UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dept. of Computer Systems and Computation

Named entitiy recognition in handwritten text images from the k best transcripts

Master's Thesis

Master's Degree in Artificial Intelligence, Pattern Recognition and Digital Imaging

AUTHOR: Giner Pérez de Lucía, José

Tutor: Sánchez Peiró, Joan Andreu

ACADEMIC YEAR: 2023/2024

Resumen

El reconocimiento de entidades nombradas (NER) en textos manuscritos antiguos es un área desafiante de la investigación en el campo de la inteligencia artificial y el procesamiento del lenguaje natural. Consiste en identificar y clasificar entidades específicas, como nombres de personas, lugares, fechas, organizaciones, etc., en textos escritos a mano en diferentes idiomas antiguos. Este proceso implica varios desafíos debido a la naturaleza variable de la escritura a mano, la evolución del lenguaje a lo largo del tiempo, la ortografía inconsistente y la presencia de abreviaturas, entre otros factores. Para abordar estos desafíos, se aplican técnicas de procesamiento de imagen y aprendizaje profundo, incluyendo redes neuronales convolucionales y recurrentes, que han demostrado resultados competitivos en la tarea de NER en la última década.

Mediante este trabajo final de máster, se ha diseñado una red neuronal recurrente para NER en unos textos manuscritos del siglo XVI pertenecientes a unas páginas de la colección de Libros de actas de decretos reales, ubicada en el Archivo General de Simancas, uno de los archivos más importantes que narra la evolución social, política y cultural de España a lo largo de la historia. Concretamente, el modelo neuronal aprende representaciones distribuidas de palabras y de caracteres (conocidas como *embeddings*), compuesto por módulos de memoria bidireccionales a corto y largo plazo (Bi-LSTM) y un campo aleatorio condicional (CRF) como capa de salida. Los resultados obtenidos a nivel de línea para las transcripciones manuales reflejan en general buenas prestaciones de reconocimiento sobre todos los tipos de entidades (puntuación F1 de 0.83 y WTER de 5.4 %), y en específico sobre nombres y apellidos de personas (puntuaciones F1 de 0.94 y 0.89 respectivamente). Por otra parte, se ha evaluado el modelo con las k-mejores transcripciones de cada línea generadas por un proceso de reconocimiento de texto manuscrito, que pueden presentar fallos en la detección de ciertas palabras. Se ha detectado un incremento del 12 % en WTER y una bajada de 0.20 puntos de F1 en las mejores decodificaciones (conocidas como las *1-best*), y un incremento de 7 % en WTER y bajada de 0.09 puntos de F1 tras considerar las 10 mejores decodificaciones (*10-best*).

Palabras clave: Reconocimiento de entidades nombradas, Aprendizaje profundo, Archivo General de Simancas, Bi-LSTM, CRF, k-mejores transcripciones

Abstract

Named Entity Recognition (NER) in ancient handwritten texts is a challenging area of research in the field of artificial intelligence and natural language processing. It consists of identifying and classifying specific entities, such as names of people, places, dates, organizations, etc., in handwritten texts in different ancient languages. This process involves several challenges due to the variable nature of handwriting, the evolution of language over time, inconsistent spelling, and the presence of abbreviations, among other factors. To address these challenges, image processing and deep learning techniques are applied, including convolutional and recurrent neural networks, which have demonstrated to be able to get competitive results for NER over the last decade.

By means of this master's thesis, a recurrent neural network has been designed for NER in manuscript texts of the XVI century belonging to some pages of the ancient collection of Books of records of royal decrees, located in the General Archive of Simancas, one of the most important archives that narrates the cultural, political and social evolution of Spain throughout history. Specifically, the neural model learns distributed representations of words and characters, (referred as *embeddings*), composed of bidirectional short and long term memory modules (Bi-LSTM) and a conditional random field (CRF) as output layer. The results obtained at line level for the manual transcriptions reflect generally good recognition performances on all types of entities (F1 score of 0.83 and WTER of 5.4%), and specifically on person names and surnames (F1 scores of 0.94 and 0.89 respectively). On the other hand, the model has been evaluated with the k-best transcriptions of each line generated by a handwritten text recognition process, which may fail to detect certain words present in the manuscripts. A 12% increase in WTER and a 0.20 drop in F1 score was detected for the best decodings (known as the 1-best), and a 7% increase in WTER and a 0.09 drop in F1 score was detected after considering the 10 best decodings (10-best).

Key words: Named entity recognition, Deep learning, General Archive of Simancas, Bi-LSTM, CRF, k-best transcriptions

Contents

List of Figures

List of Tables

CHAPTER 1
Introduction

With the large amount of digital information available today, there are many systems in charge of extracting knowledge from the data by identifying relevant patterns to automate decision making with as little human intervention as possible. One task for which these systems specialize is named entity recognition, commonly known as Named Entity Recognition (NER). [7], which is an important field of Natural Language Processing (NLP). It involves the detection of entities of interest in texts and their classification into different semantic categories. Entities represent terms that are clearly distinguishable from each other and are fundamental to understanding the meaning of a message, the most common being people, locations and organizations. Some applications of this problem are in information retrieval in legal [8, 9], financial [10] and medical documents [11, 12], automatic response systems [13] or in recommendation systems [14] where entity storage can help reveal user preferences.

Despite more recent research areas in which named entity recognition provides great successes [15, 16, 17, 18], its performance is the subject of study in the extraction of relevant information present in images of ancient manuscript text. Unlike electronic texts found on the web, historical documents require an additional step of handwritten text recognition (HTR) to obtain the textual transcriptions that can sometimes contain flaws in the recognition process caused by the difficulty in the legibility of certain words, variations in the typographical fonts or the poor state of preservation of the pages. NER systems in ancient handwritten texts typically combine optical character recognition (OCR) [19] with natural language processing algorithms. However, current OCR systems are not suitable because characters in handwritten text cannot be reliably isolated automatically. To adequately address this problem, holistic approaches are required, referred as "segmentation free off-line HTR" [20]. These approaches aim to recognize all text elements (sentences, words and characters) as a whole, without any prior segmentation of the image into these elements. However, for most historical manuscripts these systems generally fail to achieve the precision required for further use of the resulting transcripts. Transcription accuracy for a given manuscript depends on many factors that no one can reliably predict beforehand and therefore can impact on NER performance.

1.1 Motivation

NER allows to give an overview of those terms that are useful to understand the main theme of the text at hand and to be able to classify it according to the extracted entities. It involves the study and understanding of natural language by an intelligent algorithm that can learn from the common narrative structure that characterizes the texts. With an adequate training process, the system has to detect and categorize entities in texts that it has not handled before. Although it is possible to identify such entities manually by simply reading over the document, this option is not feasible when dealing with large collections of texts such as thousands of comments or reviews written by customers about a product or service, where automation in the detection of relevant entities saves valuable time. Using state-of-the-art artificial intelligence techniques, especially deep neural learning, very promising recognition results have been achieved, making these systems worthwhile to implement and investigate in different domains.

In case of processing images of historical documents, the final transcription may contain errors caused by the program or probabilistic text recognition model. Therefore, it is interesting to analyze how the quality of the transcription affects the detection of named entities when the input text is not exactly the same as the original one, where words can be deleted, mistranscribed or wrongly separated by the HTR system.

The professional motivation stems from the need to improve existing algorithms and propose state-of-the-art solutions to present opportunities in various businesses with the process of digital transformation, saving time and costs. More and more companies opt for the use of natural language processing tools in their daily activities, so providing expertise and knowledge in this field of research can be beneficial. In addition, named entity recognition in ancient manuscript texts has significant applications in the preservation and translation of historical manuscripts, academic research, preservation of cultural heritage, and the creation of linguistic databases for future generations. Despite the challenges involved, continuing advances in artificial intelligence and natural language processing techniques are improving the ability to interpret and understand handwritten texts from past eras for search purposes.

1.2 Objectives

The first objective of this project is to extract and classify named entities by means of natural language processing and deep neural learning techniques. The texts used for the task come from the Books of records of royal decrees, a collection of identity cards stored in the General Archive of Simancas, as seen in more detail in Chapter 4. From the given manually transcribed and annotated lines, an attempt will be made to design and train a deep neural network to recognize named entities and evaluate its performance on a set of test lines. We referrer to these lines as the ground-truth set because they do not contain word errors (i.e. lines are exactly the same as those found in the collection of books).

The second objective is to evaluate the above trained model with the n best transcriptions or hypotheses of the test lines, which are the result of a handwritten text recognition process. Specifically, we have the 10 best hypotheses for each line of the test set. They may have errors in certain words and the intention is to check for the drop in NER performance, while testing the effect of considering more hypotheses for the same line, this is increasing n from 1 to 10. The procedure followed to carry out this experimentation is described in more detail in Chapter 7.

Up to this point the model is trained with the ground-truth lines to extract the entities of the n best hypotheses from the test set. However, there are many faulty words generated by the HTR system that have not been seen during training, causing difficulties for the neural model to detect their correct labels. Given this situation, the third objective is to train the model with the n-best hypotheses of the training set and check if there is an improvement in performance by re-evaluating it with the n-best hypotheses of the test set.

Finally, the effect of introducing a wider context on the final classification performance is investigated. To do this, we first join the lines that form each identity card and partition them into fixed-length word segments. Such segments can consider information from several neighboring lines, which may influence on the correct identification of certain named entities.

1.3 Expected impact

Through this work, it is expected to speed up the process of document searching and retrieving in high volumes texts written in different contents and languages thanks to the most modern techniques of Artificial Intelligence. Instead of manually labeling the entities and storing the information in a database, the aim is to automate this costly work. By achieving a model capable of recognizing named entities trained with texts of a specific language and epoch, it is expected that it can be applied to other domains similar to the one of this work and even be trained with texts of different languages and types of entities. With the sentences labeled by an intelligent system in ancient texts, historians can benefit when querying certain entities whose category may be unknown and key to understand a fact of interest. Furthermore, relying on the predictions made by the model, it is expected that texts or documents can be classified according to the entities that appear in each one and find relationships between them, giving a better understanding of the document.

1.4 Methodology

To obtain the classification results and draw conclusions about the performance of the model in the different experimental scenarios, the following methodology has been used:

1. Read and store the lines of each card of the training, validation and test sets. They contain the manual transcriptions of the handwritten images along with their entity tag.

2. Preprocess the lines to remove punctuation marks, unwanted entity tags and information regarding the structure of the card that are not useful for NER.

3. Analyze basic statistics of the transcripts in the three datasets. In addition, check for the line length distributions in terms of the number of words, entity class distributions and most frequent terms by entity type.

4. Encode characters, words and tags so that they can be interpreted as inputs and outputs by the deep learning model. To do this, index matches have to be defined, allowing the original words to be displayed together with their actual and predicted tags.

5. Due to the framework used for NER, it is necessary to set a sequence length for each line, matching with the input dimension expected by the neural network. All lines have to have the same size and those that do not reach the set length, a padding is added (typically index 0). The same applies for word characters and tag sequences.

6. Define the composition of the neural network, adjust its parameters with the training set, monitor progress with the validation set and evaluate it on the test.

7. After several runs, obtain a final hyperparameter configuration that gives the best possible recognition performance on the test set in terms of the F1 score achieved.

8. With the neural model trained at the line level, evaluate the network with the n best hypotheses of the lines in the test set. For this, the WTER (Word Tag Error Rate) of the predicted labels for each hypotheses will be measured with respect to the ground-truth labels and the F1 score will be used again as a recognition performance guide across all classes.

9. Train the network with the n best hypotheses from the training lines and re-evaluate its performance on the n best hypotheses from the test set.

10. Train and evaluate the network at the identity card level. The manual transcriptions of the lines that compose each card are joined together and a non-overlapping sliding window is applied to create word segments of a specific size that the model will receive. The size of the segments is varied to consider a larger or smaller context and to see what effect it has on NER performance.

1.5 Structure of the work

The content of this work is divided as follows:

- Chapter 2 presents some popular NER competitions organized over the years, works related to the different approaches to the task and related to ancient manuscript documents. It also reviews some of the current state-of-the-art systems and defines the direction the proposal takes for the work in terms of the approaches seen.

- Chapter 3 the task of NER is analyzed in more detail. In addition, compliance with the legal and ethical framework in relation to the protection of the data to be processed is evaluated. Then, some difficulties for NER in handwritten historical documents are pointed out and possible solutions for this task related to the approaches covered in chapter 2 are discussed, indicating the chosen option.

- Chapter 4 describes the origin and properties of the dataset used. An exploratory analysis will also be performed to better understand the text collection available.

- Chapter 5 some fundamental concepts inherent to any system based on deep neural learning such as the one developed are introduced and convolutional and recurrent neural networks are presented, the latter being very popular for NER.

- The main components of the proposed NER-based neural model are described in Chapter 6. It first explains how the words and characters in each line are encoded along with their tags, then the distributed feature representations are highlighted together with their role in the neural network, the context encoder and label decoder components are explained, and a complete view of the model is provided.

- Chapter 7 discusses as part of the experimentation the effect and function of each hyperparameter of the model as well as pointing out the chosen configuration. It also discusses in more detail the selected evaluation metrics for NER and analyzes the final results obtained in the different experiments performed.

- Finally, Chapter 8 concludes the work by analyzing the effectiveness of the proposed NER model. It also refers to the code used for the task, highlights the relationship of the work with the studies undertaken during the master course, the most relevant transversal competencies that have been put into practice and proposes future lines of investigation based on the actual work carried out.

CHAPTER 2

Technological context

The term “named entity” was first introduced at the sixth edition of the Message Understanding Conference for Computer Science (MUC-6) organized in Columbia, Maryland (USA) in 1995 for the evaluation of progress in information retrieval systems. It was there that the main types of named entities including people, organizations, places and certain numerical quantities were established. Since then, there has been multiple events and advances in the mechanisms designed to reveal entities, being applied to texts in different languages and contexts. For example, the systems presented by Curran and Clark [21] to evaluate entity recognition in the CoNLL-2003 dataset consisting of English and German texts implemented different machine learning techniques such as maximum entropy models and hidden Markov models. The ACE [22] program introduced in 2004 focused on the recognition of entities, relationships between them and events in different information sources such as images, audios and texts in Arabic and Chinese, in addition to English. Entity search tasks on the web were performed in the INEX 2009 workshop [23], which consisted in classifying Wikipedia pages by detecting entities in their texts, returning a ranking with relevance of each page and some generative models were created to detect relationships between entities. Another similar event was the TREC 2010 Entity Track [24], where participants had to find the pages with the best match to different queries given based on their named entities present. The FactRuEval information extraction competition conducted in 2016 [25] took a step forward, asking its participants not only to recognize named entities in texts but also to return a list with attributes associated with each entity and to recognize relationships of different types between entities such as occupation and covenant ones.

2.1 NER approaches

The following are the main approaches that have emerged to tackle the NER challenge. More specifically, these are formed by rule-based algorithms, traditional machine learning algorithms and deep neural learning algorithms.

2.1.1. Rule-based approaches

Rule-based methods require the definition of hand-crafted patterns, which are stored in data structures such as a dictionaries or gazetteers, where various entities are registered and organized by category to obtain associated information. In relation to the processing of historical texts based on linguistic rules necessary to understand the nature of the domain, there is a paper by Grover et al. [26] where a system is designed to identify named entities in digitized records of the British parliament between the 17th and 19th centuries.

Recognition results indicated that the recognition system performs better at classifying people rather than places, especially when examined on the older set of records, indicating that the image processing software (OCR) for transcribing the documents contributes to the inaccuracy by not recognizing and distinguishing certain words well. Another approach [27] adapted a set of manual language rules on 18th century US court cases, in which some difficulties associated with entity recognition caused by variations in word order or the appearance of terms not included in the predefined list of rules are shared. The authors in [28] proposed another system of rules, using a gazetteer for entity recognition in a 19th century American newspaper dealing with Civil War issues at the time. With the transcriptions of the pages together with the labels referring to 10 different types of entities, a good recognition result was obtained on places and dates. However, more problems arose with the identification of names of individuals and newspapers. Finally, Diez et al. applied NER to a set of spanish medieval texts (between the 12th and 15th centuries) of different poetic and legal genres, developing a modular architecture that makes use of manually created lexical analyzers to detect common patterns and regular expressions in the texts. In the Figure 2.1, the relationships between system modules are illustrated, showing how the entities found in the analysis module are stored in a dictionary and in a gazetteer through the interaction between the preprocessing and variant term generation modules. The dependency analysis module detects possible relationships between entities such as family and authority relationships, and they are represented in a graph created with XML-TEI syntax.

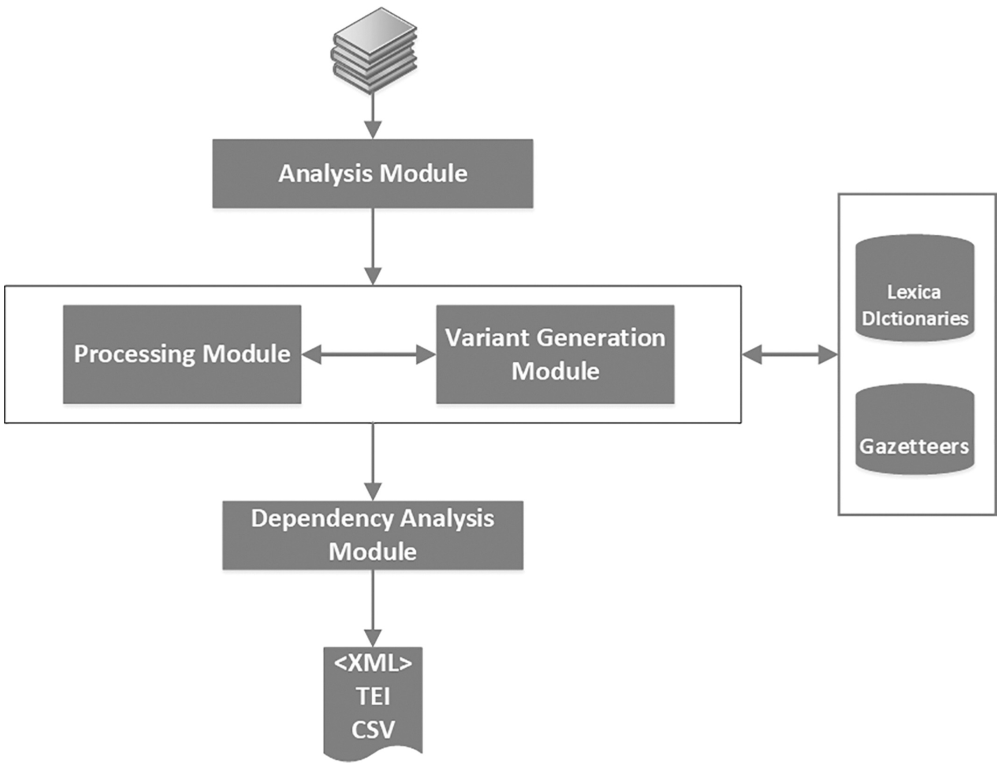


Figure 2.1: Architecture of the NER system based on modules and lexical rules presented by Diez et al [1].

2.1.2. Traditional machine learning-based approaches

Traditional machine learning algorithms analyze the input data to make predictions about the outputs, including supervised algorithms that make use of the labels of the data to adjust their parameters, and unsupervised algorithms that rely only on the input data to make decisions. Ehrmann et al. [29] used application programming interfaces (APIs) and recognition tools implemented with supervised machine learning algorithms applied to old publications of the swiss newspaper Le Temps. The presented classifiers rely on pre-trained language models for the task, evaluating their accuracy on the identification of persons and locations in the archives of different years. The results showed a generalized drop in entity detection performance (especially in people) for older publications, due to the low transcription quality of the OCR software and language variability. Packer et al. [30] designed different entity extraction systems based on ensemble models (set of models) consisting of regular expressions, word dictionaries and maximum entropy hidden Markov models on a collection of commercial documents from several sources. The behavior was compared individually for each model and for the ensemble, where the voting strategy in the ensemble models gave better results. With respect to named entities in medical reports, Keretna et al. [11] proposed a classification procedure by taking samples labeled according to the patient's medical treatment, problem and test. Such a procedure first computes word and context vectors for the reports as a feature extraction method, and then different classifiers including decision trees, nearest neighbors and CRFs were trained by taking the features of each word as input to generate the final prediction in the form of entity tag sequences.

2.1.3. Deep neural learning-based approaches

The latest advances in NER come from deep neural learning, and from there, most sophisticated models that constitute the state-of-the-art in recognition tasks are derived. Like traditional machine learning approaches, these systems rely on vector representations of words and characters to accomplish their goal, but they also have the ability to learn from the context of an entity based on nearby words due to memory units that extract past and future information. Recurrent neural architectures such as those presented by Li et al. [31], make use of these functionalities due to the bidirectional LSTM memory module installed in their layers that allows them to take into account the information before and after the time step in the input sequence. It is also common to find convolutional layers for feature extraction and a final CRF layer that models the dependencies between states or labels. That said, the CRF-layered Bi-LSTM model presented by Huang et al. [32], uses natural language and related context features along with gazetteer information to show the performance increase produced by including these additional sources, as well as incorporating deep neural learning tools. Another similar approach by Chiu and Nichols [2] was presented, in which they used a hybrid Bi-LSTM model with a convolutional network (CNN), based on word and character-level numerical representations on the CoNLL-2003 dataset. The full architecture is illustrated in the Figure 2.2, where first the vector representations of the words and additional vocabulary information are extracted, the CNN is responsible for processing the character-level information of each word to create vectors of a specific dimension, which are concatenated and fed to the Bi-LSTM. A softmax activation function is applied in the output layer of the network to return the final predictions. In contrast to other work, Yang et al. [33], implemented a recurrent neural network capable of performing multiple natural language processing tasks in different languages, including named entity recognition in texts, and without making use of numerical word representations. The network is composed of bi-directional hierarchical GRU memory modules instead of LSTM units, and a final CRF layer. Using the sequential information received from characters, hidden states are obtained in both directions for each position in the word, and the outgoing sequences of the GRU modules are concatenated to obtain the complete word representations. Recently, Wu et al. [34] presented a Bi-LSTM-CRF system with an attention mechanism for capturing long-term dependencies of information in named entity extraction in Chinese clinical texts.

This mechanism calculates the similarity between words by learning a matrix of weights from the outputs received from the Bi-LSTM layer. The system also relies on external features of each word such as its grammatical category or Part-of-Speech tag to collect semantic information. Finally, Devlin et al. [35] presented BERT, a language representation model based on a pre-trained bidirectional multilayer transformer. BERT can be adapted for different natural language processing tasks such as NER or QA, a process known as "fine-tuning". With respect to NER, the results obtained show a competitive performance compared to other state-of-the-art models.

