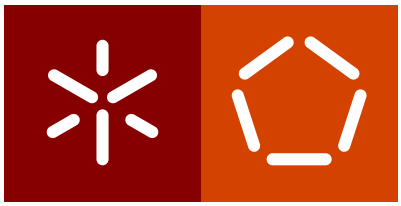


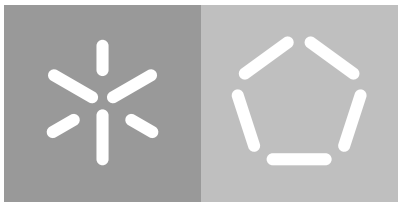
Universidade do Minho
Escola de Engenharia
Departamento de Informática

Luís Filipe da Costa Cunha

Entity Recognition in Archival Descriptions

Dissertation

February 2022



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Luís Filipe da Costa Cunha

Entity Recognition in Archival Descriptions

Dissertation

Master dissertation
Integrated Master's in Informatics Engineering

Dissertation supervised by
José Carlos Leite Ramalho

February 2022

AUTHOR COPYRIGHTS AND TERMS OF USAGE BY THIRD PARTIES

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

Insert name

Assinado por: **LUÍS FILIPE DA COSTA CUNHA**
Num. de Identificação: 15288126
Data: 2022.02.23 23:55:05+00'00'

ACKNOWLEDGEMENTS

I would like to express gratitude to my supervisor, Professor José Carlos Ramalho, whose guidance, support, and encouragement have been invaluable throughout this dissertation. I am extremely grateful for our friendly chats at the end of our meetings and your support in my academic and personal achievements.

A special thanks to my parents, who always believed in me and provided the resources to keep pursuing my dreams.

Finally, to Alexandra Mendes and all my friends for always supporting me.

ABSTRACT

At the moment, there is a vast amount of archival data spread across the Portuguese archives, which keeps information from our ancestors' times to the present day. Most of this information was already transcribed to digital format, and the public can access it through archives' online repositories. Despite that, some of these documents are structured with many plain text fields without any annotations, making their content analyses difficult. In this thesis, we implemented several Named Entity Recognition solutions to perform a semantic interpretation of the archival finding aids by extracting named entities like Person, Place, Date, Profession, and Organization. These entities translate into crucial information about the context in which they are inserted. They can be used for several purposes with high confidence results, such as creating smart browsing tools by using entity linking and record linking techniques.

In this way, the main challenge of this work was the creation of powerful NER models capable of producing high confidence results. In order to achieve high result scores, we annotated several corpora to train our Machine Learning algorithms in the archival domain. We also used different ML architectures such as MaxEnt, CNNs, LSTMs, and BERT models. During the model's validation, we created different environments to test the effect of the context proximity in the training data.

Finally, during the model's training, we noticed a lack of available Portuguese annotated data, limiting the potential of several NLP tasks. In this way, we developed an intelligent corpus annotator that uses one of our NER models to assist and accelerate the annotation process.

Keywords: Named Entity Recognition, Archival Finding Aids, Machine Learning, Deep Learning, BERT, Data Annotation

RESUMO

De momento, existe uma vasta quantidade de dados arquivísticos espalhados pelos arquivos portugueses, que guardam informações desde os tempos dos nossos antepassados até aos dias de hoje. A maior parte desta informação já foi transcrita para o formato digital e encontra-se disponível ao público através de repositórios online dos arquivos. Apesar disso, alguns destes documentos estão estruturados com muitos campos de texto livre, sem quaisquer anotações, o que pode dificultar a análise do seu conteúdo. Nesta tese, implementamos várias soluções de Reconhecimento de Entidades Mencionadas, a fim de se realizar uma interpretação semântica sobre descrições arquivísticas, extraindo entidades tais como Pessoa, Local, Data, Profissão e Organização. Estes tipos de entidades traduzem-se em informação crucial sobre o contexto em que estão inseridas. Com métricas de confiança suficientemente elevadas, estas entidades podem ser utilizadas para diversos fins, como a criação de ferramentas de navegação inteligente por meio de técnicas de entity linking e record linking.

Desta forma, o principal desafio deste trabalho consistiu na criação de poderosos modelos NER que fossem capazes de produzir resultados de elevada confiança. Para alcançar tais resultados, anotamos vários datasets para treinar os nossos próprios algoritmos de Aprendizado de Máquina no contexto arquivístico. Para além disso, usamos diferentes arquiteturas de ML tais como MaxEnt, CNNs, LSTMs e BERT. Durante a validação do modelo, criamos diferentes ambientes de teste de modo a testar o efeito da proximidade de contexto nos dados de treino.

Por fim, durante o treino dos modelos verificamos que existe pouca quantidade de dados disponíveis anotados em português, o que pode limitar o potencial de várias tarefas de NLP. Desta forma, desenvolvemos um anotador de datasets inteligente que utiliza um dos nossos modelos de NER para auxiliar e acelerar o processo de anotação.

Palavras-Chave: Reconhecimento de Entidades Mencionadas, Descrições Arquivísticas, Machine Learning, Deep Learning, BERT, Anotação de dados

CONTENTS

1 INTRODUCTION 1
- 1.1 Motivation 2
- 1.2 Objectives 3
- 1.3 Document Structure 3

2 STATE OF ART 5
- 2.1 Archival Finding Aids 5
- 2.2 OAI-PMH 10
- 2.3 Named Entity Recognition 13
- 2.4 Active Learning for Named Entity Recognition 14
- 2.5 OpenNLP 16
  - 2.5.1 Maximum Entropy 16
  - 2.5.2 Features 18
  - 2.5.3 Entropy Maximization 19
- 2.6 spaCy 20
  - 2.6.1 Transition Based NER 21
  - 2.6.2 Deep Learning framework for NLP 21
- 2.7 TensorFlow BI-LSTM-CRF 25
  - 2.7.1 Recurrent Neural Network 25
  - 2.7.2 Long Short Term Memory 26
  - 2.7.3 Bidirectional Long Short Term Memory 27
  - 2.7.4 BI-LSTM-CRF 28
- 2.8 Transformers 29
  - 2.8.1 Transfer Learning 29
  - 2.8.2 Attention Mechanism 31
  - 2.8.3 Generative Pre-Training 34
  - 2.8.4 BERT 35
- 2.9 Evaluation 37

3 ARCHIVAL FINDING AIDS PROCESSING 40
- 3.1 Data Harvest 40
- 3.2 Data Description 41
- 3.3 Data Cleaning 42
- 3.4 Data Annotation 42
  - 3.4.1 Statistical model Approach 43

3.4.2 Regex Approach 44
3.4.3 Manual Approach 44
3.5 Data Parse 44
3.5.1 Format Converter 47
3.6 Results 48
3.7 Conclusion 50
4 NAMED ENTITY RECOGNITION MODELS 52
4.1 Available Portuguese NER models 52
4.2 Training NER Models 53
4.2.1 OpenNLP 54
4.2.2 spaCy 56
4.2.3 BI-LSMT-CRF 60
4.2.4 BERT 64
4.3 Conclusion 68
5 NAMED ENTITY RECOGNITION RESULTS 70
5.1 Individual NER model per Corpus 70
5.2 Generalized NER model 72
5.2.1 BERT Model's Results 74
5.3 Overall Results 77
5.4 Conclusion 81
6 NER@DI 83
6.1 Web Platform - NER@DI 83
6.1.1 Architecture 83
6.1.2 Features 85
6.1.3 Interface 86
6.2 Smart Annotator - ARCANO 88
6.2.1 ARCANO Interface 89
7 CONCLUSION 92
7.1 Contributions 94
7.2 Future Work 94

A SUPPORT MATERIAL; LISTINGS 103
A.1 "O Século" newspaper archival fond in XML format. 103
A.2 Paróquia do Curral das Freiras archival fond in XML format 106
A.3 ARCANO Sequence Diagram. 108

LIST OF FIGURES

Figure 1	Example of description levels structure.	8
Figure 2	Archival fond of *Casa Real* (Portugal).	9
Figure 3	News from the newspaper *O Século*	13
Figure 4	Information and Entropy probabilities.	17
Figure 5	Entropy function subject to restrictions.	20
Figure 6	Transition Based NER.	21
Figure 7	Embed process.	22
Figure 8	Encode process.	23
Figure 9	Attend process.	24
Figure 10	Predict process.	24
Figure 11	Recurrent Neural Network.	26
Figure 12	Long Short Term Memory memory cell.	27
Figure 13	BI-LSTM-CRF.	28
Figure 14	Attention Mechanism.	31
Figure 15	Attention Scores.	32
Figure 16	Self-Attention Complexity.	33
Figure 17	GPT results and directionality.	34
Figure 18	BERT bidirectionally applied to token level task.	36
Figure 19	Named entities distributed by corpus and entity type.	49
Figure 20	Named entities labels' distribution.	50
Figure 21	OpenNLP event stream tagging.	54
Figure 22	OpenNLP model learning curves.	55
Figure 23	spaCy training workflow.	56
Figure 24	2d spaCy word embeddings distribution.	58
Figure 25	spaCy error gradient of the loss function.	59
Figure 26	spaCy model learning curve.	60
Figure 27	BI-LSTM-CRF Word Embeddings distribution.	63
Figure 28	BI-LSTM-CRF model learning curve.	64
Figure 29	BERT transfer learning.	65
Figure 30	BERT models learning curves.	68
Figure 31	NER results by corpus.	78
Figure 32	NER results by entity label.	81

Figure 33	NER@DI architecture.	84
Figure 34	NER@DI, NER interface.	86
Figure 35	NER@DI annotated corpora.	87
Figure 36	*ARCANO* Anotator interface.	89
Figure 37	*ARCANO* training interface.	90
Figure 38	*ARCANO* statistics and customization.	91
Figure 39	ARCANO Sequence Diagram.	109

LIST OF TABLES

Table 1	Model confidence example.	15
Table 2	Confusion Matrix.	38
Table 3	Concrete example of the confusion Matrix.	38
Table 4	Number of annotated entities per corpus.	49
Table 5	Individual NER models results.	71
Table 6	Generalized NER models validation results.	73
Table 7	Generalized model results on unseen data.	74
Table 8	Generalized BERT models results.	76
Table 9	Generalized BERT models results on unseen data.	77
Table 10	Overall models validation results.	79
Table 11	Generalized NER models results by entity label.	80
Table 12	*ARCANO*, number of annotated entities.	89

ACRONYMS

A

ABM_805 Paróquia do Jardim do Mar.

ABM_807 Paróquia do Curral das Freiras.

ACA Arquivo da Casa do Avelar.

ADAM Adaptive Moment Estimation.

ANTT Arquivo Nacional da Torre do Tombo.

API Application Programming Interface.

B

BERT Bidirectional Encoder Representations from Transformers.

BI-LSTM Bidirectional Long Short-Term Memory.

BILUO Beginning, Inside, LAST, Unit, Outside.

BIO Beginning, Inside, Outside.

C

CNN Convolutional Neural Network.

CRF Conditional Random Field.

CSV Comma-separated values.

D

DGLAB Direção-Geral do Livro, dos Arquivos e das Bibliotecas.

E

ELMO Embeddings from Language Models.

F

FAA Familia Araújo de Azevedo.

G

GPT Generative Pre-Training.

GPU Graphics processing unit.

H

HTTP Hypertext Transfer Protocol.

HWCR Handwritten Character Recognition.

I

IFIP International Federation for Information Processing.

IG1 Inquirições de Genere 1.

IG2 Inquirições de Genere 2.

J

JSON JavaScript Object Notation.

L

LSTM Long Short-Term Memory.

M

MAXENT Maximum Entropy.

ML Machine Learning.

MLM Masked Language Modeling.

N

NER Named Entity Recognition.

NLP Natural Language Processing.

O

OAI-PMH Open Archive Initiative Protocol for Metadata Harvesting.

OCR Optical Character Recognition.

R

RNN Recurrent Neural Network.

T

TT Torre do Tombo.

U

UI Instalation Unit.

ULMFIT Universal Language Model Fine-tuning.

X

XML Extensible Markup Language.

XSLT eXtensible Stylesheet Language for Transformation.

1

INTRODUCTION

Throughout the history of Portugal, there was a need to create an archive where information about the kingdom was recorded. In 1378, the first known Portuguese certificate was issued by the institution Torre do Tombo (TT), during the reign of D. Fernando. Since then, TT has been responsible for creating a national archive where information about the king, his vassals, administration of the kingdom, overseas possessions and relations with other kingdoms was recorded. This institution, over 600 years old, was installed in a tower of the castle of Lisbon until 1755, when the famous Lisbon earthquake occurred on the 1st of November, causing the tower to collapse. Despite this, the documents were recovered and moved to the Monastery of *São Bento da Saúde*. Over time, TT began to perform essential functions, taking on a fundamental role in the registration of national information (ANTT, 2017).

Besides the monarchy, another entity was responsible for producing records in antiquity, the clergy. However, in the mid-twentieth century, due to the proclamation of the Portuguese Republic, several parishes' books became in possession of the government. This led to the creation of several archives distributed throughout the country. That said, with the last century social, economic and technological evolution, the amount public and private institutions has increased considerably, generating a colossal amount of archival records.

At the moment, most Portuguese archives have online repositories making their archival finding aids available to the public in digital format. Thus, in this work, we harvested the finding aids of these archives using the OAI-PMH protocol and performed a semantic interpretation of the collected data through the application of a well-known NLP technique, Named Entity Recognition. In order to do so, several ML approaches were used to identify and classify entities of interest from unstructured archival text, with the intent of discovering which method reveals the best results. For training statistical models capable of performing in the archival domain, we have gathered training data, which consists of selecting and annotating archival corpora. Then, we selected the ML algorithms usually associated with Named Entity Recognition (NER) and have revealed state-of-art results in this task.

The first algorithm used was Maximum Entropy (Maxent), which maximizes the entropy of a given model subject to its defined features to make less implicit decisions as possible. Then we experimented with Deep Learning, starting with Convolutional and Recurrent Neural Networks. Here, while the CNNs are resource-efficient, the LSTMs can preserve longer-term dependencies with the cost of higher complexity. Finally, the latest architecture consists of a relatively recent approach in the scientific community, which has revolutionized several NLP tasks, the Transformers. This new approach presents a self-learning technique that can efficiently use the GPUs' parallel computational power to train models with hundreds of million parameters, making them have a broader and richer understanding of the language used. After the models' training, they were all subjected to validation tests to analyze their performance, comparing the results of each one.

In the end, we found out that one of the barriers to creating a NER model applied to a specific context domain is to generate data to train the model. To address this problem, a smart annotation support tool has been implemented, which aims to use ML models to speed up and assist the annotation process.

Throughout this project, several tools were generated that allowed to facilitate and support its development. In order to encourage the investigation of this area of NLP, all the produced material was made public through the creation of a Web platform, NER@DI (Cunha and Ramalho, 2021c), including some of the ML models created in this work and the smart annotator.

1.1 MOTIVATION

Due to the need to record national information, several archives emerged across the whole country, highly increasing the number of documents in Portuguese archives, even more in recent years. Nowadays, there is a colossal amount of archival data already available online. According to *Direção-Geral do Livro, dos Arquivos e das Bibliotecas* (DGLAB), the digital archive has more than 40 million images and respective archival finding aids (DGLAB, 2021). However, the search for information in these documents can become challenging to perform due to their volume and complexity.

To promote the data search on archival data, we presented a solution that aims to encourage the search of information in large volumes of archival documentation using semantic criteria. Suppose the generated ML models can extract entities from the archival finding aids with high confidence. In that case, the entities extracted

could be used to develop intelligent browsing tools that would allow the navigation between records through the relationship between the extracted entities.

This browsing mechanism would create new possibilities for processing the archives' information, enabling new approaches for exploiting the available data using the relationships between different records which in the present is a complex operation to perform.

1.2 OBJECTIVES

The main goals of this dissertation are:

- Generate annotated archival data in order to train the ML models.
- Train and fine-tune several ML models capable of recognizing the required Named Entities from archival corpora.
- Obtain the high validation F1-score values to use the extracted entities in future works.
- Compare different ML algorithms to understand which one reveals the best results for this domain.
- Implement a Web platform to allow the use of the ML models as a service.

1.3 DOCUMENT STRUCTURE

This dissertation is structured as follows:

- **Chapter 1 - Introduction**

  This chapter introduces the subject of this dissertation, identifying its primary problem, motivation and objectives.

- **Chapter 2 - State of the art**

  This chapter presents the relevant literature and related work associated with this dissertation subject. It describes the archival documents structure, different NER approaches and state-of-art ML algorithms.

- **Chapter 3 - Archival Finding Aids Processing**

  This chapter describes the archival data processing, from data harvesting to data annotation, generating the ML training data.

- **Chapter 4 - Named Entity Recognition Models**

  This chapter introduces different approaches to training ML algorithms capable of performing NER using.

- **Chapter 5 - Named Entity Recognition Results**

  This chapter contains the validation results of the NER models.

- **Chapter 6 - NER@DI**

  This chapter presents a Web platform developed to publish the generated NER models, the intelligent data annotator, and some other tools that were created along with this dissertation.

- **Chapter 7 - Conclusion**

  This chapter contains this dissertation's main conclusion, contributions and future work.

2

STATE OF ART

The main objective of this thesis is the extraction of named entities from Portuguese archival finding aids with high validation results. Therefore, a research was carried out on the most viable methods of recognizing entities in natural text.

This chapter discusses several NER approaches that have shown good results in this field, as well as technologies and other resources that can be crucial to successfully processing archival documents. It also presents the structure of the archival data and the technologies needed to access the archives' online repositories.

2.1 ARCHIVAL FINDING AIDS

In this work, we intend to perform Named Entity Recognition in archival data, more precisely in their descriptions.

Archival Finding Aids, also called archival collection guides or archival descriptions, are documents that describe archival materials. In fact, archival collections can take enormous dimensions, which can make the search for information in these documents extremely challenging. By describing the archive's content, archival finding aids establish administrative, physical and intellectual control over the archives' holdings, helping the researchers retrieve the information they are looking for much faster.

In Portugal, guidelines for the archival description have been created that describe rules for standardizing the archival finding aids. The purpose of these standards is to create a working tool to be used by the Portuguese archivist community in creating descriptions of the documentation and its entity producer, thus promoting organization, consistency and ensuring that the created descriptions are in accordance with the associated international standards to this domain. In addition, the adoption of these guidelines makes it possible to simplify the research or information exchange process, whether at the national or international level.

These guidelines are divided into three distinct parts:

- Guidelines for descriptions of archival documentation;
- Guidelines for the description of entities holding archival descriptions;
- Guidelines for choosing and building standardized access points.

First, the guidelines for document descriptions consist of creating a solid structure capable of ensuring the consistency and organization of documents, providing methods of retrieving, integrating, sharing and exchanging data by organizing information through levels of description.

Secondly, there are guidelines for describing archival authorities, which are applied to describe the entities that own or create the archives, for example, individuals, families or even institutions, adding information to contextualize the environment in which the archival documents are created.

Finally, there are guidelines for choosing and building standardized access points. This type of guidance, as its name implies, consists of determining, controlling and standardizing the method of choosing the access points of the archival documentation, for example, the names of the entities or places associated with the archival documents (Rodrigues et al., 2011).

In order to process the archival finding aids, it is essential to understand its organization as well as to be aware of its structure. Despite the existence of these standards, they act as guidelines, so it is not always possible to comply with them. It is not expected that these methods will be strictly followed by files documented 500 years ago. That said, not all organizations have their internal processes structured the same way, so many of the entities that produce archives adopt their own structures, creating mechanisms that satisfy their needs.

This can be achieved through the hierarchy system presented in Rodrigues et al. (2011). In order to create a dynamic mechanism that allows the producing entities to shape the structure of their archival documents according to their own context, a hierarchical system based on levels of description is then used.

- Fond - All archival documents of a given entity or organization constitute a fond;
- Subfond - Corresponds to a subdivision of a fond which can exist independently. For example, administrative departments or family subdivisions;
- Section - A section corresponds to a subset of a fond or subfond which does not have a high degree of autonomy. It may represent geographical, chronological, functional, thematic or a class of a classification plan. (For many archivists, the concepts of Sub-Fond and Section are equivalent);

- Subsection - Corresponds to a subdivision of a section;
- Series - A series represents a set of documents, simple (pieces) or compound (files), which were associated when they were created because they are documents of a similar nature. Usually, documents belonging to a Series are associated with a specific function or activity or have a common relationship between them;
- Subseries - This level corresponds to a subdivision of a series;
- File - Corresponds to a set of documents organized and grouped for use by its holding entity. Usually, these documents are grouped by some criteria, such as the subject or associated activity;
- Piece or Item - An item corresponds to the simplest element of this system. This can contain the data associated with a letter, images or even sound records;
- Group of Fonds - A group of fonds, as the name implies, corresponds to a set of fonds which are grouped together for some specific purpose, for example, for archival management;
- Collection - The description level Collection corresponds to a set of documents that were grouped in an artificial way through a specific criterion. These documents may belong to different fonds. In addition, a collection can be generated at different levels of description;
- Installation unit - An installation unit corresponds to a structure capable of storing and preserving the desired information. For example, books, notebooks, diskettes, cassettes, databases, etc.

Figure 1 shows a valid example of a structure that an archive can adopt.

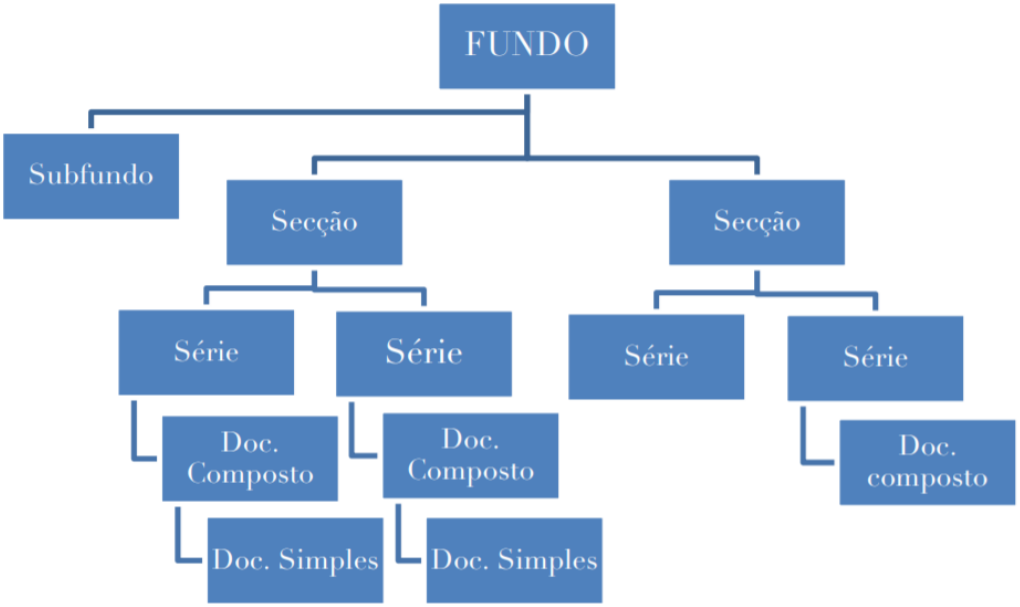


Figure 1: Example of description levels structure.

Analyzing this diagram, it is possible to observe different levels of archival description that together constitute a hierarchy of the structure of a given archive. Initially, there is a fond, which is divided into two sections and a subfond. These sections contain two series, each composed of files that have items. This system is dynamic, so it is easy to obtain different variations. Each entity that holds an archive implements a similar system with different levels of description, creating a structure that suits its context.

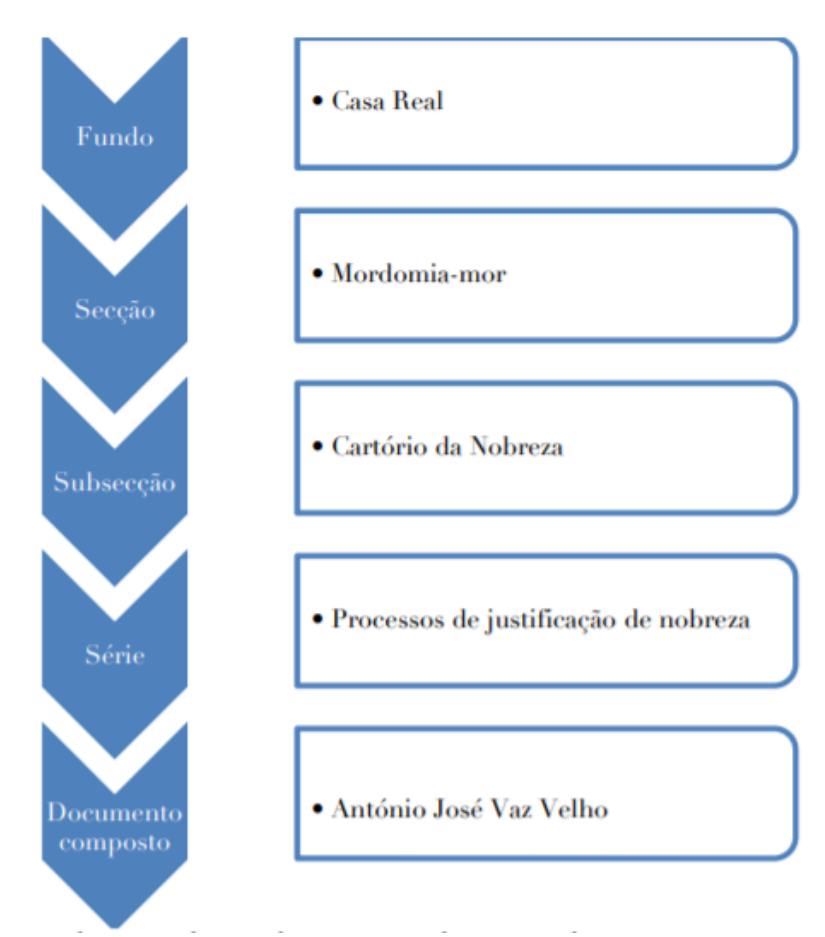


Figure 2: Archival fond of *Casa Real* (Portugal).

Figure 2 shows the structure of the description levels of the *Casa Real* archival belonging to the *Arquivo Nacional da Torre do Tombo (ANTT)* . As it can be seen, this fond, that corresponds to the Portuguese royalty, has at least one section, *Mordomia-mor*, that represents the people who were part of the royalty, associated with the palace management and administration territory. That said, there is a subsection of *Mordimia-mor*, the *Cartório da Nobreza*, which is associated with bureaucratic services of the court. Next, a series of the subsection is presented, *Processos de Justificação de nobreza*, corresponding to the nobility justification processes. Finally, there is a file associated with a singular entity, *António José Vaz Velho*, where all the nobility justification processes of this entity are grouped.

2.2 OAI-PMH

There are hundreds if not thousands of archives spread across the world. These archives keep records that contain the entire human history, so it is often requested access to their documents. In order to facilitate data sharing, most archives created their own online repository to allow access to their records. The online repositories use the OAI-PMH (Open Archive Initiative Protocol for Metadata Harvesting) protocol, ensuring the interoperability of standards, promoting broader and more efficient dissemination of information within the archival community. This protocol enables data providers to expose their structured metadata, enabling users to harvest it by using a set of six verbs invoked within HTTP requests (Lagoze et al., 2002).

- GetRecord - Verb used to retrieve an individual metadata record from a repository;
- Identify - Verb used to retrieve information about a repository.
- ListIdentifiers - This verb is an abbreviated form of ListRecords, retrieving only headers rather than records;
- ListMetadataFormats - Verb used to retrieve the metadata formats available from a repository;
- ListRecords - Verb is used to harvest records from a repository;
- ListSets - Verb used to retrieve the set structure of a repository.

In Portugal, most archival documents are already in digital format which can be found on various online repositories such as the *Portal dos Arquivos*[1] or the ANTT [2]. These documents are mapped into files with XML format, as we can see in Listing A.1, which corresponds to an archival fond related to the censorship experienced during the Salazar dictatorship in Portugal.

