FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Named entity extraction from Portuguese web text

André Ricardo Oliveira Pires

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Sérgio Sobral Nunes

Co-Supervisor: José Luís da Silva Devezas

June 27, 2017

Named entity extraction from Portuguese web text

André Ricardo Oliveira Pires

Mestrado Integrado em Engenharia Informática e Computação

June 27, 2017

Abstract

In the context of Natural Language Processing, the Named Entity Recognition (NER) task focuses on extracting and classifying named entities from free text, such as news. Entity detection enables more complex tasks, such as Relation Extraction or Entity-Oriented Search, for instance the ANT search engine. There are some NER tools focused on the Portuguese language, such as Palavras or NERP-CRF, but their F-measure is below the F-measure obtained by other available tools, for instance based on an annotated English corpus, trained with Stanford CoreNLP or with OpenNLP.

ANT is an entity-oriented search engine for the University of Porto (U.Porto). This search system indexes the information available in SIGARRA, U.Porto's information system. Currently it uses handcrafted selectors to extract entities, based on XPath or CSS, which are dependent on the structure of the page. Furthermore, it does not work on free text, specially on SIGARRA's news. Using a machine learning method allows for the automation of the extraction task, making it scalable, structure independent and lowering the required work effort and consumed time.

In this dissertation, I evaluate existing NER tools in order to select the best approach and configuration for the Portuguese language, particularly in the domain of SIGARRA's news. The evaluation was done based on two datasets, the HAREM collection, and a manually annotated subset of SIGARRA's news, which are used to assess the tools' performance using precision, recall and F-measure. Expanding the existing knowledge base will help index SIGARRA pages by providing a richer entity-oriented search experience with new information, as well as a better ranking scheme based on the additional context made available to the search engine. The scientific community also benefits from this work, with several detailed manuals resulting of the systematic analysis of available tools, in particular for the Portuguese language.

First, I carried an out-of-the-box performance analysis of some selected tools (Stanford CoreNLP, OpenNLP, spaCy and NLTK) with the HAREM dataset, obtaining the best results for Stanford CoreNLP (56.10%), followed by OpenNLP. Then, I performed a hyperparamenter study in order to select the best configuration for each tool, having achieved better-than-default results in each tool, particularly for NLTK's Maximum Entropy classifier, increasing the F-measure from 1.11% to 35.24%. I also prepared a novel Portuguese corpus, called SIGARRA News Corpus, composed of 905 annotated news, with 12644 entity annotations. Finally, using the best configuration, I repeated the training process with the SIGARRA News Corpus, having achieved F-measures as high as 86.86%, for Stanford CoreNLP. Furthermore, given this was also the out-of-the box winner, it leads me to conclude that Stanford CoreNLP is the best option for this particular context.

Resumo

No contexto da área do Processamento de Linguagem Natural, a tarefa de Reconhecimento de Entidades Mencionadas (REM) foca-se na extração e classificação de entidades mencionadas de texto livre, como notícias. A deteção de entidades permite tarefas mais complexas, como Extração de Relações ou Pesquisa Orientada a Entidades, como no motor de pesquisa do ANT. Há algumas ferramentas de REM focadas na língua portuguesa, tais como o Palavras ou o NERP-CRF, mas o seu *F-measure* está abaixo do obtido usando outras ferramentas disponíveis, por exemplo com base num *corpus* inglês anotado, treinado com o *Stanford CoreNLP* ou com o *OpenNLP*.

O ANT é um motor de busca orientado a entidades da Universidade do Porto (U.Porto). Este sistema de pesquisa indexa as informação disponível no SIGARRA, o sistema de informação da U.Porto. Atualmente usa seletores construídos manualmente para extrair entidades, baseados em XPath ou CSS, que são dependentes da estrutura da página. Além disso, não funcionam em texto livre, especialmente nas notícias do SIGARRA. Um método baseado em aprendizagem computacional permite a automatização da tarefa de extração, tornando-a escalável, independente da estrutura, diminuindo o esforço de trabalho exigido e o tempo consumido.