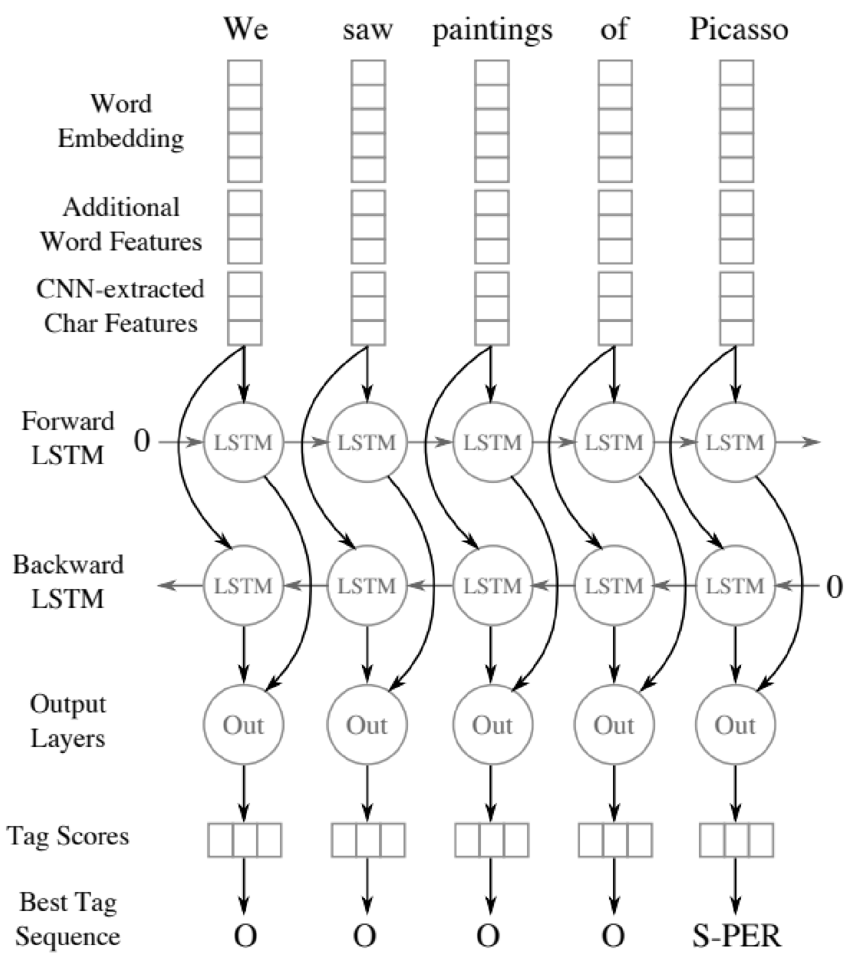


Figure 2.2: Hybrid BiLSTM-CNN neural architecture of Chiu and Nichols [2].

2.2 NER applied to ancient handwritten documents

Many works have applied NER to historical handwritten documents. For example, in [36] natural language processing tools such as spaCy, Stanza and Flair were compared in entity detection in different corpora of medieval letters and records. As in the current work, the NER models were evaluated with manual and automatic transcriptions produced by a handwritten text recognizer, showing some loss in recognition performance. The authors found that line segmentation errors in handwritten text have a greater impact than word recognition errors on the final classification results.

On the other hand, the authors of [37] proposed a transformer-based system for NER, capable of recognizing handwritten text and entities at the same time. They obtained new state-of-the-art results in the ICDAR 2017 competition for information extraction on the Esposalles database [38]. Again, they pointed out that line segmentation errors were sometimes unavoidable and quite affected the model performance. A two-stage learning technique was used, where first the model was trained only for HTR, without passing information from the entity labels, and then a fine-tunnig process was performed by associating the entities in the ground-truth text.

A comparison between two approaches to recognize entities in a corpus of medieval letter images written in different languages was performed in [39]. Specifically, a sequential technique applying first HTR and then NER, and a technique combining the two steps into one was studied. The latter showed better performance in terms of recognition of some entities such as dates, names and locations for the studied languages.

2.3 State of the art review

In most of the neural architectures presented in the state-of-the-art NER task, little emphasis is placed on the descriptive analysis of the document itself, on which the performance of the system is to be measured. Studying beforehand the composition of the texts and identifying frequent and irrelevant terms that may introduce noise to the model can be of great help in order to improve its performance. Thus, data preprocessing is an important step that cannot be missing in NLP tasks.

Like the vocabulary itself, entity tag distributions are not fully studied. In cases of document collections with unbalanced entity categories, the performance of the classifier or neural network may be affected by not giving equal importance to all types in training, so class balancing strategies may be useful in this situation.

2.4 Proposal

The neural model proposed resembles the Bi-LSTM architecture presented by Chiu and Nichols (Figure 2.2), although there are some differences. A convolutional network (CNN) is not used for the extraction of character-level information from each word, but rather short and long-term memory modules (these are referred as character-level LSTMs). In addition, decoding is done by means of a conditional random field (CRF) compared to a standard multilayer perceptron since it is a probabilistic model that achieves high classification performances in many NER tasks.

Most of these systems use representations of both words and characters already pre-trained on large collections of texts, being an advantage since prior knowledge of the language has been extracted to be adapted to another collection with a similar vocabulary. In this case, in the absence of models with distributed word representations from a collection similar to the one to be treated, the proposed neural model learns to generate the numerical representations from scratch, which are initialized with random numbers in a predefined range by an embedding layer.

CHAPTER 3
Problem analysis

The NER task aims to recognize and classify named entities in large text collections whose elements are distinguishable from each other by sharing a common semantic category of interest. Extracting knowledge from these elements involves analyzing the context in which they occur and recording similarities with respect to other terms that can help interpret their meanings. With diverse languages, spelling variations, and rapid vocabulary expansion, named entity processing has become a challenge for many researchers in the field seeking to incorporate state-of-the-art solutions to address these problems. NER techniques have been applied to different domains that consist of specific terminology such as medical, legal or informational, sometimes providing good recognition benefits. In recent years, its study has been widely developed in historical manuscript texts [40, 41, 42] to reveal relationships between entities and assist in the search and retrieval of documents.

A fundamental aspect of developing an NER-based architecture is tag annotations to categorize document terms. They normally vary depending on the collection and are focused on the context in which the events take place, but some of the most common include labels of people, geographical places and organizations, and there may be more specific subcategories within each one. In supervised machine learning models, labels allow them to make predictions about the data and in tasks such as NER, like many others, they are essential for the correct functioning of the system, in this case referring to predefined groups depending on the meaning of words. That said, this is a labeling problem, where from ordered sequences of terms with an assigned position, the system has to be able to return another sequence with the classes (type of named entity) corresponding to these terms. To do this, the system takes the labels of the words as a reference, allowing it to identify differential patterns that help predict the labels of new sequences that it has never seen.

3.1 Analysis of the legal and ethical framework

Next, legal and ethical aspects that must be met are analyzed to guarantee the protection of the data that will be used and fair decision making by the machine learning model. This is important since good practices must be followed in the processing of the available information.

3.1.1. Data protection analysis

Artificial intelligence is the future for many industries and businesses, whose models take large amounts of data to extract the maximum value from it. Much of this data is personal in nature and must be kept private throughout the entire life cycle, from creation to withdrawal. To do this, the instructions imposed by the General Data Protection Regulation (GDPR) must be followed, which includes the fundamental rights and treatment principles.

The data for this work are manual transcriptions of the lines and outputs of a handwritten text recognizer from a collection of ID cards that come from the General Archive of Simancas [43]. Much of the documentation found in the archive is restricted to the general public since it may contain sensitive information and its content can only be consulted for justified research topics. Specifically, the lines used in this work are protected. Therefore, both versions of the transcripts (manual and automatic) have been securely preserved since their collection and have been used only for the agreed purpose.

3.1.2. Ethics

In machine learning, the correct generalization of an intelligent algorithm helps to eliminate existing biases in final decisions. These biases can be introduced unconsciously by the person in charge of designing the system or be present in the input data due to their nature, resulting in controversial predictions. That is why it is necessary to analyze the patterns in the data and identify categories or attributes that are more sensitive to being confused.

3.2 Difficulties in recognizing named entities in historical texts

This type of ancient collection requires a prior processing step to extract its content into a digital format so that it can be understood by a machine. Such task is usually carried out automatically by optical recognition OCR software or by HTR techniques, which generates the best transcription using an acyclic graph called *word graph* [3]. As can be seen in Figure 3.1, this data structure allows modeling different hypotheses or paths about the text of the input image. Each edge is labeled by the corresponding word and weighted by the posterior probability of transition.

An added difficulty in processing historical documents is the amount of noise in the transcription caused by the lack of legibility of some words, causing recognition errors in the system. For example, not detecting the separation between words well or confusing characters due to the variability of typographic styles or fonts. As a consequence, ambiguous terms located outside the vocabulary are produced that can negatively affect classification performance, so the quality of textual information plays a very important role in recognizing named entities. The state of the documents also influences the digitization process since they are collections that are decades or centuries old and over time they may have deteriorated due to environmental factors (temperature, dust, humidity,...), biological factors (microorganisms, insects,...) and human factors. The inconsistency in the structuring and segmentation of the manuscript texts of the time is another factor that influences the quality of the transcription since the text can be mixed up when there are parts of a paragraph that are not well aligned, words located between two lines, drawings on the pages that cannot be interpreted and crossed out words.

In relation to NER, the lack of historical collections related to annotations or specific language models on which the neural network can rely makes this task even more complicated due to the disparity of domains, time periods and languages found in these texts. To this we must add the changes in the language and the variations in grammatical expressions that have arisen over time, which are not that trivial for a NER-based system.

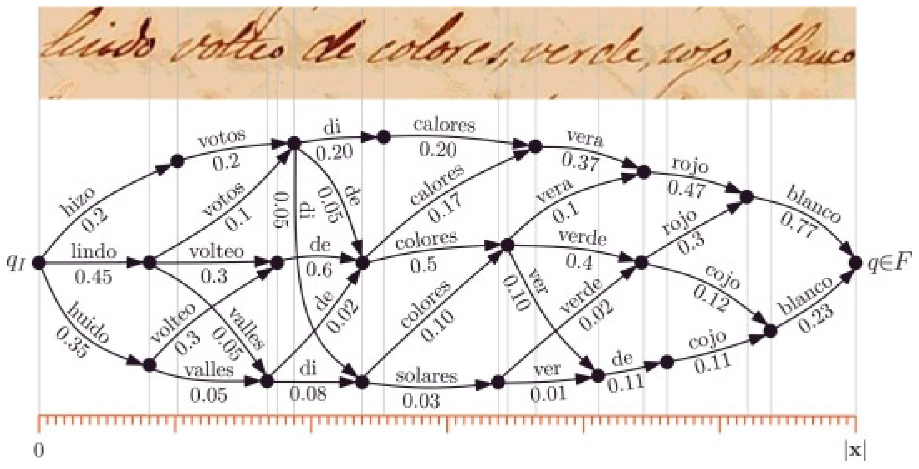


Figure 3.1: Normalized word graph obtained after decoding the line of the image with the text "lindo volteo de colores, verde, rojo, blanco". Source: [3].

3.3 Identification and analysis of possible solutions

When it comes to digital texts, the NER problem can be challenging due to the variability and complexity of natural language. Ambiguity, idioms, and linguistic diversity can make it difficult to accurately identify entities. NLP models must deal with these complexities to achieve optimal performance on the NER task in these texts.

The problem becomes even more challenging when considering the text resulting from a recognizer, such as an OCR system that converts images of manuscripts into digital text. These systems can introduce recognition errors that directly affect the quality of the information extracted. Errors such as confusing similar characters or incorrectly interpreting the layout of text in the image can lead to the misidentification of entities. The problem can be addressed in two phases: first the handwritten text is recognized and then NER is applied with the generated output, which may contain errors. Alternatively, NER may already be integrated into the HTR process. In this work a half-hearted approach is taken; with the HTR outputs that come labeled with the entities, the text is extracted and NER is applied separately by the proposed deep learning model.

The usual recognition of the text involves extracting that path or hypothesis with maximum a posteriori probability, called the *1-best*. However, there can be as many or *n-best* hypotheses, which will allow NER by considering more alternatives of the same transcription.

With text coming from handwritten text recognizers, post-processing and error correction strategies are essential. Combining NER techniques with rule-based automatic correction mechanisms or supervised learning can help mitigate the errors introduced by character recognition and therefore improve entity detection and classification.

As introduced in Section 2.1, there are various strategies for applying NER in texts of any field. The most classic one is formed by models based on grammatical rules, which are created through common context-specific patterns and are usually supported by external resources such as dictionaries and gazetteers, which contain terms grouped by their semantic nature. For example, they consist of sections with all the names of cities, people, surnames, organizations, etc. These rules are often handcrafted and capture mentions of entities with very large vocabularies. An advantage of rule-based models is that they do not require a training and validation set as they do not consist of annotated labels, saving the sometimes considerable tuning time necessary for a traditional machine learning model. However, the rules cannot be generalized to other contexts since they are conditioned by the domain of the dictionary or catalog of proper names on which they have been defined. This means that if a set of rules have been created to recognize and extract names of American cities, they will not work correctly in detecting other cities in documents with a high presence of European locations, for example, then the scope of the vocabulary is a great limitation. To this we must add the time consumed and the difficulty in defining them, since they require knowledge of the specific domain.

In the second block are traditional machine learning models, which are divided into unsupervised and supervised. Unsupervised ones do not use the labels of observations to learn patterns and rely on their similarities to extract entities according to defined groups, such as clustering technique [44]. On the other hand, the supervised ones collect the information from the labels to be predicted and in NLP tasks they are mainly based on feature engineering [45], transforming the raw data into vector representations to be taken by the model. Some of the best-known supervised models that operate with such feature representations are hidden Markov models, decision trees, support vectors and conditional random fields (CRF).

However, in recent years, systems based on deep learning, such as deep neural networks and transformers, have managed to improve the efficiency and precision of the recognition of important entities, achieving optimal results for the state-of-art [46]. These models are capable of identifying more complex hidden relationships in the data, where the level of abstraction in operation is raised due to the layer and block composition of their trainable units. Another advantage of these models is that they can also learn vector representations of words and characters that record properties in a multi-dimensional space, allowing similarities based on the distances of each component. Furthermore, the neural compositions commonly used in NER are considered as recurrent neural networks [47] since they can operate with sequential information and use memory of nearby words to analyze the context by position or timestep, uncovering dependencies between labels using of probabilistic models. In particular, these are the Bi-LSTM models that consist of distributed representations of words and characters, and an output CRF layer, being widely used for this task. Obviously there are multiple variants of this common architecture, especially in the way of extracting and combining feature representations, but the main labeling mechanism still considers the context of each word.

3.4 Proposed solution

Having analyzed the three main groups of models, the architecture designed for this work follows the concept of neural network, dominant in recent years for presenting promising results in NER tasks. Specifically, a recurrent neural model is proposed with bidirectional LSTM (Long Short-Term Memory) or Bi-LSTM cells to learn from the context of words in a phrase or sequence. Numerical representations of words and characters are used, both of which are learned in the training stage. The reason for managing character representations is that the model can infer information from words that are outside the vocabulary or that it has never seen due to morphological similarities. For example, if during the evaluation of the model the unknown word “licenciada” is presented and the same word but in masculine “licenciado” has already been handled in training, with the representations of characters useful information can be extracted from the composition of the word because it shares a common stem of a seen word that has been learned. Finally, a CRF layer will be chosen to decode the labels since it models the transition probabilities between states, in addition to being a very popular option for sequence labeling on which good classification results are obtained.

CHAPTER 4

General Archive of Simancas

The texts used to evaluate the proposed neural model come from the General Archive of Simancas (AGS) [43], located in the Simancas Archive, within the province of Valladolid. It is one of the most important historical archives in Spain and one of the oldest in the world, founded in 1540 by Charles I. In 2017, its collections were recognized as a World Heritage Site by UNESCO, classified as "Memory of the World". Inside (Figure 4.1) there are multiple documents written between the 15th and 19th centuries. Are included:

1. Documents of the royal administration: These are records of government decisions, correspondence of monarchs, officials and ambassadors, as well as documents related to tax collection and resource management of the time.
2. Ecclesiastical documents: The archive also preserves documents related to the Catholic Church in Spain, including parish records and other religious documents.
3. Court documents: Documents related to trials and legal proceedings are kept, providing a detailed view of justice at the time.
4. Cartographic documents: Maps and historical plans that represent different regions of Spain and its colonial possessions.

Some of the specific functions of the AGS are the following:

1. Custody of documents: The archive houses an immense amount of documents related to the administration and history of Spain. These documents include official records, royal correspondence, laws, decrees, maps, plans, and a wide variety of other written materials.
2. Historical research: The archive is a valuable resource for historians, genealogists and academics who wish to study the history of Spain and its colonies. The documents found in Simancas provide a detailed view of politics, society, economics and other aspects of life in different periods of Spanish history.
3. Public access: Although much of the collection of the General Archive of Simancas is reserved for academic research, access is also provided to the general public to consult historical documents. However, it is important to note that some documents may be subject to access restrictions due to their sensitivity or conservation status.
4. Archival Services: The archive offers advisory and training services in document management and conservation to other institutions, as well as individuals interested in preserving their personal archives.

Figure 4.1: Simancas Archive, Valladolid. ICAL. Source: [4].

The importance of the General Archive of Simancas lies in its role as custodian of a significant part of Spain's historical heritage. Its extensive collection of documents is fundamental for historical research and the understanding of the political, social and cultural evolution of the country over the centuries.

Specifically for this work, there are manual and automatic transcriptions of some pages of the collection of Books of records of royal decrees, which, as their name indicates, are volumes that contain the official acts or records of the decrees and real decisions. They are an important part of the General Archive of Simancas and are described in more detail in the following section.

4.1 Books of records of royal decrees

The collection of Books of records of the royal decrees are a fundamental element to know how the archive works. Divided into subseries, they collect in writing (sometimes summarized) and devoid of diplomatic elements such as complete titles, autographs and other validation elements, the real certificates issued in the different legal transactions. There are thus General Books of Navarra, Granada, Aragon, of Accounting and Finance, of Military Orders, of census redemption, correspondence, identity cards, and a subset of books of royal decrees of the house of the Empress Isabel of Portugal.

These books arrived in Simancas in several consignments as part of the documentation of the Chamber of Castile. The first books of royal decrees were incorporated into the archive as part of the documentation of the secretary Francisco de los Cobos in the 16th century. The collection of Books of acts of royal decrees is made up of 377 books, containing 279 894 pages in total. Some pages of the collection are illustrated in Figure 4.2.

Given the importance of the collection, the search and extraction of relevant semantic information from these documents (names, dates, places, ...) is of great interest not only for researchers, but also for the general public. That is why it is expected to locate all the relevant semantic information contained in the documents, allowing searches for many concepts such as people or places.

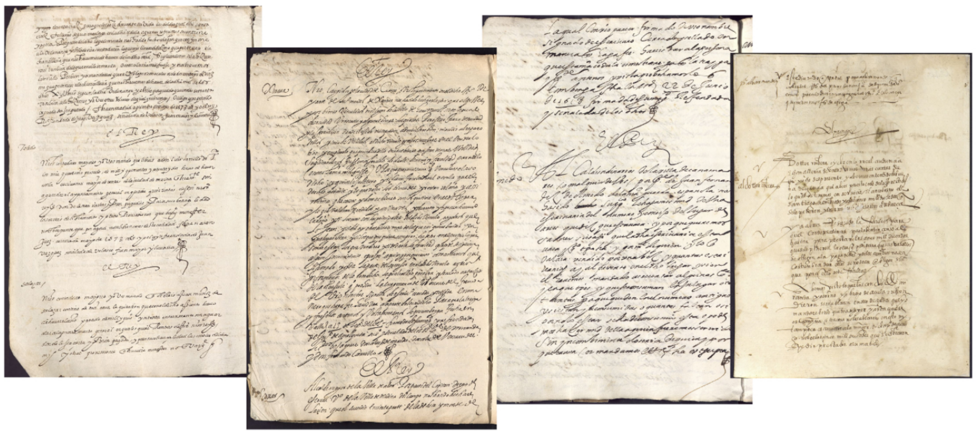


Figure 4.2: Example of some pages from the collection of Books of records of royal decrees.

4.2 Transcriptions of the texts

Firstly, the handwritten texts of certain pages of the collection in digital format have been manually transcribed by expert paleographers and the result has been provided in text files. A representative sample of documents from the Books of records of royal decrees has been selected to evaluate the capacity of the neural network to detect and classify the named entities, which have also been manually labeled. The deep learning model will use this information to adjust its parameters in training. This ground-truth process can be generated using interactive tools available at the Polytechnic University of Valencia. The information present in these texts is important in order to prepare a language model on which to rely.

The collection is structured in ID cards, where each one of them is made up of several lines. These documents were issued in the name of the monarch and had the force of law. They covered a variety of topics, such as business regulations, tax matters, official appointments, among others. Therefore, the cards contain semantic information of interest such as names and surnames of people, places, dates and institutions which the NER model must be able to locate.

For this work, the cards have been divided into two training and test partitions. A small part is extracted from the training set for validation. The transcripts have codes such as "$MI" or "$FCREL" about the structure (the logical layout) of the record. Entity labels are given at the end of each word and are expressed after the special symbol "$" along with the initial of its type in lowercase. For those words that do not have an associated tag, a special one is added (defined as "none"). Below are some lines (unsorted) from an arbitrary card in the training set (the "none" tag is omitted here):

```
CCA_CED_0045_0010.lst80_46 $FC Cámara$o de Su Alteza.
CCA_CED_0045_0010.l3McI_17 $MI Mateo$n Hacera$a.
CCA_CED_0045_0010.l0QQp_52 del chanciller$o y obispo$c y don García$n,
CCA_CED_0045_0010.l4Gtw_28 $FCREL renunciación de Juan$n de$a Azcoitia$a.
```

On the other hand, the transcriptions generated by the handwritten text recognition process are given for all lines. Obtaining the n-best hypotheses in handwritten text recognition involves generating a list of the n best possible interpretations for a handwritten input sequence. Once line images have been converted into recognized text, language models and decoding techniques are used to generate alternative hypotheses. These models can be graph-based, rule-based, or even deep learning-based, depending on the complexity and requirements of the system. During the decoding process, multiple hypotheses or interpretations are generated for the input line image. Each hypothesis has an associated score that reflects its probability of being the correct interpretation. The scores are based on a word graph (Figure 3.1), where the edges are labeled with words and weighted by the transition probability between them, and the nodes contain the segmentation boundaries of words in the image. This data structure allows modeling a large number of most probable word sequences and segmentation hypotheses in a compact and convenient way. That said, the n best hypotheses are chosen based on their scores, which represent the n most likely interpretations of the input sequence. In this case, we have the 10 best hypotheses for the training and test sets. They are ordered in descending order according to their probability. For example, for the ground-truth line: "renunciación de Juan de Azcoitia", the 10 best hypotheses generated are the following:

```
CCA_CED_0045_0010.l4Gtw_28 $FCREL renunciación de Juan$n de$a Ccotia$a.
CCA_CED_0045_0010.l4Gtw_28 $FCREL renunciación de Juan$n de$a cotia$a.
CCA_CED_0045_0010.l4Gtw_28 $FC renunciación de Juan$n de$a Ccotia$a.
CCA_CED_0045_0010.l4Gtw_28 $FC renunciación de Juan$n de$a cotia$a.
CCA_CED_0045_0010.l4Gtw_28 $FCREL renunciación de Juan$n de$a Ccotra$a.
CCA_CED_0045_0010.l4Gtw_28 $FCREL renunciación de Juan$n de$a cotra$a.
CCA_CED_0045_0010.l4Gtw_28 $FCREL renunciación de Juan$n de$a Ccotia$a
CCA_CED_0045_0010.l4Gtw_28 $FC renunciación de Juan$n de$a Ccotra$a.
CCA_CED_0045_0010.l4Gtw_28 $FCREL renunciación de Juan$n de$a cotia$a
CCA_CED_0045_0010.l4Gtw_28-10 $FC renunciación de Juan$n de$a cotra$a.
```

It should be noted that the handwritten text recognizer generates labels referring to the Named Entities, that is, it applies NER together with HTR. However, these entity tags will be ignored as NER is applied separately by the proposed system for this work. Furthermore, it is possible to find lines with less than 10 hypotheses.