In this fond, we have an Identification Zone for each description unit. This zone contains metadata about each unit, such as reference code, title, date, description level, and dimensions. For example:

1 https://portal.arquivos.pt/
2 https://antt.dglab.gov.pt/

```
<archdesc level="otherlevel" otherlevel="F">
    <did>
        <langmaterial>Português</langmaterial>
        <physdesc>
            <dimensions>c. 44.000 u.i, c. de 2500 m.l.; papel, filme</dimensions>
        </physdesc>
        <repository>Arquivo Nacional da Torre do Tombo</repository>
        <unitdate label="UnitDates" type="inclusive" certainty="False/False" normal="1880/1979">
    ca. 1880/ca. 1979</unitdate>
        <unitid identifier="1009215" countrycode="PT" repositorycode="PT-TT">PT/TT/EPJS</unitid>
        <unittitle type="Atribuído">Empresa Pública Jornal O Século</unittitle>
    </did>
...
```

Listing 2.1: Fond description level.

This XML element represents the identification zone of the archival fond, with reference code `"PT/TT/EPJS"`. This code indicates that this fond belongs to Portugal (`PT`), in the *Torre do Tombo* archive (`TT`), corresponding to a Portuguese newspaper of that time, *Empresa Pública Jornal O Século* (`EPJS`). In addition, it is also possible to check the date associated with the fond (ca. 1880 / ca. 1979) as well as its title and dimensions. That said, following the organization of the document, there is a section.

```
<c level="otherlevel" otherlevel="SC">
    <did>
        <unitid identifier="4490062" countrycode="PT" repositorycode="PT-TT">PT/TT/EPJS/A</
    unitid>
        <unittitle type="Formal">Arquivo da Redação</unittitle>
    </did>
...
```

Listing 2.2: Section description level.

This element is contained in the XML element of the archival fond, thus corresponding to a subdivision of it. In this case, this section represents the *Arquivo da Redação*, editorial archive in Portuguese, with the reference code `"PT/TT/EPJS/A"` making this the section `A` of the fond. Finally, this section consists of a series, which in this case represent the censorship cuts of this newspaper, containing installation units.

```

<c level="otherlevel" otherlevel="UI">
    <did>
        <langmaterial>Português</langmaterial>
```

```
5         <physdesc>
6             <dimensions>1 mç. (143 f.); papel</dimensions>
7         </physdesc>
8         <physloc>Empresa Pública Jornal O Século, Cortes de Censura de 'O Século', cx. 191, mç.
      242</physloc>
9         <repository>Arquivo Nacional da Torre do Tombo</repository>
10        <unitdate label="UnitDates" type="inclusive" certainty="True/True" normal="
      1959-12-01/1960-02-29">1959-12-01/1960-02-29</unitdate>
11        <unitid identifier="4490285" countrycode="PT" repositorycode="PT-TT">PT/TT/EPJS/A/2/242<
      /unitid>
12        <unittitle type="Formal">Cortes de Censura de 'O Século': maço 242</unittitle>
13    </did>
14 ...
```

Listing 2.3: Instalation Unit description level.

As can be seen, this Instalation Unit (UI) has reference `"PT/TT/EPJS/A/2/242"`. In fact, through this reference it is possible to identify each level of description of the document. Listing 2.3 refers to Installation Unit *Maço 242* (UI 242) of the censorship cuts (Series 2), from the editorial archives (section A) of the newspaper *O Século*, belonging to the TT archive, Portugal (fond `"PT/TT/EPJS"`). In addition, this level of description contains data stored in images, and its corresponding textual description, which can be seen in Listing A.1. One of the images that can be found on this UI is presented in Figure 3.

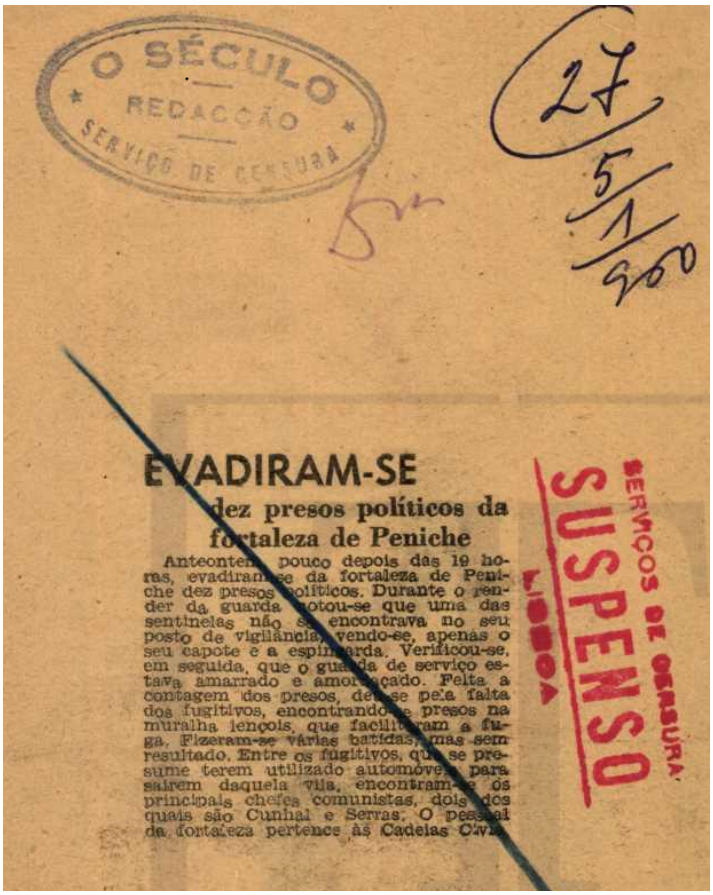


Figure 3: Censored news from the newspaper *O Século*, taken from ANTT online repository.

This image was taken from the newspaper *O Século*, which was censored by the famous *Lápis Azul*, a Portuguese symbol of censorship used in the Salazar dictatorship. This newspaper tried to publish the escape of *Álvaro Cunhal* and other political companions among the dictatorship's opposition, from Peniche fortress, in January 1960.

2.3 NAMED ENTITY RECOGNITION

One of the objectives of NLP is the classification and extraction of certain entities in textual documents. It is easy to understand that entities such as people's names, organizations, places or dates translate into crucial information about their contexts. This type of data can be used for various purposes, making this practice very popular. Therefore, a new NLP subfield rises, Named Entity Recognition.

In the past, two different approaches were taken to recognize entities in natural texts. Initially, specific regular expressions were coded to filter various entity types. In some cases, this mechanism delivered good results, mostly when there was an in-depth knowledge of the domain in which this method was intended to be applied. However, this is not always the case. In fact, such an approach was not considered exceptionally dynamic since it is necessary to rewrite a large part of the code if one wants to change the domain language. Furthermore, the existence of ambiguity between entities makes them hard to classify. For example, a person's name can be used as the name of a place.

Alternatively, statistical classifiers are used. This method consists of using ML models to predict whether a specific word sequence represents an entity. This approach has some advantages over the previous one. For example: one can now use this solution in different languages without changing much code; The model can be trained with different parameters and be adjusted to different contexts; An annotated dataset is generated that can be reused for other purposes; etc. In fact, today, several already pre-trained ML models are capable of identifying and classifying various entity types. However, the available models are generic, meaning that the entity prediction for more specific contexts can return poor results. Initially, these models were trained in a supervised fashion (Sekine and Ranchhod, 2009), however, recent research demonstrated that unsupervised methods could be used to increase the models' performance.

Despite being much more dynamic than the previous approach, using this type of model leads to some work for the experimenter. The experimenter must write down an annotated training dataset to prepare and train the model. Despite being tedious work, it has a low complexity level and therefore does not require great specialization (Ingersoll et al., 2013).

2.4 Active Learning for Named Entity Recognition

In this section, Active Learning applied to NER techniques will be addressed in order to explore methods of training Machine Learning statistical models.

Active Learning is a topic well known by the research community that aims to improve the performance of a given model, taking into account how the dataset used to train the model is chosen. In other words, this approach aims to strategically select, based on some criteria, the more informative examples from the available data. The idea is that the ML algorithm learns how to choose the best training data in order to achieve higher results with fewer annotations. With this selective method,

the model is expected to perform better than using traditional supervised learning, which consists of choosing a random set of data.

Thus, it is necessary to determine the most informative set of data for the model and how it can distinguish it. Active Learning proposes several approaches to do this, the most common one being *uncertainty sampling*. For example, Table 1 presents confidence values of a simulated model trying to find out which label, `A`, `B` or `C` the tokens `Word1` and `Word2` belong to.

	Label A	Label B	Label C
Word 1	0.1	0.1	0.8
Word 2	0.5	0.3	0.2

Table 1: Model confidence example.

Analyzing this table, the model has 0.8 of confidence that `Word1` corresponds to label `C` and 0.1 of confidence that the same word corresponds to labels `A` and `B`. The analysis is similar for the second row of the table, `Word 2`.

Thus, using the *Least Confidence* method, the word that describes more information for the model is `Word2` since the model has only 0.5 confidence that this word corresponds to label `A` and even less confidence for the other labels. Furthermore, as the probability is more distributed by the labels, it is considered that the model has difficulty in classifying the `Word2`. Choosing this word to be annotated and then used in training the model increases the likelihood that the model will be able to classify it in the future.

Another approach referred in Shen et al. (2018) is *Representativeness-Based*, which means that the model must learn to choose the information that does not add redundancy to it. It must look for data that increases its knowledge, allowing it to classify new problems. Adding repeated information to the model can generate the phenomenon of data over-fitting that can be avoided following this approach.

Finally, it is interesting to mention one more approach, *Entropy sampling* or *Uncertainty sampling* (Fang et al., 2017). The definition of entropy will be covered in the sections below, however, the idea behind this method is to choose the data with the highest entropy, i.e., the data with the greatest uncertainty. At first glance, this method looks the same as *Least Confidence*, however, to calculate the entropy of an instance, all probabilities are taken into account. In this example, it turns out that the word chosen would also be the `Word2`.

2.5 OPENNLP

One of the tools chosen for this dissertation was Apache OpenNLP, a machine learning-based toolkit implemented in Java, belonging to Apache. Essentially and as its name implies, its purpose is the processing of natural language through the use of ML algorithms. It delivers a wide range of features, such as tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing, and co-reference resolution.

In this thesis, the features associated with NER will be addressed, which depend on the tokenization task. At this time, Apache OpenNLP provides models for various tasks in several different languages such as English, Spanish, Danish, etc, however, it does not provide a pre-trained model of NER for the Portuguese language, so it will be necessary to train one model from scratch.

To understand how OpenNLP works, it is necessary to investigate what kind of ML algorithms it uses. In this case, the base algorithm used is Maximum Entropy, which will be explored in the following sections (OpenNLP, 2017).

2.5.1 *Maximum Entropy*

The concept of entropy was borrowed from physics (thermodynamics) and applied to various areas of computer science like the Information Theory or even classification algorithms, such as Maximum Entropy where entropy represents the level of uncertainty.

According to Goodfellow et al. (2016) in the *Information Theory*, the occurrence of a given event with a low probability of occurring translates into more information than the occurrence of an event with a high probability of occurring. On the other hand, there is *Information Entropy*, which, in *Information Theory* corresponds to the measure of uncertainty, i.e., the average quantity of information required to represent an event drawn from the probability distribution for a random variable. The entropy takes a low value when the probability of certainty for some event is high and takes a high value when all events are equally likely. Figure 4 illustrates these statements.

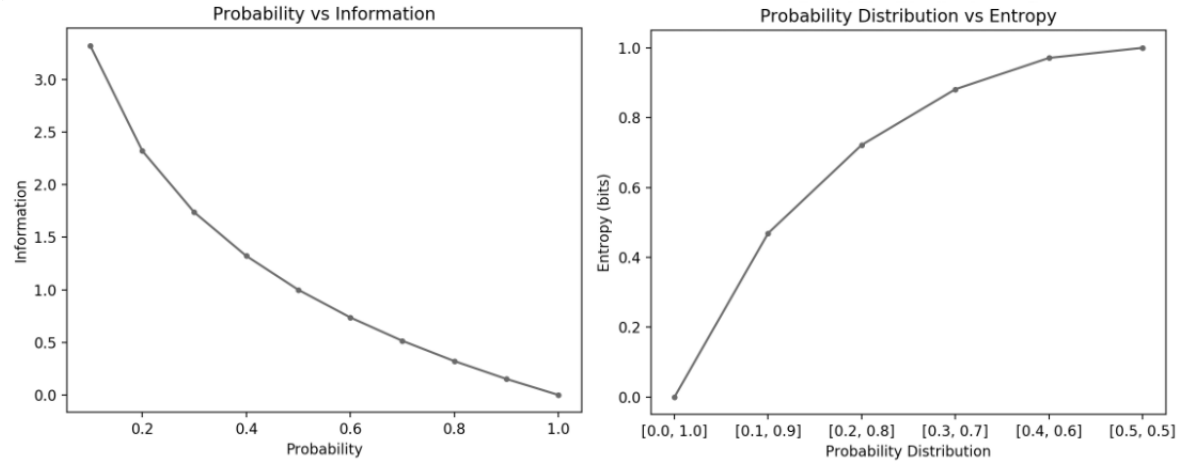


Figure 4: Information vs Probability (Left), Entropy vs Probability (Right).

Information entropy is a measure of the lack of structure or detail in the probability distribution describing your knowledge. - Jaynes, E. T. 1982.

Maximum Entropy Models are statistical models that maximize the entropy of probabilistic distribution subjected to an N number of constraints. These models reveal good results when used to model real-world problems considered hard to model. Usually, they are used to predict high dimensional data, in other words, when there is a number of possible combinations much greater than the amount of available data.

The principle behind this algorithm is that the distribution with the most uncertainty compatible with the context domain should be chosen. To do so, it is necessary to create several features that represent the information known about the domain. In fact, these features represent restrictions of the model, which help the classification of the intended target. After generating the features, it is necessary to maximize all models' entropy that satisfy these restrictions. By doing so, we are preventing our model from having features that are not justified by empirical evidence, preserving as much uncertainty as possible (Manning and Schütze, 1999).

Ignorance is preferable to error and he is less remote from the truth who believes nothing than he who believes what is wrong. - Thomas Jefeerson (1781).

In this chapter, the use of this algorithm applied to NER will be addressed, which will explore the model features as well as the maximization of entropy restricted to them.

2.5.2 *Features*

As previously stated, a feature is a way in which known information about the context is passed to the model as constraints, i.e., evidence or hints that make the model correctly classify certain specific cases.

Mathematically speaking, it can be represented as a binary function that for some given $x \in X$, that represents the class of the entities we are trying to predict, and $y \in Y$, that represents the possible contexts that we are observing, it returns the corresponding boolean value.

$$f : X \times Y \longrightarrow \{0, 1\}$$

All features correspond to functions with this signature. That said, each problem has its own characteristics, which makes it necessary to identify new features when we are faced with different contexts. We can say that features are context-dependent, allowing the algorithm to adapt to different problems. The experimenter must choose the type of information each feature adds to the model. In the function below, an example of a possible feature is represented.

$$f(a, b) = \begin{cases} 1 & \text{if a = Local and } \textit{checkLocation(b) = true} \\ \\ 0 & \text{otherwise.} \end{cases}$$

where

$$\textit{checkLocation}(b) = \begin{cases} 1 & \text{if previous word in } b \text{ is "em" and current word starts} \\ & \text{with capital letter.} \\ \\ 0 & \text{otherwise.} \end{cases}$$

In the Portuguese language, when the token *"em"* (in) precedes a word that starts with a capital letter, there is a high probability where that word corresponds to a

Place entity type (e.g., *"em Lisboa"*, *"em Inglaterra"*). Therefore, this feature would help the model to classify Place type entities.

Through this type of constraint, it is possible to restrict the model to our context, however, it is usually necessary to identify more than one feature, which leads to the problem of interdependence between them. This can be resolved iteratively so that the decision to be made, in a given iteration, takes into account previous decisions. This iterative process is really important on sequence tasks where the previous output can influence the following ones. For example, a person's name is a sequence of words starting with a capital letter. When the model classifies the first word as a person's name, there is a high probability that subsequent words starting with a capital letter also belong to that name. Thus, the model must have the ability to consider the decisions previously taken when classifying a certain token.

According to Ratnaparkhi (1998), the overlapping feature mechanism is the reason that makes the Maximum Entropy algorithm distinguish itself from other models. It makes it possible to add known information into the model and let the created features overlap to try to predict the best possible outcome.

2.5.3 *Entropy Maximization*

Following the MaxEnt algorithm, the optimal solution to this classification problem is the most uncertain distribution subject to the defined constraints. The idea behind this is to choose the model that makes the fewer implicit assumptions possible. Thus, after defining all constraints considered relevant to the context domain, the next step is to maximize the entropy of the model. In order to do so, the function of Information Entropy is used.

$$H(X) = -\sum p(a,b) \log p(a,b)$$

The maximum entropy function is a convex function, which means that the value of the weighted average of two points is greater than the value of the function in this set of points. Thus, the sum of the entropy function is also convex. A constraint on this function creates a linear subspace that corresponds to a surface that is also convex and, therefore, has only a global maximum (Manning, 2003).

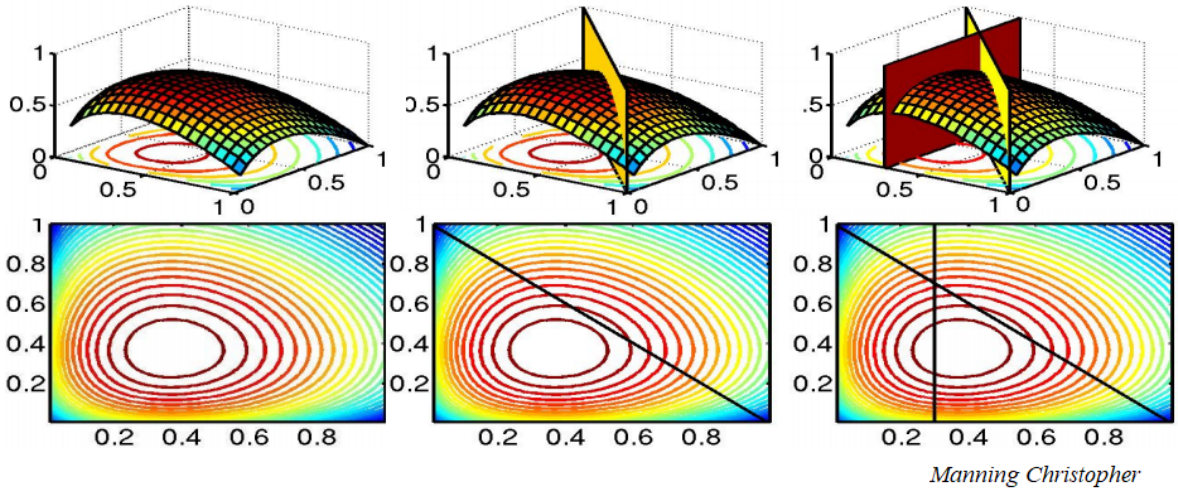


Figure 5: Entropy function subject to restrictions.

As explained in Berger et al. (1996), in order to maximize the entropy of the model subject to a limited number of features, we need to solve a constrained optimization problem. This problem might seem trivial at first glance. In fact, a problem with low complexity can be solved analytically, however, when the number of constraints increases and they overlap with each other, it is not possible to find a general solution analytically. This problem is then solved using Lagrange multipliers by forming a Lagrangian function. An example of this resolution can be found in Morais (2018).

2.6 spaCy

Another tool that was used in this dissertation is spaCy, an open-source library for advanced natural language processing, belonging to the company Explosion, founded by the creators of spaCy.

Again, this library offers several features associated with NLP, however, only those relevant to NER will be addressed. Despite having several similarities to OpenNLP, spaCy presents a very different approach to entity recognition, Deep Learning. It is no secret that Neural Networks have unlocked new possibilities in the Machine Learning context, achieving state-of-the-art results in many fields. This tool provides several base models of different languages such as Chinese, Danish, Dutch, English, French, Portuguese, etc. This is an advantage over the previous tool since we have an available Portuguese model.

This software was implemented in the python programming language and published under the MIT license. In this section, the approach taken by spaCy regarding the recognition of entities (spaCy, b) will be presented.

2.6.1 *Transition Based NER*

Most NER frameworks generally use a tagging system, which in practice translates into attaching a tag to each word of interest in the document to further classify it. Instead of using this type of structure, spaCy uses a different mechanism to deal with this problem, a transition-based approach.

Transition	Output	Stack	Buffer	Segment
	[]	[]	[Mark, Watney, visited, Mars]	
SHIFT	[]	[Mark]	[Watney, visited, Mars]	
SHIFT	[]	[Mark, Watney]	[visited, Mars]	
REDUCE(PER)	[(Mark Watney)-PER]	[]	[visited, Mars]	(Mark Watney)-PER
OUT	[(Mark Watney)-PER, visited]	[]	[Mars]	
SHIFT	[(Mark Watney)-PER, visited]	[Mars]	[]	
REDUCE(LOC)	[(Mark Watney)-PER, visited, (Mars)-LOC]	[]	[]	(Mars)-LOC

lample et al. (2016)

Figure 6: Example of Transition Based sequence applied on NER.

As we can see in Figure 6, this approach is based on transitions between different states in order to correctly classify the target. The general idea about this system is that, given some input token, the model has a set of available actions it can take in order to classify each token, transiting into the possible state configurations. In the end, the real challenge lies in predicting the actions or transitions to be made to correctly predict the token's label. To address this challenge, spaCy presents a Deep Learning framework spaCy (2017).

2.6.2 *Deep Learning framework for NLP*

This framework consists of using a statistical model based on Neural Networks to predict the actions to be taken. To apply this principle to natural text, first, we have to process the input tokens calculating representations for the words in the vocabulary. Then, to retain the words' context, it is necessary to contextualize every token in the sentence, which in practice means that we have to recalculate the word's numeric representation based on the sentence it belongs to. After that, the model comes up with a summary vector representing all the information needed to help

predict the word's label. With that vector, the model is able to predict the best action to be taken in order to transit to the next state.

In order to simplify this deep learning framework, it can be split into four different steps, Embed, Encode, Attend and Predict.

Embed

The first task of this approach is Embed. This task consists of calculating word embeddings using a word identifier in order to generate vectors for each word in the document.

The Embed task generates numeric representations of each token of the document, the word embeddings. In practice, in this stage, multidimensional vectors for each token are generated to create a numeric vocabulary that the model can reason about.

Figure 7: Embed process.

In fact, the objective of this stage is to generate different representations for words with different semantic meanings through multidimensional vectors. These vectors allow using a vector distribution so that words that refer to the same entity will have similar distribution values. This mechanism provides the model with tools that allow it to have a richer understanding of the vocabulary. For example, given the word `"student"` in the school context, other words such as `"study"`, `"book"` or `"class"` are also likely to be found in the same sequence, because they are all related to the school context. It is important that the information about the words' proximity is not lost when we create our numeric representations in order to teach the model that those words have some similarity between them. With this embedding mechanism, words like `"student"`, `"pupil"` and `"finalist"` will always be very similar in distribution.

This mechanism makes the model less limited to the annotated text, which translates into a greater capacity for learning.

Encode

In the Embedding stage, we created numeric representations for individual tokens, however, the meaning of a word is not always the same and may vary depending on

the context where it is inserted. In this way, the Encode task aims to recalculate the token embeddings, creating context-sensitive vectors.

Figure 8: Encode process.

In order to create context-dependent vectors, we have to recalculate them, taking into account the sentence where they are inserted. A frequent approach to calculate contextualized word vectors in the sentence is the use of RNNs Lample et al. (2016) which uses the whole sentence. However, spaCy's developers believe that using the whole sentence to get a word's context is not the best way to do it. According to them, calculating the word embedding based on the whole sequence will cause the model to have difficulties knowing if a specific context should be associated with the corresponding token. In some cases, this can result in data over-fitting, making the model sensitive to things it should not. Thus, spaCy approaches this problematic with CNNs, i.e., with a fixed window of N words for each side of the token, with the belief that in the vast majority of scenarios, a small window of words is all it takes to accurately represent the token's context.

In addition, with this type of Neural Networks it is possible to create a decaying effect, to define the level of importance that a given context has on the vector of a word, which is not possible with the previously mentioned method. This makes this method more dynamic and versatile in associating the context with the words, according to spaCy developers.

Furthermore, by using CNNs, one can take advantage of parallelism, which is something that is becoming more and more relevant with the arising of the GPUs computational power, something that is not possible with RNNs architectures due to their sequential nature.

Attend

Now that all the word vectors are contextualized, they can be used to help with the prediction task. The challenge now is to know what information should be taken

into account to predict the label of a certain word token. For that, spaCy introduces the Attend phase.

Figure 9: Attend process.

This phase consists of selecting all the necessary information in order to correctly classify our target. Given a query vector, the model must come up with relevant data to associate that token to its correspondent entity type.

In order to do so, spaCy utilizes a feature mechanism that dictates the way the output vectors are generated. These features are implemented to help the model find all the vectors that the query vector attends to. With the defined features, the model should collect all the data needed to classify a certain token. In order to do so, by default, spaCy take into account the first token in the buffer, the words that are immediately to the left and right of that word and the last entity previously classified by the model. These features can be defined arbitrarily, ensuring the dynamism and versatility of the model. This allows the experimenter to define new features depending on his own context to fine-tune the model for his domain.

Predict

The last step of this framework consists of the actual prediction of the action that the model should take, given the summary vector generated before.

Figure 10: Predict process.

After all the words are turned into vectors (Embbed), the vectors are contextualized within the sequences (Encode) and the feature defined are taken into account, generating the summary vector (Attend), the system is ready to make the prediction.

In fact, the predict step consists of taking the resulting vector from the Attend stage and passing it into a simple multi-layer perceptron, which returns the actions probabilities. Then the model validates all possible actions and chooses one according to the algorithm's confidence to correctly classify each token of the document.

Finally, this process is iterated through a cycle until the document is finished. It is important to emphasize that all stages of this framework are pre-computed, i.e., they occur outside the cycle, so when the model iterates through the document, fewer computations are needed.

2.7 TensorFlow BI-LSTM-CRF

Lately, Deep Learning has been the most used approach to respond to this NLP task. Neural Networks have demonstrated several advances in the NLP field, achieving state-of-the-art results in NER, surpassing previous architectures. In this way, we used Google's library TensorFlow to create an ML model and compare it to the previous tools.

Tensorflow presents several ML features that allow the creation of various Neural Networks architectures. However, this tool is not only applied to NLP tasks, but to any field where the use of Deep Learning is intended. Thus, training a NER model with Tensorflow requires greater knowledge of the entire process involved, from converting vocabulary into word embeddings to training the neural network capable of solving sequence tagging problems.

2.7.1 *Recurrent Neural Network*

One of the first approaches associated with Deep Learning in NLP was to use Recurrent Neural Networks (RNN) (Graves et al., 2013).

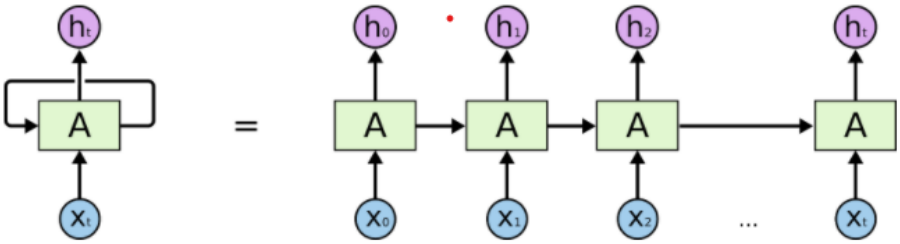


Figure 11: Recurrent Neural Network.

In Deep Learning, the first option that comes to mind when we want to process sequential tasks is the use of RNNs. In NER, the research community started to use these Neural Networks as they revealed good results compared to the existing solutions, making them the standard algorithm for this task. Despite that, RNNs are also famous for some inconveniences, like the vanishing gradients making long term dependencies difficult to deal with. Another problem is that, in order to correctly classify a token's label, one must consider the word's neighbourhood before and after it. RNNs are unidirectional, which means they can only rely on the prior context of the word, assuming that the model reads the document in linear order. In order to solve some of these problems, new features were added to this algorithm.

2.7.2 *Long Short Term Memory*

We can think of a Long Short Term Memory (LSMT) as an RNN capable of preserving Long Term Dependencies. In fact, in order to solve the RNN's memory problem, a memory component was added to it.