Nesta dissertação, eu avaliei ferramentas de REM existentes para selecionar a melhor abordagem e configuração a utilizar em relação à língua portuguesa, particularmente no domínio das notícias do SIGARRA. A avaliação foi feita com base em dois conjuntos de dados, a coleção HAREM, e um subconjunto manualmente anotado de notícias do SIGARRA, que foram usados para calcular o desempenho das ferramentas usando *precision*, *recall* e *F-measure*. A expansão da base de conhecimento existente ajudará a indexar as páginas do SIGARRA proporcionando uma experiência de pesquisa orientada a entidades mais rica e com nova informação, bem como um melhor esquema de classificação baseado no contexto adicional disponibilizado ao motor de busca. A comunidade científica também beneficia deste trabalho, com múltiplos manuais detalhados resultantes da análise sistemática das ferramentas, em particular para a língua Portuguesa.

Primeiramente, eu analisei a performance base de algumas ferramentas selecionadas (Stanford CoreNLP, OpenNLP, spaCy e NLTK) com a coleção HAREM, obtendo os melhores resultados com o Stanford CoreNLP (56.10%), seguido do OpenNLP. De seguida, efetuei um estudo aos hiperparâmetros, de modo a selecionar a melhor configuração para cada ferramenta, conseguindo alcançar melhorias em cada ferramenta, principalmente no classificador de Entropia Máxima do NLTK, em que houve melhorias no *F-measure* de 1.11% para 35.24%. Preparei, também, um *corpus* Português único, denominado de SIGARRA News Corpus, composto por 905 notícias anotadas, com 12644 anotações de entidades. Finalmente, usando a melhor configuração, eu repeti o treino com o *dataset* das notícias do SIGARRA, tendo obtido *F-measures* de 86.86%, para o Stanford CoreNLP. Além disso, dado que este foi também o melhor classificador com as configurações base, posso concluir que o Stanford CoreNLP é a melhor opção.

Acknowledgements

I would like to give special thanks to my supervisors, Sérgio Nunes and José Devezas, for their continuous support throughout all this process and for reviewing my work. Also, I would like to thank the ANT team, specially José Devezas and Yulia Karimova, for helping me prepare SIGARRA News Corpus and publish my work.

To my parents, who always believed in me and gave me the means to pursue my dreams. Finally, to Catarina Mendes, Catarina Pires and all of my closest friends for making my life worth living.

André Ricardo Oliveira Pires

Contents

CONTENTS

List of Figures

LIST OF FIGURES

List of Tables

List of Listings

Abbreviations

NE	Named Entity
NER	Named Entity Recognition
NLP	Natural Language Processing
IE	Information Extraction
ML	Machine Learning
KB	Knowledge Base
POS	Part of Speech
CRF	Conditional Random Fields
HMM	Hidden Markov Model
MEMM	Maximum Entropy Markov Model
ME	Maximum Entropy
SVM	Support Vector Machine
OBIE	Ontology-Based Information Extraction
MUC	Message Understanding Conference
CoNLL	Conference on Natural Language Learning
ACE	Automatic Content Extraction
HAREM	HAREM Avaliação de sistemas de Reconhecimento de Entidades Mencionadas
U.Porto	University of Porto

Chapter 1

Introduction

This chapter contextualizes the field of Named Entity Recognition (NER) regarding scientific production, commercial applications and the applications in the ANT project. The main motivation for this work consists in expanding beyond structure dependent information extraction in the web for knowledge base construction, in order to support entity-oriented search. I close the chapter with the structure for the remaining document.

1.1 Context

With the vastness of information made available in the web, there is a need for a method of filtering the relevant data and presenting it to the readers. Most of the Internet's information is not available in structured form. Natural Language Processing (NLP) is a field of Artificial Intelligence concerned with making human language understandable to computers. This enables obtaining structured information in a way that it can be indexed and used by a machine for knowledge-driven tasks, such as question answering.

Information Extraction (IE) and NLP are intertwined. IE's main task is to extract relevant data from documents, which can either be unstructured or structured texts. One of the main subtasks of information extraction is Named Entity Recognition. The concept of "named entity" was first introduced in 1996, at the Message Understanding Conference - 6 (MUC-6), by Grishman and Sundheim [GS96]. At that time, the concept of named entity referred to names of people, locations, organizations and numeric expressions such as money and dates. Over the years, there have been multiple redefinitions of named entities, mainly because there was a need to include other entities for specific purposes, for example DNA sequences for the biomedical area.

NER provides a means for further, more complex, tasks in information extraction. One of those tasks is Relationship Extraction, whose main objective is to identify semantic links between the entities identified in a sentence.