4.3 Exploratory data analysis

Exploratory data analysis is a necessary step to better understand the collection of texts available. It involves the extraction of knowledge prior to the modeling itself in such a way that relevant patterns can be identified through exploratory tables and graphs. With each line labeled, some of the elements to analyze include word vocabulary, entity categories, and line and ID card lengths.

Before doing so, various tasks have been carried out to pre-process the text and eliminate as much noise as possible that the neural model can receive. Specifically, it has been removed:

- Labels referring to the structuring of the ID card (e.g. "FCREL","FC","MI") as they do not influence NER.
- Common punctuation symbols present in the transcripts such as full stops, commas, dashes, and hash marks.
- Tags that are not of interest introduced in the ground-truth labeling process. For example, the "tachado" category for crossed out words in handwritten texts or "rúbrica" for highlighting sections of text. For simplicity, these are replaced with "none" tags.
- Empty lines remaining after applying the previous steps.

After pre-processing the lines, the text, labels and identifiers have been stored separately so that they can be managed to train the neural model and run the desired experiments.

Some basic statistics from the manual transcriptions of the lines are illustrated in the Table 4.1. A total of 8 366 lines remain. 67% of them are concentrated in the training set, 8% in the validation set and 25% in the test. There are on average 35 (± 8) lines per card and 10 (± 4) words per line in all partitions. Each card contains on average 340 (± 147) words. There is a higher proportion of unique words in validation of 26.3% compared to 10.4% in training and 14.5% in test, although it may be related to the size of each partition in terms of the running words. It must be stated that the vocabulary is quite reduced, consisting of 6879 unique terms across all partitions. In addition, the percentage of words outside the vocabulary (OOV) is appreciated, considered as the percentage of running words that are not defined in the training set and that therefore will be unknown by the model in inference. A high number of OOV words in the test set can be a drawback since the neural network will not have learned from the word embeddings of these terms and can only infer their meanings through the context, which can sometimes be ambiguous. However, learning from character representations and their order in the word can be helpful in such cases when a common stem is shared with another word that has been observed during training. There is a similar number of unique characters in all three sets, which are primarily alphanumeric characters.

Table 4.1: Statistics of ground-truth transcripts according to the partition.

	Train	**Validation**	**Test**
Cards	155	19	61
Lines	5 637	626	2 103
Running words	54 354	5 850	19 842
Unique words	5 633	1 539	2 886
OOV running words (%)	0	5.20	6.63
Characters	236 456	25 366	85 885
Unique characters	76	67	73

The most frequent terms in the collection are illustrated in the table Table 4.2. In it, the word "de" is the most dominant, forming 8.9% of the total words. Generally, these frequent terms in the collection do not provide information about any entity although they are widely used in the language. The interesting thing here is to see that the frequency of each term is inversely proportional to its rank, following Zipf's law [48] which confirms that few words appear quite frequently while the majority of words are mentioned a few times. This means that the product of the rank and the frequency of a term tends to a constant value, and as can be seen in the Table 4.2 for the most frequent words, it is close to 9000. This constant represents the gradient of the ordered frequency distribution on a logarithmic scale for both term rank and frequency (Figure 4.3).

The distribution of the number of words in the ID cards is shown in the histogram of the Figure 4.4. A high variability is seen, where the shortest records contain over 100 words and the longest ones has almost 1000. The lengths of most cards vary between 200 and 400 words, where each one has over 30 lines of text. The longest one contains 67 lines.

The histogram with the distribution of line lengths in terms of words is illustrated in Figure 4.5. Mostly the lengths vary between 1 and 25 words. As stated before, the lines contain over 10 words on average.

Table 4.2: Most frequent terms in the ground-truth set.

Term w	**Rank** r	**Frequency** f	$P(w)$	$r \cdot f$
de	1	7 132	0.089	7 132
y	2	4 213	0.053	8 426
que	3	3 220	0.040	9 660
la	4	2 138	0.027	8 552
en	5	2 021	0.025	10 105
a	6	1 517	0.019	9 102
los	7	1 166	0.015	8 162
por	8	1 140	0.014	9 120
el	9	1 132	0.014	10 188
del	10	1 047	0.013	10 470
dicho	11	859	0.011	9 449
se	12	839	0.010	10 068
para	13	716	0.009	9 308
lo	14	669	0.008	9 366
Y	15	669	0.008	10 035
las	16	615	0.008	9 840
con	17	539	0.007	9 163
dicha	18	533	0.007	9 594
o	19	457	0.006	8 683
os	20	446	0.006	8 920

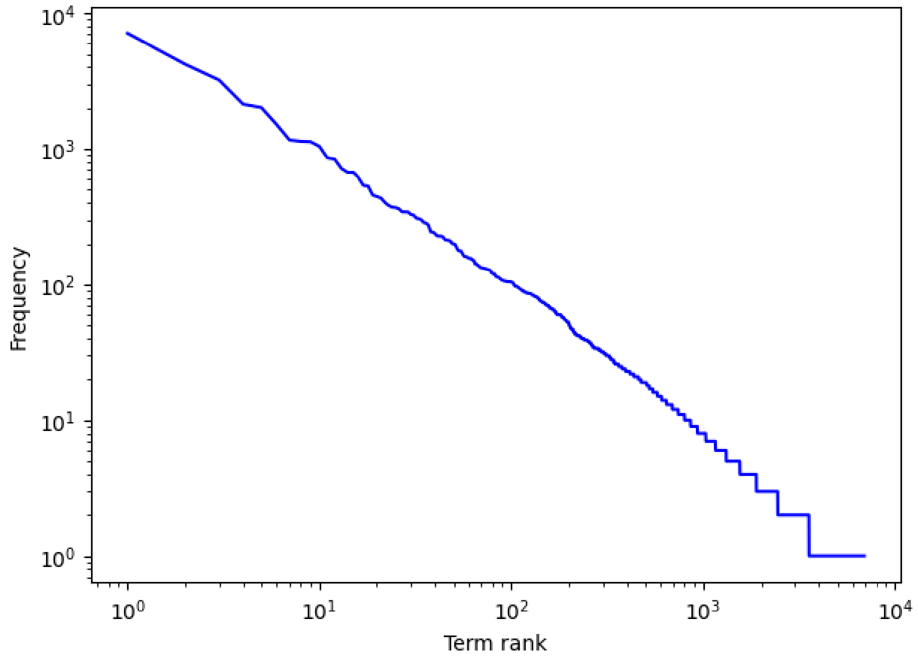


Figure 4.3: Frequency of each vocabulary term in relation to its rank, on a logarithmic scale on both axes. Zipf's law is fulfilled, where the frequency is inversely proportional to the rank.

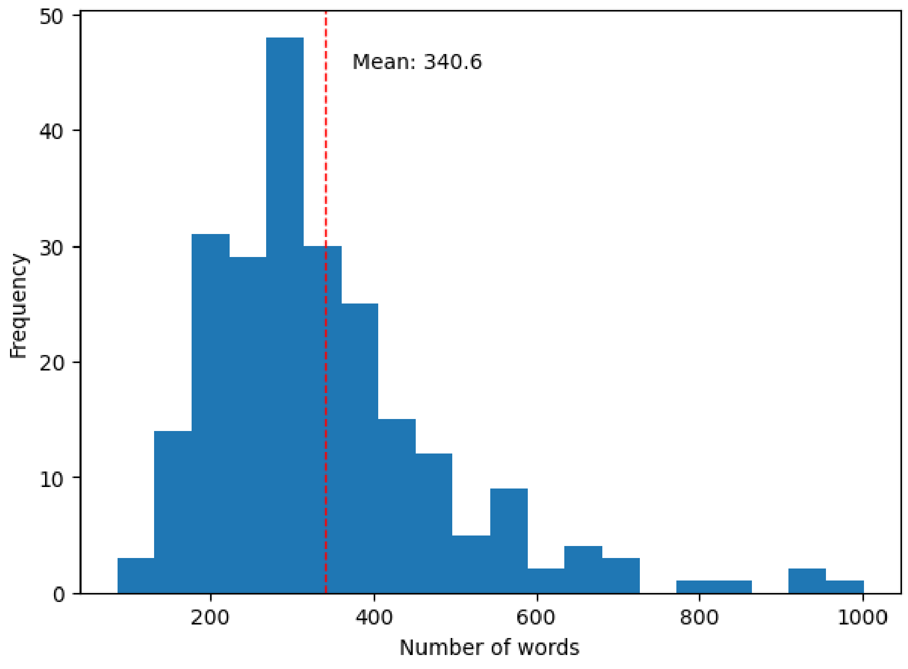


Figure 4.4: Distribution of ID card lengths in terms of the number of words.

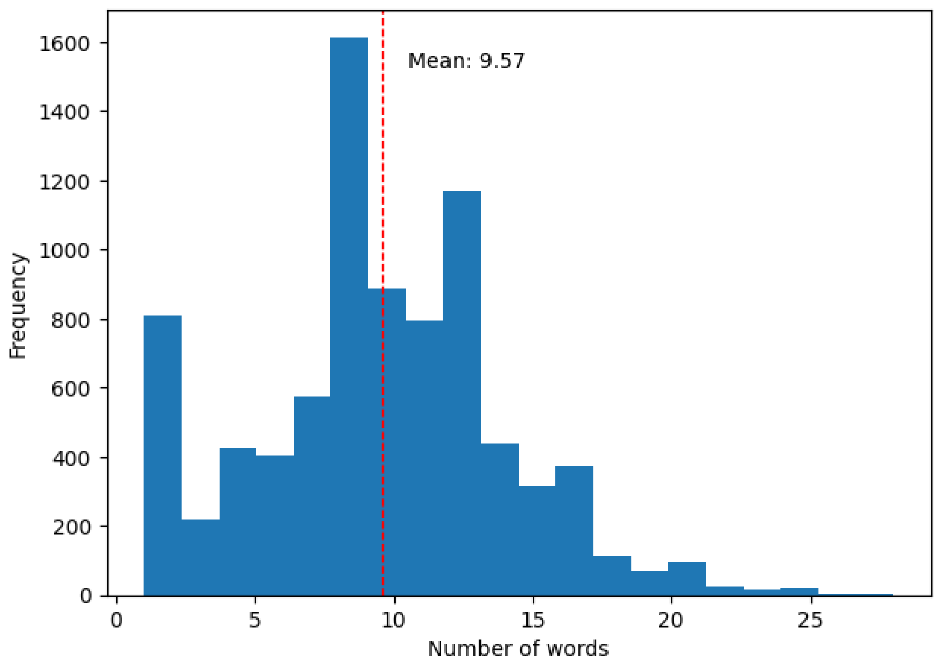


Figure 4.5: Distribution of line lengths in terms of the number of words.

The frequencies of the entity tags are observed in the Table 4.3. There are a total of 14 different tags, considering the category of "none" or words that do not refer to a specific type of entity, which is the most abundant of all, as expected. As for the rest, semantically similar categories can be found such as offices, positions and jobs that can easily be confused by the NER model, as well as dukes, counts, marquises and barons, which represent positions in the hierarchy of noble titles. Locations and institutions are also used in similar contexts throughout the collection. Additionally, the most frequent unigrams per entity type are shown in the tables 4.4, 4.5 and 4.6. There is a high presence of male names and compound surnames. Highlight terms like “de”, “del” and “y” that occur frequently in different classes. In these cases, learning from the context is key to finding the correct label assignment. Common locations include Madrid, Valladolid and Granada. There is a very small vocabulary of jobs, with only graduates and doctors.

Table 4.3: Absolute and relative frequencies of entity labels.

Entity	Frequency	Proportion
None	68 777	0.859
Surname	3 102	0.039
Office	1 741	0.022
Position	1 653	0.021
Name	1 552	0.019
Location	1 307	0.016
Institution	1 002	0.013
Roman numeral	462	0.006
Job	205	0.003
Count	102	0.001
Marquis	63	0.001
Duke	52	0.001
Cross	25	< 0.001
Baron	3	< 0.001

Table 4.4: Most frequent unigrams labeled as names, surnames, locations and institutions.

Name	*f*	Surname	*f*	Location	*f*	Institution	*f*
Juan	360	de	877	Madrid	117	de	201
Francisco	118	Vázquez	75	Valladolid	65	Cámara	58
Diego	114	Pérez	47	Granada	65	Consejo	39
Pedro	108	Velasco	46	de	48	la	29
Alonso	71	la	35	Burgos	32	Orden	29
Antonio	47	Carbajal	30	Sevilla	31	Santiago	25
Fernando	41	Cobos	30	El	26	concejo	21
Lope	34	y	28	Toledo	21	y	21
Luis	31	Guevara	26	Castilla	21	Monasterio	20
Martín	31	López	23	del	20	iglesia	19

Table 4.5: Most frequent unigrams labeled as offices, positions and jobs.

Office	*f*	Position	*f*	Job	*f*
de	195	del	138	licenciado	105
escribano	86	Consejo	127	doctor	39
y	77	de	119	Licenciado	19
secretario	67	nuestro	72	bachiller	19
la	64	los	69	licenciados	15
del	61	y	59	Doctor	5
nuestra	37	corregidor	43	cenciado	1
Casa	32	comendador	40	chiller	1
tesorero	31	general	35	Bachiller	1
número	30	presidente	31		

Table 4.6: Most frequent unigrams labeled as counts, marquises and dukes.

Count	*f*	**Marquis**	*f*	**Duke**	*f*
conde	33	de	18	duque	13
de	28	marqués	16	de	10
condesa	7	Marqués	9	del	5
Conde	3	Almazán	5	Infantado	5
Lemos	3	Mondéjar	3	Duque	4
Benavente	3	Este	2	Medinaceli	3
Luna	2	Garras	2	Medina	3
Barajas	2	La	1	Sidonia	3
Cabra	2	Guardia	1	Alburquerque	2
Torralva	2	Cañete	1	Lerma	1

CHAPTER 5
Deep neural learning

This chapter introduces some concepts of deep neural learning that are important to understand the essentials of this technology that has become the focus of attention for most modern NER systems. The most well-known architecture variants in this field are also presented, including recurrent neural networks, widely used for the processing of sequential data such as the words that make up a sentence or paragraph.

Deep learning is a branch of machine learning whose algorithms are inspired by the neural composition of the human brain. Specifically, these are neural networks and their objective is to imitate the characteristic behavior of any human being when learning from a totally new situation. Neural networks manage and analyze large amounts of data to accelerate decision-making processes. Some of the tasks for which deep learning is used are for the detection of spam, classification and recognition in images, prediction of stock prices, recognition of named entities in texts as in this case or even in the implementation of autonomous vehicles. Compared to traditional machine learning algorithms, neural networks are capable of extracting patterns that are harder to identify and have an intelligent method of learning from their own mistakes and adjusting their parameters.

5.1 Neural network composition

The basic units that make up neural networks are neurons or simply nodes, and they are organized in layers that make up the main architecture [49]. The three different types of layers are: the input layer, the hidden layer and the output layer. The input layer is where the data from which the prediction will be obtained is received. The hidden layer is responsible for storing a series of unknown characteristics of the incoming data that are useful for obtaining relevant and complex patterns. Finally, the output layer contains in each node the value resulting from applying all the necessary operations on the incoming information. As seen in Figure 5.1, the units of each layer are densely connected to the units of the next layer by a set of links. Typically, there is more than one hidden layer to capture additional relationships that require more processing, and this type of network is called a multi-layer network. The links have associated weights that determine the values that the nodes will occupy in the hidden and final layers, being the parameters that have to be fitted during training. A neural network is therefore a mathematical model that receives in each iteration a set of samples and modifies its internal parameters based on their values.

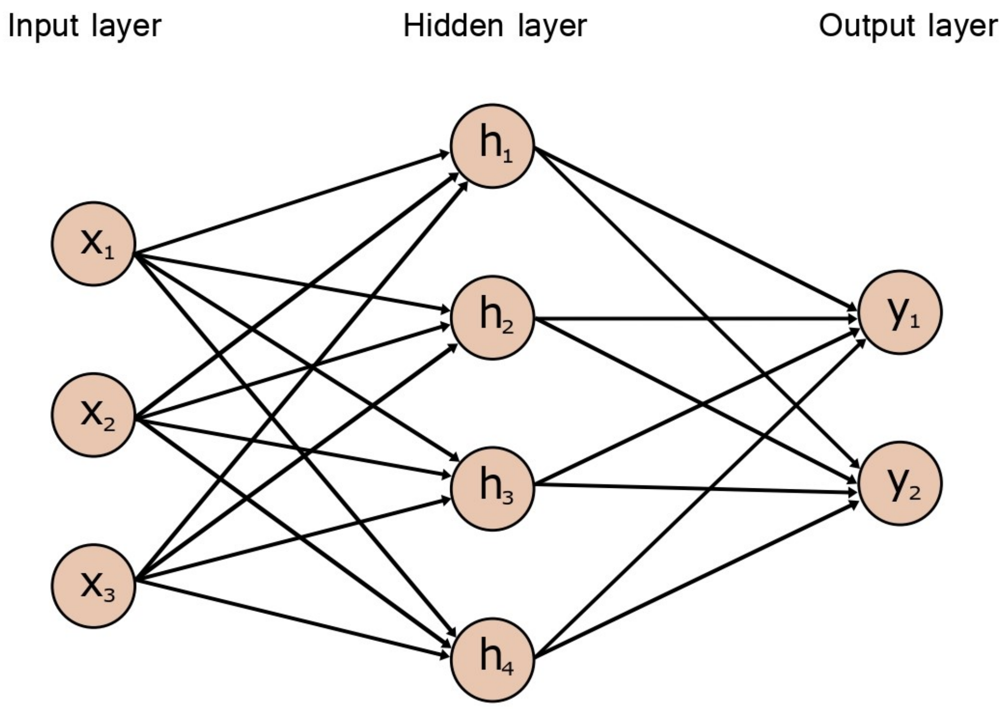


Figure 5.1: Classic composition of a neural network in its layers.

5.2 Forward pass

Once some input values are introduced to the neural network, the nodes of the input layer are initialized along with the weights of all the connections and the forward calculation process begins, which consists of sequentially determining the outputs of all the nodes in each layer until obtaining the values in the output layer. To do this, matrix multiplications are carried out between the outputs of each layer and the weights that connect them to the next layer. That is, when a hidden or output node receives the inputs from the previous layer, they are multiplied by their respective weights and added. In addition, the weight of a bias term whose value is constant is applied to this sum. The results go through an activation function that decides whether or not they are relevant to contribute to the prediction. It acts as a transformation function that normalizes the outputs of the nodes of a layer and helps in the gradient descent process discussed in the next section. The most common are the linear, the *ReLu*, which acts as a linear if it detects a positive value but converts the negative ones into zeros, the sigmoid which is a non-linear function that normalizes the input values between zero and one, and softmax function which transforms the outputs of a layer into probabilities and is often used for classification problems with multiple categories.

The output s_i^n of a node i of layer n , with $1 \leq i \leq M_n$ and $1 < n \leq N$, where M_n is the number of nodes in the layer n and N is the total number of layers, it is calculated as follows:

$$s_i^n = g(\sum_{j=1}^{M_{n-1}} \Theta_{i,j}^n \cdot s_j^{n-1}) \text{ , where:}$$

- $g(x)$ denotes an activation function on x,
- $1 \leq j \leq M_{n-1}$, where M_{n-1} is the number of nodes in layer $n-1$,
- $\Theta_{i,j}^n$ is the weight of the link that connects node j with node i of layer n,

- s_j^{n-1} is the output of node j of layer $n-1$.

When results are obtained for the output layer nodes, a prediction is generated for the input data. Depending on the problem, whether classification or regression, the number of outputs returned varies. For example, the presence of multiple values usually indicates the probability of belonging to one class or another, choosing the highest one as the final decision, and in neural networks with only one output value returned they can refer to a binary classification or a numerical prediction.

5.3 Loss function

An important factor in training a neural network is the loss function selected to evaluate the predicted outputs. The loss function quantifies how different the expected or actual outputs are from the predictions generated by the forward propagation process, calculating a prediction error. In each epoch where all the training samples pass through the network, the average loss is stored to observe its evolution. As in any optimization problem, it is about minimizing or maximizing the value of an objective function and obtaining the set of decision variables that are best placed to solve the problem. In the context of neural networks, the aim is to obtain that composition of weights that produce the least possible error.

In classification problems where the aim is to establish probabilities of belonging to each class, cross-entropy loss functions are often used that indicate how the predicted probability distributions differ from the authentic ones. In a binary classification, the binary cross entropy BCE for a sample x is determined by:

$$BCE(x) = -y(x) \cdot \log p(x) - (1 - y(x)) \cdot \log(1 - p(x)) \text{ , where:}$$

- $y(x)$ indicates the actual class to which x belongs (0 or 1),
- $p(x)$ is the probability that x belongs to class 1,
- $1 - p(x)$ is the probability that x belongs to class 0.

For classification problems with more than two classes, categorical cross entropy is used, which separately calculates the loss in each class and adds them together. Given a sample x, its categorical cross entropy CCE is calculated as follows:

$$CCE(x) = -\sum_{i=1}^{M} y_i(x) \cdot \log p_i(x) \text{ , where:}$$

- M is the total number of classes,
- $y_i(x)$ indicates whether x belongs to class i (1) or not (0),
- $p_i(x)$ is the probability that x belongs to class i.

As can be seen in the formula, low predicted probabilities for the correct class increase the error more as it is a logarithmic function that penalizes large differences between $p_i(x)$ and $y_i(x)$.

Mean square error (MSE) and mean absolute error (MAE) are often used to measure forecast error in regression problems that require a numerical quantity. The formulas for both loss functions are as follows:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

Where:

- N is the total samples,
- y_i is the real value of the sample i,
- $\hat{y}_i$ is the predicted value of sample i.

In addition to the loss functions mentioned, others that are commonly used are the Hinge loss and the Kullback-Leibler divergence for classification, and the Huber loss for regression.

5.4 Backpropagation

During the learning process, the neural network must update its weights every time it obtains predictions about the input data, whose changes are given by the loss function. This procedure is called backpropagation and indicates the direction in which the network weights have to be moved to approach a minimum loss value using a technique known as gradient descent [50]. The objective is to adjust these weights using the value of the loss function as a reference so that in the next training iterations the network outputs are closer to the real outputs, having to calculate the error in all nodes except in the layer input.