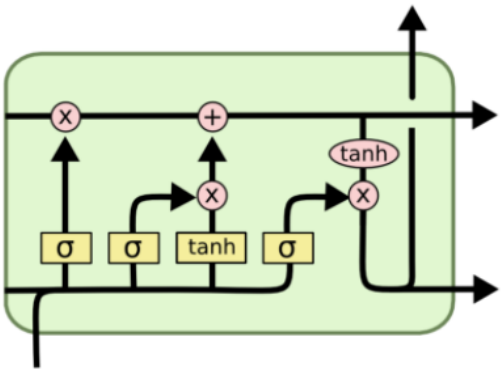


Figure 12: Long Short Term Memory memory cell.

This new memory cell keeps a state that is updated across the Neural Network chain by using input, output and forget gates (Olah, 2015). In short, these gates are responsible for regulating the data that must be updated at each time-step, the information that must be passed to the next cell and the information that must be forgotten, respectively. Using this mechanism, as we keep updating the memory cell, a notion of context is being created, enforcing the memory capability of an RNN, thus making it more capable of dealing with the Long Term Dependencies problem.

With this solution, the memory problem was attenuated, however the model can still only read the document in one direction, which means that the generated context only refers to the prior words.

2.7.3 *Bidirectional Long Short Term Memory*

To make the model consider not only the context before the token but also the context after it, a new architecture was used, a BI-LSTM.

This new approach consists of processing the document in both directions using two different LSTMs, one for the previous context of the token and another for the subsequent. As we are using LSTMs, this mechanism is capable of keeping long term dependencies which means that the resulting vector will now have information of the whole sentence. In this way, this approach allows the generation of more robust numeric representations of each token.

Although this approach reveals better results than the use of a simple RNN, it is important to emphasize that a BI-LSTM has a much higher complexity, making the model more difficult to train, needing more time and computational resources.

2.7.4 *BI-LSTM-CRF*

BI-LSTM came to solve several problems of RNNs, obtaining state-of-art results in several NLP tasks, however, its evolution did not stop there. In fact, in 2015, a new article was released by Huang et al.Huang et al. (2015), introducing a new architecture, BI-LSTM-CRF.

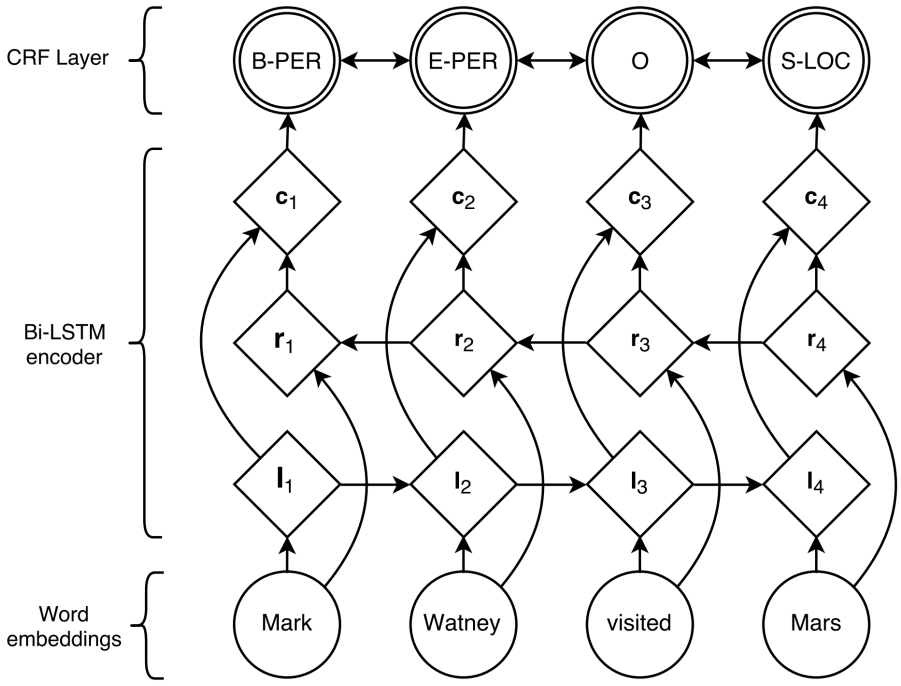


Figure 13: Bidirectional Long Short Term Memory Conditional Random Field.

This new model consists of adding a Conditional Random Field(CRF) to a BI-LSTM, which enables the model to use sentence-level tag information to help correctly classify the token. Like a BI-LSTM uses past and future features in the prediction task, the CRF layer uses past and future tags to predict the token tag efficiently. In other words, this new component receives the LSTM outputs and is in charge of decoding the best tagging sequence, boosting the tagging accuracy (Ma and Hovy, 2016).

The use of this architecture has already proved itself, demonstrating state-of-art results in several NLP tasks. In this work, we used the TensorFlow library to implement this algorithm in order to perform NER in archival finding aids.

2.8 TRANSFORMERS

For several years, LSTMs have achieved state-of-the-art results in most tag sequence modelling, making them the most used algorithm for a wide variety of NLP tasks. In fact, this deep learning technique has established itself in language modelling, making Machine Learning models finally good at remembering things. However, in December 2017, Google published a new article that would be the very beginning of a new era of NLP, "Attention Is All You Need" (Vaswani et al., 2017) introducing a new architecture, Transformers.

As we saw in the past, RNNs worked well for NLP, however, due to their sequential nature, they prevented the use of parallelism efficiently, which severely limits today's computational power. In fact, the model could only reason about a word when all the previous words were already processed. In addition, when present with long sequences, RNNs presented the Vanishing gradient problem, which was mitigated with the upgrade to LSTMs, however, this practice considerably increased the complexity of the problem, acting as a fix and not as a definitive solution.

Thus, Transformers were introduced as a better version of this approach, solving these problems by using the computational power of GPUs, processing all the tokens of the sentence in parallel, reducing the sequential computation and creating possibilities for creating more capable models. Over the last 4 years, we have witnessed an NLP revolution that has completely changed the way we create and use NLP models. We can say that this big change was due to two key factors:

- Enabling Transfer Learning in NLP models, pre-training them with unsupervised learning.
- Drastic efficiency increase in the processing of NLP models, thanks to the Attention mechanism that allows processing large amounts of text in parallel.

2.8.1 *Transfer Learning*

As the name implies, transfer learning consists of storing knowledge obtained during the resolution of a problem so that this knowledge can be reused in new problems. In computer vision, this technique has been widely used over the last decade, allowing the reuse of pre-trained models on large datasets, such as ImageNET (Russakovsky et al., 2015). This makes it possible to train robust models with less effort because, with less training data, it is possible to generate results similar or

even superior to models trained from scratch that need a higher amount of training data.

From 2013 to mid-2018, the most common method of training an NLP model was to use pre-trained word embeddings generated through algorithms such as word2vec (Mikolov et al., 2013) or Glove (Pennington et al., 2014). This method greatly impacted the NLP community, allowing ML models to create relationships between words and calculate their similarity. Despite that, these word embeddings were used to initialize only the first layer of the neural network, being necessary to train all remaining layers with training data associated with the intended task.

This method made it difficult to use transfer learning, as it is only possible to apply the knowledge previously generated in the first layer of the Neural Network. Sebastian Ruder introduces this problem by calling these word embeddings "shallow representations that trade expressivity by efficiency", which allows the model to learn only low-level features about the vocabulary (Ruder, 2018). With this method, most of the token semantic meaning and contextualization was lost, thus limiting the performance of the models.

Recent approaches introduced new methods for transferring the model's knowledge by creating fully pre-trained models. Thus, the knowledge of the model can be fully reused for different tasks, keeping both low and high-level nuances of the vocabulary, thus enabling transfer learning in NLP.

Over time, the scientific community continued to explore possibilities with LSTMs, tweaking them and adding components to them, creating models like LSTM-CNN, LSTM-CNN-CRF, etc. At the same time, unsupervised learning approaches were also coming into the scene (Dai and Le, 2015).

In 2018 a new article was released Radford et al. (2017) where an unsupervised trained LSTM was able to perform sentiment analyses. This model, trained in amazon reviews, managed to achieve state-of-the-art results in this task, drawing attention to this learning method. We can say that this was a big step in terms of applying unsupervised learning techniques on sentence classification, however it was still focused on only one specific task.

Afterwards, another research that has greatly impacted this field came out, Peters et al. (2018). Here the authors introduced Embeddings from Language Models (ELMo), presenting a feature-based approach that generates deep contextualized word vectors where the words' representations are functions of all the internal layers of the model. By taking into account not only the word's syntax and semantics, but also their context meaning, the same token would have different representations

when used in different contexts. This time, the LSTM was pre-trained on a large corpus, making it possible to achieve good performances on several NLP tasks.

Then, in Howard and Ruder (2018) the authors presented ULMFiT resulting in one of the first successful applications of transfer learning on NLP, where they pre-trained a model with a huge dataset containing more than 100 million words (Merity et al., 2017). By fine-tuning this pre-trained model, it was reused for different NLP tasks enabling generalization with only 100 labeled examples. Due to this transfer learning property, a pre-trained model fine-tuned with 10 times less annotated data could finally perform as good as a model trained from scracth.

2.8.2 *Attention Mechanism*

The new transformer architecture has revolutionized token processing. In fact, with this architecture, it is possible to make use of parallelism processing all tokens simultaneously, improving the efficiency of this process drastically. But how does this mechanism work? What is the difference that makes this solution superior to previous architectures? To answer these questions, let us address the Self Attention mechanism applied to natural text processing.

This mechanism aims to create new representations from word embeddings, capable of expressing a greater notion of context. In other words, for each sentence, it is intended to associate a degree of relevance between a certain token and the remaining tokens of the phrase. For this, a methodology that makes use of queries (Q), keys (K) and values (V) vectors was introduced.

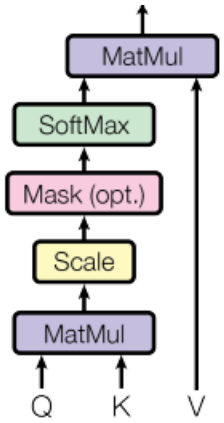


Figure 14: Attention Mechanism, taken from Vaswani et al. (2017).

In practice, for each Query vector, we perform a dot product between this vector and all the key vectors, thus generating the attention scores, which represent how much each token attends to the other tokens. Then, the softmax function is used to normalize these values between 0 and 1, creating a probabilistic distribution, where the high values are heightened and the low values are depressed in order to create weights with a stronger opinion. Finally, in order to calculate the final output, another dot product is performed between the previously calculated weights and the Values vectors (Alammar, 2018). This final output consists of an improved version of the initial word embeddings since this new representation has a notion of contextualization of each token in the sentence.

In Figure 15 taken from Bahdanau et al. (2015), we can observe the attention scores of a translation task, where it can be seen which words are attending to what.

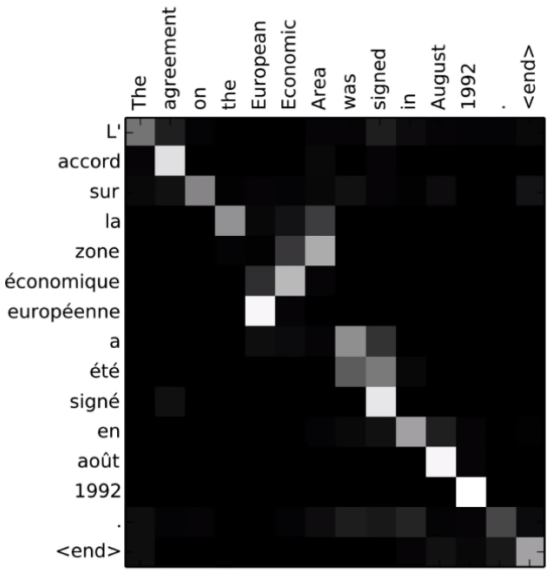


Figure 15: Attention Scores.

As can be seen in this example, the order of the words in the English-translated sentence is different from the sentence translated into the French language. We can verify that the attention scores correctly represent this change in the order of the words.

In order to keep track of the words' position in the sentence, positional information is added to word embedding before this Self-Attention process. This makes it possible to process all sentence tokens in parallel, which changes everything.

In Transformers, the attention mechanism is performed in the Multi-Head-Attention layer. This layer consists of using 8 different heads, i.e., 8 self-attention mechanisms with 8 different Query, Key and Value matrices. In this way, the model can learn 8 different semantic meanings from attention, such as grammar, vocabulary, etc. Once again, all this can be done in parallel. Currently, with the computational power of parallelism provided by technological advances in GPUs, calculations such as the matrix dot product are considered a cheap operation, making this entire mechanism extremely efficient. This allows training models in much larger amounts of text than previous architectures.

Complexity Analyses

Figure 16, taken from Vaswani et al. (2017) allows us to compare the self-attention mechanism complexity with the architectures used previously. Here, n corresponds to the sequence length, d to the representation dimension, k is the kernel size of convolutions.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Vaswani et al, 2017

Figure 16: Self-Attention Complexity compared with RNN and CNN.

First, we have the complexity per layer. Self-Attention obtains better numbers when the dimension of the representations (d) is smaller than the sequence length (n), which is usually the case. Then there are the sequential operations, which has always been one of the biggest problems of RNNs. However, Self-Attention has a constant value when it comes to the amount of minimal sequential operations required, thus drastically increasing its efficiency. Finally, we have the Maximum Path length, which is intended to measure the performance of the algorithms in the management of Long Term Dependencies. As seen before, the dependencies between words significantly impact several NLP tasks, such as NER, so the models must efficiently memorize the dependencies between the words. Once again, we have that Self-Attention is the algorithm that presents the best results in this aspect, with a constant maximum size.

2.8.3 *Generative Pre-Training*

With the advances in the application of transfer learning in NLP models and the introduction of transformers, it was a matter of time until a model capable of using the best of both worlds was created. In June 2018, Radford et al. (2018) released the first version of the Generative Pre-Training (GPT). This model presents a semi-supervised approach which consists of using an unsupervised component for the model pre-training and manual text annotations for the fine-tuning. This work extended ULMfit and ELMo, with the primary objective of creating a general model that could transfer its knowledge to a wide range of NLP tasks with little fine-tuning.

In order to do so, GPT was created based on the Transformer decoder, which uses a masked-self-attention mechanism. It differs from a normal self-attention in how the attention scores are calculated. This approach uses a mask to hide the future context of the token, so the attention scores are calculated based only on the past context. In practice, the model is pre-trained by predicting the following word in the sentence in an auto-regressive way. For example, given the sentence `"Joseph likes to run"` the model starts with the token `Joseph` while all the other tokens are masked. Then, on the second step, it tries to predict the next word but can only attend to words that have already been generated, in the case, only the token `"Joseph"`. In the third step, the attention scores are calculated based on both tokens `"Joseph"` and `"likes"` and son on.

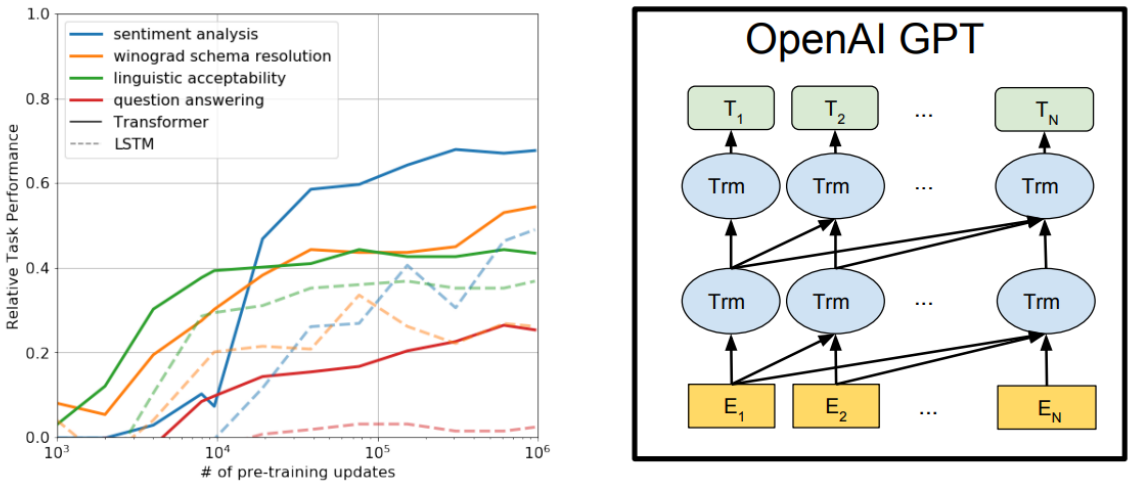


Figure 17: GPT NLP tasks results. GPT left-to-right Transformer.

As we can see in Figure 17 (Right), the decoder cannot access the information of both sides of the token, making this model unidirectional. This makes it suitable for text generation tasks, however, for Named Entity Recognition, it is essential to consider both future and past contexts so we can say that this model is not optimal for our task.

Despite this, this GPT model obtained state-of-the-art results in several NLP tasks, proving that by combining unsupervised learning for pre-training and supervised learning for fine-tuning, we could create models that can perform a wide range of tasks with "little adaptation".

In Figure 17 (Left), we can observe the performance in several tasks of two pre-trained models, LSTM and Transformer. According to Radford et al. (2018), due to the Self-Attention mechanism, transforms manage to retain Long Term Dependencies more effectively than an LSTM, which makes the pre-trained model obtain a more comprehensive knowledge of the language. Thus, the information transferred from the model's pre-training has higher quality, having a more significant impact on the tasks where it is applied. Therefore, the model pre-trained with Transformer outperformed the LSTM model.

2.8.4 *Bidirectional Encoder Representations from Transformers*

Right after the launch of the GPT model, a new Transformers based model also came out, Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019), however, this model was based on the transformer encoder architecture.

An encoder's main purpose is to create numerical representations that contain the meaning of the words, contextualized within the sentence, using the self-attention mechanism. The main difference between the encoder and decoder is that, while the decoder is unidirectional, the encoder is bidirectional (Figure 18 (Left)). As we saw in the GPT model, in the pre-training, the model could only attend to the words on the left to predict the next word in the sentence due to its auto-regressive property. An encoder uses a different method, Masked Language Modeling (MLM), which masks a random word in the sentence while letting the model predict the missing word. By training the model to predict hidden words, it can use both the left and the right context to make that prediction, thus generating a bidirectional model. In this way, the generated representations are affected by all the words in the sentence, creating meaningful representations of the whole sequence.

As stated in Devlin et al. (2019) a model that is restricted to attend only to the previous words (left to right) can achieve great performance in sentence-level NLP

tasks, however, for token level tasks (Figure 18 (Right)), it is important to attend to both directions. For example, in the token classification task, the context after the token can be crucial to correctly classify it.

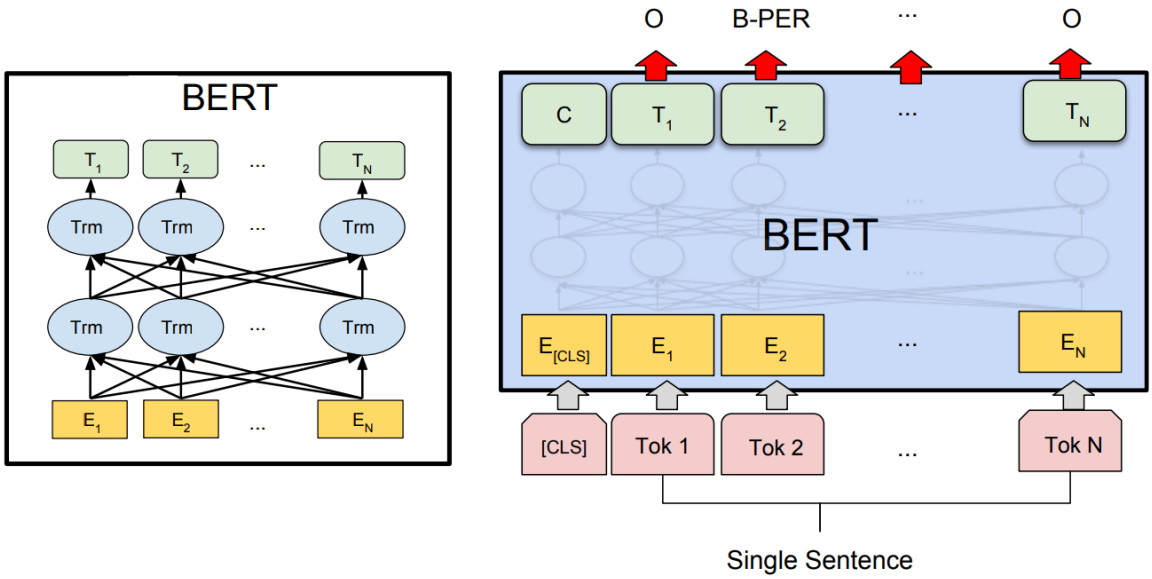


Figure 18: BERT bidirectionally applied to token level task.

BERT model was pre-trained with MLM on the Book Corpus (Zhu et al., 2015) and English Wikipedia making a total of 3300M words. It is a bidirectional model that needs low effort for fine-tuning, which outperformed many task-specific architectures, obtaining state-of-art results in 11 NLP tasks, both token and sentence level.

Subword Tokenize

For an ML statistical model to be able to interpret natural text, it is necessary to transform the text into numerical representations capable of transmitting the meaning of that text to the machine. These representations are the starting point for the ML models, so the higher the amount of information contained in these numerical representations, the better their interpretation will be, conditioning the performance of the entire ML process.

One of the techniques for creating these representations is tokenization. In NLP, this technique has been studied and widely used in several areas of token processing, such as NER. In this case, this technique usually uses a word-based tokenizer, i.e., defining a fixed size N for the vocabulary and then associating an id for the N most frequent words of that vocabulary. This method has shown good results in several

contexts, however, it has several limitations. Due to the fact that the number of words is limited, ML models have difficulties dealing with out of vocabulary words or even words that are rarely used. One solution for this problem is to increase the number of vocabulary words (N), however, this would lead to other problems such as making the computational model heavier, increasing the number of rare words.

On the other hand, as each distinct word has a different id, similar words have entirely different meanings, which causes information about the relationship of the words to be lost during this phase, decreasing the performance of the models.

Thus, in order to solve these limitations and increase the meaning of the numerical representations, models like BERT use a different technique, sub-word tokenization.

This method aims to decompose rare words into sub-words, keeping the most frequent words intact. In fact, observing a given vocabulary, words like `"tokenization"` can be decomposed into two sub-words `"token"` and `"ization"`. Using a word-based tokenizer, the words `"token"` and `"tokenization"` would have representations with entirely different meanings, however, with a sub-word tokenizer, by splitting the word `"tokenization"` into two sub-words, the model can learn that this word is composed of sub-words that it already knows. In this case, the model associates the word tokenization to the most frequent word `"token"` thus answering the problem of out-of-vocabulary words and maintaining a vocabulary with a reasonable size (HuggingFace, 2021).

Models that use this method have increased their accuracy, mainly in the classification of unknown words (Sennrich et al., 2016).

2.9 EVALUATION

When one wants to evaluate the quality of a NER model, it is important to understand which metrics should be considered, as the words that represent the named entities correspond to a small minority of the words in a document.

The process of NER can be seen as a problem of binary classification, considering that a given token belongs or not to a certain type of entity. In this way, a token can take a negative value if it does not correspond to a named entity or positive value otherwise. Normally, most of the tokens in a given document correspond to negative tokens. That said, a confusion matrix can be used, which allows visualization of the performance of a ML algorithm.

	Predicted Negative	Predicted Positive
Actual Negative	True Negative	False Positive
Actual Positive	False Negative	True Positive

Table 2: Confusion Matrix.

Observing Table 2, each row represents the actual instances of a certain named entities and the columns represent the instances of the predicted ones. Calculating metrics for NER based on the total number of tokens can be misleading. Suppose that one intends to use accuracy to validate a sentence with ten tokens, in which only one of them corresponds to a named entity that the model did not detect. Table 3 represents this scenario.

	Predicted Negative	Predicted Positive
Actual Negative	9	0
Actual Positive	1	0

Table 3: Concrete example of the confusion Matrix.

The accuracy metric consists of dividing the sum of True Positives and True Negatives by the total number of tokens.

$$Accuracy = \frac{TruePositive + TrueNegative}{Total}$$

Thus, the model obtained 90% accuracy, even though no entity was recognized correctly. In fact, the objective of NER is to find named entities, so being able to correctly classify tokens that are not entities is useless in this context, despite obtaining high accuracy. That said, the metric accuracy does not satisfy the needs of this area of NLP.

So, in order to solve this problem, two different metrics are introduced, precision and recall. Unlike accuracy, these metrics measure the quality and quantity of entities found by the model.

Precision represents the relation between True Positive and the sum of True Positive and False Positive. This represents the percentage of entities that the model correctly recognized. So to achieve high precision values, the model must be meticulous in its decision making, marking only tokens as entities when it has a high degree of certainty. This metric benefits models that implement a careful selective approach and undermines models that return to many False Positive results.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

Finally, recall is a metric used to measure the relation of the number of entities found and the number of entities that should have been found. To obtain high recall values, the model must consider uncertain entities to increase the probability of finding the largest possible number of correct entities.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

It is easy to understand that there are specific criteria that allow cheating this validation system to obtain high values for each of these metrics in separate. Suppose the model is less flexible and only classifies entities which it is sure about their class. In that case, it tends to have high precision, however, a low recall as it is likely that, in order to reduce the risk of classifying wrong entities, there are many False Negative tokens. The opposite also happens, by using a very flexible system, we get a high recall value and low precision.

One way to solve this problem is to create a combination of both metrics, which gives rise to a new metric, the F1-score.

$$F1 = 2\,\frac{Precision \times Recall}{Precision + Recall}$$

This metric is based on a weighted harmonic mean between precision and recall, representing a balance between both. This means that, for a model to have a good F1-score, also known as F-measure, it has to obtain high values in both metrics.

Finally, it is important to note that the F-measure alone is insufficient to validate a statistical model. All these metrics complement each other and should always be used together to judge the model's performance (Derczynski, 2016).

3

ARCHIVAL FINDING AIDS PROCESSING

To train and validate new models capable of recognizing entities in the archival domain, creating annotated data with the respective named entity labels is necessary.

In fact, for our ML models to be able to learn how to correctly identify the intended named entities, they need to be trained with numerous data samples. The more representative the examples used, the greater the scope and generalization of the models created. Therefore, training data must be generated from archival context datasets. As we did not find any archival finding aids datasets with already annotated entities, it was necessary to build our own annotated archival corpora.

In this chapter, we will cover all the processing of this data, from its harvesting from the archival repositories, data cleaning, annotation and finally, the data parse in order to generate the annotated corpora that will be used throughout this work.

3.1 DATA HARVEST

The first stage of data treatment was data harvesting. This process consisted of accessing the online repositories of the archives and extracting the intended archival finding aids.

For this, the OAI-PMH protocol was used. This protocol allows access to the requested data by using verbs injected in the query string. For example, to extract archival finding aids from an archival repository, the following verb can be used: `"GetRecord&metadataPrefix=ead&identifier=oai:PT/ABM/:807"`. This verb allows us to harvest the archival finding aids of the archival fond with reference code `"PT/ABM/:807"` which in this case corresponds to a parish named *Paróquia do Curral das Freiras*. This specific archival descriptions are kept in the repository of the *Arquivo Regional e Biblioteca Pública da Madeira*. The repositories return an answer in XML format, containing the metadata regarding the identified fond. An example of the *Paróquia do Curral das Freiras* archival finding aids can be seen in Listing A.2

In addition, we also used an online repository[1] that congregates a set of archival finding aids from several Portuguese archives and serves them via OAI-PMH. This method was repeated to obtain the archival finding aids used in this dissertation.

3.2 DATA DESCRIPTION

The data used to test the algorithms referred in this dissertation correspond to datasets from two national archives, the *Arquivo Distrital de Braga*[2] and the *Arquivo Regional e Biblioteca Pública da Madeira*[3].

Firstly, there is a dataset of a fond that shows a pioneering period in computing history between 1959 and 1998. This fond (`PT/UM-ADB/ASS/IFIP`), produced by the International Federation for Information Processing (IFIP), contains a section corresponding to the Technical Committee 2, which has a subsection corresponding to Working Group 2.1. This subsection is composed of several series where different archival descriptions are organized, for example, correspondences, meeting Dossiers, news from newspapers, etc.