The launch of Google's Knowledge Graph, in May 2012 [Sin12], or even Facebook's Open Graph[1], boosted the focus on entity-oriented search. The ANT[2] project represents a similar effort. This project is being developed at InfoLab, at INESC TEC and Faculty of Engineering of the University of Porto. Its main objective is to index and make available entities in the domain of the University of Porto, such as students, staff or departments. This dissertation's main focus will be on information indexed by ANT and based on SIGARRA, the information system for the University of Porto. In particular, experimenting with the HAREM golden collection and SIGARRA News Corpus, with multiple tools, to identify the best tool for NER for Portuguese.

1.2 Motivation and goals

Currently, entity extraction in the ANT project is made using handcrafted rules, with selectors such as XPath and CSS. This means that ANT's developers have to look into all of SIGARRA's web pages DOM to find out which selector rule to use for the extraction. This is extremely time consuming and requires a huge manual effort. Furthermore, given their strong structure dependence, the extraction rules only work for specific pages, making it non-generalizable . In addition, if the page's structure changes, the extraction rules have to change accordingly, which means the process of looking into the pages' DOM has to begin again. This also leads to not being able to extract entities from free text, such as SIGARRA's news.

The main goal of this dissertation is to automate the extraction process, on free text, making it less time consuming and greatly diminishing the amount of work required. Moreover, the new extraction method will be less structure dependent, which will potentially increase its reusability beyond the SIGARRA domain.

This will provide context to the ANT search engine, enabling it to improve entity ranking as well as to provide contextual information about the entities and their connections. Given SIGARRA is a Portuguese information system, the extraction methods will focus on this language.

There is a lot of Information Extraction software available, which already has significant performance in this field. A study of some of the state of the art approaches will be performed in order to decide on the best approaches to process the textual content available through ANT.

1.3 Document structure

This document is structured in seven additional chapters. In Chapter 2, I present the state of the art for Named Entity Recognition and Relation Extraction, covering the extraction techniques and evaluation methods. In Chapter 3, I describe the HAREM golden collection and the SIGARRA News Corpus I prepared, along with the NLP tools selected I will assess. The transformation of the HAREM dataset for the tools is explained in Chapter 4 and in Chapter 5 I present the evaluation method and the obtained results. Next, in Chapter 6, I present the results of a hyperparameter study,

[1] See http://ogp.me/. Accessed: 2017-06-27.
[2] See http://ant.fe.up.pt. Accessed: 2017-06-27.

carried in order to understand their individual impact in the respective classifier. Using the results from the hyperparameter study, I performed NER with the SIGARRA News Corpus; the results are available in Chapter 7. Finally, the conclusions, with the achieved objectives, contributions, such as trained NER models for Portuguese, a novel annotated Portuguese dataset and guidelines for NER for each used tool, future work, are presented in Chapter 8.

Chapter 2

Named Entity Recognition and Relation Extraction

Named Entity Recognition (NER) and Relation Extraction (RE) can be performed using many different approaches. This chapter sums up the methods of extraction and the current evaluation techniques. I will start by briefly explaining what are NER and RE, along with the main extraction implementations. NER methods can be divided in two distinct approaches. The first approach hinges on creating a set of hand-coded rules to extract entities and the second approach falls in the category of machine learning systems, which is where a computer estimates the parameters using a set of algorithms.

The machine learning methods can be divided into three categories based on the data required as input by the training algorithm. The learning algorithms can either be supervised, where the system needs an already annotated corpus, semi-supervised, where the main technique is bootstrapping, that is to say it only requires a small set of marked examples and it can then further extrapolate for unmarked examples, and unsupervised which does not require annotated examples. An ontology-based extraction method is similar to the other methods, but takes into account an ontology to guide the extraction, mainly substituting the usual entity types with a predefined ontology.

2.1 Named Entity Recognition

Named Entity Recognition is a sub-field of Information Extraction (IE). Its main purpose is to identify entities from unstructured text, such as news articles, or semi-structured text, such as Wikipedia articles. This extraction is the first part of a typical information extraction pipeline, supporting further information extraction tasks, such as semantic analysis or relation extraction.

The concept of "entity" varies from approach to approach. It mainly depends on what is considered relevant to extract in each case. As already stated, the first categories of named entities

emerged in MUC-6, where the entities were categorized as persons, locations, organizations, time expressions and numeric expressions. These have been the most common categories extracted in the field. However, some other categories are used in different, more specific, domains. Some examples are biomedical entities, such as drugs or DNA sequences, as well as entities used for military purposes, like vehicles or weapons.