The weights are updated by layers in reverse order, that is, from the output to the input. First the network calculates the prediction error and the derivative of the activation function for each node in the output layer. Multiplying both results, the error is established at the node i, δ_i^2 (assuming a multilayer perceptron with an input layer, a hidden layer and an output layer):

$$\delta_i^2 = g'(s_i^2)(t_i - s_i^2)\text{ , where:}$$

- $g'(s_i^2)$ is the derivative of the activation function at the output layer over the output at node i, s_i^2,
- t_i is the actual value at node i.

With these errors, the outputs of the nodes of the previous hidden layer s_j^1 and a learning factor p, the change that occurs in each weight Θ_{ij}^2 is determined :

$$\Delta\Theta_{ij}^2 = p\delta_i^2 s_j^1$$

Therefore, the new weight Θ_{ij}^2 will be:

$$\Theta_{ij}^2 = \Delta\Theta_{ij}^2 + \Theta_{ij}^2$$

The calculation of the new weights of the hidden layer Θ_{ij}^1 follows a similar process. In this case, the errors of the hidden nodes δ_i^1 are calculated by multiplying their derivatives in the activation function by the previously calculated errors of the output layer δ_r^2 per node r, together with the weights Θ_{ri}^2:

$$\delta_i^1 = g'(s_i^1)\sum_{r=1}^{M_2}\delta_r^2 \cdot \Theta_{ri}^2\text{ ,where:}$$

- $g'(s_i^1)$ is the derivative of the activation function in the hidden layer over the hidden output s_i^1,

- M_2 is the number of nodes in the output layer.

Again, the outputs of the previous layer are considered, which are the input data x_j, and the same learning factor p to update the weights:

$$\Delta\Theta_{ij}^1 = p\delta_i^1 x_j$$

So the new weights are:

$$\Theta_{ij}^1 = \Delta\Theta_{ij}^1 + \Theta_{ij}^1$$

With these calculations, all the parameters of a multilayer perceptron have already been updated. Although the error improvement with this new parameter configuration may not be entirely significant, the network repeats the backpropagation process with new data it receives and in more iterations. An important factor to control in this step is weight overfitting, which occurs when the neural network pays too much attention to the set of training samples presented to it and is therefore not able to generalize well to other data that it has never seen before. To address this problem, a set of validation samples is defined on which only forward propagation is performed to obtain their predictions and measure the error, but no backpropagation is done to adjust weights. Signs of model overfitting appear when the evolution of the training error progresses adequately while remaining the same or not improving for the validation set.

5.5 Types of neural networks

Once described the composition and classical learning mechanisms of neural networks, other popular variants will be presented that are created for more specific tasks and that, compared to a simple multilayer perceptron, tend to obtain better performance when evaluating other attribute structures such as images or data sequences. The types of networks to highlight are convolutional neural networks and recurrent neural networks.

5.5.1. Convolutional neural networks

Convolutional neural networks (CNNs) specialize in image feature processing and extraction tasks or in computer vision, although they are also used in speech recognition and automatic language translation. Unlike classical neural networks that operate with one-dimensional input data vectors, CNNs receive data or pixels in three dimensions, each referring to the height, width and number of channels (one for grayscale and three for RGB).

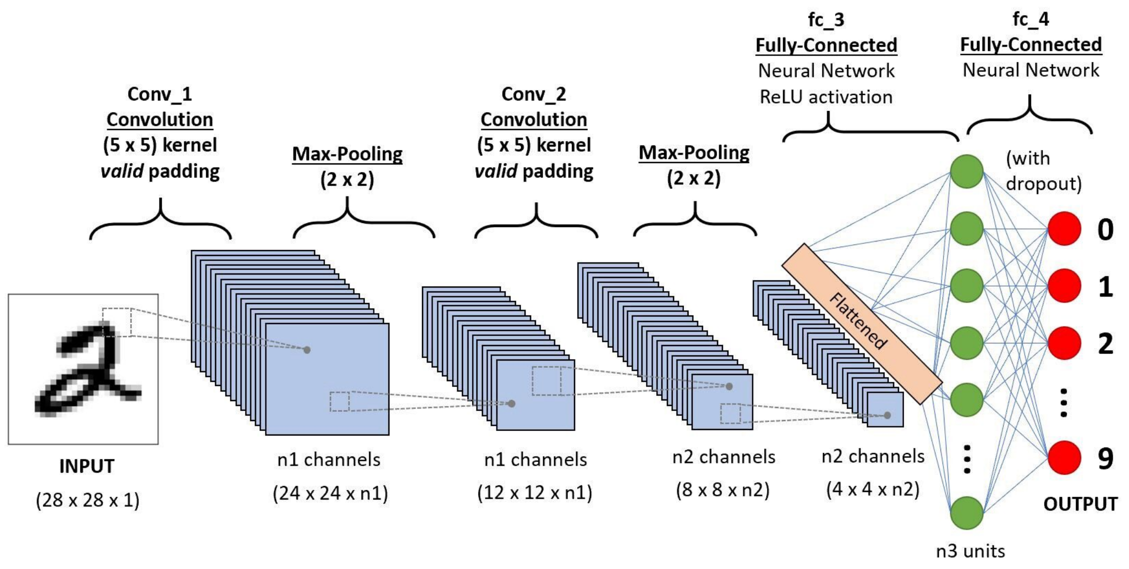


Figure 5.2: Architecture of a convolutional neural network for image classification. Source: [5].

Given an input image, the convolutional network passes a series of filters over it that capture different properties. As seen in Figure 5.2, the objective is to aggregate the information received in a way that reduces the number of elements that have to be operated and at the same time help classify the content from image. Each layer in the network is responsible for processing certain types of features, from low-level ones like edges or corners, to higher-level ones that include objects and shapes. The three most common types of layers are the convolutional layer, the pooling layer, and the densely connected layer.

5.5.2. Recurrent neural networks

Recurrent neural networks (RNNs) are a type of neural network that processes temporal or sequential data, being widely used especially for tasks related to natural language processing such as sentiment analysis, machine translation, speech generation and text tagging, including recognition of named entities. A classical neural network receives inputs that are independent of each other and that correspond to samples of the data set with the purpose of returning predictions associated with each one, but sometimes there may be dependencies between these observations that it is not able to model, giving rise to to recurring networks. What makes these network variants different is that they have memory of prior information that helps determine predictions about data at later positions within the sequence. Related to this, RNNs can also forecast the behavior of time series where identifiable cyclical patterns and trends exist.

The essence of any RNN is to store a series of hidden states associated with each time step to not only be used as the output of that position, but also to serve as input to calculate the next hidden state of the step in the sequence (Figure 5.3). This means that a decision is made by combining the input values from the current step and information carried over from previous inputs. As in artificial neural networks, all inputs and outputs are weighted by weight matrices but now these are the same in all temporal layers. The weights are adjusted during training by backpropagation through time (BPTT) process based on gradient descent.

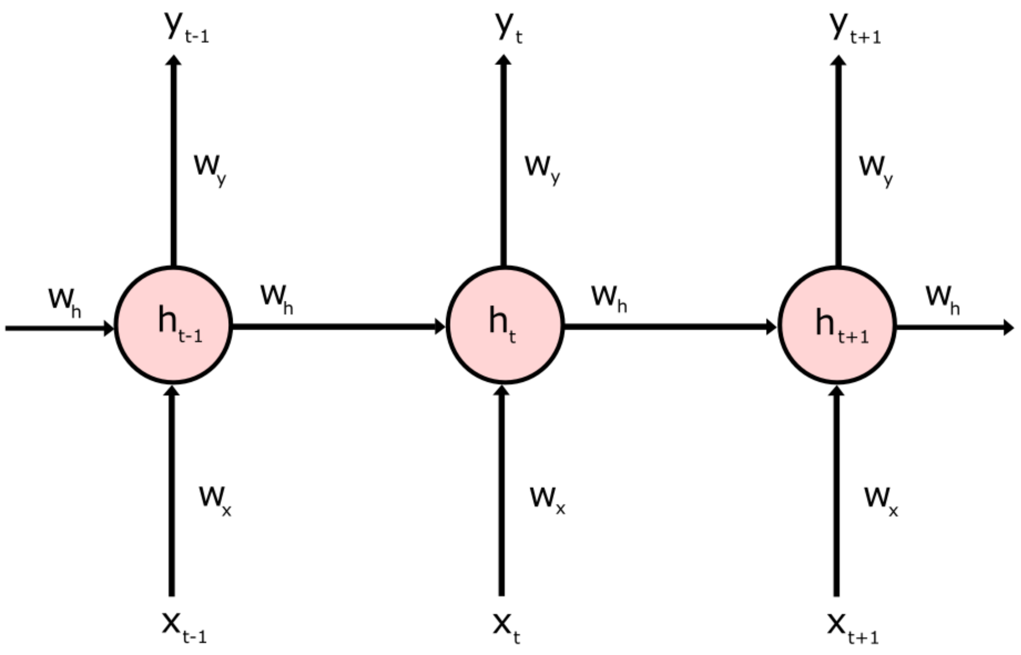


Figure 5.3: Example of a many-to-many composition of an RNN.

Backpropagation through time

The weight updating mechanism in an RNN works the same as in an artificial neural network, where first the forward step is carried out in the network to obtain outputs and a prediction error at each position of the sequence, and then the backward step to determine how the weights have to be modified based on the derivative of the error. The difference is that previous temporal dependencies are considered and the error accumulates for each step in the sequence. Assuming that there is only one layer of hidden states, there are three weight matrices to consider:

- W_x: Represents the weight matrix that connects the inputs at each step x_t with the hidden state layer,
- W_y: Represents the weight matrix that connects each hidden state h_t with its output y_t,
- W_h: Represents the weight matrix that connects the hidden state of one step to the state of the next step.

Before seeing how backpropagation works over time, we must check how each output y_t of the recurrent network is determined along with its errors E_t. First, the hidden state h_t is calculated by applying an activation function on the sum of the current step input x_t and the previous hidden state h_{t-1}, weighted by their respective matrices:

$$h_t = g(x_t \cdot W_x + h_{t-1} \cdot W_h)$$

The output y_t is then the product of the resulting vector in h_t with the weight matrix W_y, where an activation function can also be applied on the result:

$$y_t = h_t \cdot W_y$$

With y_t and the actual or expected output d_t, the error E_t is determined depending on the selected loss function. For example, with the quadratic difference it is calculated as:

$$E_t = (d_t - y_t)^2$$

To update each matrix, it is necessary to find the partial derivatives of the loss function at a specific time step with respect to the weight matrix itself. The above time steps are considered when modifying each matrix using gradient descent. That being said, the calculation to adjust W_y is as follows:

$$\frac{\mathrm{d}E_t}{\mathrm{d}W_y} = \frac{\mathrm{d}E_t}{\mathrm{d}y_t} \cdot \frac{\mathrm{d}y_t}{\mathrm{d}W_y}$$

In case of W_h, we must consider all the hidden states that contribute to the output y_t and therefore their gradients accumulate:

$$\frac{\mathrm{d}E_t}{\mathrm{d}W_h} = \sum_{i=1}^{t} \frac{\mathrm{d}E_t}{\mathrm{d}y_t} \cdot \frac{\mathrm{d}y_t}{\mathrm{d}h_i} \cdot \frac{\mathrm{d}h_i}{\mathrm{d}W_h}$$

As in W_h, to fit W_x one needs to consider all hidden state contributions to the output y_t:

$$\frac{\mathrm{d}E_t}{\mathrm{d}W_x} = \sum_{i=1}^{t} \frac{\mathrm{d}E_t}{\mathrm{d}y_t} \cdot \frac{\mathrm{d}y_t}{\mathrm{d}h_i} \cdot \frac{\mathrm{d}h_i}{\mathrm{d}W_x}$$

A problem with this weight adjustment method is that the error gradient can grow uncontrollably over the network as a consequence of there being multiple hidden states on which their derivatives are calculated (a problem known as gradient explosion [51]). In this situation, what is usually done is to establish a maximum threshold for the gradient and if it is exceeded, a normalization is applied (gradient clipping). Conversely, if the error is propagated over many time steps, its accumulated gradient may be very small and the contribution of later steps is lost as the sequence progresses. This is the gradient vanishing problem [52], which is very common in any RNN and can cause the loss of long-term temporal dependencies. Long short-term memories or LSTM were specifically designed to solve this problem, being used in the designed NER model and will be discussed in greater detail in the next chapter.

CHAPTER 6

Model Description

This chapter will describe the components of the proposed neural architecture and the interaction between them to achieve the recognition and classification of the named entities in the collection of ID cards inside the Books of records of royal decrees. The components that make up the network are:

1. A layer of distributed feature representations formed by high-dimensional vectors of real numbers, for both words and characters. These vectors are known as **word embeddings** and **character embeddings**, and they are learned in the training process.

2. The context encoder that uses the representations of each word and its component characters to extract knowledge based on other nearby words in the input sequence. It is made up of two LSTM layers (**Bi-LSTM**) that process information in both directions of the sequence and thus try to capture the context in each word position.

3. The tag decoder to obtain the best output sequence of named entity tags, using a **Conditional Random Field** (**CRF**).

6.1 Representation of text and labels

Before explaining the operation of each component, it is convenient to point out how the textual information input to the model and the labels to be predicted are represented. First, each word on a line is assigned a unique numerical value or index to be encoded, and these correspondences are stored in a dictionary. This operation is necessary since neural networks always operate with numerical values and not with strings of words, therefore omitting this step will result in an error during training. Once the dictionary is established, the indexes are applied to the words in all lines to obtain numerical sequences.

Since the lines have different numbers of words and the input layer to the network is of a fixed length, all sequences must necessarily have the same length and match with the expected input dimension, so a length must be commonly established for all of them. This length is a hyperparameter of the model that has to be set before training, although in the previous histogram of Figure 4.5 it has been verified that the majority of lines have a length of at least 30. For those sequences that do not reach the agreed size, a common technique called *padding* is used, which consists of filling them with a reserved index (usually zero) that does not represent any specific term. Figure 6.1 illustrates the application of the entire representation process on a line of the collection.

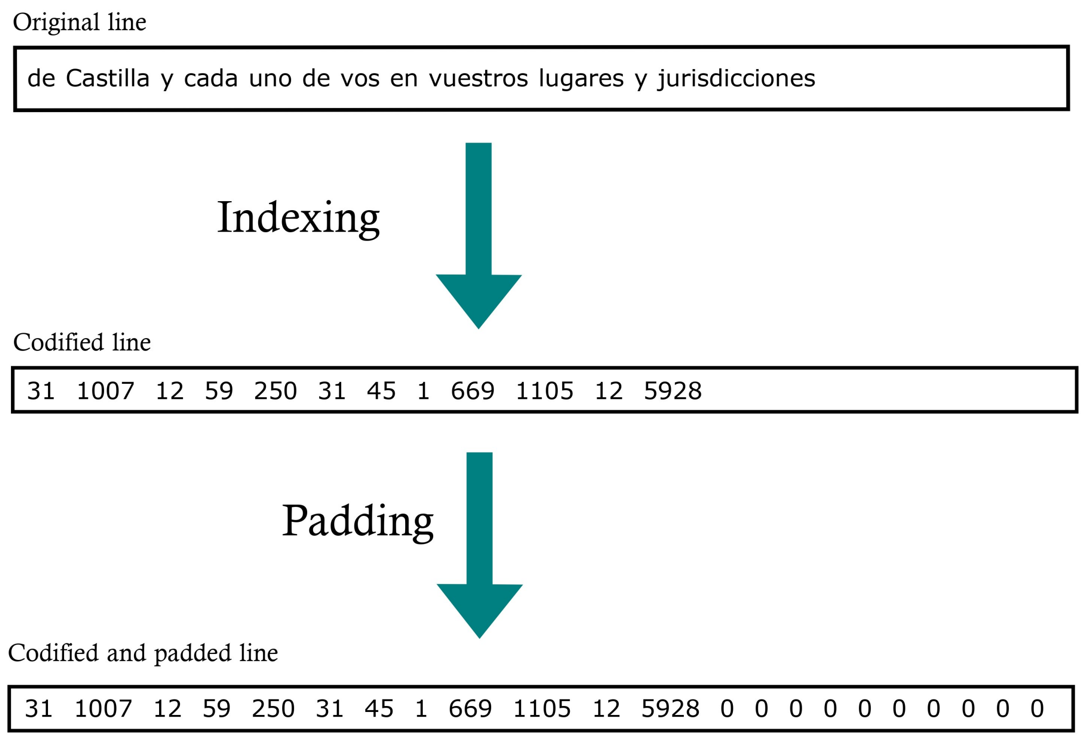


Figure 6.1: Process of representing a line of an arbitrary ID card. The terms are first indexed and then the sequence is filled with zero indices until the desired length is reached. Filling can alternatively be done at the beginning of the sequence.

The process of representing characters is analogous to that of words, where the correspondences between character and index are also stored in a dictionary, and a shorter sequence length is set. What changes is the form of representation, which is no longer composed of one-dimensional vectors, but rather matrices (one per line) that contain the coded characters of each word in their rows, that is, its dimension is the length of the word sequence by length of the character sequence. Padding is applied for words that do not exceed the maximum character length. In Table 6.1, the encoded character sequences associated with the words on the line used in Figure 6.1 are illustrated.

Table 6.1: Example of character-level encoding for words on a line.

Word	**Index**													
de	13	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Castilla	14	15	9	10	7	12	12	15	0	0	0	0	0	0	0
y	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0
cada	6	15	13	15	0	0	0	0	0	0	0	0	0	0	0
uno	3	2	5	0	0	0	0	0	0	0	0	0	0	0	0
de	13	1	0	0	0	0	0	0	0	0	0	0	0	0	0
vos	24	5	9	0	0	0	0	0	0	0	0	0	0	0	0
en	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0
vuestros	24	3	1	9	10	11	5	9	0	0	0	0	0	0	0
lugares	12	3	4	15	11	1	9	0	0	0	0	0	0	0	0
y	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0
jurisdicciones	17	3	11	7	9	13	7	6	6	7	5	2	1	9	0

Tags are also encoded in the same way as words (Table 6.2), associating an index to each different label, and with a sequence length equivalent to that of the words. The selected padding index remains the same, and these positions are masked. Masking consists of indicating to the layers in charge of processing the sequential information that certain time steps, those with the reserved index zero, are missing and that therefore they should not be taken into account when adjusting their distributed representations, being omitted in the final decoding.

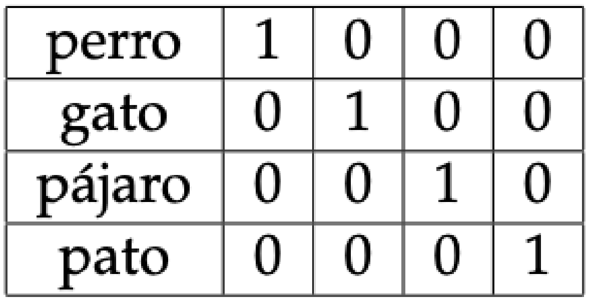
Table 6.2: Encoding of entity tags for words on a line. Padding positions with zero indices have been omitted.

Word	Tag	Index
de	none	1
Castilla	loc	6
y	none	1
cada	none	1
uno	none	1
de	none	1
vos	none	1
en	none	1
vuestros	none	1
lugares	none	1
y	none	1
jurisdicciones	none	1

6.2 Distributed Feature Representations

In deep learning applied to NER, it is quite common to have distributed representations of language elements, which collect semantic and syntactic information about each one. A distributed representation is a dense vector of real numbers associated with an element, for example a word, where each dimension refers to a hidden property. The main idea behind this concept is to determine the proximity between observations in a space of n components, whose distances depend on the similarity between these n numerical characteristics and in this way determine groups of interest. During training in each iteration, the vectors will be adjusted as the model learns new relationships, the representations being at the word and character level.

Unlike distributed representations, local or one-hot representations have a length equivalent to the size of the vocabulary $|V|$, where each dimension indicates the presence of a defined term in the vocabulary. An example of this representation is shown in Figure 6.2. For a specific term, its associated vector will contain as many zeros as terms other than it exist in the vocabulary ($|V| - 1$), and a one for its corresponding position. This means that no matter how similar two words are, they will be orthogonal in the vector space as they have totally different representations, causing contextual dependencies not to be captured. Another disadvantage is that it requires a lot of memory for large vocabulary sizes since the number of dimensions grows linearly. For these reasons this representation has not been chosen.



perro	1	0	0	0
gato	0	1	0	0
pájaro	0	0	1	0
pato	0	0	0	1

Figure 6.2: One-hot representation for the terms in V= {dog, cat, bird, duck}. Source: [6].

6.2.1. Word-level representations

The first type of distributed representations that the neural network learns are for words. Many systems presented in other works implement adjusted distributed representations of words by models such as *Word2Vec* based on multilayer neural networks trained with two possible architectures: the *skip-gram* or grammatical omission model, where for a central word, an attempt is made to predict the words that surround it based on the word vectors, and the *continuous bag of words* (CBOW), which for a sequence of context words, the central word is predicted.

In the proposed architecture, the word representations are stored in an embedding layer, which, based on the incoming sequences of encoded words, searches for real vectors of fixed length through the received indices. Before training the network, the representations are initialized with random numbers and as the language is learned, the values are adjusted. The operation in this layer is similar to that of a lookup table that associates the indices of each term with their respective feature vectors. The embedding layer is actually the first hidden layer of the model and requires three main parameters for its implementation:

- The size of the vocabulary or length of the dictionary with the correspondences between words and indexes, including pad words.
- The dimension of the representation vectors that define the number of characteristics that we want to obtain from each word.
- The length of the input encoded sequences to the model.

Additionally, an optional Boolean masking parameter (known as *mask zero*) is included to inform the model that some words are missing in the input sequence and that these are indexed with the value of zero. It is important to specify this parameter since it indicates to the upper layers of the network that these indices must be ignored since it is not interesting to know the predictions of non-existent words whose only function is to match the length of the sequences of integers with the expected input dimension to the model.