Secondly, there are two datasets corresponding to a series (`PT/UM-ADB/DIO/MAB/006`) from the archival fond *Mitra Arquiepiscopal de Braga*, which contains genre inquiries. The archival descriptions in this series contain witnesses' inquiries to prove applicants' affiliation, reputation, good name or "blood purity". One of the datasets has a very standardized structure, while the other contains many natural text elements.

Thirdly, there is a historical dataset corresponding to the fond (`PT/UM-ADB/FAM/ACA`) of the *Arquivo da Casa do Avelar (ACA)*, which depicts the family history of *Jácome de Vasconcelos*, knight and servant of King *D. João I*. This family settled in Braga around the years 1396 and 1398 with a total of 19 generational lines, up to the present time. This fond is composed of subfonds and subsubfonds that contain records associated with members of this family with a patrimonial, genealogical and personal domain.

Fourth, there is the dataset of the *Familia Araújo de Azevedo* fond (FAA), also known as *Arquivo do Conde da Barca*. This archive, produced from the year 1489 to the year 1879 by Araújo de Azevedo's family, who settled in Ponte da Barca and Arcos de Valdevez (district of Viana do Castelo) at the end of the 14th century, contains records predominantly associated with foreign policy and diplomacy across borders. This fond is composed of several subfonds composed of archival descriptions with

1 http://portal.arquivos.pt:3000/
2 http://pesquisa.adb.uminho.pt
3 https://arquivo-abm.madeira.gov.pt/

information from members of the FAA family, such as requirements, letters, royal ordinances, etc.

Fifth, there is a dataset that characterizes the streets of Braga in the year 1750. This corpus contains elements that characterize the history, architecture and urbanism of each artery in the city, which help understand the main lines of its evolution.

Finally, two datasets from the *Arquivo Regional e Biblioteca Pública da Madeira* were used which correspond to two archival fonds, more precisely to *Paróquia do Jardim do Mar* and *Paróquia do Curral das Freiras*, both parishes from Madeira archipelago. These fonds consist of three series each, representing registrations of weddings, baptisms, and deaths. Each series consists of files that correspond to the year of each record, and finally, each file has a set of pieces with archival descriptions.

3.3 DATA CLEANING

As we can see in Listing A.2 and in Listing A.1, the XML file retrieved from the archive repositories contains quite a few elements, describing in detail various properties of the respective archival documents. It does not make sense to recognize entities in text attributes that are already completely discriminated, so at this stage, we selected the elements of the XML tree that contain unstructured text. For example, fields like `scopecontent` and `unittitle` contain text descriptions, where it is possible to extract entities of interest.

After selecting the intended fields, their content was extracted using regular expressions, XSLT scripts and python scripts. In the end, plain text datasets with loose sentences were obtained.

3.4 DATA ANNOTATION

Data annotation consists of adding label tags to tokens in a dataset. In the data cleaning phase, archival datasets with natural text were generated. In this phase, the intention was to generate the ML algorithms training data by identifying and classifying entities of the following type: Names of People, Organizations, Places, Dates and Professions.

In Ingersoll et al. (2013) two distinct methods for annotating documents were identified. The first method consists of annotating only one type of entity in each dataset, creating several versions of the same corpus, annotating only one type of entity in each one. The second approach is to annotate all entity types in the same document at once.

The author argues that the first approach has several strengths, such as the ability to identify nested entities and greater flexibility in choosing which model we want to use. In fact, using the first method, the generated annotations may contain tokens with more than one tag associated with them due to the existence of nested entities. This allows datasets to retain a greater meaning from these entities. On the other hand, annotating only one type of entity in each document makes it possible to train models to identify only that type of entity, which increases its flexibility. A model with unnecessary information can lead to inefficient use of computational resources.

Despite this, this method introduces some drawbacks. Training a model with more than one entity makes it learn to disambiguate entities from each other. For example, given the sequence `"José Pacheco LDA"`, we are dealing with a named entity Organization, however, a model that does not know Organizations will be tempted to annotate this entity as Person `"José Pacheco"`. Providing more information to the model makes it learn more, thus achieving better results. Furthermore, one quickly realizes that, in a normal case where we want to extract all kinds of entities, training several models instead of one is much more time-consuming and computationally expensive. Looking at newer architectures, even fine-tuning a model can take hours, so repeating this process would drastically increase energy and time consumption. In addition, by using a single entity model to classify multiple entity labels, we have to perform an entity merge at the end of the entity recognition, increasing the complexity of the problem.

That said, we verify that reference datasets such as CoNLL-2003 (Sang and Meulder, 2003) or wikigold (Balasuriya et al., 2009) contain more than one annotated entity. In this way, we decided to annotate all entities in the same document.

With this problem solved, we now move on to the actual annotation process. Due to the size of the datasets, the annotation process is usually expensive and time-consuming so, in order to speed up this process, different annotation methods were used.

3.4.1 *Statistical model Approach*

The first approach consists of using an ML model trained in generic Portuguese texts to identify entities in unlabeled text. However, since this model has not been trained in the archival context, it cannot recognize all the intended entities. A problem associated with this method is that there are usually many false positives, which can generate more work for the experimenter. However, the idea is that, as annotated texts are generated, this text is used to retrain the ML model, increasing

its efficiency and autonomy at each iteration. It is important to note that this method acts as a support mechanism for the annotation process, and it always needs validation by the experimenter.

3.4.2 *Regex Approach*

Another approach used when annotating datasets was the use of regular expressions. This method consists of creating filters and conditions that intercept certain patterns in the text. Despite allowing the extraction of many entities quickly, this method is not very flexible, and it is not easy to disambiguate entities. For example, a person's name can be contained in an entity of the type Organization or Location, which is difficult to distinguish with this solution. However, some of the used archival finding aids contain some patterns in their structure, for example, records of baptisms, marriages, or, in the simplest case, date-type entities. Thus, although this method could not identify all document entities, it was used to support the annotation process, speeding up the annotation of identified patterns and most of the dates present in the documents.

3.4.3 *Manual Approach*

Effectively, if the previous methods managed to recognize all entities in the texts, there was no point in creating the final NER model. In the end, everything boils down to the experimenter expertise. The experimenter has to create the right regex rules and feed the ML model with the right annotations to combine all these methods with the last method, manual annotation. This method implies that a person, who is knowledgeable of the domain, manually writes down each token of the corpora. Despite being slow and quite tedious, this method is the one that yields the best results. In order to facilitate this process, a JavaScript application was created that allowed selecting and annotating an entity with a simple key-press. In this thesis, manual annotation was the predominant annotation method due to the variety of archival contexts. However, it was always assisted by previous methods.

3.5 DATA PARSE

After we have all our datasets annotated, these are ready to be used in the ML model's training. In this work, different ML toolkits and ML architectures were used,

which use different input formats. Thus, it is necessary to perform a data parse for each tool.

OpenNLP Format

In order to train an ML statistical model in the Portuguese archival domain, OpenNLP needs annotations that represent the context in which it will be used to recognize entities. Thus it is necessary to create data samples in the following format:

```
<START:Pessoa> Willem van der Poel <END> envia carta ao <START:Profissao> Presidente <END> da <START:Organizacao> IFIP <END> informando que gostaria de ser dispensado deste mandato, mas tem orgulho de fazer parte da criação do primeiro relatório sobre a linguagem algorítmica.
```

Listing 3.1: OpenNLP annotation from the IFIP corpus.

```

O autor, <START:Profissao> Coronel <END> , <START:Profissao> Comandante <END> do regimento de artilharia 3 e Encarregado do <START:Organizacao> Governo <END> interino da Praça de <START:Local> Elvas <END> , implora a protecção de <START:Pessoa> António de Araújo de Azevedo <END> para integrar a futura lista de promoções, visto que ultimamente foi preterido três vezes por oficiais modernos e que até foi recusado pelo <START:Profissao> Marechal <END> <START:Pessoa> Beresford <END> para integrar a expedição ao <START:Local> Rio Grande <END> .
```

Listing 3.2: OpenNLP annotation from the *Família Araújo de Azevedo* corpus.

These annotations correspond to real examples of our document annotations. All annotated entities are marked by tags `<Start:Entity Type>` at the beginning and `<END>` at the end. It is also interesting to note that the annotations have different types of entities. This is something that was not supported by OpenNLP release versions, however, this feature was later added.

Despite this, writing down just two examples is not enough. According to (OpenNLP, 2011), in order to create a NER model that has satisfactory performance with the OpenNLP toolkit, close to 15000 annotations are needed.

spaCy Format

Another NLP toolkit used to find entities in archival finding aids is spaCy, which like OpenNLP, also requires data samples associated with the archival domain. spaCy accepts new data samples in the following format:

```
("- Mónica , filha de João de Sousa e Custódia Costa , da freguesia da Sé .", {"entities":[(2,
    8, "Pessoa"),(20, 33, "Pessoa"),(36, 50, "Pessoa"),(69, 71, "Local")]}),
("Que fez a Confraria de São Lázaro da cidade de Braga com D. Apolónia Maria Ribeiro .", {"
    entities":[(10, 33, "Organizacao"),(47, 52, "Local"),(57, 82, "Pessoa")]}),
("Contém o despacho dado em Lisboa pelo Arcebispo , autenticado com o selo.", {"entities":[(26,
    32, "Local"),(38, 47, "Profissao")]}),
```

Listing 3.3: spaCy annotations from *Arquivo da Casa Avelar* corpus.

In this case, we have an example of annotations from the archival finding aids of *Arquivo da Casa Avelar* archival fond.

Here, a python dictionary format was used to annotate the named entities. This dictionary keeps all the sequence tokens, followed by an array with the entity labels and their delimiting spans, registering the beginning and end of each named entity.

BIO Format

In addition to the NLP toolkits mentioned above, in this thesis, other Deep Learning architectures were created to recognize entities in this domain. To train these architectures, we used the BIO data format (Beginning, Inside or Outside of the named entity), a common tagging format for tagging tokens, not only in Named Entity Recognition but also in several token level NLP tasks.

```
O Ficou            O muito                  O que                    O que
O muito            O com                    O o                      O pode
O contente         O o                      B-Profissao ministro     O segurar
O por              B-Profissao Marechal     B-Pessoa  António        O a
O saber            B-Pessoa  Beresford      I-Pessoa  de             O dignidade
O que              O e                      I-Pessoa  Araújo         O da
O o                O que                    I-Pessoa  de             O nação
O dito             O ambos                  I-Pessoa  Azevedo        O .
O embaixador       O são                    O é
O tem              O da                     O o
O conferenciado    O opinião                O único
```

Listing 3.4: BIO annotation from the *Família Araújo de Azevedo* corpus.

These listings correspond to real annotations taken from our annotated archival corpus, archival finding aids of *Familia Araujo de Azevedo* fond. As we can see, each token is tagged with a label. The label `"B-Entity"` marks the beginning of the named entity, followed by the label `"I-Entity"` tagging the token inside of the named entity. All the out of entity tokens are tagged with the label `"O"`.

CSV format

Another format widely used in this area is CSV. During the research carried out on NER models, several architectures were tested, with datasets annotated in different contexts. Some of these architectures used this format to train their ML models.

```
Sentence #,Word,Tag   ,o,0                Sentence: 97,Registo,0   ,de,I-Pessoa
,Zemanek,I-Pessoa     ,pagamento,0        ,de,0                    ,Félix,I-Pessoa
,envia,0              ,ao,0               ,batismo,0               ,de,I-Pessoa
,carta,0              ,Dr,0               ,n,0                     ,Sousa,I-Pessoa
,a,0                  ,Turski,B-Pessoa    ,º,0                     ,Mãe,0
,J,B-Pessoa                               ,2,0                     ,Francisca,B-Pessoa
,Carter,I-Pessoa                          ,Manuel,B-Pessoa         ,Pereira,I-Pessoa
,concordando,0                            ,Pai,0                   ,da,I-Pessoa
,com,0                                    ,Manuel,B-Pessoa         ,Silva,I-Pessoa
```

Listing 3.5: CSV annotation sample of IFIP and ABM_807 corpora.

Here, the dataset is divided into sentences with the separator `"Sentence #,Word,Tag"` where we keep the sentence number, the first token and its corresponding label. Then, for the rest of the sequence, a label in BIO format is associated with each token.

3.5.1 *Format Converter*

As already seen, each of these approaches has a different annotation format, however, the entities annotated are the same. So in the annotation process, only one format was used to avoid repetition of work. The chosen format was OpenNLP since it was the easiest to manually annotate named entities due to its simple structure. In fact, to annotate an entity in spaCy's format, we would have to register all the entity spans, making this process slower. As for the BIO and CSV format, one would have to tokenize and tag each document token, even the Out of Entity tokens which is not optimal.

In this way, after having all the data annotated in OpenNLP format, we have to convert it to BIO, CSV and spaCy formats. In order to do so, we developed three parsers.

The first one consists of parsing OpenNLP data into spaCy's format by using regular expressions to find the annotations and determine the entity label, token and span. Then, after extracting all the necessary data, we used it to generate a new

dataset in the spaCy data format, containing all the needed information. In this way, we were now ready to use all our corpora for training spaCy NER models.

As for parsing the data into the BIO format, we have to tokenize all the sentences into tokens and then tag all the resulting tokens with a BIO label. The way the words are tokenized can affect the Model results, so, in order to experiment with different approaches, three tokenizers were used, Keras API tokenizer, spaCy Portuguese tokenizer and a simple regex tokenizer. After the tokenization process, we mapped all the entities labels to the new tokenized file. In this way, three versions of the BIO format were created for each dataset. After testing them in the ML models, spaCy's tokenizer showed better results, probably because it is optimized for the Portuguese language.

As for CSV format, since their structure is very similar to the BIO format files, we just had to take the BIO annotated datasets and change their structure a bit to generate the new CSV files.

3.6 RESULTS

At the end of the data processing step, eight annotated datasets with five different entity labels were obtained. Then, the annotated files were parsed into all data formats. Table 4 shows the amount of annotated entities.

In total, the annotated corpora contain 164478 tokens that make up 6302 phrases. As for entities, we annotated more than 28 thousand of entities in total. All the annotated corpus are available to the public in Cunha and Ramalho (2021c).

Although we can annotate a reasonable number of entities to train our models, we found that our datasets are not balanced in terms of the number of entities per type. Figure 19 illustrates the number of entities of each type for each dataset (the *Curral das Freiras* dataset is not included in the plot since it only has entities of the Person type).

In fact, datasets like IFIP, IG1 IG2, ABM_807 have a big difference in the number of entities of each type. The predominant entity type in the archival corpora corresponds to the Person named entity with a total of 17279 annotated names. Following that, we have Place and Date named entities with a total of 6604 and 2980 entities identified, respectively. Then with a much lower number, we have Professions with 978 and Organizations with only 843 annotated named entities.

One of the reasons for this discrepancy is the nature of datasets. In fact, some of these datasets are quite old, so it is normal not to have many organizations at that

Table 4: Number of annotated entities per corpus.

Corpus	Person	Place	Date	Profession or Title	Organization	Total
IFIP	1503	325	100	40	318	2286
Familia Araújo de Azevedo	369	450	118	428	94	1459
Arquivo da Casa Avelar	465	239	141	118	91	1054
Inquirições de Genere 1	2002	3713	121	0	0	5836
Inquirições de Genere 2	692	10	54	0	0	756
Jardim do Mar	2393	574	1762	1	2	4732
Curral das Freiras	8729	0	0	0	0	8729
Ruas de Braga	1126	1293	684	391	338	3832
Total	17279	6604	2980	978	843	28684

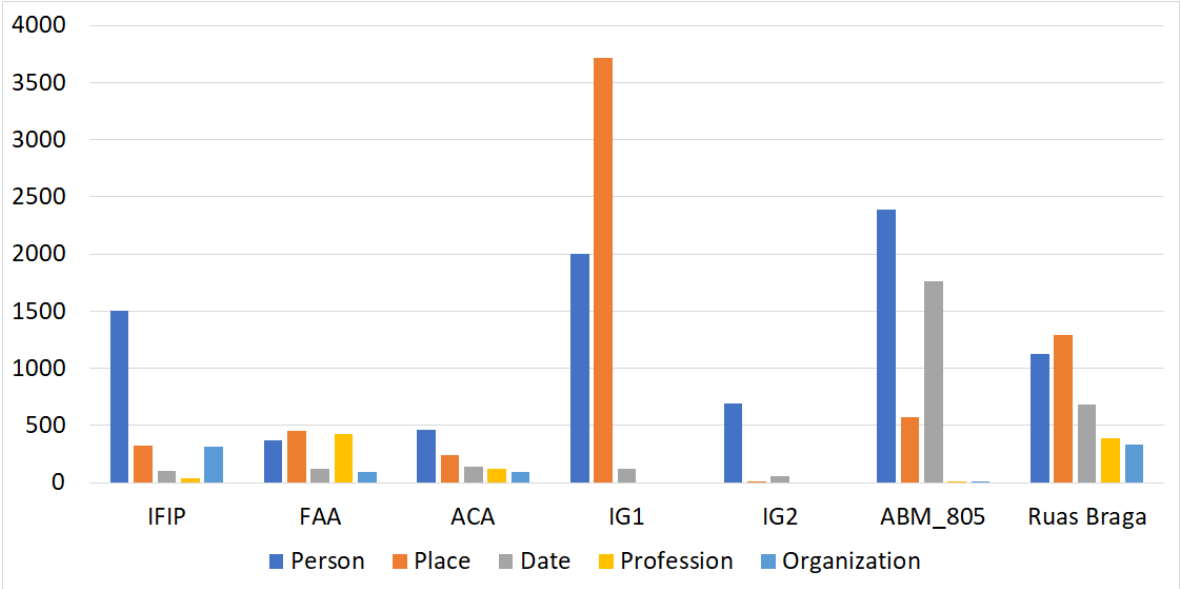


Figure 19: Named entities distributed by corpus and entity type.

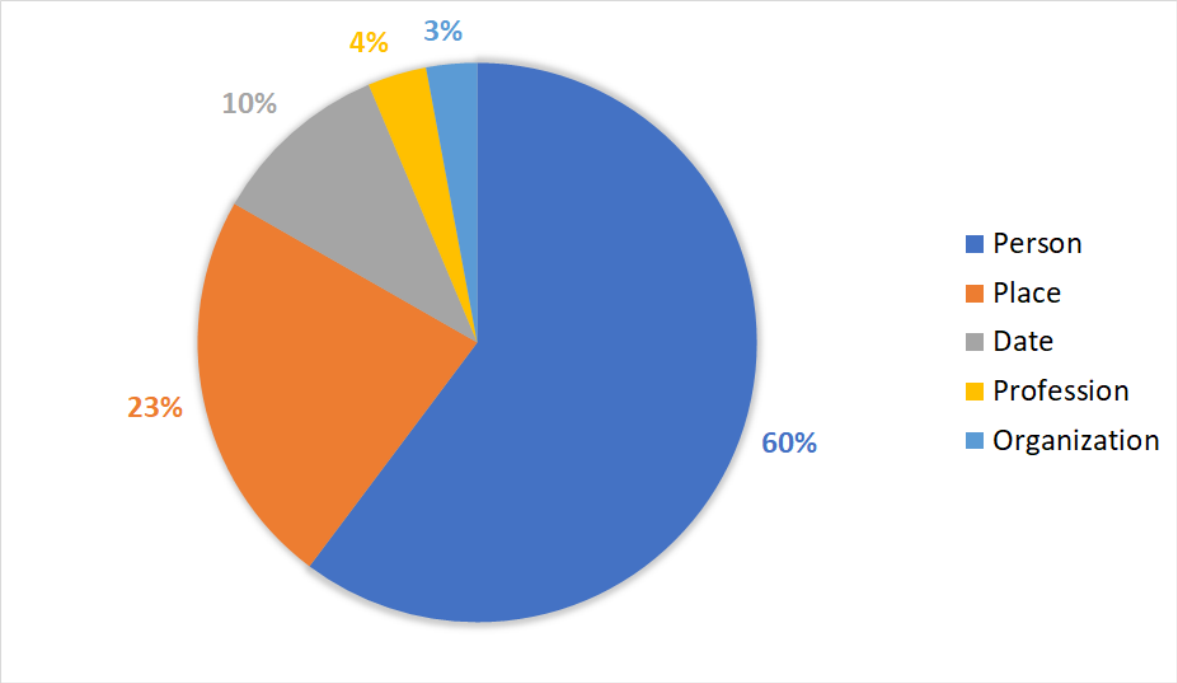


Figure 20: Named entities labels' distribution.

time. On the other hand, parish datasets only have records of people's names, dates and places, causing the discrepancy between entities to increase even more.

Figure 20 allows us to analyze the distribution of the entities types, where we can see that 60% of the identified entities correspond to peoples' names, 23% Places, followed by Dates with 10% and finally Organizations and Professions with a small fraction of 4% and 3% respectively.

This imbalance in the entity types can lead to the model having difficulties identifying entities with a smaller number of samples. The implications of this phenomenon will be analyzed in the following chapters.

3.7 CONCLUSION

In this chapter, we generated our annotated archival corpora composed of 8 different datasets. As we can see in the results Section 3.6, we annotated more than 160 thousand of tokens that contain a total of 28684 named entities. Once annotated, all datasets were parsed into 4 different formats so they could be used as the ML models input. With this data, we can advance to the training of ML algorithms in order to create NER models capable of extracting archival finding aids entities.

To improve the entities distribution balance, it would be interesting to annotate other datasets with a greater number of organizations and professions in order to balance the number of entities by type of entity.

It is important to note that creating annotated corpora like this is an important contribution to the scientific community as they can be reused for purposes besides NER.

4

NAMED ENTITY RECOGNITION MODELS

In this thesis, we intend to extract information from Portuguese archival finding aids through the recognition of the following named entities: *Pessoa* (Person), *Organização* (Organization), *Local* (Place), *Profissões ou títulos* (Professions or titles) and *Data* (Date).

For this, several approaches were tested, where the power of ML is used to process archival texts. The quickest and simplest way to apply NER to Portuguese textual data is to use ML-based toolkits specific to this task, which are ready to be used without the need for complex fine-tuning or high knowledge in the NLP field. Another approach is the creation of the ML model, with ML libraries such as Tensorflow and Pytorch, where it is necessary to generate the entire NER process, from data processing to training and validation of ML models. This chapter will present the approaches used to train various ML models.

4.1 AVAILABLE PORTUGUESE NER MODELS

The first approach consisted of using pre-existing NER models trained in Portuguese textual documents. The models used were trained in two datasets, Second HAREM Golden Collection and SIGARRA.

HAREM is a NER contest applied to Portuguese text. It had two different events which took place in 2005 (Santos et al., 2006) and 2008 (Freitas et al., 2010), resulting into two golden collections. In this dissertation, the second golden collection was used, which consists of a Portuguese annotated corpus that contains about 147991 words, retrieved from 129 different textual documents, with a total of 7836 manually annotated entities (Carvalho et al., 2008). The textual content of this corpus has been collected from several sources such as newspapers, Web pages, emails, expositories and political and fiction contexts (Santos and Cardoso, 2006). As for entity type, ten categories have been annotated, Person, Organisation, Time, Place, Works of art,

Event, Abstraction, Thing, Value and Misc. As we can see, there are no Professions entity types, which can negatively influence the results of the classifications.

The other corpus, SIGARRA (Pires, 2017), corresponds to the information system of the University of Porto data. This corpus was created to assist the entity-oriented search engine of the University of Porto, however, an annotated dataset can be used for several Portuguese language NLP tasks. Its content consists of news from 17 organic units from the University of Porto, where a sample with 905 news was annotated. The identified entities were Hour, Event, Organization, Course, Person, Location, Date and Organic Unit. In total, this corpus has a total of 12644 named entities that were manually annotated.

In Pires (2017) several implementations of NER models trained with these datasets were created and made available to the public[1], which can be used with NLP tools like OpenNLP, spaCy, NLTK and Stanford-CoreNLP. In this work, we used the spaCy and OpenNLP NER model implementations and tested these models against our validation data. Unfortunately, the results obtained did not correspond to the intended ones. A reason for this could be that, although these models were trained with Portuguese textual data, they were not trained in the archival context. Another factor is that most of the entities labels used to train these models do not coincide with the ones used in our validation data.

4.2 TRAINING NER MODELS

Another approach used in this dissertation was to create our own ML models using our training data. In order to train ML models capable of recognizing archival context entities, two different approaches were used:

- Creating models from scratch with supervised learning;
- Fine-tuning existing models pre-trained in a huge amount of texts in a self-supervised fashion.

Training models from scratch consists of initializing the model weights randomly, which means that all the knowledge that the model will get lies in the training data. Another approach is to initialize the model with another model's weights. With this approach, the model uses the knowledge of previously trained models and fine-tune that knowledge into a specific classification task. Both approaches require annotated data to train and validate ML models. The datasets used for this correspond to the

1 https://rdm.inesctec.pt/ro/dataset/cs-2017-006

annotated data referred in Chapter 3. In this way, we performed a shuffle for each dataset, and then we split all the annotated corpora, 70% was used for the model's training while 30% was reserved for the models' validation.

4.2.1 *OpenNLP*

In order to train a MaxEnt model with OpenNLP directed to NER, several steps are required. First, it is necessary to process the training dataset. After loading the annotated data into memory, it is necessary to tokenize it, splitting all the document sequences into tokens. In order to do so, OpenNLP provides pre-trained tokenizer models optimized for several languages, Portuguese in our case. It is important to emphasize that the tokenization method used in the training and validation must be the same so that the model processes the words in the same way. Using different methods can lead to a decrease in the model performance.

Then, *Name Samples* are created, consisting of data structures that hold named-entity spans and tokens, and are later used to create a stream of events in order to train the model.

0	1	2	3	4	5	6	7	8	9	10	11
"	It	is	a	familiar	story	,	"	Jason	Willaford	said	.
other	other	other	other	other	other	other	other	start	continue	other	other

Figure 21: OpenNLP event stream tagging.

OpenNLP associates each token with a *Start*, *Continnue* or *Other* tag, which represents whether a given token corresponds to the beginning, middle, or is not associated with a named entity, respectively.

With the data processed, we have to define the model features. In this case, we used the default defined features, which are presented below.

```
AdaptiveFeatureGenerator featureGenerator = new CachedFeatureGenerator(
        new AdaptiveFeatureGenerator[]{
                new WindowFeatureGenerator(new TokenFeatureGenerator(), 2, 2),
                new WindowFeatureGenerator(new TokenClassFeatureGenerator(true), 2, 2),
                new OutcomePriorFeatureGenerator(),
                new PreviousMapFeatureGenerator(),
                new BigramNameFeatureGenerator(),
                new SentenceFeatureGenerator(true, false),
        });
```

Listing 4.1: OpenNLP features.

These features can have a great impact on the performance of the statistical model, so they must be defined according to our NER problem.

Firstly we have a *WindowFeature* in conjunction with a *TokenFeature*. The *WindowFeature* tells the model the range of words to the left and right of a token that it must put focus on. The *TokenFeature* concerns the token itself, helping the model to learn the language vocabulary. The combination of both features makes the model observe the two words on the left and right so that it learns which words usually appear together, creating a notion of context. After that, *WindowFeature* is used with *TokenClassFeature*. *TokenClassFeature* refers to the class of a given token, i.e., its shape, if it is a numeric token or if it starts with a capital letter, if it contains special characters, etc. In order to find the word's class, regular expressions are used. Again, this feature will be applied to the four words closest to the token due to the *WindowFeature*. In addition to these, the *PreviousMapFeature* feature is also used, which consists of considering the result of the entity recognition performed on the last processed words.

OpenNLP provides a set of features that can be used in conjunction with each other, and it is up to the experimenter to choose the ones that fit his context. The created features will then help the model predict the correct tag for each token.

After that, we started the training of the model which consists of an iterative process, so it is necessary to define the number of iterations. It is this iterative process that allows the feature overlap behavior. By default, this hyper-parameter has a value of 100, however, we registered the model results with different number of iterations in order to analyze its behavior.