2.2 Relation Extraction

Relation Extraction is also a sub-field of Information Extraction. Its objective is to recognize relations between the entities in a text. The type of relations extracted are usually highly dependent on the domain. Some of the most common relations are in the linguistic domain: hyponyms — a word with a more specific meaning —, for example, *car* and *vehicle*; antonyms — a word with contrary meaning —, for example, *happy* and *unhappy*; and meronyms — a part-whole relationship —, for example, *chapter* and *book*. Other relation types can be used in the film industry, such as stars-in, director or film genres, or the food industry such as type of food, ingredients, quantities, or the social domain, such as parent-of, child-of, and geographical domain, such as capital-of.

The main methods for relation extraction can be divided into similar categories as those used in the named entity recognition domain. That being said, extraction can be done using hand-coded patterns or machine learning methods. While NER extraction methods only focus on extracting entities, RE focuses on extracting the relationships between the entities. Relation extraction often requires previous text analysis, which involves POS tagging, syntactic parsing, NER and feature engineering.

2.3 Extraction methods

In this section, I will present and explain the main methods for extracting entities and relations, which can be divided into hand-coded techniques, machine learning techniques and ontology-based techniques.

2.3.1 Hand-coded techniques

There are two approaches which can be considered hand-coded techniques, namely, rule-based, where patterns are used to extract entities, and dictionary-based, where tokens are matched with a gazetteer to recognize entities.

2.3.1.1 Rule-based

The first approaches to NER systems were based on the extraction of Named Entities (NEs) using grammar rules. These approaches focus on matching words using patterns, such as regular expressions. One of the first works in this field was by Lisa Rau [Rau91], in which she extracts company names from text using a set of manually created rules and heuristics such as capitalization of

words or detection of company suffixes, like "Inc." or "Corp.", and also generating acronyms for already existing company names. An example of a pattern could be a title match, such as "Mr.", followed by one or more capitalized-tokens, indicating that those tokens are probably a NE with the person type. Mikheev et al. [MMG99] named these rules as "sure-fire", as they would most likely perform as expected and be successful, and used it in their system, where a match would only classify the words as *likely* candidates. Later they used a Maximum Entropy model to decide on the definitive NE classification, which is a Machine Learning (ML) technique.

Another example of entity extraction using this technique is PAMPO [RJS+16], which is an entity extractor for the Portuguese language. It extracts entities in two phases: first, using a set of regular expressions to gather common candidate entities with typical expressions such as capitalized words or personal titles ("president" or "deputy"); then, it performs POS tagging using OpenNLP's POS tagging module for Portuguese. In this second phase, using a new set of regular expressions, now on the POS tags instead of the words, it discards some candidates. While it is important to note that PAMPO only extracts named entities and does not perform classification (assign entity classes), it reported an F-measure of 73.6% against the HAREM collection.

This kind of pattern-matching approach can also be used for relation extraction, with patterns like "*person* lives in *location*". For instance, Barrière [Bar16] used a set of regular expressions to model lexico-syntactic patterns. Hearst [Hea92] also used lexico-syntactic patterns to extract hyponym relations. Three starting patterns were defined and later, using a list of terms for which a specific relation is known to hold, more patterns can be created, either by handcoding or by bootstrapping from an existing lexicon or Knowledge Base (KB), using ML techniques. Berland and Charniak [BC99] attempted to extract meronym relations from text, using patterns. They began by identifying two words *building* and *basement* with close proximity from a corpus. From that they extracted patterns, for example, using the possessive as in "building's basement", and other patterns. Although they managed to extract some correct relationships, their overall accuracy was low, at 55%.

Another example, using the relation *(author, title)* for books as use case, Brin [Bri99] developed DIPRE, whose algorithm worked as follows: given a small seed set of *(author, title)* pairs, find its occurrences in the web and recognize patterns where they appear; then, using the extracted patterns, new *(author, title)* pairs can be extracted; this steps can be repeated until some criteria is met. The approach can be extended for other relation types by providing different pairs as a seed, so that the algorithm can find meaningful patterns.

Although hand-coded patterns are not ideal, they can provide acceptable results. Using patterns requires building them for each entity and relation, which is hard to write and hard to maintain. Furthermore, it is infeasible to write all the required patterns since there can be multiple distinct ways of expressing entities and their relations. Finally, hand-coded patterns are usually domain-dependent, meaning that different vocabularies are used in different domains.