The output of the layer is a matrix that contains the vector representations, with as many rows as there are words in the vocabulary and with as many columns as the characteristics dimensions are to be considered. With this matrix, the representations of the words in the incoming sequence are selected, which will be passed to the bidirectional LSTM memory unit along with the character encodings, if operating with them. In Figure 6.3 the representations learned in the embedding layer of words belonging to the training set are observed. The representations have been reduced to two dimensions with the t-SNE technique [53] to allow their visualization. The graph shows well-defined groups of

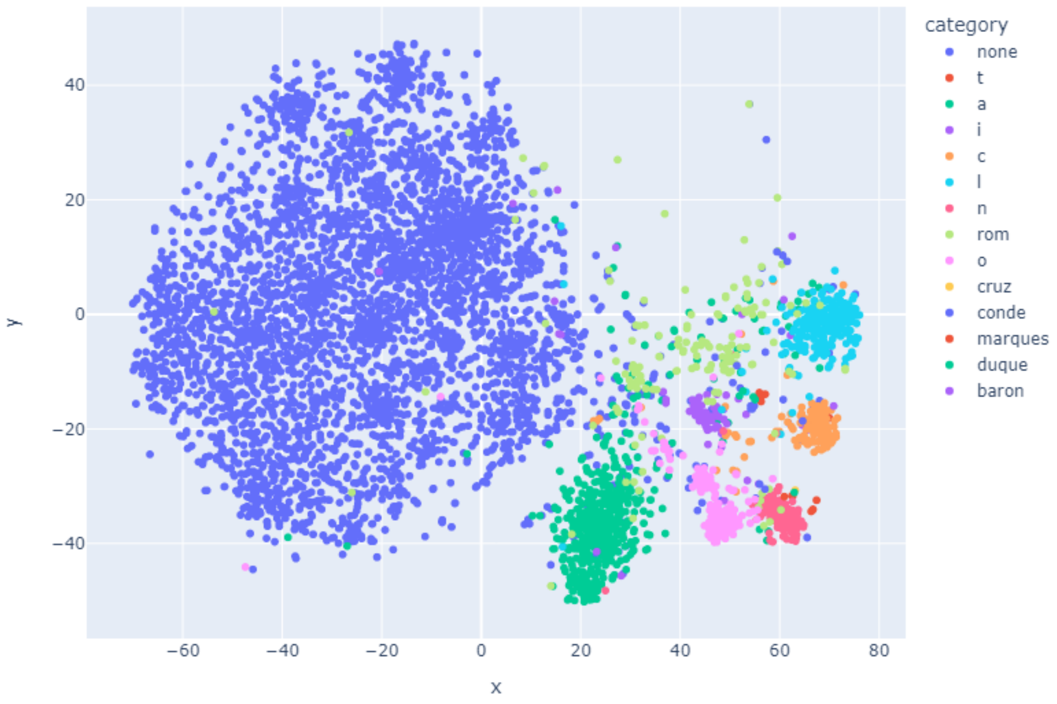


Figure 6.3: Word embeddings across two dimensions learned by the neural network in the ground-truth training lines colored by their corresponding category or entity type.

words according to their entity tag and they tend to be located close to each other in the representation space as a result of the learning process. The most interesting groups are found in the lower right, where it can be seen how the person names (n), surnames (a), offices (o), locations (l), institutions (i) and positions (c) are clearly separated. Perhaps, it is striking that the roman numerals (rom) are somewhat more dispersed.

6.2.2. Character-level representations

Word embeddings have led to good classification results for named entities on different annotated collections of texts, by finding associations between words based on their meaning. However, a recent study by Ha et al. [54] highlights the importance that characters have in determining syntactic similarities and therefore improving the performance of the models. Combining the distributed representations of words with those of characters implies recognizing words that have not been seen since their embeddings alone are limited to the vocabulary of the collection. For example, when testing the classifier, a misrecognized word or variant of another word found in training may not have an associated numerical vector in the word embedding layer, and this is when its internal structure can be used to provide useful information about its meaning thanks to the character representations.

Like words, character representations may already be pre-trained from language models such as *char2vec*, but again, these will be learned by the network since the dependencies found in these models cannot be be generalized to the data available for this work. The numerical vectors of each letter and symbol are initialized with random numbers in a defined range and stored in an embedding layer. The parameters to be introduced for the creation of the layer refer to the same elements as before but related to the characters and are the following:

- The number of characters to represent, which is the size of the dictionary with the correspondences between indexes and characters. Again, consideration should be given to including filler characters.
- The dimension of the representation vectors that define the number of properties that want to be obtained from each character. This dimension is usually smaller than that of the vectors of word representations.
- The length of the encoded character sequences input to the model, referring to words.

The masking condition is also added on the padded characters and thus ignore their representations. In this case, the output of the character embedding layer is a three-dimensional vector containing the distributed character representations for each word in the incoming sequence.

Once this layer has been trained, the representations of the different characters present in the training lines reduced to two-dimensional vectors with t-SNE have been illustrated in Figure 6.4. Groups of characters are appreciated according to their type. For example, digits are located at the top right of the representation space, while lowercase letters are at the top left and uppercase letters are at the bottom.

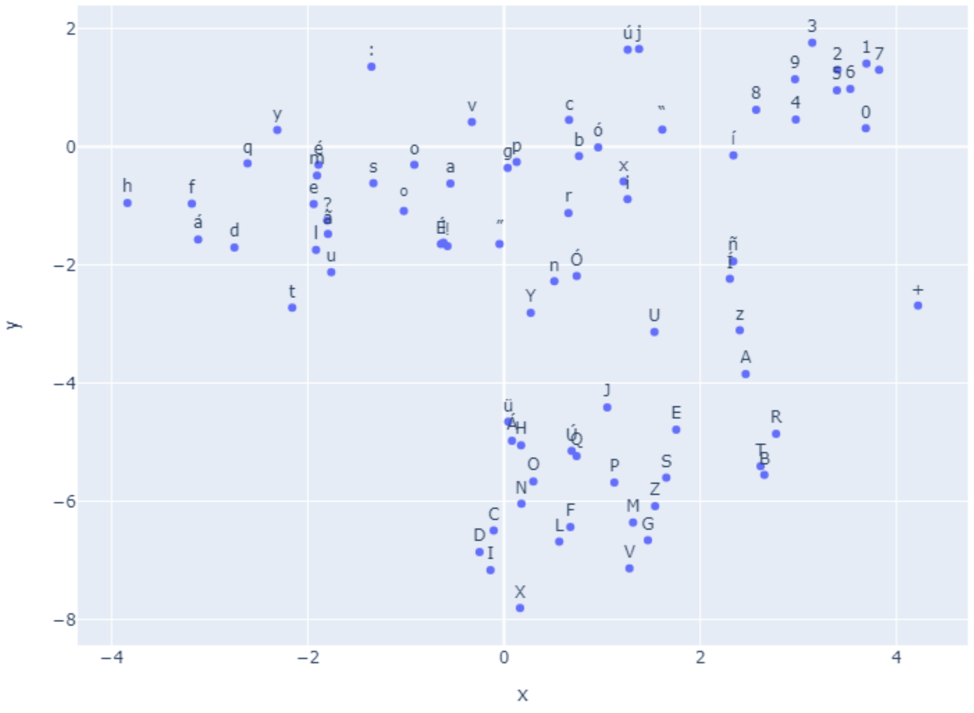


Figure 6.4: Character embeddings across two dimensions learned by the neural network in the ground-truth training lines.

The character representations then have to be combined with the word representations and be able to be used together by the context encoder. Since they have different dimensions, the character-level information per word is passed to an LSTM memory unit (not the bidirectional one) to encode the word into a low-dimensional vector. This operation is assimilated to a flattening function on a matrix, which in this case are the representations of the characters of a word, the difference is that its number of elements is not conserved. A common and alternative method to LSTM to extract character-level

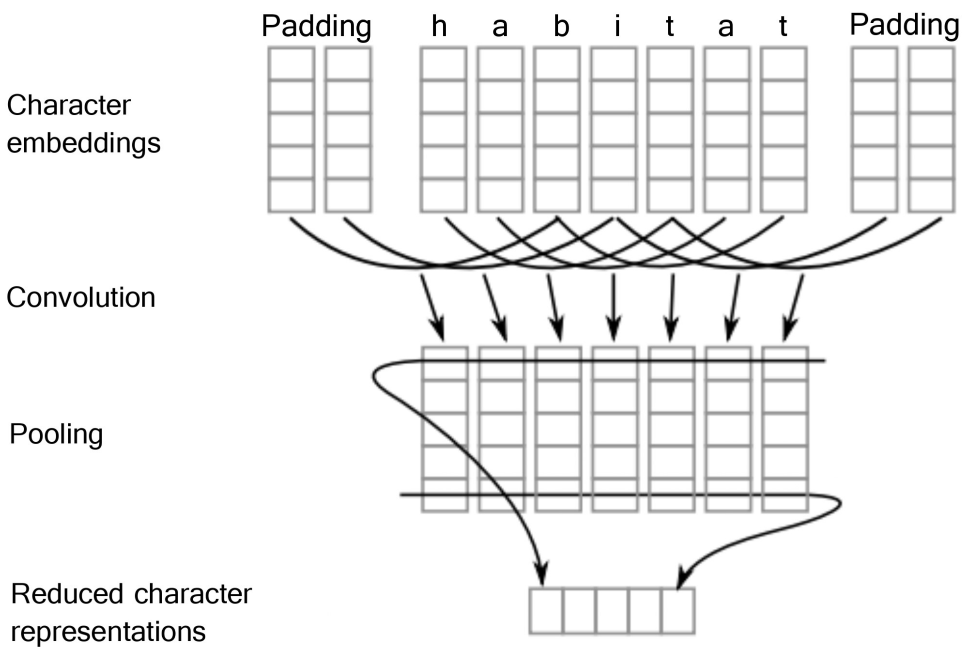


Figure 6.5: CNN that extracts the distributed representations of the characters in a word and creates a reduced feature vector. Image adapted from Chiu and Nichols article [2].

representations and convert them into a single vector is to use a convolutional neural network (CNN). The CNN captures orthographic properties of the word by passing different filters on the distributed representations of its characters and applies a max-pooling to reduce the output dimension. This process is illustrated in Figure 6.5.

Before passing these vectors to the context encoder, a value dropout function (known as spatial dropout) is applied with a certain probability. The purpose of this operation is to reduce possible overfitting of the model in the embeddings layers and to see how the network would act if complete word representations were not available.

6.3 Context encoder

Once the transformed character vectors have been concatenated with the distributed representations of learned words, the next step is to introduce them to the context encoder, a block with memory units in charge of capturing word information based on the context. Before explaining how it works, it is necessary to introduce what the long and short term memory units (LSTM) are, since their outputs determine the final representation of each word.

6.3.1. Long Short Term Memory (LSTM)

Recurrent neural networks (RNN) have the drawback that during the backpropagation process, the error gradient that adjusts the weights fades or approaches zero as the time steps increase (gradient vanishing problem). If the gradient is very small, it means that we are not fully learning from the error and therefore the learning effect is null, especially in the layers corresponding to the first steps of the sequence. LSTMs are a particularity of RNNs when it comes to dealing with sequential data of a specific order and within the field of natural language processing they are widely used for their ability to remember

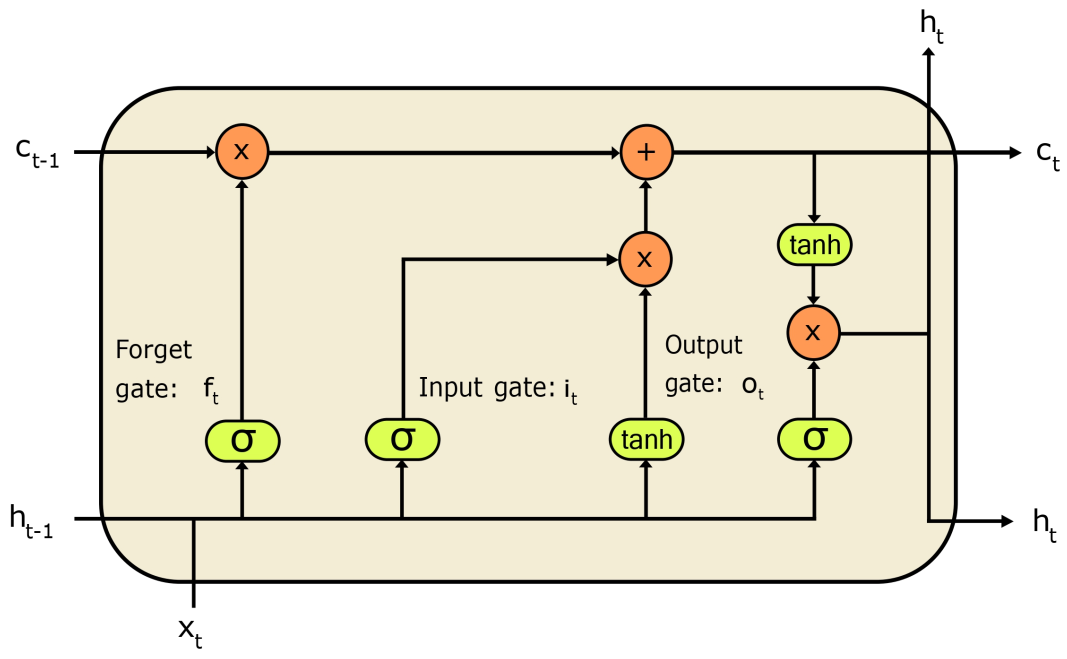


Figure 6.6: Internal composition of an LSTM. Given the hidden state from the previous step h_{t-1}, the input in the current step x_t, and the cell state from the previous step c_{t-1}, the new hidden state is computed h_t and the cell state c_t is updated.

short and long term information, created as a solution to the gradient vanishing problem seen in RNNs.

As seen in Figure 6.6, an LSTM is made up of three fundamental gates that learn which information is relevant in the current step to store or that which is not so important to be ignored, a cell state c in charge of storing the relevant information throughout the sequence, the input x in the current timestep, and the hidden states h of the previous and current step. The key to retaining or forgetting this information is the sigmoid and hyperbolic tangent activation functions that are present in the gates, these being forget gate, input gate and output gate. The following subsections indicate the operations performed by each gate and the updating of the cell state.

Forget gate

This gate is responsible for deciding what information should be discarded from the cell state c_{t-1}. The information from the previous hidden state h_{t-1}, and the input corresponding to the current time step x_t is combined and passed through a sigmoid activation function, creating a forget vector f_t:

$$f_t = \sigma(U_f \cdot h_{t-1} + W_f \cdot x_t + b_f) \text{ , where:}$$

- $\sigma(x)$ denotes the sigmoid function on x,
- U_f,W_f are weight matrices associated with the hidden state in the previous step and the input of the current step, respectively,
- b_f is a vector of independent terms.

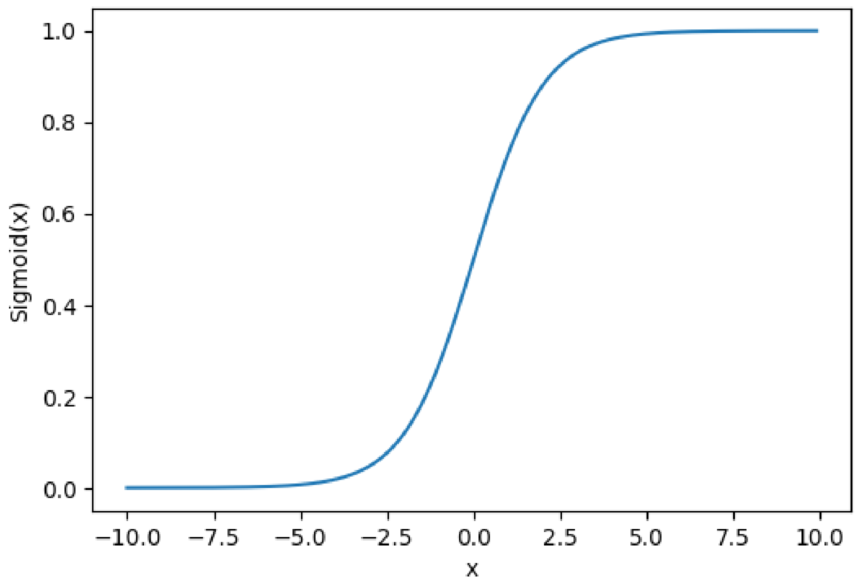


Figure 6.7: Sigmoid function for input values between -10 and 10.

Figure 6.7 illustrates the sigmoid function in a range limited between 0 and 1 for any real number. A value closer to 1 means the feature is saved and closer to 0 means it is discarded.

Input gate

The input gate decides what information should be stored in the cell state. As in the forget gate, the information from the previous hidden state and the current input is again passed through another sigmoid function to create the input vector i_t and decide which values are still saved and which are not. The difference is that now a hyperbolic tangent function is applied to normalize the values between -1 and 1, creating a vector of possible candidates $\overline{c}_t$ to add to the cell state. This result is then multiplied by the output of the sigmoid function to update the cell state. Below are the operations involved in determining i_t and $\overline{c}_t$:

$$i_t = \sigma(U_i \cdot h_{t-1} + W_i \cdot x_t + b_i),$$

$$\overline{c}_t = \tanh(U_c \cdot h_{t-1} + W_c \cdot x_t + b_c)$$

Where:

- $\tanh(x)$ denotes the hyperbolic tangent function on x,
- U_i, W_i, U_c, W_c are matrices of weights in the different operations associated with the hidden state in the previous step and input of the current step.
- b_i, b_c are vectors of independent terms.

In Figure 6.8 the hyperbolic tangent function is represented, presenting the same shape as the sigmoid function, the difference is that the sign of the input value is preserved after being applied.

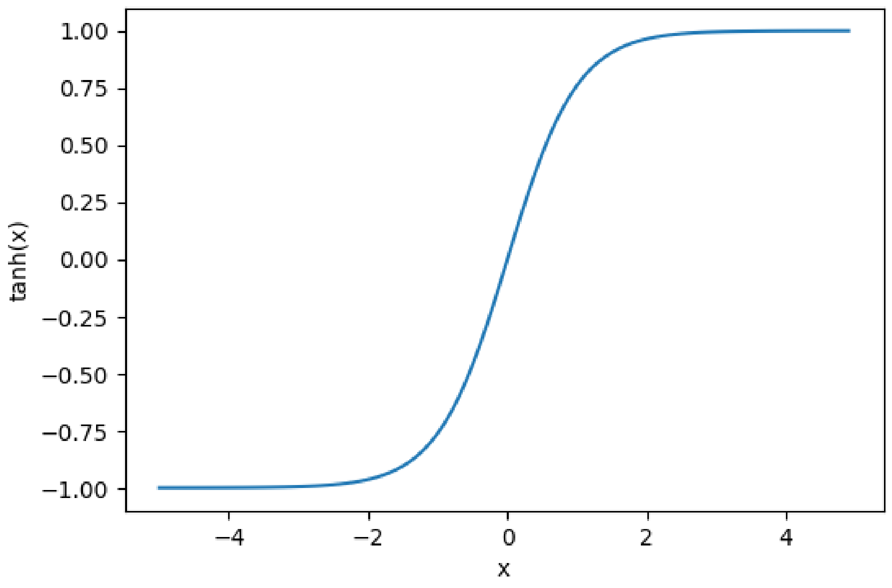


Figure 6.8: Hyperbolic tangent for input values between -5 and 5.

Cell state update

Once the relevant information has been detected and that to be ignored, the next step is to make changes to the cell state, which carries relevant information found at each step along the sequence. It is updated in such a way that its content from the previous time step c_{t-1} is multiplied by the forget vector f_t to discard values in the current step and then an addition operation is done with $i_t * \bar{c}_t$, which contains the information of the new candidates scaled by their relevance. With this, the new cell state c_t is obtained:

$$c_t = f_t * c_{t-1} + i_t * \bar{c}_t$$

Output gate

Finally, the output gate defines which hidden state h_t will be passed to the next memory unit. To determine the new hidden state, on one hand, a sigmoid function is applied to the previous hidden state h_{t-1} combined with the current input x_t to obtain the output vector o_t, and on the other hand, the hyperbolic tangent is applied to the updated cell state c_t. The results of the two functions are multiplied to define the new hidden state h_t, which will be passed along with the cell state to the next timestep. Specifically, the operations carried out by this gate are the following:

$$o_t = \sigma(U_o \cdot h_{t-1} + W_o \cdot x_t + b_o),$$

$$h_t = o_t * \tanh(c_t)$$

Where:

- U_o, W_o are weight matrices associated with the hidden state in the previous step and input of the current step, respectively,
- b_o is a vector of independent terms.

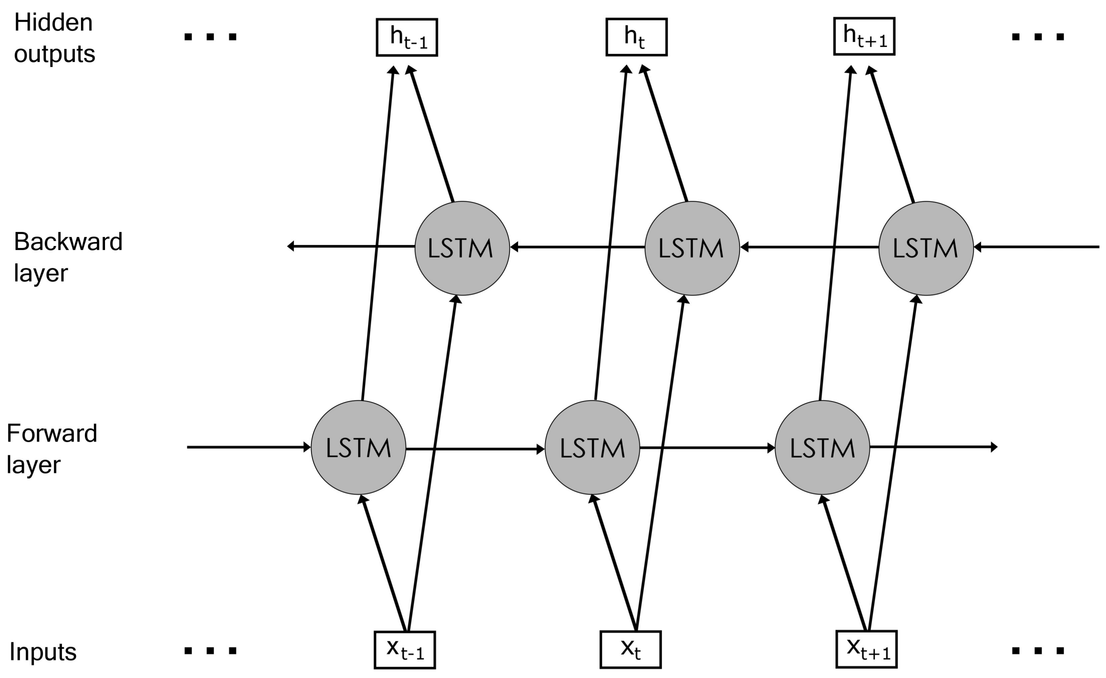


Figure 6.9: Composition of a Bi-LSTM. The memory modules in both layers receive as inputs the sequential data in addition to the previous or subsequent hidden state. The outputs of each pair of modules are concatenated to produce the final representations.

6.3.2. Bidirectional LSTM

Although LSTMs are capable of remembering long and short-term information, the hidden output states at each time step are based on stored information from previous states, meaning that only inputs from the past are available, without considering the flow of information that comes from subsequent steps. In a task like NER, words after a named entity can provide an important clue about its meaning and semantic category in addition to those before it, so installing a single layer of LSTM modules will not be enough. For this reason, two LSTM layers are used to process the information in both directions of the sequence, forming the context encoder of the neural network.

At a particular time step, one layer processes the inputs received on the left or previous inputs, and the other processes the word representations on the right. This composition of the LSTM is called Bi-LSTM or bidirectional LSTM because it refers to the information on both sides of each term to specify its context. The common structure of a Bi-LSTM is illustrated in Figure 6.9, where the vectors with the word representations are introduced into the two LSTMs associated with the corresponding step, and produce a contextual vector of the word, created from a concatenation operation on the hidden states of each module. Note that the contextual vectors are not the final outputs since they have to go through the label decoder, so they are marked as hidden and can now be used to label the incoming sequence.