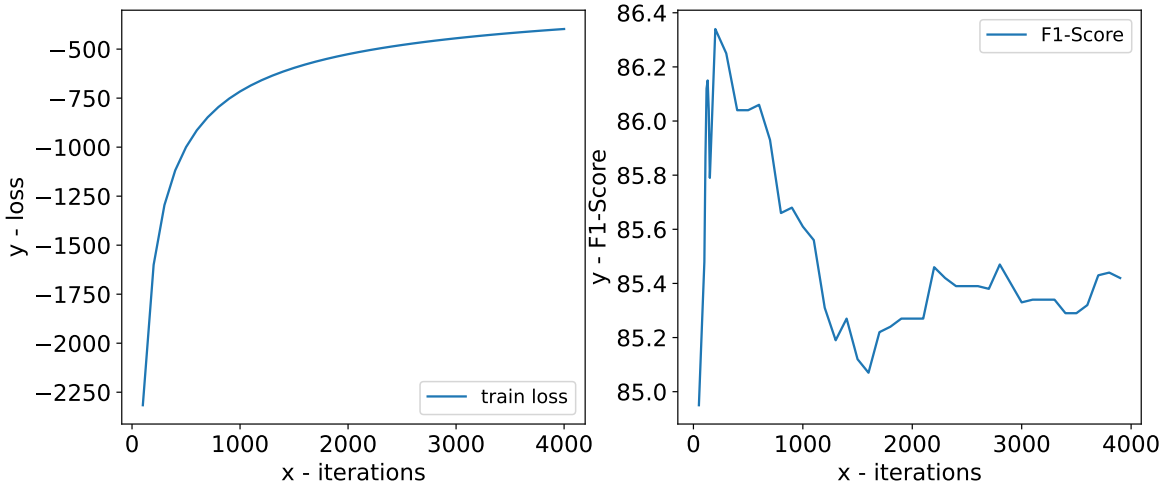


Figure 22: OpenNLP model learning curves.

In Figure 22 we have the learning curves of our model. These learning curves show our model's training loss and F1-Score (calculated on the evaluation data) by training iteration. As we can see, the loss absolute value is decreasing indefinitely, however, the F1-score rises until iteration 120 and then starts dropping, even with the loss value still decreasing. This means that, after iteration 120, the model starts over-fitting the data, learning the dataset noise. Thus, our final model was trained with 120 iterations.

In addition, there is yet another parameter to pay attention to, the cutoff. This variable corresponds to the minimum number of times a feature must occur in the model to be considered. In fact, by default, if a feature occurs less than five times, it will not be considered, which helps to reduce noise in the model. Finally, after the model is trained, it is converted into a binary file and saved to the disk (Ingersoll et al., 2013).

4.2.2 *spaCy*

With the data annotated and parsed into spaCy's format, we are ready to begin the NER model training. In Figure 23 we can see the workflow of the training process.

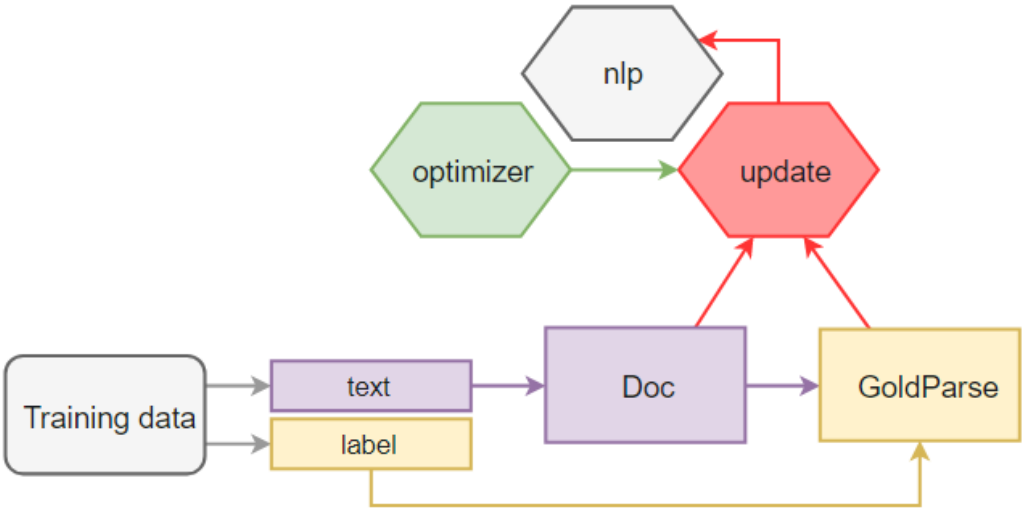


Figure 23: spaCy training workflow.

Initially, the labelled samples are loaded into memory, where the tokenization process takes place, generating Doc objects. To manage the way annotations are

stored, spaCy implements a "Gold Parse", which is responsible for storing the textual documents and their corresponding tags. In order to make the memory usage as efficient as possible, spaCy implements C-level structures to store the training data.

```
1
2 doc = Doc(Vocab(), words=["Merecendo", "grande", "destaque", "a", "fuga", "Alvaro", "Cunhal", "e
      ", "outros", "companheiros"])
3 gold = GoldParse(doc, entities=["O", "O", "O", "O", "O", "B-Pessoa","L-Pessoa", "O", "O","O"])
```

Listing 4.2: "*Gold Parse* entities tagging."

Observing Listing 4.2, each token in the Doc object is associated with a tag that identifies whether the token is a named entity or not. These tags were created using the *BILUO* scheme, which allows to classify the tokens and specify their position in the entity, i.e., if they are at the beginning (*Beginning*), in the middle (*In*), at the end of an entity (*Last*), if the entity is composed by only one token (*Unit*) or if the token does not belong to any entity type (*Out*) (spaCy, a).

Then we performed a shuffle of the training data and created the data batches. In fact, several iterations are performed during the training process, where each iteration influences the next one. By shuffling the data batches, we prevent our model from gaining dependencies on the samples order.

After that, we used spaCy's NER pipeline, which allowed us to create our NER models. In fact, spaCy is equipped with pre-trained word embeddings for multiple languages. In the case of the Portuguese language, spaCy provides a vocabulary with about 500 thousand unique vectors trained in datasets such as Rademaker et al. (2017) and Nothman et al. (2017). The use of pre-trained embeddings makes the model have a broader knowledge of the vocabulary of the Portuguese language, managing to extract relationships between the words. Through dense distribution vectors, the model can calculate the similarity of the words, thus knowing which words are close to each other in distribution. An example of this can be seen below.

```
1
2 def most_similar(word, topn=5):
3     ms = nlp.vocab.vectors.most_similar(np.asarray([nlp.vocab.vectors[nlp.vocab.strings[word]]])
      , n=topn)
4     return [nlp.vocab.strings[w] for w in ms[0][0]]
5
6 most_similar("rei", 7)
7 #output
8 ['rei', 'monarca', 'príncipe', 'grão-príncipe', 'príncipe-herdeiro', 'príncipe-regente', '
      imperador']
```

Listing 4.3: "*Gold Parse* entities tagging."

In Listing 4.3 we can see a function that is returning the seven words most similar to the word *"rei"* (king). The returning words are *"rei"*, *"monarca"*, *"príncipe"*, *"grão-príncipe"*, *"príncipe-herdeiro"* ,*"príncipe-regente"* and *"imperador"* which translate to king, monarch, prince , grand prince, heir prince and emperor, respectively. If we repeat this process for different context words, we get the distribution presented in Figure 24.

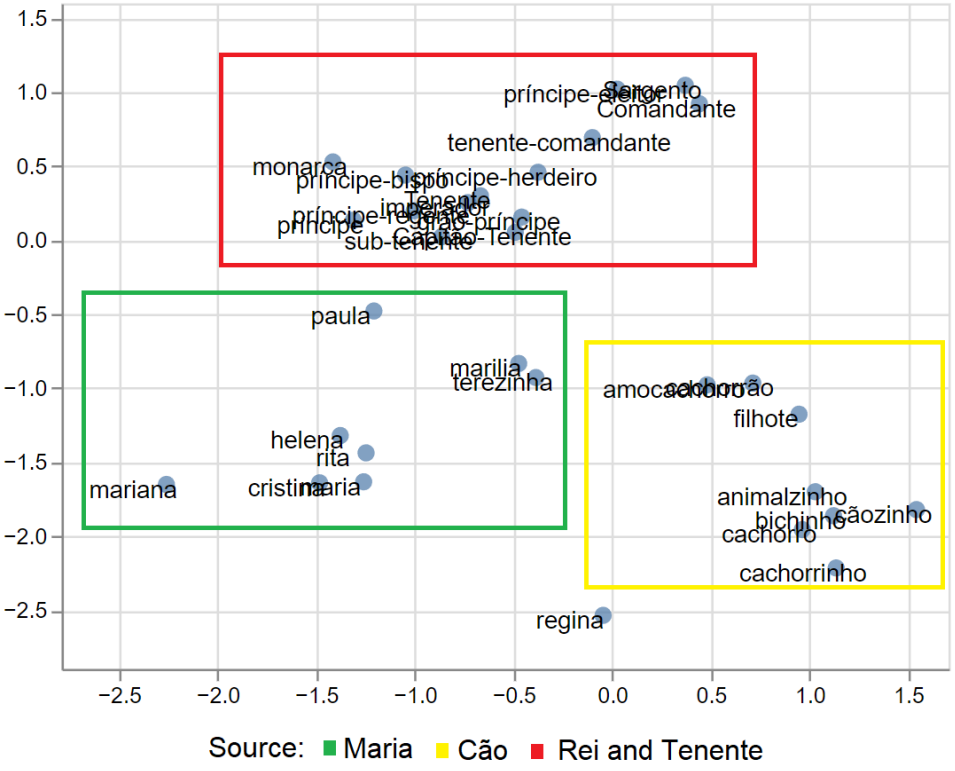


Figure 24: 2d spaCy word embeddings distribution.

Here we calculated the seven most similar words to the following tokens: *rei* (king), *tenente* (lieutenant), *maria* and *cão* (dog). From Figure 24, we can see that similar words are closer to each other. For example, words like king, lieutenant, prince, monarch, etc., are very close in the distribution as they are all related to people' titles, thus sharing semantic meaning. On the other hand, we can also see that most of the words that relate to *"Maria"* and *"Cão"* are also close between them. It is easy to understand that such a mechanism can help the model extract relationships between words. In this way, all our spaCy models were initialized with this word embedding technique.

Finally, we started the training iterations in order to update the model weights according to the defined spaCy features so that the model learns how to classify the desired named entities. Through these iterations, we calculated the error by comparing the model predictions with the annotated data. However, we do not want the model to overfit the data by memorizing only the given samples. The idea is to create a generalized model capable of performing in new similar contexts across unused data.

In spaCy, this problematic is addressed by using the error gradient of the loss function. This mechanism makes the model's input information less accurate, so it has more difficulties memorizing it.

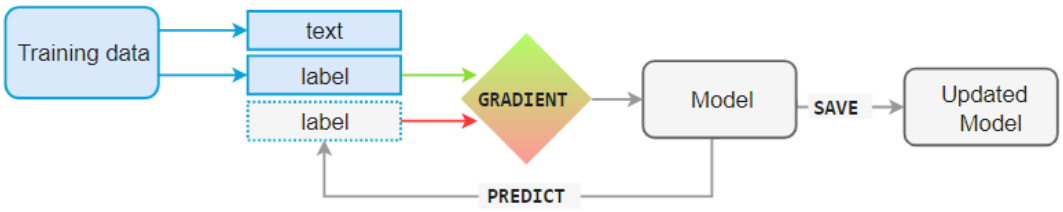


Figure 25: spaCy error gradient of the loss function.

With this approach, spaCy attenuates the data over-fitting problem by passing incomplete data to the model, giving it more freedom in its decisions. Furthermore, this technique is also used to help the model disambiguate words. For example, the word *Sameiro* may be pre-annotated as a person's name, however, this word may refer to *Santuário do Sameiro*, a Portuguese sanctuary. Therefore, in this case, the model must be able to discern this situation through the context in which the word is inserted. The model must realize what kind of entity a word refers to, based on the words in its neighbourhood and not only on the annotated samples.

Thus, spaCy let us set up this dropout mechanism as a customizable hyper-parameter, which prevents the model from being limited to the annotated dataset, generating space for new entity recognition possibilities, generalizing it to new contexts (spaCy, c).

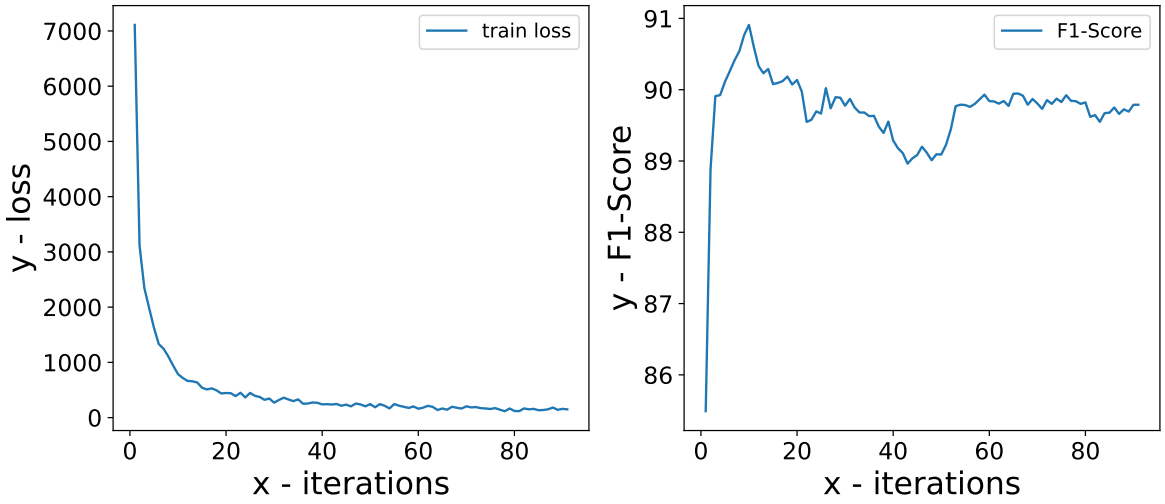


Figure 26: spaCy model learning curve.

Analysing Figure 26 (Left), we can see that the loss of the model decreases over the performed training iterations, increasing its ability to classify the named entities in the training data. On Figure 26 (Right), we have the F1-Score that is calculated on the evaluation data. In fact, the F1-score value increased from iteration 1 to iteration 10, where it peaked. Then is started to decrease even with the loss value getting lower. We can say that after iteration 10, the model starts to overfit the training data by learning its noise and putting too much focus on its details. This behaviour negatively impacts the performance of the model in unseen data. In this way, spaCy saves the model state of each training iteration in order to select the best model fit at the end of this process.

4.2.3 *BI-LSMT-CRF*

Another approach used in this dissertation to recognize archival finding aids entities is the use of the Tensorflow library to create a BI-LSTM-CRF model. Unlike the previous tools, Tensorflow is an ML library applied to various ML areas and not just NLP. Therefore, to train a NER model, we first have to handle the entire data transformation process.

To do this, initially, we have to load the training data. Once we have the data in memory, the next step is to convert it into numeric representations so the model can process them. The ML algorithms do not know how to process Portuguese language text, so we first generated the model's vocabulary. In order to do so, we created lookup tables where the word tokens and labels were mapped to a unique integer id.

Therefore, two dictionaries were created, one that contains the words (Listing 4.5), and another with the labels ids (Listing 4.4). Later, this table was used to convert the numeric representations back to textual words.

```

{'Data': 1, 'Local': 2, 'O': 3, 'Organizacao': 4, 'Pessoa': 5, 'Profissao': 6}
```

Listing 4.4: "Labels dictionary."

```
{'de': 1,          'Natural': 13,           'Meringolo': 9177,   'Adelina': 9189,
'e': 2,            'Filiação': 14,          'Pardo': 9178,       'Lbânia': 9190,
'do': 3,           'distrito': 15,          '2633': 9179,        'Rufino': 9191,
'ou': 4,           'º': 16,                 '2016': 9180,        'Espírito': 9192,
'em': 5,           'o': 17,                 'Atente': 9181,      'Prazeres': 9193,
'a': 6,            'n': 18,        (...)    'Joanesburgo': 9182, 'Etelvina': 9194,
'da': 7,           'que': 19,               'Gavela': 9183,      '1933': 9195,
'Maria': 8,        'Registo': 20,           'Calanga': 9184,     '1988': 9196,
'concelho': 9,     'Manuel': 21,            'Mambiça': 9185,     'Jesuína': 9197,
'país': 10,        'Pai': 22,               'Sotero': 9186,      'Sara': 9198,
'actual': 11,      'Mãe': 23,               '1951': 9187,        'Libânia': 9199
'residente': 12,   'para': 24,              'Bairros': 9188,     'terceiras': 9200}
```

Listing 4.5: BI-LSTM-CRF words' dictionary.

As we can see in Listing 4.4, the dictionary of labels is pretty small, as we only want to classify 6 different labels (including the out of entity label). However, the dictionary of words represents the vocabulary of our model, i.e., all the words that it knows, the reason why it usually has a large number of entries. In order to generate the word dictionary, a word-based tokenizer was used, where an id was associated only with the most frequent N words.

In fact, in Listing 4.5, the tokens are ordered according to the number of times they appear in the training dataset. As we can see, the most used words correspond to conjunctions, prepositions, determinants and pronouns. In addition, we also have the presence of people's names *"Maria"* and *"Manuel"*, names that are widely used in Portugal, specially in older times. At the end of the list, appear the less frequent words, such as *"Adelina"*, *"Rufino"* or even dates that were mentioned a few times.

With the created dictionaries, we can start to create numerical representations of the text sequences. In Listing 4.6 we have an example of this process

```
O Escritura      O datada          I-Data  1588        B-Pessoa  Maria
O de             O de              O entre             I-Pessoa  Vaz
O venda          B-Data  19        B-Pessoa  Gonçalo   O e
O e              I-Data  de        I-Pessoa  Pires     B-Pessoa  André
O respectiva     I-Data  Fevereiro O e                 I-Pessoa  Fernandes
```

```
O posse          I-Data  de           O mulher             O e
                                                           (...)

words = [[2125, 1, 1482, 2, 2126, 695, 426, 1, 165, 1, 560, 1, 2755, 271, 1038, 347, 2, 225, 8,
    357, 2, 958, 106, 2, (...), 0, 0, 0, 0, 0], (...)]

labels = [[3, 3, 3, 3, 3, 3, 3, 3, 1, 1, 1, 1, 1, 3, 5, 5, 3, 3, 5, 5, 3, 5, 5, 3, (...), 0, 0,
    0, 0, 0], (...)]
```

Listing 4.6: "Text sequence encoding."

This listing has an example of sentence processing where we converted the BIO-format data into an array numerical representation.

With the lookup tables ready for back-conversion, we proceeded to generate the data batches. All batches must have the same length, however, our sentences are composed of variable length sequences of tokens. Thus, we defined a maximum sequence length and applied a masking technique to the batches. As we can see in Listing 4.6, both the labels and words vectors have zeros at the end. We added them in order to create vectors of the same size, however, these masking values will be ignored in the model training. We also truncate sequences that have lengths higher than our maximum sequence lengths.

After encoding all text sequences into numeric arrays, the next step is to generate the word embeddings.

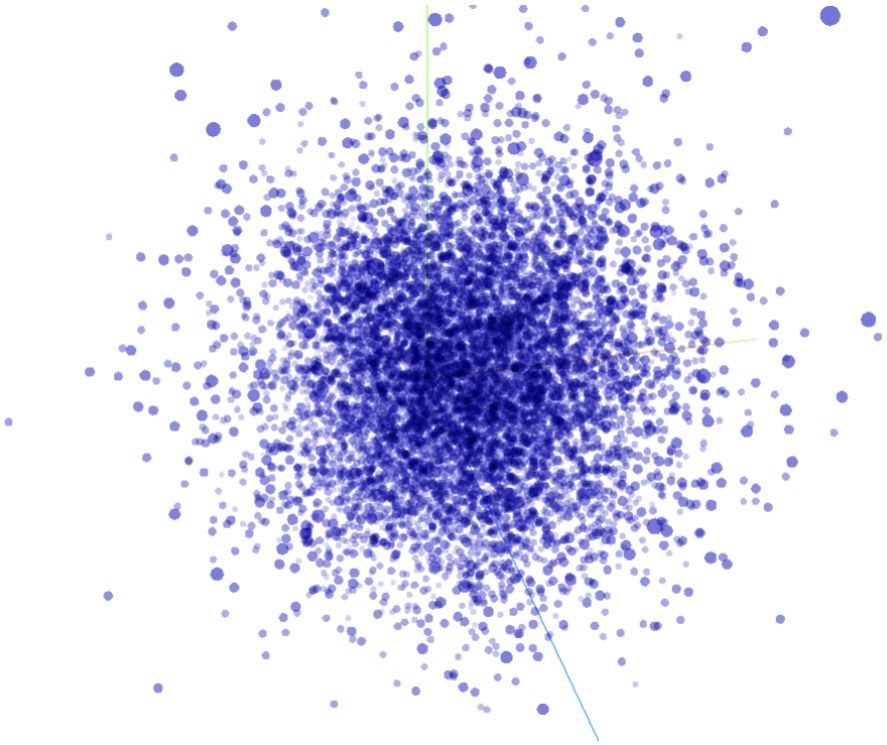


Figure 27: BI-LSTM-CRF Word Embeddings distribution.

These embeddings are dense low-dimensional representations of vocabulary data. By using more than one dimension, these representations can capture more meaningful information of each token, increasing the model knowledge about the vocabulary. For example, each dimension can express sentiments, grammatical features, genders, etc. In order to generate these embeddings, we mapped each token index to a random initialized dense vector. During the model training, the vectors' values will be updated so that words with similar meanings will have similar distribution values. In Figure 27, taken from Tensorboard, we can see the word embeddings distribution of the model.

Then, we used the Tensorflow library to implement two LSTMs, one for the forward context and the other for the backwards context. After that, we applied a CRF layer to the LSTMs output in order to process the sequence labels. As for the loss optimizer, we used Adaptive Moment Estimation (Adam), as it helps the model to converge faster and has revealed very good results in several NLP tasks.

During model training, we created model checkpoints on every 100 training steps. In addition, the training and evaluation loss were recorded to generate the Learning Curves of the model.

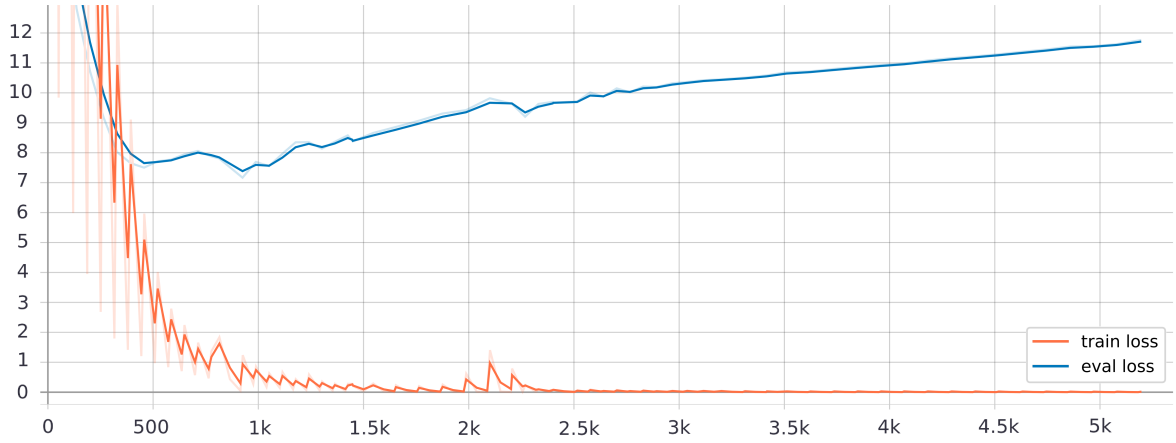


Figure 28: BI-LSTM-CRF model learning curve.

Observing Figure 28, the train loss curve (orange) is calculated using the training dataset and helps us to understand how the model is learning. The evaluation loss curve (blue) is calculated with the validation dataset and gives us information about how the model is generalizing.

In this way, we can see that the training curve converges close to the 1200 training steps. As for the evaluation curve, it drops until approximately 1000 training steps and then starts rising indefinitely. This indicates that the model has already learned the training data too well and started over-fitting it. This behaviour makes the model learn training data errors or noise too closely and prevents the model from generalizing. In this way, we used the checkpoint model at 1000 training steps for the NER prediction task.

4.2.4 *BERT*

Until now, we used approaches that consisted of training NER models from scratch. Now we will present a different model, BERT, which consists of using pre-trained models with hundreds of thousands of parameters and leveraging the knowledge acquired during the pre-train by fine-tuning them on a specific task, in our case, NER.

Models such as BERT (Devlin et al., 2019) are pre-trained using self-supervised learning, using a word masking technique, allowing the model to access both the past and the future context of the token. This mechanism creates a bidirectional model, an important feature for the token classification task. Since these models are created with many data, they have large dimensions, reaching billions of parameters, which

makes their training process resource and time expensive. This process requires a large number of GPUs, and can take several days or even weeks consuming high amounts of energy, however, it only has to be performed once due to the model's reusability property.

By training models from scratch, we randomly initialise the models' weights, which means that the model only learns from the training data. However, with this fine-tuning approach, we transferred the knowledge of models pre-trained in huge amounts of textual data. In practice, we used the pre-trained model weights, which provided the model with a statistical understanding of the language, and then we fine-tuned it with a task-specific classifier.

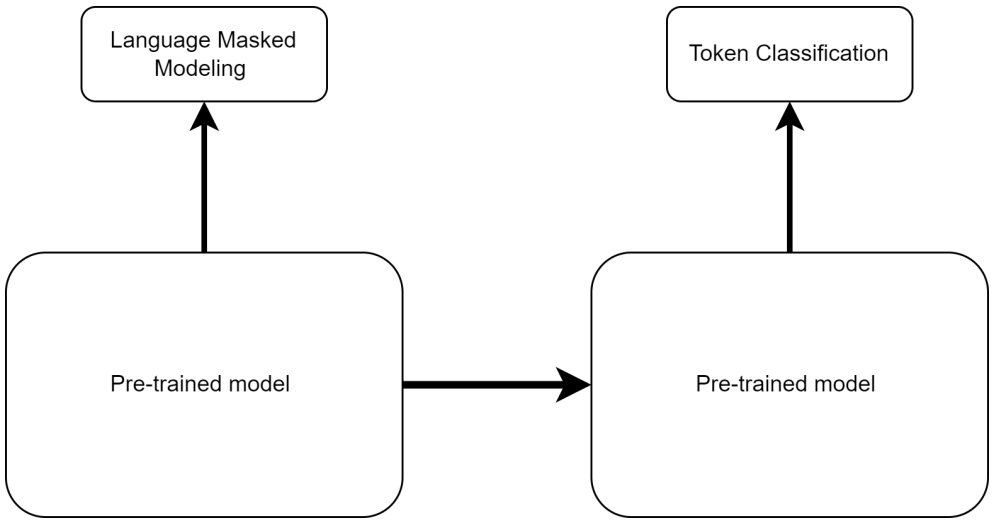


Figure 29: BERT transfer learning.

Because of this, in order to make the most of the pre-train model knowledge, we have to use a BERT model that was pre-trained in a text vocabulary similar to our training data. Thus, the first step consisted of finding a self-supervised pre-trained BERT model in a huge quantity of Portuguese textual data. Therefore, we selected three different candidates from the Hugging Face Hub (Wolf et al., 2020). First, we have the `"bert-base-portuguese-cased"` and `bert-large-portuguese-cased"` models, with 110M and 330M parameters respectively (Souza et al., 2020). These models were trained in a Brazilian Portuguese corpus composed of 2.7 billion (Filho et al., 2019) tokens. Secondly, there is a multilingual model, bert-base-multilingual-cased (Devlin et al., 2018) with 110M parameters, trained on the largest Wikipedia texts, making this model capable of processing texts in 104 different languages, Portuguese included.

Out of curiosity, at the moment, the largest existing model was created by the cooperation between Microsoft and Nvidia, published in October 2021, Megatron-Turing Natural Language Generation (Alvi and Kharya, 2021), a model with 530 billion parameters trained with 4480 A100 80GB GPUs in a set of 15 datasets consisting of a total of 339 billion tokens.