2.3.1.2 Dictionary-based

Many approaches rely on an already existing KB to extract entities from other texts. This KB is usually called *gazetteer* [SO06, KT08], which is a dictionary of a collection of entities. The main approach usually consists of matching the words in a text with the gazetteer and, if a match occurs, the word is annotated as an entity. Wikipedia can be used as a KB, because it provides an enormous amount of entities. Gattani et al. [GDL+13] developed a Wikipedia-based approach for NER in social media, where the relevant words from the text were linked to a Wikipedia page. This approach was used to classify and tag tweets.

SIEMÊS [Sar06], a participating system in HAREM [Car06], used similarity rules to obtain soft matching between the entities and a gazetteer. The used gazetteer was REPENTINO [SPC06], a publicly available gazetteer for Portuguese. After identifying possible candidates as NE, the system uses similarity rules to formulate judgments about the possible classes of a NE. So instead of multiple hard-coded rules over the gazetteer, it only has a small set of rules, such as exact matches, partial matches either on the beginning, the end or subsets of the NE candidate, and a check for frequent words in certain subclasses, to score a match in REPENTINO.

Using gazetteers proves to be a simple method for NER. However, the entities recognized are dependent on whether an entity exists in the gazetteer, which means that even very large gazetteers only contain a portion of all used entities. Finally, the performance of the NER system may also be affected by the introduction of new entities.

2.3.2 Machine learning techniques

There are multiple machine learning techniques applied in NER. The most used are probabilistic techniques, such as Hidden Markov Models, Maximum Entropy Markov Models and Conditional Random Fields. Next, I present them, along with some use cases.

Hidden Markov Model (HMM) A HMM is a statistical Markov Model, in which the state is not directly visible. In NER, HMM states are usually a name of an entity class, with an extra state for the current word not being an entity. Each state transition is dependent only on the current state, and represents the probability for the next word to be of a specific category.

As Ponomareva et al. [PRPM07] explain, let $o = \{o_1, o_2, ..., o_n\}$ be a sequence of words from a text with length n. Let S be a set of states in a finite state machine, each associated with a label (categories for entities). Let $s = \{s_1, s_2, ..., s_n\}$ be a sequence of states that correspond to the labels assigned to words in the input sequence o. HMM defines the joint probability of a state given an input sequence to be:

$$P(s,o) = \prod_{i=1}^{n} P(o_i|s_i)P(s_i|s_{i-1}) \tag{2.1}$$

So, in order to train the HMM, the following probabilities have to be set:

1. Initial probabilities $P_0(s_i) = P(s_i|s_0)$ to begin from a state i;

2. Transition probabilities $P(s_i|s_{i-1})$ to pass from a state s_{i-1} to a state s_i;

3. Observation probabilities $P(o_i|s_i)$ of an appearance of a word o_i conditioned on state s_i.

These probabilities are calculated using a training corpus. An example of the use of this approach is Nymble, by Bikel et al. [BMSW97], achieving an F-measure of 93% for English and 90% for Spanish. Another example by Zhou and Su [ZS02], in which the HMM is based on the mutual information independence assumption, instead of the conditional probability independence assumption after Bayes' rule, achieving an F-measure of 96.6%. Both used the MUC-6 dataset, and the latter also used the MUC-7 dataset.

Maximum Entropy Markov Model (MEMM) MEMM works in the same way as HMM. However, it no longer assumes that features are independent, which means there can be correlated features. While HMM is a generative model, MEMM is a discriminative model. In other words, HMM learns the joint probability distribution $P(s,o)$, while MEMM learns the conditional probability distribution $P(s|o)$.

The probability of the state sequence given the observation can be computed as:

$$P(s|o) = \prod_{i=1}^{n} P(s_i|s_{i-1}, o_i) \tag{2.2}$$

MENE [BSAG98] is an example of a maximum entropy framework for NER. It participated in the Message Understanding Conference 7 (MUC-7), obtaining an F-measure of 92.20%. Also, inspired by MENE's results, Carvalho [Car07] developed a maximum entropy framework for NER for the Portuguese language, using HAREM as training set, obtaining an F-measure of 42.48% by HAREM's evaluation standards.

