recognition with neural networks*. 2014.

[79] C. Cardellino, *Spanish Billion Words Corpus and Embeddings*, Aug. 2019. [Online]. Available: https://crscardellino.github.io/SBWCE/.

[80] N. Reimers and I. Gurevych, "Optimal hyperparameters for deep lstm-networks for sequence labeling tasks," *arXiv preprint arXiv:1707.06799*, 2017.

[81] J. Liu *et al.*, "Relation classification via bert with piecewise convolution and focal loss," *Plos one*, no. 9, e0257092, 2021.

[82] D. Escobar-Grisales *et al.*, "Colombian dialect recognition based on information extracted from speech and text signals," in *Proceedings of ASRU*, IEEE, 2021, pp. 556–563.