It is important to note that, by transferring all the pre-trained information to our model, we are increasing its language knowledge, however, all the errors, noise or even bias are also transferred. Brown et al. (2020) states that the pre-trained models can be biased to their training data. In this paper, they analysed the GPT-3 model's biases towards gender, race, and religion, however, they state that the model may express other categories of bias. In practice, regarding the religion bias, this behaviour made the model associate words such as `"Islam"` with `"terrorism"`. As for genre bias, the `"female"` word was usually associated with tokens such as `"naughty"` or `"beautiful"` while the `"male"` word is associated with the tokens `"large"`, and `"lazy"`.

Data processing

After selecting our models' checkpoints, we can start the data loading and transformation process similar to the approach presented in the BI-LSTM-CRF model. First, we loaded the training and validation data into memory and then, we started to tokenization process, mapping all the tokens into their corresponding ids in the pre-trained vocabulary. The training samples must comply with the pre-trained vocabulary, having the same format and structure, which means that the tokenizer used for the pre-training must be the same for the fine-tuning.

Transformer-based models usually use sub-word tokenizers, meaning that the rarest words of the vocabulary are usually split into sub-words.

```
#Original
['Pelo', 'qual', 'foram', 'identificados', 'os', 'bachareis', 'que', 'seriam', 'desembargadores', 'da', 'Relação', 'e', 'Casa', 'do', 'Porto', '.']

#Tokenized
['[CLS]', 'Pelo', 'qual', 'foram', 'identificados', 'os', 'ba', '##char', '##ei', '##s', 'que', 'seriam', 'desembar', '##gadores', 'da', 'Rela', '##ção', 'e', 'Casa', 'do', 'Porto', '.', '[SEP]']
```

Listing 4.7: "Text sequence encoding."

In fact, even if the training data is already tokenized in BIO format, we have to tokenize it again to create the correct correspondence between the training data and

the pre-train vocabulary id. In Listing 4.7 we can see that the tokens *"bachareis"* and *"desembargadores"* were divided into smaller sub-tokens. Consequently, since we changed the sequence vectors lengths by splitting tokens into sub-tokens, we also have to remap the labels' vectors to ensure that each token is associated with its correct label. After that, we had to pad and truncate all the sequences according to the pre-trained maximum length parameters.

Fine-tunning

With the data processed, we can start fine-tuning the model for the archival context. In order to do so, we used the Hugging Face library that allows us to fine-tune several state-of-art NLP models with Keras and Pytorch API.

The fine-tuning process removes the pre-trained model head (last layer) focused on the masked language objective and replaces it with a new randomly initialized head. The idea is to create a new classifier specialized in recognizing entities from archival data, with N outputs, where N is equal to the number of labels.

To fine-tune the model, we only used NVIDIA GEFORCE GTX 1070 Ti GPU. Because of this, we had to adjust the batch size for the bigger models to avoid out-of-memory errors. The fine-tuning of each model lasted from 30 minutes to 2 hours. During this process, we logged the training and evaluation loss in order to generate the model's learning curves.

In Figure 30 we can see the loss values per epoch of the three generated models, bert-large-portuguese, bert-base-portuguese and bert-multilingual. In fact, the models start to overfit between 2 and 4 epochs. In order to retrieve the models with the highest capacity of generalisation, the Huggingface library keeps several model checkpoints during the training phase and selects the best one at the end. Only the best checkpoint will be used for the actual Named Entity Recognition.

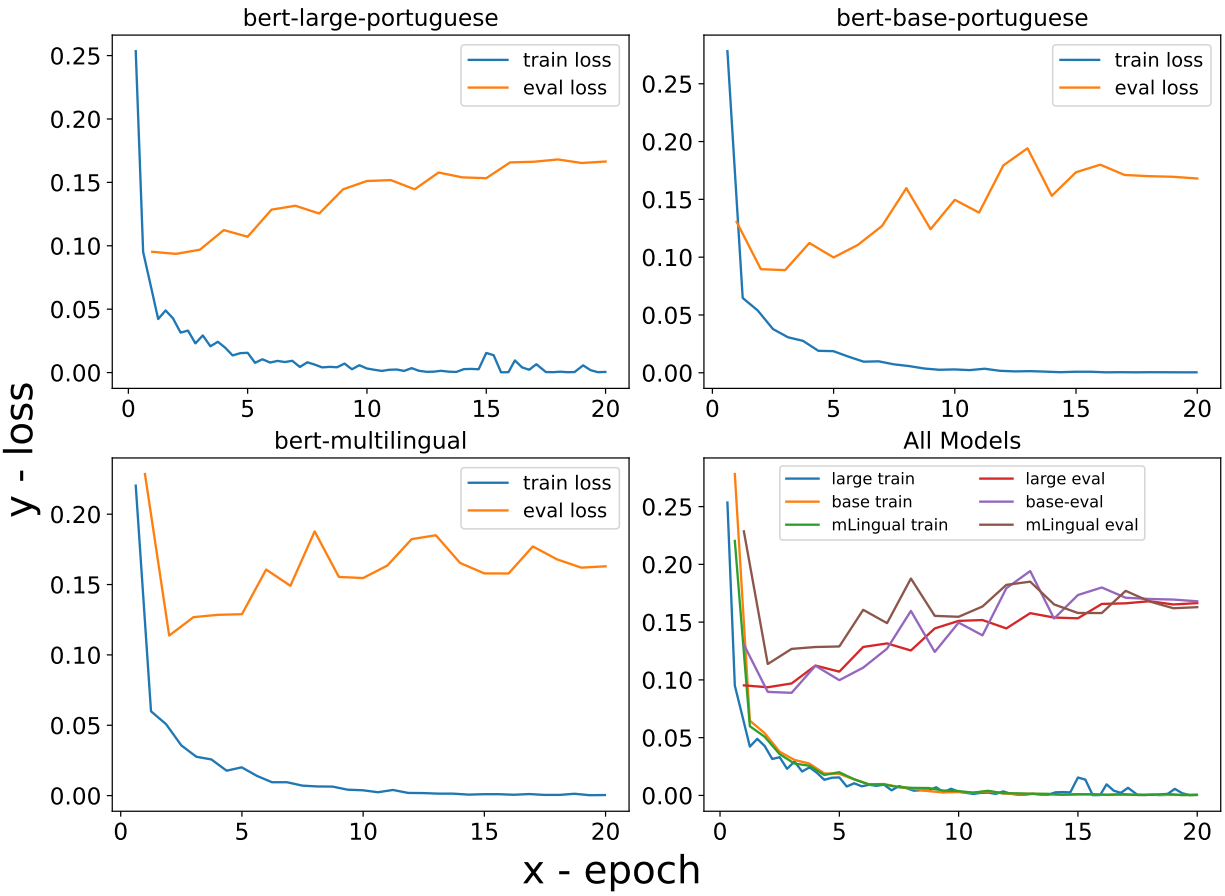


Figure 30: BERT models learning curves.

4.3 CONCLUSION

In this chapter, we presented several approaches to generate NER models capable of recognizing entities in the archival context.

The first approach consisted of selecting available pre-trained models in Portuguese language documents. Despite being a quick and easy way to obtain NER models, these models were not trained in archival content documents, containing identified entities of no interest. In the end, the obtained results were underwhelming.

The second approach consisted of training our own ML models, with the annotated corpora generated in Chapter 3. Here, 4 different algorithms were used, trained with different methodologies, from creating models from scratch up to using pre-trained word embeddings or even self-supervised learning proceeded by fine-tuning. Each

implementation used different techniques in order to try to obtain the best possible results, which will allow conclusions to be drawn during validation.

Finally, all these models are available to the public via the Cunha and Ramalho (2021c) and the hugging face hub[2].

2 https://huggingface.co/lfcc

5

NAMED ENTITY RECOGNITION RESULTS

In this chapter, we will present the validation results of all models trained in this thesis. The metrics used to measure NER models' performance are Recall, Precision and F1-score since the accuracy metric does not satisfy the needs of this NLP field (Derczynski, 2016).

The main focus of this work was to create a mechanism that could recognize entities from archival finding aids with enough confidence. In this way, several NER models were trained with different data contexts. The closer the context of training data and the validation data, the better NER results will be, as the model can rely more on its learning. Thus, in order to test this hypothesis, the first approach consisted of generating individual models for each dataset.

However, even if we can achieve good results within the same corpus, that is not really impressive or useful, as one would have to annotate part of the corpus to extract named entities from it. In this way, the next step was to test the model generalization. In order to do so, we tested how a single model trained with data from all datasets would perform, compared to the last approach. Can the merge of contexts cause the model to get confused? Will it create a more powerful model? In this step, we intended to test those hypotheses. After this, we tested the generalized model with unseen data. In fact, the whole point of generating such a model is to recognize named entities from new data, so it is important to understand how will the model behave in this environment.

5.1 INDIVIDUAL NER MODEL PER CORPUS

The first approach of performing NER in archival data was to create one NER model for each annotated corpus from the data processing Chapter 3.6, splitting the data for training and validation, 70% and 30% respectively. In this section, we used our OpenNLP, spaCy, and Tensorflow models and presented their validation results.

Table 5: Individual NER models results.

Train	Validation	Model	Precision(%)	Recall(%)	F1-Score(%)
IFIP	IFIP	OpenNLP	87,08	82,61	84,79
		spaCy	88,16	89,90	89,02
		TensorFlow	96,12	98,67	97,00
Familia Araújo de Azevedo	Familia Araújo de Azevedo	OpenNLP	72,56	72,30	72,43
		spaCy	74,41	72,82	74,09
		TensorFlow	88,98	87,28	86,32
Arquivo da Casa Avelar	Arquivo da Casa Avelar	OpenNLP	80,15	79,85	80,00
		spaCy	87,82	87,18	87,50
		TensorFlow	89,25	93,25	90,63
Inquirições de Genere 1	Inquirições de Genere 1	OpenNLP	99,93	98,87	99,90
		spaCy	97,35	95,08	96,20
		TensorFlow	100	100	100
Inquirições de Genere 1	Inquirições de Genere 2	OpenNLP	63,17	62,17	62,67
		spaCy	89,66	85,98	87,78
		TensorFlow	98,86	98,95	98,78
Jadim do Mar	Jardim do Mar	OpenNLP	100	99,86	99,93
		spaCy	100	100	100
		TensorFlow	100	100	100
Jardim do Mar	Curral das Freiras	OpenNLP	93,37	99,84	96,50
		spaCy	99,97	99,90	99,93
		TensorFlow	100	100	100

Looking at Table 5, we can conclude that the created NER models were able to successfully classify most of the intended entities. It appears that in most cases, the BI-LSTM-CRF model generated with TensorFlow obtains the best results with an F1-score between 86,32% and 100%, followed by spaCy with an F1-score between 70,09% and 100%, and finally OpenNLP with an F1-score between 62,67 and 100%. As we can see with these results, the introduction of Deep Learning on NER reveals significant advances in this field.

It is important to note that only one model was created to validate datasets with high proximity in the context domain, for example, the *Inquirições de Genere 2* and *Curral das Freiras* datasets.

As we can see, with the OpenNLP model, when using the corpus *Inquirições de Genere 2* to validate the model trained on the *Inquirições de Genere 1* dataset, the results obtained were lower (62,67% F1-score) in comparison to the other tools (87,78% and 98,78% F1-score). In this case, it turns out that deep learning has demonstrated a greater capacity for transfer learning.

Finally, analyzing Table 5 we see that the FAA dataset is the one in which the models presented the lowest results. One reason for this could be that it contains very long sequences. In fact, as previously seen, a Bi-LSTM-CRF is more prepared to deal with Long Term Dependencies, presenting better results than the other tools (86.32% F1-score).

5.2 Generalized NER model

After obtaining the individual model approach results, an attempt to create a generalized model with annotations from all datasets was made.

Annotating a fraction of a dataset where this technology is to be applied is not always practical, so it would be interesting to create a generalized model capable of adapting to new contexts of similar nature.

In fact, by training a model with data from several contexts, it is able to learn more features about the language vocabulary. By doing so, we intend to prepare the model to perform in a wider variety of contexts, which includes text sequences it has never seen. However, this new approach must not obtain worse results than the individual models approach due to the models' degree of generalization.

For this method, the generalized models were trained with 70% of each dataset to be later validated individually with the remaining 30% of each one. This procedure was firstly tested with OpenNLP, spaCy and our Tensorflow implementation, obtaining the following results.

Table 6: Generalized NER models validation results.

Corpus	Model	Precision(%)	Recall(%)	F1-Score(%)
IFIP	OpenNLP	89.43	83.60	86.41
	spaCy	86.99	88.71	87.84
	TensorFlow	92.84	96.85	94.08
Família Araújo Azevedo	OpenNLP	81.94	63.67	71.66
	spaCy	75.19	76.78	75.98
	TensorFlow	78.22	82.47	78.89
Arquivo da Casa Avelar	OpenNLP	88.84	81.68	85.11
	spaCy	87.18	87.18	87.18
	TensorFlow	86.83	92.21	87.99
Inquirições de Genere 1	OpenNLP	99.60	99.53	99.57
	spaCy	98.31	96.74	97.52
	TensorFlow	100	100	100
Inquirições de Genere 2	OpenNLP	74.70	65.61	69,80
	spaCy	79.96	92.21	87.26
	TensorFlow	93.70	98.34	94,82
Jardim do Mar	OpenNLP	99.71	99.71	99.71
	spaCy	99.15	100	99.57
	TensorFlow	100	99.60	99.72
Curral das Freiras	OpenNLP	93.49	99.69	96.49
	spaCy	99.98	99.90	99.94
	TensorFlow	100	100	100

As can be seen, the results obtained by this model are similar to the previous ones, so we can say that the NER performance has not decreased. In this case, the BI-LSTM-CRF model obtained an average of 93.64% F1-Score, followed by the spaCy model with 90.76% F1-score and finally the OpenNLP model with an average of 86.96%.

Now, to measure its performance in a different context, we had to test it with new data. Observing our annotated corpora Table 4, we can see that we did not used one of the annotated corpus in the models' training. The corpus in question corresponds to the *Ruas de Braga* dataset, which contains brief notes of the city of Braga streets (the year 1750), with a considerable amount of named entities. We did

that to test the performance of the model on an unseen context domain, which is a crucial objective of this NLP task. In Table 7 we can see the results of this approach.

Table 7: Generalized NER models results on *Ruas de Braga* corpus.

Corpus	Model	Precision(%)	Recall(%)	F1-Score(%)
Ruas de Braga	OpenNLP	73.09	61.09	66.55
	spaCy	75.39	62.62	68.42
	TensorFlow	50.50	58.80	53.00

Comparing these results to the previous ones, the results obtained in this case are lower. The best model obtained an F1-score of 68.42.

In fact, in addition to the model not having been trained with any part of this dataset, it contains a lot of Organization and Profession type entities. As can be seen in Table 4, the model was trained with few instances of this type, thus their recognition may prove challenging.

On the other hand, it appears that this time, the model generated with BI-LSTM-CRF obtained worse results than the other models. One of the reasons for this could be that this model has a vocabulary restricted to his training data which can lead to a high amount of out of vocabulary words. In fact, both deep learning approaches presented in this thesis represent words through word embeddings, however, spaCy uses pre-trained word embeddings from a Portuguese corpus which makes it have a much larger vocabulary. On the other hand, in the BI-LSTM-CRF implementation, we created our own word embeddings during the model's training, which means that the model only knows the words contained in its training data. Therefore, while the spaCy model can assign semantic meaning to words not present in its training data, the BI-LSTM-CRF will identify such words as unknown words, making it hard to reason about them. To conclude, we can say that having a larger vocabulary can become a valuable tool when evaluating corpus with contexts that vary from the model's training.

5.2.1 *BERT Model's Results*

After testing our individual model approach against the generalized model, we understood that, in order to achieve this thesis goal, it would be more beneficial to keep working with generalized models only.

In this way, in this subsection, we present the results of our BERT models, which were fine-tuned and validated with the exact same annotated corpora as the previous generalized models.

For the sake of simplicity, we abbreviated the models' name, i.e., we used the name `BERT-large` to represent `bert-large-portuguese-cased`, `BERT-base` to represent `bert-base-portuguese-cased` and `BERT-multilingual` to represent `bert-base-multilingual-case`.

Observing Table 8, we find that, in general, the results obtained are satisfactory, ranging between 69.47% and 100% F1-Score, which is not surprising since there is great proximity between the training data and validation data, facilitating the models' classification task.

We can also see that there is a new model called `spaCy-BERT-base`. In fact, during the development of this work, spaCy released its 3.0 version, implementing the new transformers mechanism. In this way, we used spaCy's BERT implementation to compare it with other BERT models. By coincidence, spaCy uses one of the models already presented in this dissertation for the BERT model pre-train in Portuguese language, `bert-base-portuguese-cased`.

It is also verified that the model trained with a higher number of parameters presents better results. Here BERT-large obtained in average an F1-score of 88,45% followed by spaCy-BERT-base with 87,942%, BERT-base with 86,75, and BERT-multilingual with 85,932%. A larger model has a richer and more nuanced understanding of language, making it able to generate more accurate semantic interpretations.

Interestingly, some datasets obtained almost perfect results, close to obtaining an F1-Score of 100%, which is not usual in NER tasks. In fact, this happens due to the nature of the datasets in question, i.e., these datasets have a more structured internal organization and present a low variety of types of entities, facilitating the models' work.

Table 8: Generalized BERT models results.

Corpus	Model	Precision(%)	Recall(%)	F1-Score(%)
IFIP	BERT-multilingual	89.67	92.27	90.95
	BERT-large	91.06	93.43	92.23
	BERT-base	87.80	90.95	89.35
	spaCy-BERT-base	88.24	91.00	89.60
Família Araújo Azevedo	BERT-multilingual	74.17	85.03	79.23
	BERT-large	70.09	83.39	76.16
	BERT-base	64.27	75.59	69.47
	spaCy-BERT-base	80.73	83.51	82.09
Arquivo da Casa Avelar	BERT-multilingual	84.39	88.27	86.28
	BERT-large	84.47	87.47	85.94
	BERT-base	79.02	83.11	81.01
	spaCy-BERT-base	87.02	90.18	88.57
Inquirições de Genere 1	BERT-multilingual	99.95	100	99.99
	BERT-large	100	100	100
	BERT-base	99.95	100	99.98
	spaCy-BERT-base	100	100	100
Inquirições de Genere 2	BERT-multilingual	76.67	98.05	86.05
	BERT-large	94.81	98.11	96.43
	BERT-base	84.54	98.11	90.82
	spaCy-BERT-base	93.77	97.49	95.59
Paróquia do Jardim do Mar	BERT-multilingual	99.56	99.72	99.64
	BERT-large	99.53	99.90	99.71
	BERT-base	99.53	99.87	99.70
	spaCy-BERT-base	98.87	99.57	99.22
Paróquia do Curral das Freiras	BERT-multilingual	99.98	99.96	99.96
	BERT-large	99.99	99.99	99.99
	BERT-base	99.97	99.95	99.96
	spaCy-BERT-base	99.98	99.97	99.97

Finally, to see how these models would perform in a new context, we used the same approach as before. i.e., validated the generalized BERT models with the annotated corpus *Ruas de Braga*.

Table 9: Generalized BERT models results on *Ruas de Braga* corpus.

Corpus	Model	Precision(%)	Recall(%)	F1-Score(%)
Ruas de Braga	BERT-base	72.37	73.60	72.98
	BERT-large	75.38	74,67	75.03
	BERT-multilingual	72.55	63.70	67.84
	spaCy-BERT-base	71.18	74.09	72.61

From Table 9 we can see that the validation results with the *Ruas de Braga* dataset were lower compared to the other datasets, ranging from 67.84% to 75.03 F1-score. The `BERT-large` model was the one that got the best results. However, comparing this results with Table 7, we can see that the BERT models were able to perform better with unseen data. Again, the knowledge from the models' pre-train has revealed to be an important asset in this scenario.

5.3 Overall Results

In this section, we will present the overall results of our models, analysing and comparing their performance by dataset and named entity label.

Firstly in Figure 31 we can compare the models F1-score results against all our annotated corpora. In this case, we verified that the model created with BERT architecture obtained an average F1-Score of 90.69%, followed by the BI-LSMT-CRF with 88.56%, the CNN with 87.96% and finally, the MaxEnt model with an average F1-Score of 84.41. Despite this, at first glance, we can see that sometimes CNN and BI-LSTM-CRF models got better results than BERT, however, there is a good reason for that.

Seven of these datasets were used to train the models, however, as we saw before, the *Ruas de Braga* dataset was used only for validation, thus creating a more challenging environment for entity recognition. Furthermore, this dataset has many profession-type entities, which is a difficult entity type to classify due to its low number of occurrences in the training corpora. In this scenario, the transformers achieve better results compared to the other algorithms. This happens mostly because the BERT model was trained in huge amounts of text, making it have a higher knowledge of the vocabulary of the Portuguese language, which becomes a

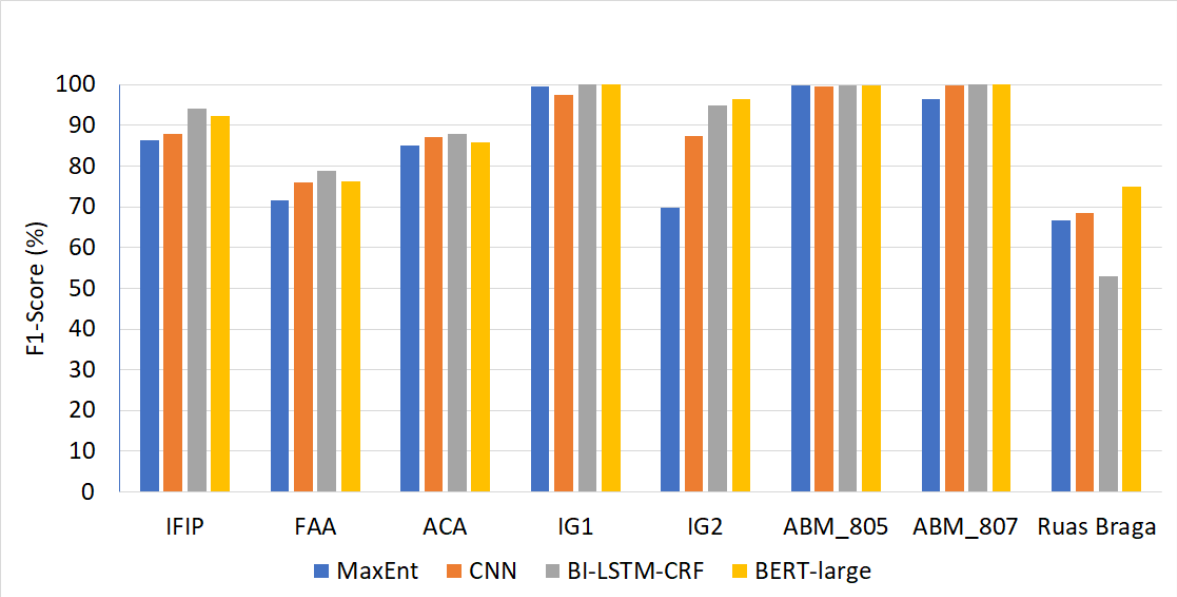


Figure 31: NER results by corpus.

powerful weapon when processing new documents which the models have never seen during their training. We can say that BERT models are less dependent on annotations in order to perform well.

Despite this, the other models continue to be a good alternative when we want to extract entities from smaller and more concrete contexts. In fact, smaller models can be used to better understand a specific context by learning the particular context features with greater detail. They also avoid the noise from the models pre-train.

Then, we decided to merge all the validation data into one dataset in order to generate the overall F1-scores of each architecture. After that, we used the generalized models against this validation dataset, which generated the results in Table 10.

Table 10: Overall models validation results.

Model	Precision(%)	Recall(%)	F1-Score(%)
OpenNLP	86.66	85.66	86.15
spaCy	88.75	92.13	90.41
Tensorflow	88.88	92.99	90.89
BERT-base	91.42	95.03	93.19
BERT-large	93.30	95.80	94.53
BERT-multilingual	89.86	94.85	92.29
spaCy-BERT-base	91.66	93.59	92.61

Analyzing Table 10 we can say that, the models that obtained the best results were the BERT models with BERT-large achieving an F1-score of 94.53%, followed by BERT-base with 93.19%, spaCy-BERT-base with 92.61% and BERT-multilingual with 92.29% F1-score. Then we have the other models, with the BI-LSTM-CRF achieving an F1-score of 90.98%, spaCy with 90.41% and finally, the worst model was OpenNLP with an average of 89.15% F1-score.

After that, we calculated the F1-score of each entity label per model, which can be analyzed in Table 11. From this table, we can observe the Precision, Recall and F1-score of each entity type per model. In general, we can say that the models demonstrated less difficulties in recognizing entities such as Person, Place and Date. As for the remaining entity types, Organization and Profession, the F1-scores are lower, which means that these entity types are more likely to be poorly classified by our models. One hypothesis for this could be the unbalance of the entities types in the annotated corpora, as we have less annotated Organization and Profession entity labels. In practice, the model has fewer samples of this type to learn how to correctly classify them, which can negatively impact the results.

Table 11: Generalized NER models results by entity label.

Entity Type	Model	Precision(%)	Recall(%)	F1-Score(%)
Organizacao	OpenNLP	72.64	66.96	69.68
	spaCy	67.40	79.13	72.80
	Tensorflow	76.14	84.81	80.24
	BERT-base	63.42	84.46	72.44
	BERT-large	68.28	80.31	73.81
	BERT-multilingual	60.29	82.35	69.61
	spaCy-BERT-base	72.27	74.78	73.50
Pessoa	OpenNLP	90.74	91.40	91.07
	spaCy	91.68	97.06	94.30
	Tensorflow	99.30	98.16	98.73
	BERT-base	92.98	97.20	95.04
	BERT-large	96.58	97.43	97.00
	BERT-multilingual	89.09	96.69	92.73
	spaCy-BERT-base	93.62	96.86	95.21
Local	OpenNLP	89.53	87.89	88.70
	spaCy	89.07	89.72	89.40
	Tensorflow	90.45	92.31	91.37
	BERT-base	93.67	94.87	94.27
	BERT-large	92.97	96.89	94.89
	BERT-multilingual	93.51	94.90	94.20
	spaCy-BERT-base	93.05	93.39	93.22
Profissão	OpenNLP	63.21	53.17	57.76
	spaCy	76.19	76.19	76.19
	Tensorflow	80.85	69.09	74.71
	BERT-base	72.13	74.16	73.13
	BERT-large	76.09	78.65	77.35
	BERT-multilingual	71.17	76.70	73.83
	spaCy-BERT-base	80.91	84.13	82.49
Data	OpenNLP	80.99	83.86	82.40
	spaCy	94.05	92.58	93.31
	Tensorflow	68.71	91.80	78.60
	BERT-base	98.99	98.74	98.87
	BERT-large	99.24	99.12	99.18
	BERT-multilingual	99.48	99.09	99.29
	spaCy-BERT-base	95.67	94.92	95.29

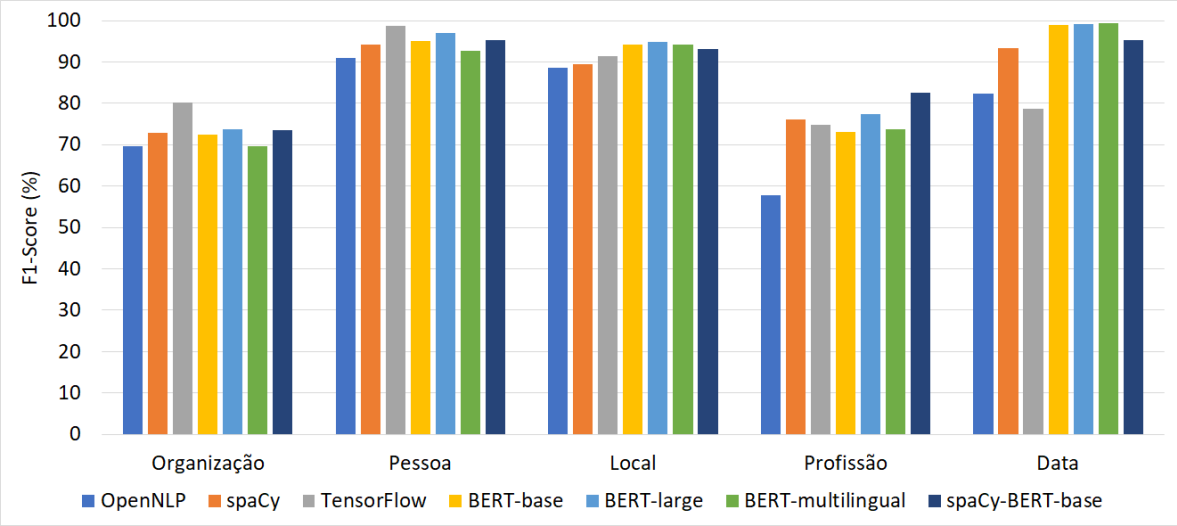


Figure 32: NER results by entity label.