Conditional Random Fields (CRFs) CRFs [LMP01] work similarly to HMMs but are not constrained with local features. This means that CRFs are able to deal with a much larger set of features. Furthermore, while HMM's probabilities must satisfy certain constraints, in CRFs there are no restrictions. As Teixeira et al. [TSO11] states, according to Lafferty et al. [LMP01] and McCallum and Li [ML03], let $o = \{o_1, o_2, ..., o_n\}$ be a sequence of words from a text with length n. Let S be a set of states in a finite state machine, each associated with a label (categories for entities). Let $s = \{s_1, s_2, ..., s_n\}$ be a sequence of states that correspond to the labels assigned to words in the input sequence o. CRFs define the conditional probability of a state given an input sequence to be:

$$P(s|o) = \frac{1}{Z_o} \exp\left(\sum_{i=1}^{n}\sum_{j=1}^{m} \lambda_j f_j(s_{i-1}, s_i, o, i)\right) \tag{2.3}$$

where Z_o is a normalization factor of all state sequences, $f_j(s_{i-1}, s_i, o, i)$ is one of the m functions that describes a feature, and λ_j is a learnt weight for each feature function.

Kazama and Torisawa [KT07] developed a CRF NER, using Wikipedia as external knowledge, to help classify entities and thus improve the accuracy of their NER model. They performed

entity linking to a Wikipedia article, and extracted a category label from the first sentence of a Wikipedia article to use it as a feature. Another use of a CRF for NER was proposed by Amaral and Vieira [dAV13], called NERP-CRF. This algorithm was trained using the HAREM corpora. First, sentence segmentation and POS tagging was performed, so that the complexity in applying the CRF method was decreased. Multiple features for the CRF algorithm were used, such as words around the NE and the capitalization of the NE. Using the HAREM corpora to evaluate, it scored 57.92% and 48.43% for F-measure, respectively.

In Teixeira et al. [TSO11], they present a bootstrapping method using CRF. Firstly, using a dictionary-based approach, a set of non-annotated news items is annotated. In this phase, only entities with two or more words are considered. The sentences, where all the capitalized words are annotated, are used as a seed corpus to infer a CRF model. This model, is used to annotate the same corpus used in the first stage, resulting in an increase of annotated sentences. These are then used to infer a new CRF model. This cycle is repeated until the model stabilizes.

Support Vector Machines (SVMs) SVMs [CV95], known as a large margin technique, defines an optimal hyperplane which separates categories. The SVM algorithm is based on finding the best hyperplane which gives the maximum margin between the decision border and the closest objects from the classes. Although, originally, an SVM can only deal with linearly-separable tasks, by using kernel functions it can transform non-linearly separable data, in its original space, into a higher dimension space, where the data becomes linearly separable. SVMs can only deal with binary classification. However, it can be used in non-binary classification tasks (such as NER), by using methods like the *one-against-one* approach [DB96], in which multiple classifiers are constructed and each one deals with two different classes. Afterwards, using a voting strategy, it chooses the best category for the present object.

Mididiú and Duarte [MD07] is an example of SVM use for the Portuguese language, achieving an F-measure of 88.11%, although not with the HAREM corpus. Other examples are by Ekbal and Bandyopadhyay [EB10] for Bengali and Hindi, using lexical patterns as features for the SVM, achieving an F-measure of 84.15%, and by Asahara and Matsumoto [AM03] for the Japanese language, proposing a character-based chunking method, achieving an F-measure of 87.2%.

2.3.3 Ontology-based

The concept of Ontology-Based Information Extraction (OBIE) is relatively new. Wimalasuriya and Dou [WD10] defined OBIE, in 2010, as:

> "An ontology-based information extraction system: a system that processes unstructured or semi-structured natural language text through a mechanism guided by ontologies to extract certain types of information and presents the output using ontologies."

The main goal of OBIE systems is to identify concepts, properties or relations expressed in ontologies. Being NER a sub-task of IE, some techniques may also fall in this category. For

instance, Pandolfo et al. [PPA16] developed a framework for automatic population of ontology-based digital libraries. They used some of OpenNLP's [Fou] modules to perform NER using an ontology as input so that the entities have the same name of the ontology classes considered for the automatic ontology population. Their triple extractor module can extract triples from text and add them to their knowledge base. Each triple represents a relation between the extracted entities, based on a gazetteer of verbs.

This approach is particularly useful when dealing with specific domains, mainly because the most common entity categories may not be sufficient in such cases. Yasavur et al. [YALR13] defined an ontology-based approach for the domain of behaviour and lifestyle change, creating a behavioural health ontology to model world knowledge. They also used WordNet to extend the ontology for NER purposes. Dictionary-based approaches to OBIE consist in having a gazetteer whose entities follow a particular ontology. An example of this approach is Saggion et al. [SFMB07], which adapted GATE's ANNIE module using its own gazetteer containing countries and regions gathered from multiple sources.