6.4 Label decoder

The last block of the model is formed by the label decoder. As its name indicates, it produces a sequence of labels from the contextual vectors extracted from the Bi-LSTM, which correspond to the input word sequence. There are two very common architectures to obtain the final labels, one of them and the most direct is a multilayer perceptron that extracts the probabilities of belonging to each category, applying a softmax activation function on the nodes of the output layer and the other is the conditional random fields or CRFs, this being the option chosen to extract the final tags of each word.

6.4.1. Conditional random field

The conditional random field (CRF) was introduced in 2001 by Lafferty et al. [55] and consists of a statistical model based on an undirected graph, whose objective is to determine a sequence Y of unknown variables given an input sequence X of observations using conditional probabilities $P(Y|X)$. With the contextual information of each entry x_i ($1 < i < n$, where n is the length of the sequence), we try to predict which is the most convenient label y_i. For this, the model has to be trained with different inputs to adjust their weights and maximize the conditional probability distribution. If a term appears in a sequence with a context that is usually not the one around it, prior knowledge of other instances can be used to predict its label based on its conditional probabilities. The CRF is represented as an acyclic graph $G = (V, E)$, with V being the set of nodes formed by all input observations and entity tags, and E the set of edges that consider the label emission and transition probabilities. The nodes are contained in two disjoint sets A and B, with $A = \{x_1, x_2, ..., x_n\}$ and $B = \{y_1, y_2, ..., y_n\}$ containing each element of the input and output sequences respectively. The structure of the CRF is illustrated in linear form in Figure 6.10.

Unlike hidden Markov models (HMM), a CRF is a discriminative model. This means that it models the conditional probabilities instead of the joint ones $p(x, y)$ and the distributions of observations $p(x)$ to make predictions. Furthermore, a CRF does not introduce the independence assumptions of HMMs so the context of each observation is considered as additional information and the dependencies are not only calculated with respect to the previous state but with any state present in the sequence, even with the final state. Also, CRFs usually perform better than HMMs in generating the final sequence of hidden states in most NER tasks.

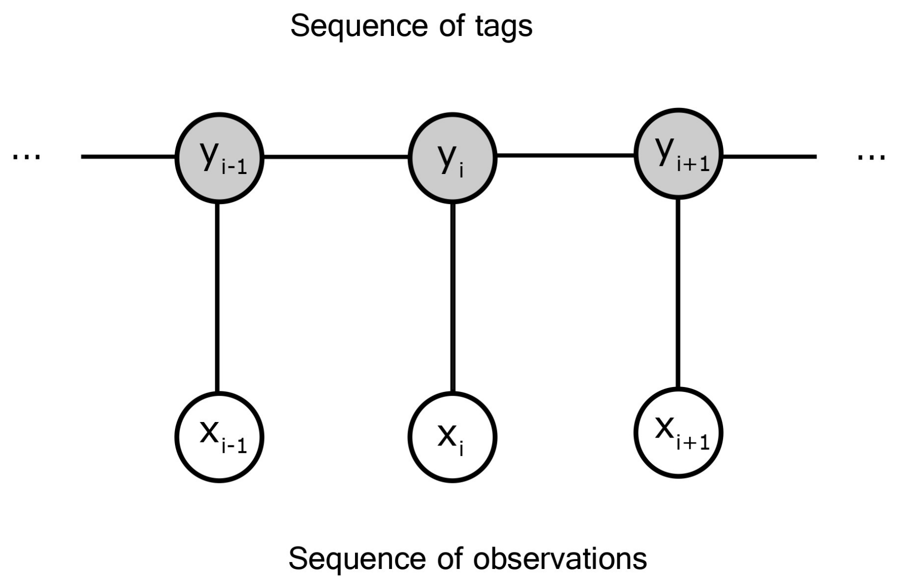


Figure 6.10: Linear-chain CRF that associates each input x_i with its corresponding label y_i.

CRFs are widely used in tasks such as NER to label word sequences of a certain length. The fact that the prediction for a word depends on the predictions of neighboring words makes the CRF a very valuable tool for modeling conditional probabilities. If there were no dependencies between words, the probability of emitting their sequence of labels would be the product of the emission probabilities of each individual state. Then, the probability of generating the sequence of states Y given some observations X of length T additionally considers some transition probabilities between labels and is denoted by:

$$P(Y|X) = \prod_{t=1}^{T} P(y_t|x_t) \cdot P(y_t|y_{t-1}), \text{ where:}$$

- $P(y_t|x_t)$ is the emission probability of state y given the input x at step t,
- $P(y_t|y_{t-1})$ is the transition probability to state y_t from y_{t-1}.

Note that the inputs x_t received by the CRF layer are the contextual vectors that come out of the bidirectional LSTM at each timestep.

6.4.2. Viterbi algorithm

To obtain the final predicted sequence, the Viterbi algorithm [56] is used, which determines the path of states with the highest probability through dynamic programming. At each step t the maximum probability of generating a prefix of observations of length t is calculated by traversing a sequence of states ending in s. This operation is equivalent to selecting the path with the highest probability that emits the prefix of length $t-1$ ending in state s' and multiplying it by the transition to s and by the emission of x_t in s. Of all the states s' evaluated, the one that gives the maximum probability is chosen:

$$V(s,t) = \max_{s'} \quad V(s',t-1) \cdot T(s',s) \cdot U(s,x_t), \text{ where:}$$

- $V(s,t)$ is the probability of the most likely sequence of states of length t ending in state s,
- $T(s',s)$ is the probability of transition from state s' to s,
- $U(s,x_t)$ is the probability of emitting the observation x_t in s.

When $t = 1$, the initial probability $I(s)$ of each state and the emission probability of the first observation x_1 are considered:

$$V(s,1) = I(s) \cdot U(s,x_1)$$

Finally, the probability of the entire sequence is determined by the maximum probability among all paths that produce the chain of observations of length T ending in each state s:

$$\hat{P}(Y|X) = \max_{s} \quad V(s,T)$$

6.5 Bi-LSTM-CRF model

Once all the components of the network have been defined, its complete structure is illustrated in Figure 6.11. It is a Bi-LSM-CRF model that takes, on one hand, lines with their encoded words and, on the other hand, the encoded characters of each word present in that line, to extract distributed representations of specific lengths. The character embeddings are passed through the LSTM, which is responsible for combining and encoding them in a reduced dimension. The resulting representations are concatenated with the word embeddings, with the aim of providing information about morphological composition. These vectors are fed into the bidirectional LSTM, creating contextual vectors. Finally, the CRF predicts the tags at each timestep with the Viterbi algorithm.

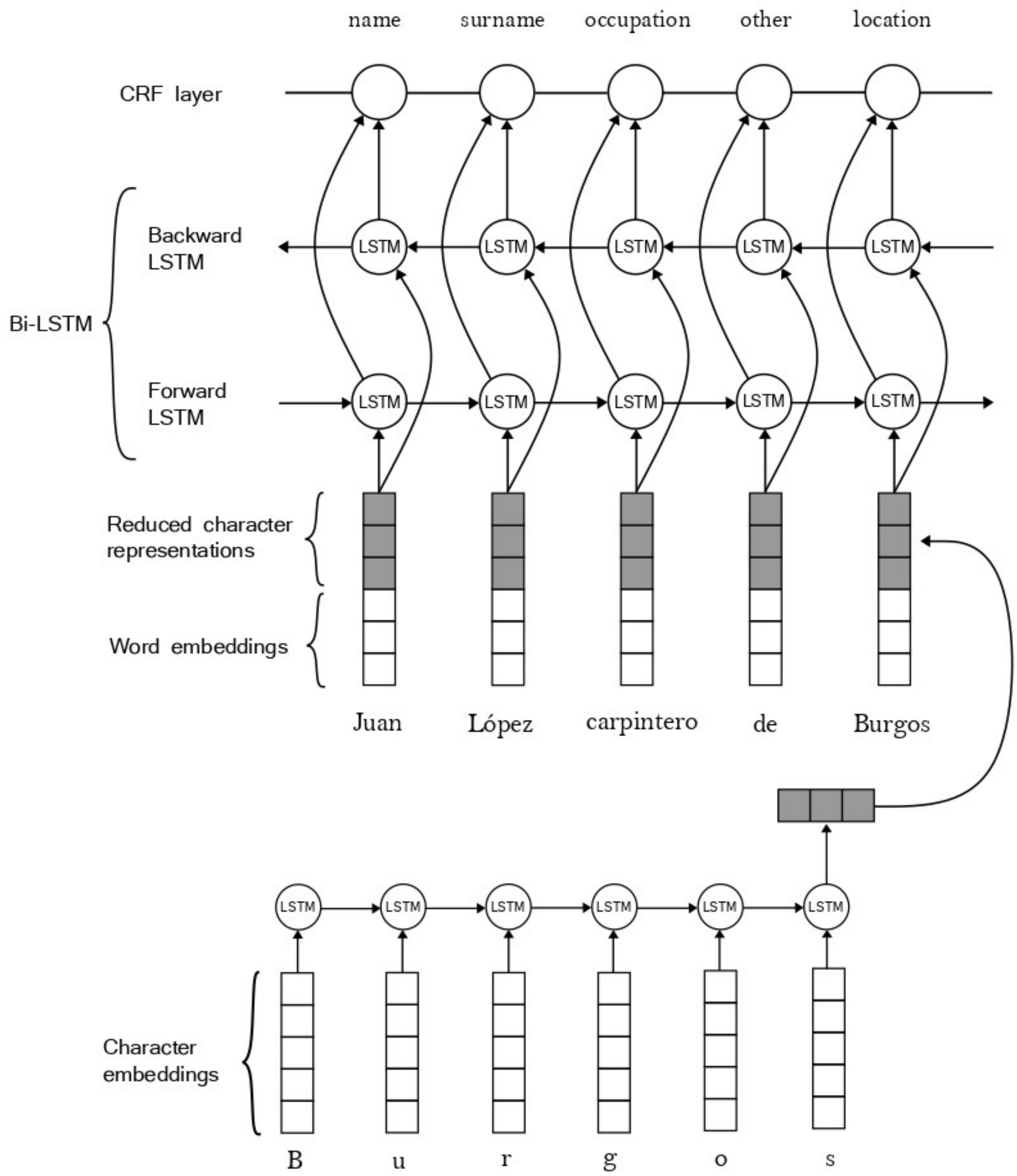


Figure 6.11: Complete Bi-LSTM-CRF model for predicting the named entities of a given word sequence. At the bottom, the LSTM extracts the reduced character-level representations and they are concatenated with the word embeddings to consider morphological features. Then context is extracted in the Bi-LSTM and CRF layer field returns the best sequence of labels.

CHAPTER 7

Experiments and results

With the model for named entity recognition defined, the next step is to train and evaluate it with the available transcripts. To do this, one first has to select the hyperparameters that produce the lowest possible classification error, define a loss function to adjust the network weights, and choose the most appropriate evaluation metrics. The following sections describe these steps and illustrate the results obtained in the different experiments carried out.

7.1 Hyperparameter selection

The model hyperparameters as characteristic of the training process have a significant effect in the classification success of named entities. Compared to model parameters whose values can be learned from the input data as the weights of a neural network, the hyperparameters are external to the model, meaning that they are not estimated from the data but have to be manually selected to control and optimize the learning process. A bad decision in any of them can condition the results obtained regardless of the complexity of the neural architecture. Therefore, this section analyzes the effect they have on the learning evolution, indicating the final configuration used.

7.1.1. Learning rate and optimizer

As seen in the backpropagation process in Chapter 5, during the gradient descent that obtains the set of network parameters that minimizes the value of the loss function in training, the error produced by each weight is scaled by a learning factor called the learning rate. The learning rate is arguably the most important hyperparameter of any deep learning model. It is limited between 0 and 1, determines the size of the weight modification, where a value very close to 0 indicates a slow convergence of the error caused by poor parameter adjustment and therefore many iterations or training steps will be needed to reach a minimum. If, on the other hand, the learning factor is very high, the error converges quickly and tends to oscillate around the minimum since the update of weights is noticeable; it can even increase the value of the loss function and overflow if the changes are very abrupt. There is no method that specifies what the optimal learning factor is, although there are multiple techniques such grid search that determines the best value for a given range. The problem with these methods is that they require a lot of adjustment time by training the model with a different configuration of hyperparameters and the improvement produced may not be entirely significant. In many cases the default value is usually 0.01 or 0.1 and is defined together with the optimizer, which is an algorithm that defines a parameter update strategy to reduce the error in the model.

There are two main groups of optimizers, those based on gradient descent and adaptive ones, which modify or adapt the learning rate in each iteration according to the change produced in the parameters. Adaptive optimizers have an advantage since in datasets with a lot of attribute variability, the value of some specific parameters may have to be modified to a greater magnitude and considering a dynamic learning factor is a good solution to change the training pace.

By testing different optimizers and learning factors, Adam's adaptive optimizer has been chosen with a learning factor (α) of 0.01. Adam generally achieves very good results in deep neural learning due to its ability to quickly converge to the minimum and to deal with sparser gradients caused by differences in the data. However, there are others such as RMSProp and AdaDelta that perform similarly in some characteristics but Adam is considered the most effective in many cases. In addition, it consists of two other parameters that model the rate of reduction of the learning factor (β_1 and β_2) and an epsilon (ϵ) to avoid divisions by zero in the calculations. These additional parameters have been kept to their default values.

7.1.2. Batch size

The batch collects a number of samples to process before the model parameters are updated. Instead of going sample by sample to perform backward and forward steps in the network that can be very costly in large data collections, it is worked with groups of predefined sample sizes to speed up these processes. Batch prediction error is calculated by averaging or accumulating the individual errors of each sample and then propagating the error and updating weights. It is very common that the last batch is not completely filled because the total number of samples is not divisible by the size of the batch.

The choice of the size of these groups is an important hyperparameter to consider in any deep learning model. An excessively large size can cause loss of generalization in the model by adjusting its parameters based on a very small error and requires more time to complete each step but obtains better gradient estimates. On the other hand, using very small sizes improves the speed of convergence but it is not guaranteed to achieve it, requiring a smaller learning factor due to very dispersed gradients. Due to memory storage issues, batch sizes are usually powers of 2, with the most used being small sizes of 16, 32 and 64, which have presented the best results in many recognition tasks [57, 58] although it highly depends on the total number of samples and the model itself. In this case, a batch size of 32 samples has been chosen.

7.1.3. Epochs

In contrast to the batch size, the number of epochs is a hyperparameter that determines how many times the model receives the entire set of training samples to update its parameters. During a training epoch, all batches of samples are passed through the neural network one by one, back and forth. In programming terms, this can be thought of as a nested loop, where each epoch loops through the batches that make up the dataset and ends when the batches are processed in the last epoch. There is no formulation to establish a specific number of epochs, although it is very common that the model overfits when trained with too many of them. Thus, the number of epochs used to train the neural model is 10, point at which the error improvement is no longer noticeable and the validation loss begins to increase.

7.1.4. Sequence length

Starting with the specific parameters of the Bi-LSTM-CRF model, the length of the sequences is a hyperparameter to be determined, which indicates the number of words or tokens to be considered to extract their labels and therefore is also the input dimension of model.

As seen previously in Figure 4.5 in the exploratory analysis of the data, the lengths of the lines vary from 1 to 30 words, following a normal distribution. Due to this variability, the ideal length is one that collects the maximum amount of information possible without adding excessive padding to the shortest lines since it involves training the model with additional positions that will not be taken into account to generate its labels. Hence, the length of the sequences has been set to 30, which is the maximal length. Progressively lowering this amount means wasting terms that can be useful to determine the context of certain words, especially for longer lines. As for ID card level, the sequence length will be varied to consider wider contexts as part of the experimentation.

To select the length of the character sequences necessary in the extraction of morphosyntactic information from words, the same reflection has been followed, selecting a size of 15. Words with more than 10 letters rarely occur in the collection.

7.1.5. Dimension of representation vectors

The next hyperparameter to determine is the size or dimension of the representation vectors of the words and characters, defined in the embedding layer. These numerical representations group elements by their meaning and the number of components to be considered highly depends on the size of the vocabulary, since the larger it is, the more diversity of terms or symbols there are, implying more properties that have to be captured. The problem with high-dimensional spaces is that the correlation between observations is lost when considering components that can introduce noise to the model and increase the number of parameters to train, while with very small spaces the lexical information is greatly compressed. Typically, this hyperparameter is selected by trial and error, although a common starting point is defined by the root to the fourth of the vocabulary size [59].

Due to the reduced vocabulary (6879 unique words), the word embeddings have 20 dimensions as they are sufficient to model a high amount of features. In the case of characters, since they have a smaller vocabulary size, their dimensions have been reduced to 10.

7.1.6. Hidden units in the LSTM

Each LSTM layer has to return an output vector of a specific size to be used in subsequent layers. The designed model has two main LSTM blocks for remembering short and long-term information: One is the character-level LSTM and another that is the bidirectional LSTM also known as the context encoder.

The goal of the character-level LSTM is to produce more compact representations of characters that have to be fused with the word embeddings. This is done by passing the learned character embeddings of the words as inputs to the LSTM, that will memorise the order of the characters to return a reduced representation of the word in terms of its internal composition and not by its meaning (as it occurrs with word embeddings). Since we already have representations of words of dimension 20, the number of outputs in this block is also set to 20 and thus obtain resulting vectors of size 40 that consider characteristics of the meaning and the internal composition of the word in equal parts.

The bidirectional LSTM consists of two LSTM layers to store relevant information in opposite directions of the sequence and the outputs of each also have to be combined to generate the contextual vectors according to the word. At this point, it is interesting that the size of these outputs is large enough to record all the attributes that relate words to each other, but at the same time introduce the right units to limit the number of operations involved in calculating the hidden states and update the cell state of the LSTMs. For this reason, 64 output units have been required in each layer.

7.1.7. Dropout

In machine learning, dropout is a technique to reduce model overfitting by causing arbitrary neurons or nodes in a certain layer of the network to be deactivated or ignored. The idea behind this is that there is a dependency in the outputs of each node that means that the detected patterns cannot be correctly generalized to other samples that have not been used in training. If some nodes are not taken into account during the forward and back-propagation processes in the network, all their connections leaving or reaching them are ignored and therefore fewer parameters need to be updated. With this, possible dependencies between weights are broken and the remaining nodes also load the output values of the deactivated nodes. The hyperparameter is then the probability of node omission in a specific layer of the network.

With respect to recurrent networks, the cell state and the hidden state that will be passed to the next temporal position with a certain probability are omitted. Specifically in the proposed model, this takes place in the different gates that make up the LSTM, where the matrix and activation operations of each one are ignored. In the research work of Ghal and Ghahramani [60], another alternative was presented to apply the *dropout* technique in recurrent networks that consists of masking or ignoring all units of random time steps in the sequence, in instead of arbitrary hidden and cell states. The skip probability is 0.3 in the two LSTM blocks of the model, meaning that approximately one-third of the recurrent connections are not considered to determine the hidden states at each step. A similar process but over a two-dimensional space (called spatial dropout) is applied to the fused vectors with semantic and morphosyntactic features. The probability is also 0.3 with the intention of eliminating dependencies between properties of each term.

7.2 Loss function

The loss function that the model uses to update its parameters is the CRF loss itself of the label decoding layer. The goal is for the estimated probability of the actual sequence of tags to be as close to 1 as possible to minimize loss or error. In the computation of this objective function, the negative log-likelihood of the expected output sequence is calculated considering all possible combinations of sequence paths. As stated by Arnaud Stiegler [61], this is:

$$\text{loss} = -\log\Big(\frac{\exp(\sum_{t=1}^{n} U(y_t|x_t) + \sum_{t=1}^{n} T(y_t|y_{t-1}))}{\sum_{y'} \exp(\sum_{t=1}^{n} U(y'_t|x_t) + \sum_{t=1}^{n} T(y'_t|y'_{t-1}))}\Big) \tag{7.1}$$

The numerator of the logarithm refers to the probability of obtaining the expected sequence y from the emission probabilities U and transition probabilities T between states. The probabilities of all possible state paths accumulate in the denominator exponentially with the length of the sequence. Therefore, the loss will be smaller the more notable the probability of the expected sequence is with respect to the others.

7.3 Evaluation metrics

Accuracy is the most popular and reference metric in any classifier as it indicates the proportion of samples or observations that have been correctly assigned to their respective categories. In NER it is also important to show the percentage of word labels correctly recognized by the model. However, for unbalanced classes where there is typically a majority class (in this case with "none" tags that form 85% of the whole collection), the accuracy is no longer the best metric to consider because it does not reflect the quality of recognition in the more relevant categories which in this case can be names, locations or positions. For this reason, the precision, recall and the F1 score specific to each type of label have been taken into account.

The precision of a specific label is the number of times it has been correctly recognized over the total number of predictions made for that same label in the entire set. For example, if 500 surnames are predicted in the test set and 450 of those predictions correspond to real surnames, its precision is 0.9. On the other hand, recall indicates how many labels of a specific type have been correctly recognized out of the total number of labels of that type. Again, if there are 600 surnames in the test set and 450 are those identified without error, its recall is 0.75. Finally, the F1 measure is a function that takes balance between precision and recall. Weighted version of the F1 score reflects the true performance of the model when the dataset is unbalanced like the one we have for this occasion, being one of the reasons why it is such a common metric to use. Standard F1 score is calculated as the harmonic mean of the precision and recall:

$$\text{F1 score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7.2}$$

This score can be modified to give more weight to precision or recall metrics, depending on the task needs. From the metrics related to each class, the average is obtained taking into account all the types of labels in the collection of ID cards. The averages of precision, recall and F1 show the generalized performance of the model. To obtain these evaluation metrics, it is necessary that the lines are tagged with the associated entity types, as is the case with transcriptions that do not contain errors. However, in the outputs generated by the handwritten text recognizer there may be variations in the original lengths caused by insertions and deletions of some words. For this reason, the edit or Levenshtein distance [62] between the predicted and reference tag sequences has been calculated. This distance has been normalized by the length of the reference sequence to determine the Word Tag Error Rate (WTER). WTER is the standard WER metric (Word Error Rate) but applied to entity tags. Since there are n-best hypotheses for each line of the test set, the individual WTER of a line is given by the hypothesis whose predicted labels have the smallest editing distance with respect to the ground-truth labels from the manual transcripts. Therefore, the final WTER for the n-best hypotheses is calculated as the average of all individual WTERs per line:

$$\text{WTER}_n = \frac{1}{L} \sum_{i=1}^{L} \min_{1 \le j \le n} \text{WTER}(\text{hyp}_{ij}, \text{ref}_i), \tag{7.3}$$

$$\text{WTER} = \frac{S + B + I}{N} \cdot 100 \tag{7.4}$$

Where:

- hyp_{ij} are the predicted labels for the j-best hypothesis of line i,
- ref_i are the reference labels of line i,
- L is the total number of lines in the test set,
- S is the number of substitutions,
- B is the number of deletions,
- I is the number of insertions,
- N is the number of words that the reference line has.