Finally, in Figure 32 we have a plot that displays the models' F1-score results per entity type. Here we can analyze and compare the disparity between the entity labels recognition results per model. In general, the models demonstrated more difficulty in recognizing Profession and Organization entity types. Beside that, some models achieved better results in the recognition of specific entity types. For example, the BI-LSTM-CRF model was the best model in recognizing the Organization entity type with an F1-score of 80.24%. Additionally, all the BERT models demonstrated superiority in identifying Date entity labels. On the other hand, we have the OpenNLP model, which was the model that achieved lower results in almost all the named entity labels, which makes sense since it was the model that showed the worst results overall in Table 10.

5.4 CONCLUSION

In the end, we were able to achieve satisfactory results by annotating part of the datasets we wanted to recognize entities on. In the first approach, where we trained individual models for each dataset, the results were promising, achieving high F1 scores in most annotated corpora.

Following that, with the creation of the generalized models, we confirmed that by joining all the training corpora into a single dataset, we could create models that could perform on a wider variety of contexts, with no noticeable performance loss.

As for the entities recognition in contexts that differ from the model's training, in our experiment, we achieved results of 75% F1-score, which is lower than the other experiments. Here we concluded that models such as BERT, with previous knowledge from their pre-train, could achieve better results than training from scratch approaches. In fact, pre-trained models have a much better knowledge of the vocabulary which means that, with less annotations, they can achieve better results in the entity recognition.

In order to achieve better results, one could annotate more data associated with the archival context. Another factor that could improve our models' performance would be to improve the entity balancing of our annotated corpora. As for our models, there are also several improvements we could try. For example, regarding the BILSTM-CRF model, the addition of pre-trained word embeddings could help the model correctly process a wider variety of words. The use of this method has already shown improvements in this field.

As for BERT models, we could train our own BERT model from scratch instead of using a pre-train model in a different context. The model used in this work was trained on Brazilian Portuguese data, which is not the ideal solution for our Portuguese archival context. Since there is a colossal amount of Portuguese archival information available to the public, it could be used to train a new model with a greater knowledge of the archival context vocabulary.

6

NER@DI

6.1 WEB PLATFORM - NER@DI

With the NER models generated, we decided to create a platform in order to make them available to the public. Thus NER@DI was born, a web platform that provides several tools that were developed throughout this work, such as the annotated corpora, parsers to convert datasets into different formats and, of course, the ML models that make it possible to process archival finding aids, extracting relevant named entities.

6.1.1 *Architecture*

This platform was created with the intent of being complemented with new features in the future. Thus, a micro-service architecture was used, promoting looser coupling, more flexibility and portability.

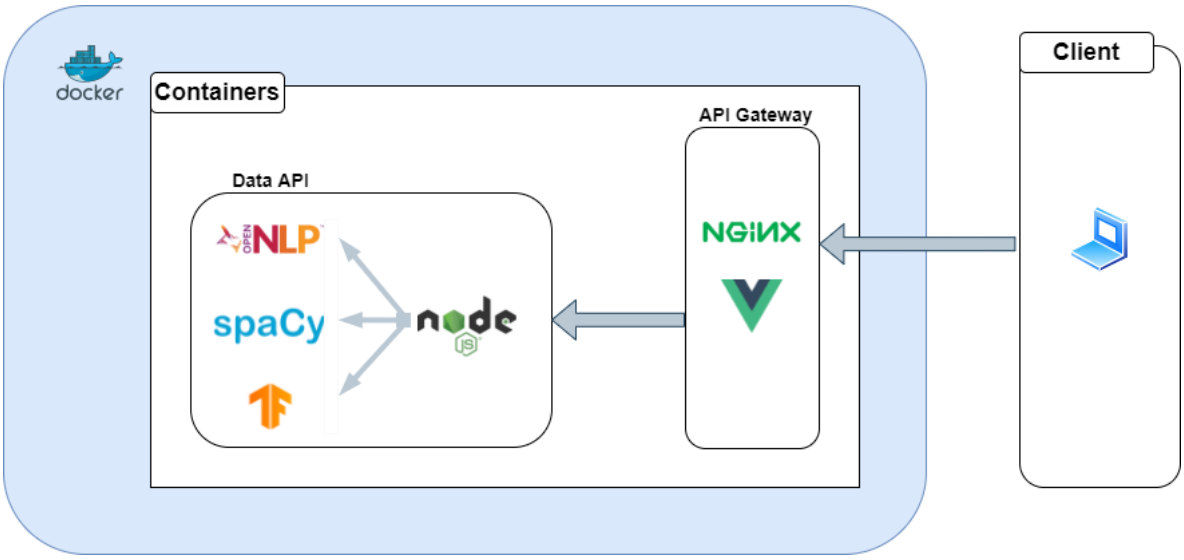


Figure 33: NER@DI architecture.

At the moment, it has two containers that correspond to the Data API and the API Gateway.

The API Gateway is implemented with an Nginx web server containing the client application, developed in Vue.js, a framework that uses reactive interfaces. The use of an API Gateway pattern makes the client less coupled to the micro-services, i. e., it does not need to know the internal structure of the server to communicate with the application. The use of this pattern means that there are no direct references between the client and the microservices, so the refactoring or maintenance of these will not affect the client. On the other hand, if a gateway server was not used, all microservices would be exposed to the public, which could lead to security issues.

Then, Vue.js was used to create the client application. It has a small learning curve, so it is fairly approachable, allowing the creation of maintainable interfaces due to its reusable components mechanism that allows isolating all logic from the views. To complement this framework, Vuetify was used, which consists of a UI library that provides several pre-made reactive components.

The second container is the Data API, which is responsible for receiving, processing and responding to HTTP requests, in this case, associated with the extraction of entities. For this, a Node.js server was used, complemented by Express.js (n.d) library, which works like a broker that is responsible for managing the API routes and delegating the NER processing to the corresponding tools. The Machine Learning models were implemented with OpenNLP, spaCy and Tensorflow so, to process NER requests, the Node.js server uses the *child_process* (Node.js, n.d) library, creating

new processes to execute programs in python and java. When the execution of the programs finishes, the created processes return the output to the Node.js server, which is responsible for returning the response to the client in JSON format.

Finally, in order to deploy this platform, each micro-service was wrapped with a docker container. These containers promote the isolation, scalability, agility and portability of each micro-service since it is really easy to install a containerized application in any system that has docker running. At the moment, NER@DI is hosted on the servers of the University of Minho's Informatics Department, at Cunha and Ramalho (2021c).

6.1.2 *Features*

During this dissertation, several support tools were generated that helped to create and optimize NER's ML models. Thus, some of these tools were selected to be implemented in NER@DI, presenting the following features:

- Performs Named Entity Recognition with three different ML models, pre-trained in Portuguese archival finding aids.
- Implements sorting and filtering functionality that enables, for example, to sort entities by label and remove repeated extracted entities.
- Supports text file import in order to process them with ML models and export the obtained results in CSV and JSON formats.
- Provides tools to parse annotated corpus into different formats, for example, CSV and BIO.
- Provides download methods to all the annotated corpora making it available to the public.
- Presents a set of results from previous entity extractions so that it is possible to verify real case applications of each model in several different corpora.

Thus, NER@DI can be used by various types of users, for example, historians wishing to extract relevant entities from archival documents or even other developers or researchers with the intent of reusing the annotated datasets in other contexts.

6.1.3 *Interface*

In this section, we will present some views of the NER@DI platform. In Figure 34 we can see the results of our spaCy NER model when applied to the *Arquivo da Casa Avelar* corpus.

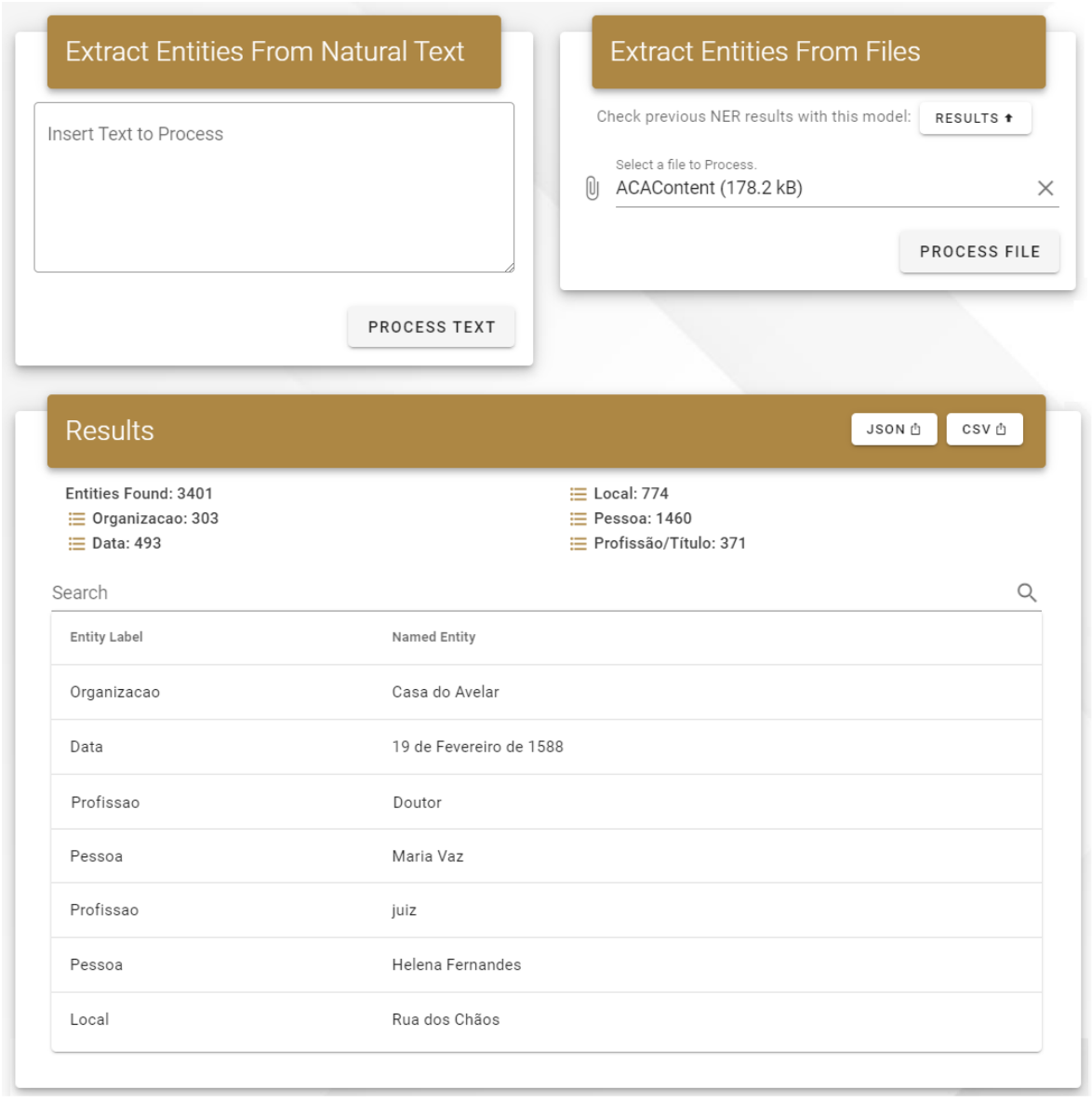


Figure 34: NER@DI, spaCy NER model results on *Arquivo da Casa Avelar* dataset.

Firstly, we have two cards that allow us to insert natural text or a text file as input to the NER models. Then, the data is passed to the models, which will identify and extract the named entities and present their output into the result card.

The results card displays the total number of recognised entities per label. There is also a table with all the identified named entities and a search text field that we can use to filter them. In addition, we can sort the entities alphabetically or by entity label and remove the repeated entities. Then, in the top right corner, we have two buttons that allow us to export the NER results into JSON and CSV format.

Then, in Figure 35 we have another interface view, where we published our annotated corpora.

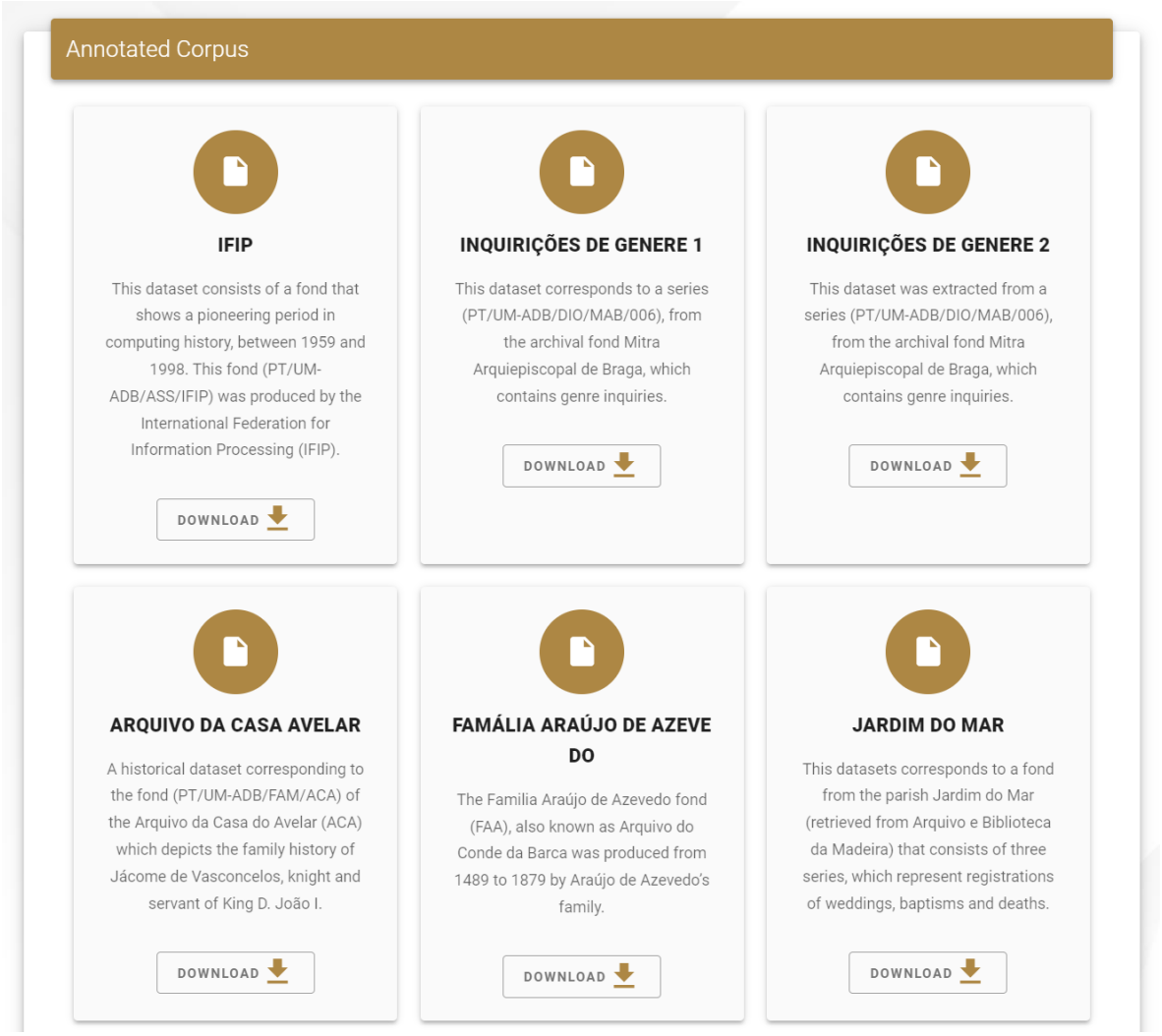


Figure 35: NER@DI annotated corpora.

In this view, we have a list of annotated datasets, with a brief description that corresponds to all the annotated corpora created in this work. In total, NER@DI has 9 annotated datasets.

6.2 SMART ANNOTATOR - ARCANO

The key to obtaining good results in this subfield is the training data quality. The closer the context of the training data to the context in which this technology is intended to be used, the better the results will be. Good training material is not always available, which creates the need to annotate text. Normally, this activity is performed manually by an experimenter who is knowledgeable about the domain of the documents. However, this task can become time-consuming and tedious despite its low complexity. Thus, the idea of developing an intelligent tool to support text annotation, *ARCANO* Cunha and Ramalho (2021c), was born.

This annotator aims to intelligently assist the entire annotation process using ML models to try to predict named entities of the texts we want to annotate. The idea is to use a generic model to find entities in a small fraction of the target dataset. Then, by correcting the entities found, it is intended to teach the model to annotate the concrete context, iteratively.

The sequence diagram in Figure A.3 illustrates the annotation process flow in *ARCANO* .

Initially, the experimenter imports the entire target dataset for it to be annotated. Then, this dataset is divided into N batches that will be used to train the ML model. Firstly, the first batch is sent to the server to identify and classify the entities it can find. Then, the result is sent to the client to be corrected by the experimenter. Now that we have validated training data, we can use it to refine the ML model. Thus, the correctly annotated data is sent to the server to train the model as well as the second batch so that the trained model can extract new entities from it. The more annotated batches, the higher the amount of training data, making the model learn how to classify entities even better. This makes the experimenter annotate fewer and fewer entities manually. The greater the autonomy degree of the model, the lesser the work of the experimenter. Finally, *ARCANO* joins all the previously annotated data into a file, enabling to export the fully annotated dataset.

This tool was used to annotate a corpus from the *Arquivo Nacional da Torre do Tombo*, the *Arquivo de Oliveira Salazar*'s archival finding aids. As we can see in Table 12, in total, 71397 tokens were annotated, making more than 7000 different entities.

In the end, the annotation process was considerably easier and faster due to the intelligent entity recognition system.

Table 12: Annotated entities of *Arquivo de Oliveira Salazar* corpus.

Corpus	Person	Place	Date	Profession or Title	Organization	Total
Arquivo de Oliveira Salazar	2641	1807	279	1414	1258	7399

6.2.1 *ARCANO Interface*

ARCANO's visual interface was developed to make the named entities annotation process easy, intuitive, and fast. Figure 36 shows the its main interface.

Figure 36: *ARCANO* Anotator interface.

This figure shows an example of a record from the *Arquivo de Salazar* finding aids file, which was automatically annotated by the model and later validated by the experimenter. To remove a token's label, we just have to select that token with the right mouse button. As for annotating a named entity, we need to identify the token or set of tokens we want to annotate and press the key associated with the corresponding entity.

Once we have the entire sentence correctly noted, we can either accept this sentence by pressing the green button or reject this sentence by pressing the red button. These actions decide whether or not this phrase should be used to re-train the *ARCANO*'s ML model.

The style used to highlight the named entities was based on Displacy, a spaCy library that visually shows the identified entities.

After annotating a set of 100 sentences, we can re-train our model. In Figure 37 we have the interface that allows us to do it.

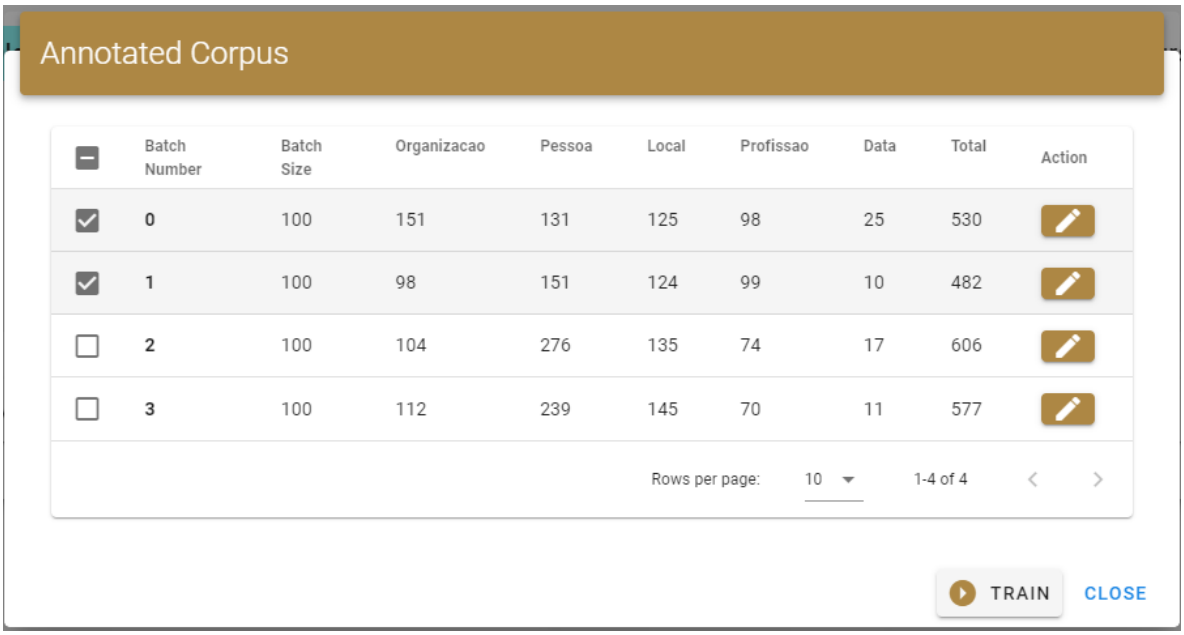


Figure 37: *ARCANO* training interface.

In the table represented in this figure, *ARCANO* lists the annotated data batches available in memory. Here, we can analyze the number of entities annotated per label, as well as the total number of entities in each batch. Finally, it allows us to select the batches that we intend to use to train the model, so it becomes more and more prepared to recognize entities in our context. In this way, the most representative

batches of the dataset should be chosen. With the data batches sectioned, we can proceed with the model's training.

Furthermore, *ARCANO* still has some others features, such as allowing some customization and showing some statistics, as can be seen in Figure 38.

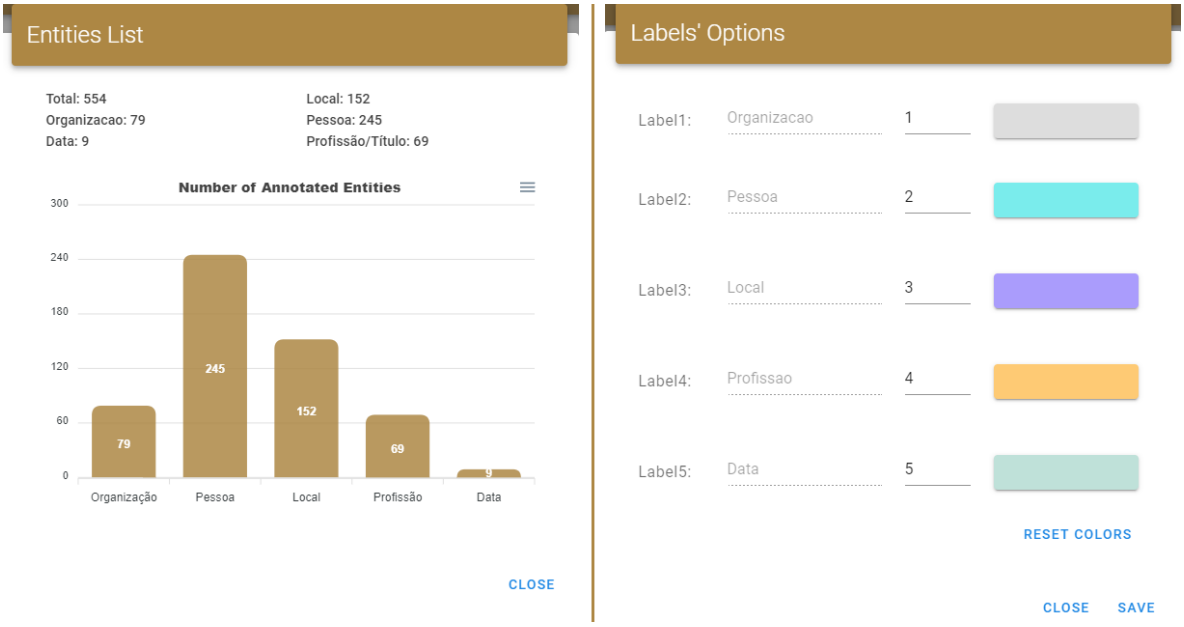


Figure 38: ARCANO statistics (LEFT) and customization (RIGHT).

On the left, we can see the entity labels distribution of the current data batch. The entity labels balance can be valuable indicators for selecting the batches we use for the model training. Finally, on the right, *ARCANO* lets its users customize the label's colours and customize the keyboard keys associated with each label, which are used in the annotation process in the view corresponding to Figure 36.

7

CONCLUSION

The main goal of this thesis was to create a working tool that was able to recognize entities from archival finding aids with enough confidence so the extracted entities could be used in future works. In order to accomplish this, we applied machine learning techniques to a widely used NLP task, NER.

The archival finding aids used in this thesis contain a very specific context, which means that available generic NER models may present lower results than intended. In addition, there is a language barrier, i.e., the amount of Portuguese annotated data available to train this type of model is limited. Despite this, in this work, we demonstrated that by training our own models with annotated archival data, it was possible to obtain satisfactory NER results.

In order to train our own ML models, we started by harvesting all our training data from Portuguese archives' online repositories. For this, we used the OAI-PMH protocol that allowed access to several archival fonds and the harvest of their corresponding finding aids. With the data collected, we proceeded to annotate their named entities, generating annotated corpora composed of 8 different archival datasets, 164478 tokens, and more than 28 thousand annotated entities.

Then we created our NER models with different tools and algorithms to compare them and select the one that fits our needs. The first introduced tool was OpenNLP, which uses the MaxEnt algorithm for the NER task. Then, we moved to a Deep Learning approach, as this method has been showing state-of-art results in several NLP tasks in the last decade. Here, we created a NER model using spaCy, with a CNN approach complemented by pre-trained word embeddings. In addition, we also used Tensorflow, where we implemented a BI-LSTM-CRF model. Finally, our last approach consisted of using the last trend that has revolutionized several NLP tasks, in terms of efficiency and results, Transformers, in our case, a BERT implementation. In this approach, we used large pre-trained models with a Masked Language Modeling objective and fine-tuned them to our NER task using the Huggingface library.

In total, we used 4 different algorithms, training and evaluating several NER models with each of them to compare their results.

In the models' validation, we implemented different context environments in order to test our hypothesis. Here, we wanted to understand the influence that the context proximity of the training data had on the entity recognition itself. In order to do so, we tested two different approaches: training individual NER models for each annotated dataset and creating a single generalized model with data from all the annotated datasets. We concluded that, for our case, it was more beneficial to use a generalized model, as it can learn more features about the language vocabulary, performing better on new context domains.

In the end, we saw that some models performed better than others. In fact, the BI-LSTM-CRF model, trained without pre-trained word embeddings, achieved pretty good results on data with a high degree of similarity with its training data. However, on unseen corpora, it performed poorly. As for BERT models, it is not a coincidence that the attention mechanism is being adopted in most NLP tasks, marking an incredible advance on NLP in general. In our experiments, BERT models also showed some improvements compared to the other tools, mainly when applied to new contexts domains, which is crucial in the NER task. We want our models to be able to apply their knowledge in textual documents that they did not see during their training. In fact, the efficient use of the GPUs' parallel computational power has allowed creating bigger and more capable language models providing the machine with a higher knowledge of the human language.

Overall, our model results were pretty satisfactory, ranging between 86.66% and 94.53% F1-score.

With our experiments, we understood that one important factor that greatly influences the NER model's performance is the amount and quality of the annotated corpora. Generating training data so the models can learn from the annotations can be a tedious and slow process. In order to attenuate this, we developed an intelligent annotator that uses ML to support text annotation, *ARCANO*. This annotator tool uses one of our ML models to try to find named entities in a new archival corpus. In order to test this tool, we annotated a new corpus with 71397 tokens identifying a total of 7399 new named entities. In the end, we concluded that this tool was able to speed up the annotation process.