2.4 Evaluation and datasets

Evaluating NER systems allows us to know if new systems are evolving in a positive way, getting higher precision and recall. There is a need for having systematic evaluations, so that all NER systems have the same standards when evaluating their performance.

There are multiple proposed techniques to rank NER systems based on their ability to annotate text correctly. These techniques were defined and used in conferences, such as MUC, CoNLL, ACE or HAREM. There are lots of conferences in this area, I will only talk about these four, which are the most relevant for this dissertation. These conferences not only differ in their evaluation techniques but also on what is considered an entity, so they have different entity classes. This makes it hard to compare different tools which participated in different conferences.

The evaluation task's main objective is to compare the output of the NER system to an actual correct output by human linguistics (gold standard). The most common metrics used to rate classification tasks are:

- *Precision*: the ratio of correct answers (True Positives) among the answers produced (Positives). This means checking if the answers marked as positive are truly positive.

$$precision = \frac{TP}{TP + FP} \tag{2.4}$$

- *Recall*: the ratio of correct answers (True Positives) among the total possible correct answers (True Positives and False Negatives). This means checking if all the positives are marked.

$$recall = \frac{TP}{TP + FN} \tag{2.5}$$

- *F-Measure*: the harmonic mean of precision and recall.

$$F_1\text{-}measure = \frac{2 * precision * recall}{precision + recall} \tag{2.6}$$

The four different classes of classification results are [Kon12]:

- *True Positive (TP)*: predicted value was positive and the actual value was positive.
- *True Negative (TN)*: predicted value was negative and the actual value was negative.
- *False Positive (FP)*: predicted value was positive and the actual value was negative.
- *False Negative (FN)*: predicted value was negative and the actual value was positive.

As an example, let's use the following correctly annotated text, marked up according to MUC guidelines, where `Porto` is identified as a location, `James Jr` as a person, `Acme` as an organization, `Richard Doe` as a person and `Software Inc` as an organization:

```
<ENAMEX TYPE="LOCATION">Porto</ENAMEX> received multiple personalities, such as <
    ENAMEX TYPE="PERSON">James Jr</ENAMEX>, CEO of <ENAMEX TYPE="ORGANIZATION">Acme
    </ENAMEX>, and <ENAMEX TYPE="PERSON">Richard Doe</ENAMEX>, CTO of <ENAMEX TYPE=
    "ORGANIZATION">Software Inc</ENAMEX>.
```

Listing 2.1: Example of a correctly annotated text.

Now, imagine the following output produced by a NER system, where `Porto` was identified as a person, `James Jr` as a person, `Acme` was not identified, `Richard` was identified as a person, although with the wrong boundary, `CTO` as an organization and `Software Inc` as an organization:

```
<ENAMEX TYPE="PERSON">Porto</ENAMEX> received multiple personalities, such as <
    ENAMEX TYPE="PERSON">James Jr</ENAMEX>, CEO of Acme, and <ENAMEX TYPE="PERSON">
    Richard</ENAMEX> Doe, <ENAMEX TYPE="ORGANIZATION">CTO</ENAMEX> of <ENAMEX TYPE=
    "ORGANIZATION">Software Inc</ENAMEX>.
```

Listing 2.2: Example of an annotated output.

As we can see in Table 2.1, the system outputted four different errors and three different correct answers. The main objective now is to figure out what score to give to this output. The NER conferences have different ways of dealing with this, which will be detailed in the following sections.

2.4.1 Message Understanding Conference

The Message Understanding Conference 6 (MUC-6) [GS96] was the first conference to introduce the NER task. Consequently this corresponded to the first evaluation technique definition for this

Table 2.1: Types of named entity recognition errors.

Correct solution	System output	Error
<Location> Porto </Location>	<Person> Porto </Person>	The system recognized an entity but assigned it the wrong label.
<Organization> Acme </Organization>	Acme	The system did not recognize the entity.
<Person> Richard Doe </Person>	<Person> Richard </Person>	The system recognized an entity but assigned it the wrong boundaries.
CTO	<Organization> CTO </Organization>	The system hypothesized an entity where there is none.

task. Its evaluation metrics were chosen based on other information retrieval tasks. In MUC events, a system is scored according to two separate axes. One regarding its ability to find the correct class (category) regardless of the boundaries, and another to check whether the entity boundaries are correct, regardless of its class. For both of these axes, three measures were kept: COR (correct answers), POS (number of original entities) and ACT (number of guesses). Both precision, recall and F-measure are calculated using the sum of these two axes. This evaluation scheme gives partial credit to errors occurring only on one axis, full credit for having both axes correct, and zero credit for errors in both axes. MUC considered the following named entities: personal names, organizations, locations and, at a later stage, temporal entities, such as date and time, and numeral measurements, such as currency and percentage expressions. MUC focused only on the English language.