7.4 Results

First, the results are shown over the ground-truth lines from the test set when training the neural network with the ground-truth lines from the training set. These transcriptions exactly represent the content of the original manuscript texts and therefore do not contain errors derived from the recognition process. The individual evaluation metrics of each entity tag along with the averaged metrics for all tags are illustrated in the Table 7.1. The F1 score of 0.83 achieved indicates that the system has generally been able to correctly recognize the entities existing in the handwritten texts. It uses the meaning and analyzes the context of words to assign their labels correctly in most cases. Note that the metrics of the "none" tags have been excluded from the table because they distort the calculation of the averaged metrics.

As for the individual categories, there are some that have been recognized quite accurately such as names, surnames, roman numerals, crosses, dukes and jobs, proof of this are the high F1 values obtained. However, the model has a harder time detecting others such as positions, institutions, locations and offices, all of them with an F1 of less than 0.72. A more complete analysis of the classification errors can be seen in the confusion matrices of the labels normalized by rows (real labels) and columns (predicted labels) in figures 7.1 and 7.2 , respectively. In the first, the recall of each label can be seen on the diagonal and in the second, the precisions are shown. These values differ from those seen in Table 7.1 since tags of type "none" have not been considered for their calculations, which is why the recall and precision metrics are higher. In both matrices, it can be seen that institutions (i) are often confused with locations (l) and offices (o) with positions (c), something that was expected due to their similarities in context. Many of the words tagged as institutions contain a location, for example: "Monasterio de León" or "Catedral de Málaga". This causes the model to predict that the terms "León" and "Málaga" are locations rather than institutions, although it does not always fail if the context is well captured by the Bi-LSTM. Interestingly, many surnames are incorrectly identified as locations. This is caused by compound surnames that include the word "de" like "Francisco de Vargas" or "Juan de Medina", where "Vargas" and "Medina" are predicted as locations. This confusion makes sense as some surnames are formed by the birthplace of that person. Jobs have optimal recognition results because these entities are very specific as seen before in Table 4.5.

Table 7.1: Evaluation results specific to each entity type on the manual transcripts of the test set.

Entity tag	Precision	Recall	F1 score	Support
Surname	0.909	0.859	0.884	830
Position	0.650	0.798	0.716	361
Count	0.636	1.000	0.778	7
Cross	1.000	0.818	0.900	11
Duke	1.000	1.000	1.000	23
Institution	0.733	0.550	0.628	404
Location	0.636	0.805	0.710	343
Marquis	0.909	0.769	0.833	13
Name	0.956	0.916	0.936	405
Roman numeral	0.958	0.958	0.958	95
Office	0.774	0.587	0.667	479
Job	1.000	1.000	1.000	43
Macro avg	**0.847**	**0.838**	**0.834**	**3014**

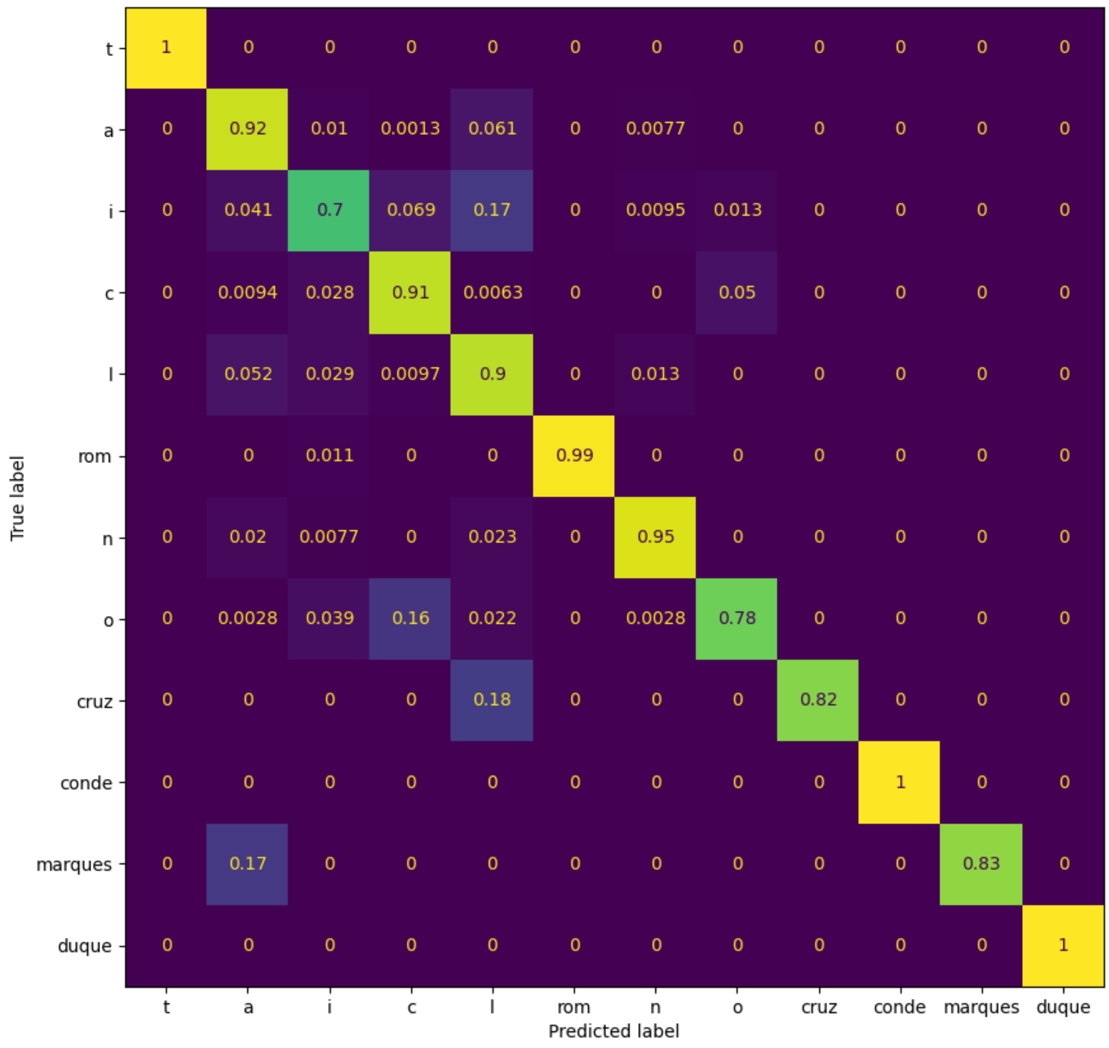
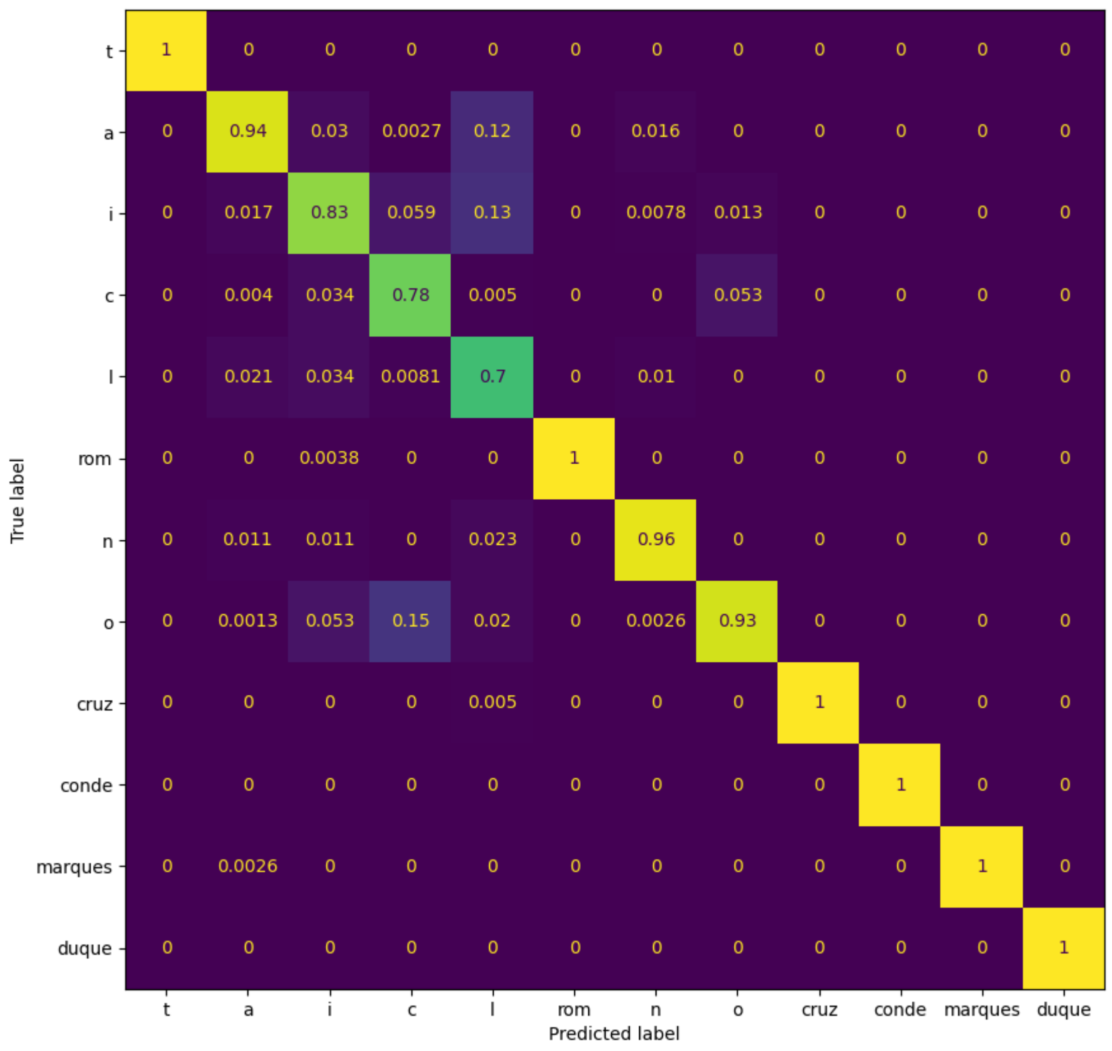


Figure 7.1: Confusion matrix normalized by rows or by true labels for jobs (t), surnames (a), institutions (i), positions (c), locations (l), roman numerals (rom), names (n), offices (o), crosses (cruz), counts (conde), marquises (marques) and dukes (duques).



True label \ Predicted label	t	a	i	c	l	rom	n	o	cruz	conde	marques	duque
t	1	0	0	0	0	0	0	0	0	0	0	0
a	0	0.94	0.03	0.0027	0.12	0	0.016	0	0	0	0	0
i	0	0.017	0.83	0.059	0.13	0	0.0078	0.013	0	0	0	0
c	0	0.004	0.034	0.78	0.005	0	0	0.053	0	0	0	0
l	0	0.021	0.034	0.0081	0.7	0	0.01	0	0	0	0	0
rom	0	0	0.0038	0	0	1	0	0	0	0	0	0
n	0	0.011	0.011	0	0.023	0	0.96	0	0	0	0	0
o	0	0.0013	0.053	0.15	0.02	0	0.0026	0.93	0	0	0	0
cruz	0	0	0	0	0.005	0	0	0	1	0	0	0
conde	0	0	0	0	0	0	0	0	0	1	0	0
marques	0	0.0026	0	0	0	0	0	0	0	0	1	0
duque	0	0	0	0	0	0	0	0	0	0	0	1

Figure 7.2: Confusion matrix normalized by columns or by predicted labels for jobs (t), surnames (a), institutions (i), positions (c), locations (l), roman numerals (rom), names (n), offices (o), crosses (cruz), counts (conde), marquises (marques) and dukes (duques).

Through these results, it can be said that the neural model uses the embeddings or learned representations to determine the meaning of the words and with the context encoder, the knowledge of neighboring words in the sequence is used, which is very useful for this task. For example, knowing that names almost always come together with surnames, and that it is possibly accompanied by a job, position or profession in relation to a person, is very valuable information that the model captures. In addition, the CRF models conditional probabilities to figure out the correct output sequence from observations so a complete model for named entity recognition is achieved. Finally, with a good choice of hyperparameters and careful training monitoring, competitive classification results can be obtained.

Now, the results are shown for the recognition of entities in the n-best hypotheses of the test set lines using the previous model trained with the ground-truth set of lines from the training set with the same hyperparameter configuration. Then, the neural network has been trained with the 10-best hypotheses of the training set lines already labeled by the HTR system, and the inference has been performed again on n-best of the test lines to check for an improvement. NER performances for all these experiments are summarized below in Table 7.3.

Using the formula 7.3 to determine the global WTER of the system, it is noteworthy that as the number of best hypotheses n increases in the test set, the resulting WTER decreases as a consequence of exploring more tag sequence alternatives that can be more similar to the reference tags from the ground-truth lines. That said, the 1-best or the best hypothesis generated by the recognizer does not necessarily have to be the sequence that produces the lowest WTER. In fact, that is the reason for the error improvement seen in the Figure 7.3, where indeed, a drop in the WTER is seen as the number of hypotheses to be considered increases. The decrease in the error is less remarkable with greater n value since it is more difficult to produce an improvement as more hypotheses are considered. Furthermore, a slight improvement close to 1% in WTER occurs when the network is trained with the 10-best hypotheses of the training lines. The tags used to train the NER model are the ones generated by the HTR system, which can be incorrectly assigned to the recognized words. To adequately tag the hypotheses, word insertion labels must be taken into consideration. This improvement in WTER occurs because there are failures in the recognition of certain words in the training set that also occur in the test set. Therefore, the model learns new word embeddings, adjusts its parameters according to the new labeled hypotheses and correctly matches the entity categories in the inference more frequently. In this occasion for simplicity, all hypotheses have been given equal weight in training but could be weighted according to the posterior probability returned by the HTR system. This is, the model pays more attention to the tags in the most likely sequences rather than equal attention in all possible word paths. Also, mention that the WTER obtained with the ground-truth test set is 5.4%, so NER quality has been lost in both scenarios, as expected.

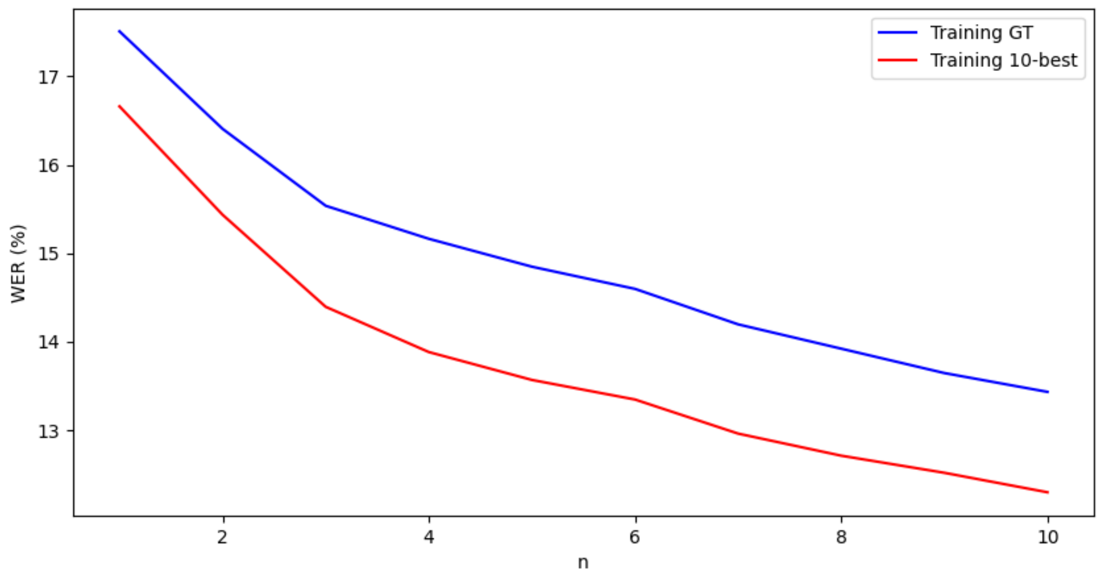


Figure 7.3: Evolution of the WTER on the n-best hypotheses of the test lines when training the neural network on the manual transcriptions and the transcriptions generated by the handwritten text recognizer for the train set lines.

After training the model with the 10-best hypotheses of the training set lines, the recognition results on the best hypotheses or 1-best of the test lines, which would be the final decodings generated by the handwritten text recognizer, are illustrated in Table 7.2. The reference tags used to evaluate the system's performance are those from the ground-truth set given that a similarity threshold value in recognized word and actual word is surpassed. Otherwise, "none" tag is used as reference. It should be noted that the number of samples in each class and in total are less than those in Table 7.1 for the ground-truth set (14% less) since deletions of certain entities have been produced in the HTR process. Again, "none" tags have been discarded from the results. In general, quality has been lost in entity detection. There is a considerable drop in the global metrics, in addition to the individual ones by type of entity. The system's F1 score has dropped to around 0.2 which indicates that it is having a harder time identifying entities. Highlight some important ones such as surnames, institutions and occupations, whose F1 values have decreased by more than 0.2 points.

Table 7.2: Evaluation results for the 1-best set of the test lines.

Entity Tag	Precision	Recall	F1 score	Support
Surname	0.772	0.570	0.656	730
Position	0.647	0.513	0.572	339
Count	0.571	0.667	0.615	6
Cross	0.875	1.000	0.933	7
Duke	0.857	0.261	0.400	23
Institution	0.628	0.235	0.342	324
Location	0.497	0.570	0.531	291
Marquis	1.000	0.231	0.375	13
Name	0.844	0.690	0.759	313
Roman numeral	0.930	0.988	0.958	81
Office	0.486	0.410	0.445	424
Job	0.943	0.892	0.917	37
Macro avg	**0.754**	**0.586**	**0.625**	**2588**

Figure 7.4 shows the evolution of the F1 of the system over the n-best of the test set when trained with the ground-truth transcriptions and the 10-best hypotheses for the training set generated by the HTR system. The evaluation is done in a similar way to the formula 7.3, only that the hypothesis (predictions and references) with the highest F1 is selected for each line to obtain the average score at every value of n. An increase in the F1 value of more than 0.1 is seen when considering up to the 10-best hypotheses in the two experiments, demonstrating the effectiveness of having more alternatives to examine. However, the improvement is not as notable when training with the 10-best, even for low values of n, the F1 is slightly higher when training with the ground-truth transcripts. This could be related with the tags used to train the system as they are predicted from the HTR process meaning that errors are present. Its quite surprising the increase in F1 produced when including the 7 best decodings.

The noise introduced into the vocabulary by the handwritten text recognizer causes the neural model to fail to correctly detect some named entities. Despite learning from the context in training and taking advantage of word embeddings, there are many new terms created with these hypotheses that can occur in ambiguous contexts. There are a considerable number of words that are unknown and given this, the model can use the information from the character composition to reveal part of their meaning given the LSTM that memorizes the order of appearance of symbols and extracts the reduced features. Even so, a low quality of recognition of the handwritten text can make the NER task difficult as we have seen.

A summary of the results obtained for the mentioned experiments is shown in Table 7.3. Indeed, there is a significant loss in performance when testing over the outputs generated by the HTR system. However, these last set of results do reflect more in reality how the NER performance would look like under a scenario where there are many collections of ancient handwritten books where indexing and searching based on entities is

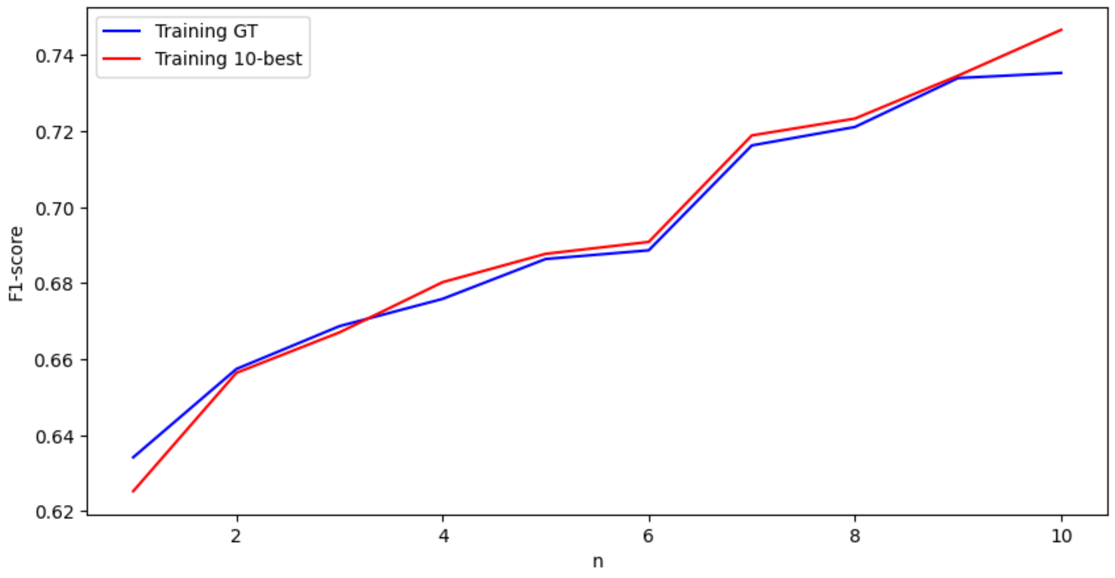


Figure 7.4: Evolution of the F1 score for the n-best hypotheses of the test lines when training the neural network on the ground-truth transcriptions and the transcriptions generated by the handwritten text recognizer for the train set lines.

required. This is because NER systems are not evaluated over ground-truth transcripts as they are very costly to obtain, rather on the automatic decodings produced by the HTR system. Both F1 and WTER improve as the number of best hipotheses increases up to 10 during inference. Training with the 10-best set makes a small improvement that is better reflected by the decrease in WTER close to 1%.

Table 7.3: Summary of NER performances of the proposed system for all the experiments carried out at line level.