Finally, in order to promote future work in this NLP field, we developed a Web platform that allowed us to share some of the developed tools to the public, such as the implemented NER models, all the annotated corpora, and *ARCANO* annotator.

7.1 CONTRIBUTIONS

In this dissertations we able to provide several contributions to the scientific community, mainly in NER field applied to the Portuguese language:

- Annotated and published a total of 9 Portuguese archival corpora, which can be used in a wide number of different NLP tasks.
- Demonstrated that we could achieve good NER results in our own data context domain by training or fine-tuning our own NER models.
- Developed a smart annotator tool that speeds up the annotation process in order to encourage the community to generate more Portuguese annotated data, which is something that we lack compared to the English language.
- Developed and deployed a web platform that contains several tools developed in this work, such as the NER models, the intelligent annotator, all the annotated corpora, etc.
- Published a paper on SLATE'21 called "NER in Archival Finding Aids" Cunha and Ramalho (2021a)
- Published a paper on Linked Archives 2021, International Workshop called "Towards Entity Linking, NER in Archival Finding Aids" Cunha and Ramalho (2021b)

7.2 FUTURE WORK

Several approaches could be put into practice in order to improve the obtained results. For example, we could improve our annotated corpora quantity and quality.

In fact, the corpora used to train the models was not annotated by an archival expert. Some of these documents were written centuries ago, which means that, even if the used language is the same, Portuguese, it is possible that we were not able to identify all the named entities due to vocabulary deviations as a consequence of the language evolution over the years. Because of this, the quality of the annotated data can be negatively affected.

On the other hand, archival documents represent a wide variety of contexts, from documents of public administration, private organizations, religious organizations, etc. In order to create a NER model that can perform in all these different domains, the ML model would need a higher amount of annotated data samples, generating

a more representative training environment. This way, it could learn how to classify a broader set of context domains.

As for the actual ML models, we have seen that models with some pre-training achieved better results in unseen data. However, the models used in this work were pre-trained in generic domains. Thus, there is some room for improvement, as we could pre-train our models in the archival context, allowing them to learn more features about the vocabulary used. For example, the BERT model used in this work was trained on Brazilian Portuguese data, which is not ideal. Since a vast amount of Portuguese archival information is available to the public, it could be used to pre-train a new model in an unsupervised fashion. This could provide the model with greater knowledge of archival context vocabulary.

Although there are already many archival documents in digital format, some of them have not yet been transcribed, so they are only available as digitized images. Thus, in order to obtain its content and use it in the model's training, one could use computer vision, more precisely, handwritten character recognition (HWCR) or optical character recognition (OCR).

As for the extracted entities, they could be used for numerous purposes, such as creating a Knowledge Graph. In order to do so, we could perform an entity linking task to gather the entities relationships.

Besides that, entity linking could also be performed to make it possible to browse between different archival documents but related by some entity. It would also be interesting to use the extracted entities to create toponymic and anthroponomic indexes to understand the impact this tool could have on browsing archival finding aids.

BIBLIOGRAPHY

Jay Alammar. The illustrated transformer, 2018. URL http://jalammar.github.io/illustrated-transformer. Accessed in 18-07-2021.

Ali Alvi and Paresh Kharya. Using deepspeed and megatron to train megatron-turing nlg 530b, the world's largest and most powerful generative language model, Oct 2021. URL https://www.microsoft.com/en-us/research/blog/using-deepspeed-and-megatron-to-train-megatron-turing-nlg-530b-the-worlds-largest-and-most-powerful-generative-language-model/. Accessed in 15/10/2021.

ANTT. História do arquivo nacional da torre do tombo, 2017. URL http://antt.dglab.gov.pt/inicio/identificacao-institucional/6-2/. Acedido a 2020-12-15.

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. 2015.

Dominic Balasuriya, Nicky Ringland, Joel Nothman, Tara Murphy, and James R. Curran. Named entity recognition in wikipedia. 2009. doi: 10.3115/1699765.1699767.

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 1996. ISSN 08912017.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. volume 2020-December, 2020.

Paula Carvalho, Hugo Gonçalo Oliveira, Diana Santos, Cláudia Freitas, and Cristina Mota. *Segundo HAREM: Modelo geral, novidades e avaliação*. 01 2008.

Luís Filipe Cunha and José Carlos Ramalho. NER in Archival Finding Aids. In Ricardo Queirós, Mário Pinto, Alberto Simões, Filipe Portela, and Maria João Pereira, editors, *10th Symposium on Languages, Applications and Technologies (SLATE 2021)*, volume 94 of *Open Access Series in Informatics (OASIcs)*, pages 8:1–8:16, Dagstuhl, Germany, 2021a. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-202-0. doi: 10.4230/OASIcs.SLATE.2021.8. URL https://drops.dagstuhl.de/opus/volltexte/2021/14425.

Luís Filipe Cunha and José Carlos Ramalho. Towards entity linking, ner in archival finding aids. In Carla Teixeira Lopes, Cristina Ribeiro, Franco Niccolucci, Irene Rodrigues, and Nuno Freire, editors, *Proceedings of Linked Archives International Workshop 2021 co-located with 25th International Conference on Theory and Practice of Digital Libraries (TPDL 2021)*, pages 22–29, 2021b. URL http://ceur-ws.org/Vol-3019/LinkedArchives_2021_paper_12.pdf.

Luís Filipe Cunha and José Carlos Ramalho. Ner@di, 2021c. URL http://ner.epl.di.uminho.pt/. Accessed in 09-10-2021.

Andrew M. Dai and Quoc V. Le. Semi-supervised sequence learning. volume 2015-January, 2015.

Leon Derczynski. Complementarity, f-score, and nlp evaluation. In *Proceedings of the 10th International Conference on Language Resources and Evaluation, LREC 2016*, 2016. ISBN 9782951740891.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. volume 1, 2019.

DGLAB. Direção geral do livro dos arquivos e das bibliotecas - arquivos, 2021. URL http://dglab.gov.pt/area-arquivos/. Accessed in 24-09-2020.

Express.js. Express - node.js web application framework, n.d. URL https://expressjs.com/. Accessed in 10-04-2021.

Meng Fang, Yuan Li, and Trevor Cohn. Learning how to active learn: A deep reinforcement learning approach. In *EMNLP 2017 - Conference on Empirical*

Methods in Natural Language Processing, Proceedings, 2017. ISBN 9781945626838. doi: 10.18653/v1/d17-1063.

Jorge A. Wagner Filho, Rodrigo Wilkens, Marco Idiart, and Aline Villavicencio. The brwac corpus: A new open resource for brazilian portuguese. 2019.

Cláudia Freitas, Cristina Mota, Diana Santos, Hugo Gonçalo Oliveira, and Paula Carvalho. Second HAREM: Advancing the state of the art of named entity recognition in Portuguese. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May 2010. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2010/pdf/412_Paper.pdf.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning (Adaptive Computation and Machine Learning series)*. The MIT Press, nov 2016. ISBN 0262035618. URL https://www.xarg.org/ref/a/0262035618/.

Alex Graves, Abdel Rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2013. ISBN 9781479903566. doi: 10.1109/ICASSP.2013.6638947.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. volume 1, 2018. doi: 10.18653/v1/p18-1031.

Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging, 2015.

HuggingFace. Summary of the tokenizers, 2021. URL https://huggingface.co/transformers/tokenizer_summary.html. Accessed in 23/08/2021.

Grant S. Ingersoll, Thomas S. Morton, and Andrew L. Farris. *Taming text: how to find, organize, and manipulate it*. Manning, Shelter Island, 2013. ISBN 9781933988382. OCLC: ocn772977853.

Carl Lagoze, Herbert Van de Sompel, Michael Nelson, and Simeon Warner. Open archives initiative - protocol for metadata harvesting - v.2.0, Jun 2002. URL https://www.openarchives.org/OAI/2.0/openarchivesprotocol.htm.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *2016 Conference of the North American Chapter of the Association for Computational Linguistics:*

Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference, 2016. ISBN 9781941643914. doi: 10.18653/v1/n16-1030.

Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Long Papers*, 2016. ISBN 9781510827585. doi: 10.18653/v1/p16-1101.

Christopher Manning. Maxentmodels and discriminative estimation, 2003. URL https://web.stanford.edu/class/cs124/lec/Maximum_Entropy_Classifiers.pdf. Accessed in 22-01-2021.

Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, London, England, 1999.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. 2017.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. 2013.

Mike Morais. Neu 560: Statistical modeling and analysis of neural data: Lecture 8: Informationtheory and maximum entropy, 2018. URL http://pillowlab.princeton.edu/teaching/statneuro2018/slides/notes08_infotheory.pdf. Acedido a 20-10-2020.

Node.js. Node.js v16.4.0 documentation, n.d. URL https://nodejs.org/api/child_process.html. Accessed in 17-03-2021.

Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R Curran. Learning multilingual named entity recognition from wikipedia, Oct 2017. URL https://figshare.com/articles/dataset/Learning_multilingual_named_entity_recognition_from_Wikipedia/5462500/1.

Christopher Olah. Understanding lstm networks, August 2015. URL http://colah.github.io/posts/2015-08-Understanding-LSTMs/. Accessed on March 10, 2021.

Apache OpenNLP. Welcome to apache opennlp, 2017. URL https://opennlp.apache.org/. Accessed in 18-10-2020.

Developers Community OpenNLP. *Apache OpenNLP Developer Documentation.pdf*. 2011. ISBN 2555680587443.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://aclanthology.org/D14-1162.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. volume 1, 2018. doi: 10.18653/v1/n18-1202.

André Ricardo Oliveira Pires. Named entity extraction from portuguese web text. Master's thesis, Faculdade de Engenharia da Universidade do Porto, 2017.

Alexandre Rademaker, Fabricio Chalub, Livy Real, Cláudia Freitas, Eckhard Bick, and Valeria de Paiva. Universal dependencies for portuguese. In *Proceedings of the Fourth International Conference on Dependency Linguistics (Depling)*, pages 197–206, Pisa, Italy, September 2017. URL http://aclweb.org/anthology/W17-6523.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. Learning to generate reviews and discovering sentiment, 2017.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. 2018.

Adwait Ratnaparkhi. Maximum entropy models for natural language ambiguity resolution. 1998.

Ana Maria Rodrigues, Catarina Guimarães, Francisco Barbedo, Glória Santos, Lucília Runa, and Pedro Penteado. Orientações para a descrição arquivística, May 2011. URL https://act.fct.pt/wp-content/uploads/2014/05/ODA-3%C2%AA-vers%C3%A3o.pdf.

Sebastian Ruder. NLP's ImageNet moment has arrived. https://ruder.io/nlp-imagenet/, 2018. Accessed in 07-10-2021.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115, 2015. ISSN 15731405. doi: 10.1007/s11263-015-0816-y.

Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task. 2003. doi: 10.3115/1119176.1119195.

Diana Santos and Nuno Cardoso. A golden resource for named entity recognition in portuguese. In Renata Vieira, Paulo Quaresma, Maria das Graças Volpe Nunes, Nuno J. Mamede, Cláudia Oliveira, and Maria Carmelita Dias, editors, *Computational Processing of the Portuguese Language*, pages 69–79, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-34046-1.

Diana Santos, Nuno Seco, Nuno Cardoso, and Rui Vilela. HAREM: An advanced NER evaluation contest for Portuguese. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy, May 2006. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2006/pdf/59_pdf.pdf.

Satoshi Sekine and Elisabete Ranchhod. *Named Entities: Recognition, classification and use*. John Benjamins Publishing Company, july 2009.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. volume 3, 2016. doi: 10.18653/v1/p16-1162.

Yanyao Shen, Hyokun Yun, Zachary C. Lipton, Yakov Kronrod, and Animashree Anandkumar. Deep active learning for named entity recognition. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018.

Fábio Souza, Rodrigo Nogueira, and Roberto Lotufo. BERTimbau: pretrained BERT models for Brazilian Portuguese. In *9th Brazilian Conference on Intelligent Systems, BRACIS, Rio Grande do Sul, Brazil, October 20-23 (to appear)*, 2020.

spaCy. Annotation specifications · spacy api documentation, a. URL https://spacy.io/api/annotation#biluo.

spaCy. spacy 101: Everything you need to know · spacy usage documentation, b. URL https://spacy.io/usage/spacy-101. Accessed in 07-01-2021.

spaCy. Training spacy's statistical models · spacy usage documentation, c. URL https://spacy.io/usage/training. Accessed in 14-01-2021.

spaCy. Model architecture, 2017. URL https://spacy.io/models. Acedido a 14-01-2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. volume 2017-December, 2017.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27, 2015. doi: 10.1109/ICCV.2015.11.

A

SUPPORT MATERIAL; LISTINGS

A.1 "O SÉCULO" NEWSPAPER ARCHIVAL FOND IN XML FORMAT.

```
<ead
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="urn:isbn:1-931666-22-9">
    <eadheader>
        <eadid url="https://digitarq.arquivos.pt/details?id=1009215"/>
        <filedesc>
            <titlestmt>
                <titleproper/>
            </titlestmt>
        </filedesc>
        <profiledesc>
            <descrules>ISAD(G): Norma Geral Internacional de Descrição Arquivística: adoptada
    pelo Comité de Normas de Descrição, Estocolmo: Suécia, 19-22 de Setembro de 1999. Conselho
    Internacional de Arquivos; Trad. Grupo de Trabalho para a Normalização da Descrição em
    Arquivo. 2ª ed. Lisboa: IAN/TT, 2004. ISBN: 972-8107-69-2. Também disponível a partir de
    http://www.iantt.pt NP 405-1. 1994, Informação e documentação - Referências bibliográficas:
     documentos impressos. Lisboa: IPQ. 49 p. Elaborada por Comissão Técnica para Informação e
    Documentação / Instituto da Biblioteca Nacional e do Livro. Ed. 1995. Norma harmonizada com
     a ISO 690 (1987), termo de homologação DR, III Série, nº 128 de 1994-6-3 NP 405-4. 2002,
    Informação e documentação - Referências bibliográficas: documentos electrónicos. Lisboa:
    IPQ. 26 p. Elaborada por Comissão Técnica para Informação e Documentação / Instituto da
    Biblioteca Nacional e do Livro. Ed. 2003. Norma harmonizada com a ISO 690-2 (1997), termo
    de homologação DR, III Série, nº 143 de 2002-7-9 PORTUGAL. Instituto dos Arquivos Nacionais
     / Torre do Tombo - Orientações para a descrição arquivística. Grupo de Trabalho para a
    Normalização da Descrição em Arquivo. 1ª versão. Lisboa: IAN/TT, 2006. 124 p..ISBN972
    -8107-88-9. Elaboradas no âmbito do Programa para a Normalização da Descrição em Arquivo.</
    descrules>
        </profiledesc>
    </eadheader>
    <archdesc level="otherlevel" otherlevel="F">
```

```
18          <did>
19              <langmaterial>Português</langmaterial>
20              <physdesc>
21                  <dimensions>c. 44.000 u.i, c. de 2500 m.l.; papel, filme</dimensions>
22              </physdesc>
23              <repository>Arquivo Nacional da Torre do Tombo</repository>
24              <unitdate label="UnitDates" type="inclusive" certainty="False/False" normal="
    1880/1979">ca. 1880/ca. 1979</unitdate>
25              <unitid identifier="1009215" countrycode="PT" repositorycode="PT-TT">PT/TT/EPJS</
    unitid>
26              <unittitle type="Atribuído">Empresa Pública Jornal O Século</unittitle>
27          </did>
28
29          ...
30
31          <dsc>
32              <c level="otherlevel" otherlevel="SC">
33                  <did>
34                      <unitid identifier="4490062" countrycode="PT" repositorycode="PT-TT">PT/TT/
    EPJS/A</unitid>
35                      <unittitle type="Formal">Arquivo da Redação</unittitle>
36                  </did>
37                  <relatedmaterial>
38                      <ref xlink:role="parent">1009215</ref>
39                  </relatedmaterial>
40                  <odd>
41                      <p>Registo migrado a partir do sistema CALM em 2008-12-27. CALM:Autor:
    Cribeiro</p>
42                  </odd>
43
44          ...
45
46      <c level="otherlevel" otherlevel="UI">
47          <did>
48              <langmaterial>Português</langmaterial>
49              <physdesc>
50                  <dimensions>1 mç. (143 f.); papel</dimensions>
51              </physdesc>
52              <physloc>Empresa Pública Jornal O Século, Cortes de Censura de 'O Século', cx. 191,
    mç. 242</physloc>
53              <repository>Arquivo Nacional da Torre do Tombo</repository>
54              <unitdate label="UnitDates" type="inclusive" certainty="True/True" normal="
    1959-12-01/1960-02-29">1959-12-01/1960-02-29</unitdate>
55              <unitid identifier="4490285" countrycode="PT" repositorycode="PT-TT">PT/TT/EPJS/A
    /2/242</unitid>
56              <unittitle type="Formal">Cortes de Censura de 'O Século': maço 242</unittitle>
57          </did>
```

```
58          <relatedmaterial>
59              <ref xlink:role="parent">4490094</ref>
60          </relatedmaterial>
61          <note>
62              <p>Nota ao elemento de informação “Dimensão e suporte”: Na medida em que a maioria
        dos recortes das provas tipográficas apresenta notícias truncadas e fragmentadas,
        geralmente restritas aos trechos censurados, não foi possível determinar a quantidade exata
         de documentos existentes, razão pela qual fica indicado apenas o total de folhas.</p>
63          </note>
64          <odd>
65              <p>Registo migrado a partir do sistema CALM em 2008-12-27. CALM:Cota Actual:Empresa
        Pública Jornal ’O Século’, Cortes de Censura de ?O Século?, cx. 191, mç. 242 e mç. 243CALM:
        Dimensão e Suporte:1 cx.</p>
66          </odd>
67          <scopecontent>
68              <p>Contém recortes de provas tipográficas de notícias que foram alteradas ou
        suprimidas, previamente à respetiva edição do jornal, por três instâncias da Censura, cujos
         carimbos constam habitualmente de cada folha onde foram colados os mencionados recortes, a
         saber: "O Século - Redação - Serviço de Censura", "Serviços de Censura - Comissão de
        Lisboa" e "Serviços de Censura (sede)".
69
70                ...
71
72              Outros assuntos de relevo, embora mais esparsos, dizem respeito à discussão de
        problemas agrícolas na Assembleia da República (15 de dezembro de 1959) e ao apoio à
        candidatura de Aquilino Ribeiro para o Prémio Nobel (2 de fevereiro de 1960). Há também
        atualização sobre julgamentos de presos políticos ("No Plenário Criminal: dez condenações a
         pena maior e três em prisão correcional suspensas por três anos" - réus acusados de
        atividades subversivas, do dia 22 de dezembro de 1959), mas merecendo grande destaque a
        fuga de Álvaro Cunhal e outros companheiros, numa notícia que é repetida ("Evadiram-se dez
        presos políticos da fortaleza de Peniche", dos dias 5 e 6 de janeiro de 1960). Finalmente,
        é comunicada alteração no comando da Polícia Internacional e de Defesa do Estado ("O sr.
        capitão Neves Graça deixa a direção da PIDE", do dia 25 de fevereiro de 1960).</p>
73          </scopecontent>
74          <dao xlink:linktype="simple" xlink:href="https://digitarq.arquivos.pt/vault/?id=THUMB/
        F2C4630AAAFFC6B15B2283B3A835AAE0&a=false"/>
75      </c>
76                  </c>
77              </c>
78          </dsc>
79      </archdesc>
80  </ead>
```

Listing A.1: "O Século" newspaper archival fond in XML format.

A.2 PARÓQUIA DO CURRAL DAS FREIRAS ARCHIVAL FOND IN XML FORMAT

```
<ead xmlns="urn:isbn:1-931666-22-9" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xlink="
    http://www.w3.org/1999/xlink">
  <eadheader>
    <eadid url="https://arquivo-abm.madeira.gov.pt/details?id=807" identifier="PT/ABM/PCML02"
    countrycode="PT" mainagencycode="PT-ABM">PT/ABM/PCML02</eadid>
    <filedesc>
      <titlestmt>
        <ead:titleproper xmlns:ead="urn:isbn:1-931666-22-9">Paróquia do Curral das Freiras</ead:
    titleproper>
      </titlestmt>
    </filedesc>
  </eadheader>
  <archdesc level="collection"><did><langmaterial><language>por
</language></langmaterial><physdesc><dimensions>316 u.i. (312 liv., 4 mf.)</dimensions></
    physdesc><repository>Arquivo Regional e Biblioteca Pública da Madeira</repository><unitdate
    >1813/1911</unitdate><unittitle>Paróquia do Curral das Freiras</unittitle><unitid>PT/ABM/
    PCML02<extptr xlink:type="simple" xlink:href="https://arquivo-abm.madeira.gov.pt/details?id
    =807"/>
        </unitid></did><scopecontent><p>Livros de registo de batismos, casamentos e óbitos.</p>
    </scopecontent>
<dsc>
<c level="series">
      <did>
         <physdesc>
            <dimensions>105 u.i. (104 liv., 1 mf.)</dimensions>
         </physdesc>
         <repository>Arquivo Regional e Biblioteca Pública da Madeira</repository>
         <unitdate>1813/1911</unitdate>
         <unittitle>Registo de casamentos</unittitle>
         <unitid>PT/ABM/PCML02/002<extptr xlink:type="simple" xlink:href="https://arquivo-abm.
    madeira.gov.pt/details?id=4444"/>
         </unitid>
      </did>

<c level="file">
      <did>
         <langmaterial>
            <language>por
</language>
         </langmaterial>
         <physdesc>
            <dimensions>1 liv.: 16 f. num. e rub.</dimensions>
         </physdesc>
         <repository>Arquivo Regional e Biblioteca Pública da Madeira</repository>
         <unitdate>1882/1882</unitdate>
```

```
37          <unittitle>Livro de registo de casamentos do Curral das Freiras do ano de 1882</
      unittitle>
38          <unitid>PT/ABM/PCML02/002/00024<extptr xlink:type="simple" xlink:href="https://arquivo-
      abm.madeira.gov.pt/details?id=1010"/>
39          </unitid>
40        </did>
41
42 <c level="item">
43        <did>
44           <langmaterial>
45              <language>por
46 </language>
47           </langmaterial>
48           <physdesc>
49              <dimensions>1 f.</dimensions>
50           </physdesc>
51           <repository>Arquivo Regional e Biblioteca Pública da Madeira</repository>
52           <unitdate>1882/1882</unitdate>
53           <unittitle>Registo de casamento n.º 1: Jesuíno Rodrigues c.c. Justina de Jesus</
      unittitle>
54           <unitid>PT/ABM/PCML02/002/00024/000001<extptr xlink:type="simple" xlink:href="https://
      arquivo-abm.madeira.gov.pt/details?id=502239"/>
55           </unitid>
56        </did>
57     </c>
58 <c level="item">
59        <did>
60           <langmaterial>
61              <language>por
62 </language>
63           </langmaterial>
64           <physdesc>
65              <dimensions>1 f.</dimensions>
66           </physdesc>
67           <repository>Arquivo Regional e Biblioteca Pública da Madeira</repository>
68           <unitdate>1882/1882</unitdate>
69           <unittitle>Registo de casamento n.º 2: Manuel Rodrigues c.c. Antónia de Jesus</
      unittitle>
70           <unitid>PT/ABM/PCML02/002/00024/000002<extptr xlink:type="simple" xlink:href="https://
      arquivo-abm.madeira.gov.pt/details?id=502240"/>
71           </unitid>
72        </did>
73     </c>
74 <c level="item">
75        <did>
76           <langmaterial>
77              <language>por
```

```
78 </language>
79          </langmaterial>
80          <physdesc>
81             <dimensions>1 f.</dimensions>
82          </physdesc>
83          <repository>Arquivo Regional e Biblioteca Pública da Madeira</repository>
84          <unitdate>1882/1882</unitdate>
85          <unittitle>Registo de casamento n.º 3: António de Canha c.c. Victorina de Jesus</
      unittitle>
86          <unitid>PT/ABM/PCML02/002/00024/000003<extptr xlink:type="simple" xlink:href="https://
      arquivo-abm.madeira.gov.pt/details?id=502241"/>
87          </unitid>
88        </did>
89    </c>
90 ...
```

Listing A.2: *Paróquia do Curral das Freiras* archival fond.

A.3 ARCANO SEQUENCE DIAGRAM.

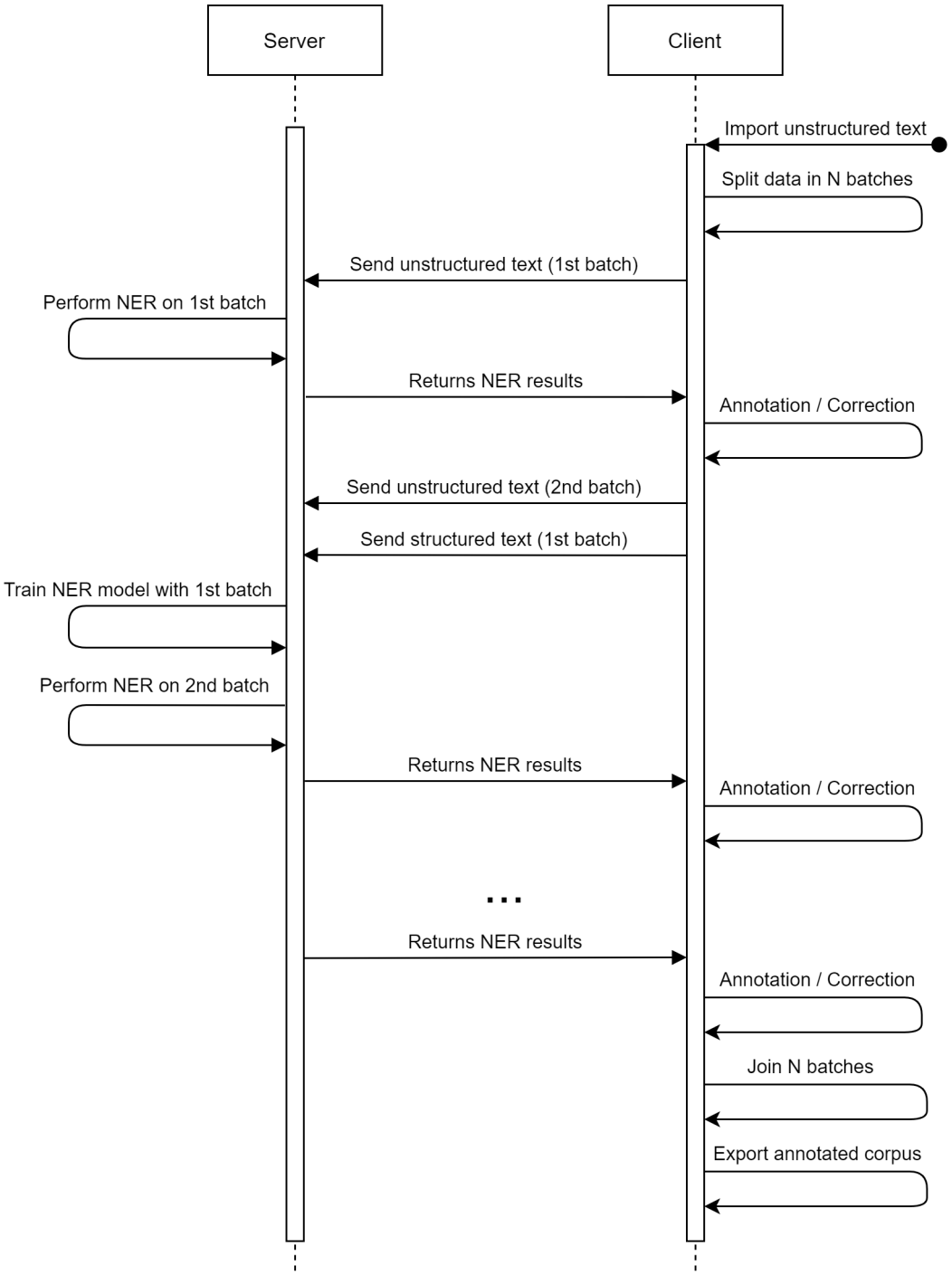


Figure 39: ARCANO Sequence Diagram.