Training set	Test set	F1 score	WTER (%)
Ground-truth	Ground-truth	0.834	5.4
Ground-truth	1-best	0.635	17.5
Ground-truth	10-best	0.729	13.6
10-best	1-best	0.625	16.7
10-best	10-best	0.748	12.3

In the Table 7.1 classification results obtained for the ground-truth transcriptions of the test lines, there are some entity recognition failures caused by not having enough preceding or following information since the context is limited to the line of the manuscript. This is a drawback for those cases in which there are dependencies between entities located on different lines, especially when there are entities composed of several words that are incomplete due to line breaks. For example, for the test line: “de Soria es una persona que nos ha servido mucho y está en deuda con nuestros servidores y servidores”, the term “Soria” is incorrectly recognized as a location instead of a surname. If the model had had the information from the previous line with the name of the person in question, this error could possibly be avoided by the CRF. That is why the neural network has been evaluated by modifying the amount of context that is passed as input, limited to the card length. Figure 7.5 shows the evolution of the F1 metric by varying the size of the window applied to the cards in terms of the number of words contained. The best performance is achieved with a size between 30 and 50 words, obtaining F1 scores greater than 0.85. There comes a point where introducing more context degrades the overall accuracy of

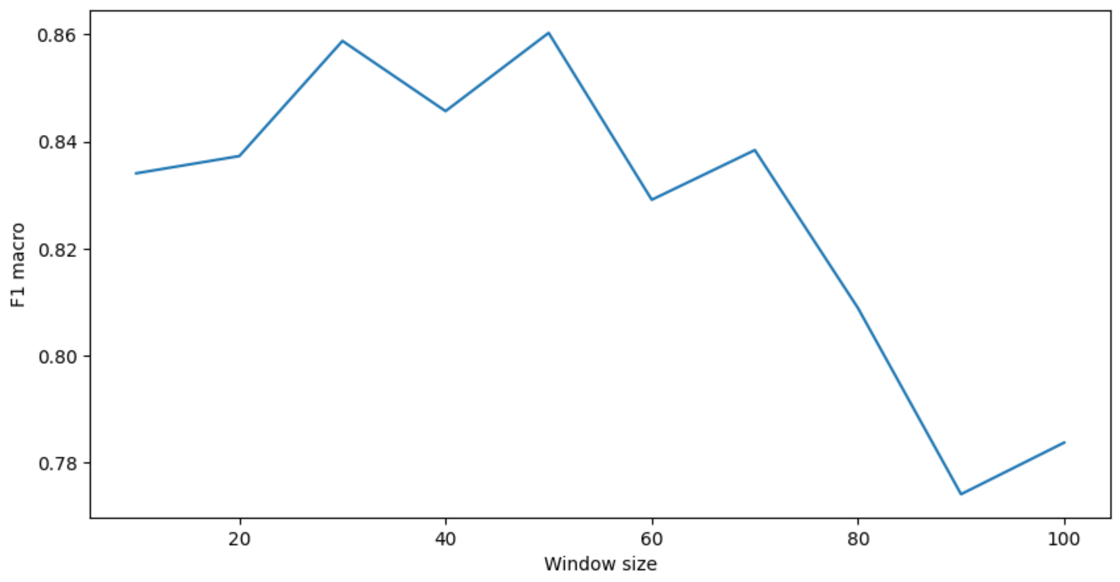


Figure 7.5: Evaluation results when modifying the size of the window that is applied to each ID card, ranging from 10 to 100 words in steps of 10.

the system since there are no significant long-term dependencies to model and noise is introduced into the Bi-LSTM and CRF. Mention that the same hyperparameter configuration has been maintained in all tests so that the comparison is fair (except for the length of the word sequences, which is the hyperparameter to compare).

As a conclusion to the results obtained in all the experiments, each layer of the neural model contributes to the success of recognition of named entities in handwritten texts by processing sequential data to extract a final label vector. With this many-to-many architecture characteristic of recurrent neural networks, good entity classification performance has been achieved in the original transcripts of the identity cards to identify the most relevant entities. Although it is true that the model has difficulty identifying certain entities such as institutions, locations and positions, which can create further confusion, the F1 score obtained at the line level of 0.83 indicates that high precision and coverage can be achieved at the same time for all classes, evidencing the good entity recognition features. However, the context at the line level is sometimes incomplete to accurately determine the label of an entity. Considering the information from the neighboring lines has helped to achieve a slight improvement in terms of classification results, although an excess of context produces losses in the F1 score, as seen in Figure 7.5.

In relation to the best hypotheses generated by the handwritten text recognition system, the WTER has been evaluated to measure the similarity in the predicted labels and reference line labels, indicating that very few operations are needed on average for these to be equal. NER quality has been lost compared to the ground-truth testing experiment because the WTER obtained has increased, something that was expected. As there are insertions and deletions of words caused by errors in the HTR process, the editing distance increases even if the predictions are perfect. The generated outputs closely fit each word as a result of learning representations of features and context from the ground-truth training lines, although there are cases where the network has a harder time identifying entities due to errors in transcription. This is when the morphological composition of the unknown vocabulary and the probabilities of emission and transition between states learned in the conditional random field are taken advantage of for the final decoding. Even so, the context can in some cases be ambiguous causing recognition difficulties in the entities. The error has been reduced by 1% by training the system with the n best hypotheses of the lines belonging to the training set, showing that learning from a larger vocabulary in which there can be different versions of the same word helps in inference when failures that occur in the HTR process have already been seen in training. Additionally, the error has decreased by 5% as the value of *n* increases from 1 to 10 of the *n* best hypotheses of the test lines. This demonstrates the importance of not considering only the best transcription or the one with the highest a posteriori probability when evaluating the system since there may be another less likely path generated by the handwritten text recognizer with greater success in detecting entities by the proposed neural model.

CHAPTER 8

Conclusions

With the results obtained and analyzed, the main objective of this work, consisting of the design of a Bi-LSTM-CRF neural model capable of identifying the named entities of interest in a subset of ID cards located in the collection of Books of records of royal decrees, has been satisfactorily completed. By having the labeled manual transcriptions, the neural network has managed to correctly detect and classify the entities, being able to confirm that there are many words labeled according to their context, either at the line level or considering the information from more neighboring lines in the card. Through a composition of multiple layers that process sequential data with each one fulfilling a different task, it has been possible to create an effective and efficient grammatical tagging system to recognize entities in handwritten texts. The short-term and long-term dependencies between words in the input sequence are successfully modeled by the bidirectional LSTM and the distributed representations of learned features contribute to the distinction between words based on a set of numerical properties. Furthermore, providing a probabilistic model that does not consist of strict independence assumptions in the input data, such as a CRF, is an advantage when decoding the labels since it involves assigning conditional probabilities so that the most likely sequence is finally emitted given contextual vectors that contain information about neighboring words. Based on the best hypotheses of the original lines, it has been proven that there is a drop in performance on WTER and F1 score caused by recognition failures of certain words, which sometimes are inevitable due to the nature of the handwriting. However, these metrics have been improving as more hypotheses have been introduced into the evaluation, which shows that relying on more transcription alternatives for a given line positively influences the detection of named entities. For example, the model may fail to recognize an entity in all the generated hypotheses for a given line except for one, which does not have to be the one with the highest posterior probability.

8.1 Reproducibility

The neural model created to recognize entities along with the results of all the experiments carried out and the data available can be found in my GitHub repository [63]. Alternative scenarios where the NER system can be taken advantage of include in medical and legal texts to detect people or material of interest in an automated way, in search engines for indexing documents according to the entities in user queries, in the extraction of financial data that is costly to determine in large collections of unstructured documents and in the selection of candidates for a job based on the detection of relevant entities in the resumes such as companies in which they have worked, universities and skills.

8.2 Relationship of the work developed with the studies completed

Throughout the master's degree, many technologies have been covered and knowledge acquired in different subjects. Of all of them, the ones that have been put into practice the most during the work are programming in Python, especially with the frameworks or the open source libraries of Keras and Tensorflow to create custom neural networks. They have been studied and implemented in the subject of neural networks but due to the great variety of functionalities and types of existing model architectures adapted for specific tasks, new techniques have been required that have not been taught, as is usual in many AI projects.

On the other hand, the concepts related to natural language processing introduced and put into practice in the subjects of computational linguistics have also been very useful, such as words embeddings and morphological labeling or *POS tagging* which also deals with collections of annotated documents. Furthermore, the iterative parameter optimization processes taught in the shape recognition subjects, including the gradient descent used in the neural network backpropagation algorithm, have been very beneficial for understanding the training process involved in the proposed RNN. Finally, highlight the importance of probabilistic models based on structured predictions studied such as conditional random fields and HMM as they determine the label sequences for given observations with the Viterbi algorithm.

The most significant transversal skills that have been needed for the development of the work are:

1. **CT2 – Application and practical thinking**: Advantages and disadvantages of all approaches have been identified in relation to NER. The option selected to meet the proposed objectives is a recurrent neural network that provides one of the best classification performances that constitutes the state-of-the-art results in many data collections, which is why the proposal is considered the most suitable after having analyzed reference results of other systems. Although in deep learning NER models have many variations in terms of the final composition, the most popular techniques and elements in the field of research have been chosen, showing their performance on an old collection of texts with specific evaluation metrics.

2. **CT3 – Analysis and problem resolution**: The solution presented has gone through many correction phases starting from an initial idea. As new alternatives have been explored and known that are highly valued by the scientific community, the model has been expanded with additional components until a series of desired results have been achieved. With all the pieces put together, a powerful system has been created to recognize entities in handwritten texts.

3. **CT11 – Permanent learning**: Many concepts related to the neural network and especially to the processing of sequential data were unknown until now. The methods and related techniques learned throughout the master's degree are not enough to successfully complete this task, so it is needed to invest time in research and reading other proposals to implement the best neural network possible. To solve programming errors that have arisen along the way, consulting forums has sometimes been key to get the model fully working as desired.

4. **CT12 – Planning and time management**: In a task such as a final master's thesis, the organization of time is a very important factor to ensure its correct completion. This implies dedication and following a scheme for developing subtasks. At first, it

has been difficult to balance course studies with research on the content including time reading about related work and code implementation. However, as the master's subjects have been completed, the development of the project has been greater and more follow-up meetings have been held.

8.3 Future work

As the time to carry out this work is limited, future directions are proposed to explore based on the study completed in the detection of named entities in handwritten texts. Based on the original proposed objectives, the possibility of evaluating and comparing the performance of the model with another collection of AGS documents is proposed or even test with another sample of pages of the Books of records of royal decrees. Since they are books that have been written in different years and by different authors, it could be studied if there is a variation in the language, verify how it affects the quality of text recognition and therefore obtain new NER performances.

As earlier stated, NER has been done in two separate steps for this work, where first handwritten text is recognized and then entities are found by the Bi-LSTM-CRF model. However, many HTR systems are integrated with NER techniques to perform both tasks simultaneously. Thus, it is proposed to explore the combination of HTR and NER in just one step. In addition, the WTER and F1 metrics used to evaluate NER performance on the automatic transcripts generated by HTR may not be suitable in certain cases where a single word named entity is wrongly separated in two or even three parts. This is, the NER model can correctly tag those tokens based on the entity they refer but because insertions are produced, WTER increases. Therefore, it is highly encouraged to do more research into the way to evaluate NER systems in old manuscripts to adequately measure recognition accuracy.

Bibliography

[1] M. L. Díez Platas, S. Ros Munoz, E. González-Blanco, P. Ruiz Fabo, and E. Alvarez Mellado, "Medieval spanish (12th–15th centuries) named entity recognition and attribute annotation system based on contextual information," *Journal of the Association for Information Science and Technology*, vol. 72, no. 2, pp. 224–238, 2021.

[2] J. P. Chiu and E. Nichols, "Named entity recognition with bidirectional lstm-cnns," *Transactions of the association for computational linguistics*, vol. 4, pp. 357–370, 2016.

[3] A. H. Toselli, E. Vidal, V. Romero, and V. Frinken, "Hmm word graph based keyword spotting in handwritten document images," *Information Sciences*, vol. 370, pp. 497–518, 2016.

[4] Canal Patrimonio, "El Archivo de Simancas es reconocido por la Unesco como Memoria del Mundo." `https://www.canalpatrimonio.com/los-fondos-del-archivo-de-simancas-son-reconocidos-por-la-unesco-como-memoria-del-mundo/`. [Accessed: 2023-06-29].

[5] Datapeaker, "Arquitectura de red neuronal convolucional." `https://datapeaker.com/big-data/arquitectura-de-red-neuronal-convolucional-arquitectura-cnn/`, 2020. [Accessed: 2023-09-26].

[6] M. Esteve, "Word embeddings." Retrieved from `https://marescas.medium.com/word-embeddings-8e6efd145e2e`, 2019. [Accessed: 2022-05-22].

[7] M. Marrero, J. Urbano, S. Sánchez-Cuadrado, J. Morato, and J. M. Gómez-Berbís, "Named entity recognition: fallacies, challenges and opportunities," *Computer Standards & Interfaces*, vol. 35, no. 5, pp. 482–489, 2013.

[8] C. Dozier, R. Kondadadi, M. Light, A. Vachher, S. Veeramachaneni, and R. Wudali, "Named entity recognition and resolution in legal text," in *Semantic Processing of Legal Texts*, pp. 27–43, Springer, 2010.

[9] E. Leitner, G. Rehm, and J. Moreno-Schneider, "Fine-grained named entity recognition in legal documents," in *International Conference on Semantic Systems*, pp. 272–287, Springer, 2019.

[10] J. C. S. Alvarado, K. Verspoor, and T. Baldwin, "Domain adaption of named entity recognition to support credit risk assessment," in *Proceedings of the Australasian Language Technology Association Workshop 2015*, pp. 84–90, 2015.

[11] S. Keretna, C. P. Lim, D. Creighton, and K. B. Shaban, "Enhancing medical named entity recognition with an extended segment representation technique," *Computer methods and programs in biomedicine*, vol. 119, no. 2, pp. 88–100, 2015.

[12] K. Xu, Z. Zhou, T. Hao, and W. Liu, “A bidirectional lstm and conditional random fields approach to medical named entity recognition,” in *International Conference on Advanced Intelligent Systems and Informatics*, pp. 355–365, Springer, 2017.

[13] A. Jiao, “An intelligent chatbot system based on entity extraction using rasa nlu and neural network,” in *Journal of Physics: Conference Series*, vol. 1487, p. 012014, IOP Publishing, 2020.

[14] N. Dingwall and V. R. Gao, “Enhancing gazetteers for named entity recognition in conversational recommender systems,” in *Proceedings of the Joint KaRS & ComplexRec Workshop. CEUR-WS*, 2021.

[15] I. J. Unanue, E. Z. Borzeshi, and M. Piccardi, “Recurrent neural networks with specialized word embeddings for health-domain named-entity recognition,” *Journal of biomedical informatics*, vol. 76, pp. 102–109, 2017.

[16] C. Malarkodi, E. Lex, and S. L. Devi, “Named entity recognition for the agricultural domain.,” *Res. Comput. Sci.*, vol. 117, no. 1, pp. 121–132, 2016.

[17] J. M. Giorgi and G. D. Bader, “Towards reliable named entity recognition in the biomedical domain,” *Bioinformatics*, vol. 36, no. 1, pp. 280–286, 2020.

[18] J. C. S. Alvarado, K. Verspoor, and T. Baldwin, “Domain adaption of named entity recognition to support credit risk assessment,” in *Proceedings of the Australasian Language Technology Association Workshop 2015*, pp. 84–90, 2015.

[19] S. Mori, C. Y. Suen, and K. Yamamoto, “Historical review of ocr research and development,” *Proceedings of the IEEE*, vol. 80, no. 7, pp. 1029–1058, 1992.

[20] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, “A novel connectionist system for unconstrained handwriting recognition,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 5, pp. 855–868, 2008.

[21] J. R. Curran and S. Clark, “Language independent ner using a maximum entropy tagger,” in *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pp. 164–167, 2003.

[22] G. R. Doddington, A. Mitchell, M. A. Przybocki, L. A. Ramshaw, S. M. Strassel, and R. M. Weischedel, “The automatic content extraction (ace) program-tasks, data, and evaluation.,” in *Lrec*, pp. 837–840, Lisbon, 2004.

[23] G. Demartini, T. Iofciu, and A. P. d. Vries, “Overview of the inex 2009 entity ranking track,” in *International Workshop of the Initiative for the Evaluation of XML Retrieval*, pp. 254–264, Springer, 2009.

[24] K. Balog, P. Serdyukov, and A. P. d. Vries, “Overview of the trec 2010 entity track,” tech. rep., NORWEGIAN UNIV OF SCIENCE AND TECHNOLOGY TRONDHEIM, 2010.

[25] A. S. Starostin, V. V. Bocharov, S. V. Alexeeva, A. A. Bodrova, A. S. Chuchunkov, S. Dzhumaev, I. V. Efimenko, D. V. Granovsky, V. F. Khoroshevsky, I. V. Krylova, *et al.*, “Factrueval 2016: evaluation of named entity recognition and fact extraction systems for russian,” 2016.

[26] C. Grover, S. Givon, R. Tobin, and J. Ball, “Named entity recognition for digitised historical texts,” in *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC’08)*, 2008.

[27] K. Bontcheva, D. Maynard, H. Cunningham, and H. Saggion, "Using human language technology for automatic annotation and indexing of digital library content," in *International Conference on Theory and Practice of Digital Libraries*, pp. 613–625, Springer, 2002.

[28] G. Crane and A. Jones, "The challenge of virginia banks: an evaluation of named entity analysis in a 19th-century newspaper collection," in *Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries*, pp. 31–40, 2006.

[29] M. Ehrmann, G. Colavizza, Y. Rochat, and F. Kaplan, "Diachronic evaluation of ner systems on old newspapers," in *Proceedings of the 13th Conference on Natural Language Processing (KONVENS 2016)*, pp. 97–107, Bochumer Linguistische Arbeitsberichte, 2016.

[30] T. L. Packer, J. F. Lutes, A. P. Stewart, D. W. Embley, E. K. Ringger, K. D. Seppi, and L. S. Jensen, "Extracting person names from diverse and noisy ocr text," in *Proceedings of the fourth workshop on Analytics for noisy unstructured text data*, pp. 19–26, 2010.

[31] J. Li, A. Sun, J. Han, and C. Li, "A survey on deep learning for named entity recognition," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 1, pp. 50–70, 2022.

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

[33] Z. Yang, R. Salakhutdinov, and W. Cohen, "Multi-task cross-lingual sequence tagging from scratch," *arXiv preprint arXiv:1603.06270*, 2016.

[34] G. Wu, G. Tang, Z. Wang, Z. Zhang, and Z. Wang, "An attention-based bilstm-crf model for chinese clinic named entity recognition," *Ieee Access*, vol. 7, pp. 113942–113949, 2019.

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

[36] C. B. Monroc, B. Miret, M.-L. Bonhomme, and C. Kermorvant, "A comprehensive study of open-source libraries for named entity recognition on handwritten historical documents," in *International Workshop on Document Analysis Systems*, pp. 429–444, Springer, 2022.

[37] A. C. Rouhou, M. Dhiaf, Y. Kessentini, and S. B. Salem, "Transformer-based approach for joint handwriting and named entity recognition in historical document," *Pattern Recognition Letters*, vol. 155, pp. 128–134, 2022.

[38] V. Romero, A. Fornés, N. Serrano, J. A. Sánchez, A. H. Toselli, V. Frinken, E. Vidal, and J. Lladós, "The esposalles database: An ancient marriage license corpus for off-line handwriting recognition," *Pattern Recognition*, vol. 46, no. 6, pp. 1658–1669, 2013.

[39] E. Boroş, V. Romero, M. Maarand, K. Zenklová, J. Křečková, E. Vidal, D. Stutzmann, and C. Kermorvant, "A comparison of sequential and combined approaches for named entity recognition in a corpus of handwritten medieval charters," in *2020 17th International conference on frontiers in handwriting recognition (ICFHR)*, pp. 79–84, IEEE, 2020.

[40] M. Ehrmann, A. Hamdi, E. L. Pontes, M. Romanello, and A. Doucet, "Named entity recognition and classification in historical documents: A survey," *ACM Computing Surveys*, vol. 56, no. 2, pp. 1–47, 2023.

[41] V. M. Ngo, G. Munnelly, F. Orlandi, P. Crooks, D. O'Sullivan, and O. Conlan, "A semantic search engine for historical handwritten document images," in *International Conference on Theory and Practice of Digital Libraries*, pp. 60–65, Springer, 2021.

[42] M. Humbel, J. Nyhan, A. Vlachidis, K. Sloan, and A. Ortolja-Baird, "Named-entity recognition for early modern textual documents: a review of capabilities and challenges with strategies for the future," *Journal of Documentation*, vol. 77, no. 6, pp. 1223–1247, 2021.

[43] Wikipedia, "Archivo general de simancas." https://es.wikipedia.org/wiki/Archivo_General_de_Simancas. [Accessed: 2023-06-30].

[44] X. Han, C. K. Kwoh, and J.-j. Kim, "Clustering based active learning for biomedical named entity recognition," in *2016 International joint conference on neural networks (IJCNN)*, pp. 1253–1260, IEEE, 2016.

[45] J. Heaton, "An empirical analysis of feature engineering for predictive modeling," in *SoutheastCon 2016*, pp. 1–6, IEEE, 2016.

[46] A. Prasad, H. Déjean, J.-L. Meunier, M. Weidemann, J. Michael, and G. Leifert, "Bench-marking information extraction in semi-structured historical handwritten records," *arXiv preprint arXiv:1807.06270*, 2018.

[47] H.-g. Lee, G. Park, and H. Kim, "Effective integration of morphological analysis and named entity recognition based on a recurrent neural network," *Pattern Recognition Letters*, vol. 112, pp. 361–365, 2018.

[48] S. T. Piantadosi, "Zipf's word frequency law in natural language: A critical review and future directions," *Psychonomic bulletin & review*, vol. 21, no. 5, pp. 1112–1130, 2014.

[49] G. Panchal, A. Ganatra, Y. Kosta, and D. Panchal, "Behaviour analysis of multilayer perceptrons with multiple hidden neurons and hidden layers," *International Journal of Computer Theory and Engineering*, vol. 3, no. 2, pp. 332–337, 2011.

[50] S. Du, J. Lee, H. Li, L. Wang, and X. Zhai, "Gradient descent finds global minima of deep neural networks," in *International conference on machine learning*, pp. 1675–1685, PMLR, 2019.

[51] S. Kanai, Y. Fujiwara, and S. Iwamura, "Preventing gradient explosions in gated recurrent units," *Advances in neural information processing systems*, vol. 30, 2017.

[52] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[53] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[54] P. Ha, S. Zhang, N. Djuric, and S. Vucetic, "Improving word embeddings through iterative refinement of word-and character-level models," in *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 1204–1213, 2020.

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

[56] G. D. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[57] I. Kandel and M. Castelli, "The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset," *ICT Express*, vol. 6, 05 2020.

[58] Y. Yao, J. Wang, P. Long, M. Xie, and J. Wang, "Small-batch-size convolutional neural network based fault diagnosis system for nuclear energy production safety with big-data environment," *International Journal of Energy Research*, vol. 44, no. 7, pp. 5841–5855, 2020.

[59] TensorFlow, "Introducing tensorflow feature columns." `https://developers.googleblog.com/2017/11/introducing-tensorflow-feature-columns.html`, 2017. [Accessed: 2022-06-30].

[60] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," *Advances in neural information processing systems*, vol. 29, 2016.

[61] A. Stiegler, "Exploring conditional random fields for nlp applications." `https://hyperscience.com/tech-blog/exploring-crfs-for-nlp-applications/`, 2021. [Accessed: 2023-11-22].

[62] R. Haldar and D. Mukhopadhyay, "Levenshtein distance technique in dictionary lookup methods: An improved approach," *arXiv preprint arXiv:1101.1232*, 2011.

[63] J. Giner, "Ner with bi-lstm-crf model." `https://github.com/JoseGiner67/TFG/tree/main/NER%20AGS`, 2023. [Accessed: 2023-11-22].