While the MUC-6 only centered on NER, the MUC-7 [CM98] provided one extra task regarding the identification of relations among categories (Template Relation). The main relationships were of *employee_of*, *product_of* and *location_of*. In MUC-6 the leading participant obtained an F-measure of 96.42%, while in MUC-7 it obtained 93.39%, in the NER task.

2.4.2 Conference on Natural Language Learning

The Conference on Natural Language Learning (CoNLL) [TD03] provided evaluation for systems either in English or German. Contrary to the MUC evaluation, CoNLL only gives credit to exact matches. That is to say, it only gives credit when both the boundaries and the class of the guessed entity are correct, corresponding to an exact match in the ground truth. Consequently, having only one of these axes correct, results in zero credit.

This is a simple evaluation scheme, underestimating the system score. This technique is often too restrictive, giving zero credit to an "almost good" answer, which sometimes is enough for the

task at hand. This conference concentrates on four classes of named entities, namely persons, locations, organizations and miscellaneous. Although, this is an annual conference since 1999, only the 2002 and 2003 editions focused on the NER task. In the 2003 edition of CoNLL, results were as high as 88.76% (F-measure).

2.4.3 Automatic Content Extraction

The Automatic Content Extraction (ACE) [DMP+04] succeeded MUC, in 1999, and has a different view of the NER task. This program relates to English, Arabic and Chinese texts and considers multiple classes of NEs, such as persons, organizations, locations, facilities, weapons, vehicles and geo-political entities. It also considers subclasses for these classes, for example, fictional characters is a subclasses of person. The ACE program not only deals with NER, but also relation detection and event extraction, so they have different evaluation techniques for each of these tasks. ACE's 2003 relation types are *ROLE*, corresponding to a role a person plays in an organization, *PART*, part-whole relationships, *AT*, location relationships, *NEAR*, relative locations, and *SOCIAL*, such as parent.

Its evaluation is based on a complex weighting scheme where each distinct NE class and each type of error have a different weight. Partial matches are allowed to a certain extent. The final score is 100% minus the sum of the penalties from these weights. The relation detection and event extraction are evaluated using the same scheme. This is a powerful method of evaluation due to its ability to customize the cost of error. However, given the complexity of this method with multiple parameters, it is difficult to compare different systems.

2.4.4 HAREM Avaliação de Reconhecimento de Entidades Mencionadas

HAREM is an evaluation contest for NER in Portuguese. There were two main HAREM events, in 2005 and 2008. In the HAREM conference, there were types for entities, but also categories and subtypes. It evaluated the task of identification, the task of morphological classification and the task of semantic classification. The first event [Car06, SSCV06] gave partial credit to producing a correct type identification of an entity and having wrong boundaries, and to producing wrong type identification and having the correct boundaries. This partial credit is given through the equation:

$$score = 0.5\frac{n_c}{n_d} \tag{2.7}$$

where n_c represent the number of common terms, and n_d the number of distinct terms between the output entities and the ones in the HAREM's golden collection, being 0.5 the maximum partial credit. The full credit is given when both type and boundaries are identified correctly. Besides precision, recall and F-measure, the HAREM event also used other metrics, such as under and over-generation and combined error.

The second event [FMS+10] had a different approach. This was due to the introduction of a new system of classification. This new system allowed for the classification with multiple alternatives in each entity, using the *ALT* category, for example in Listing 2.3.

```
<ALT><Barcelona Olympic Games> | <Barcelona> <Olympic Games></ALT>
```

Listing 2.3: Example of `ALT` tag application.

in which it can output the "Barcelona Olympic Games" as an event, or "Barcelona" place and "Olympic Games" event. This second event only has a single new measure, which is an extension of the combined measure of the first HAREM, taking into account the existence of subtypes and the optionality of all values. The second event not only included the NER task but also the task of identifying the semantic relations between the NE - ReRelEM [FSM+09] track. The relations defined in this event are *Identity*, *Inclusion*, *Location* and *Other*. Relations were scored as correct, missing or incorrect. That being said, only correct identifications received one point and the remaining received none, where a correct identification only considered the triples which linked the correct NE and whose relation was well classified. The HAREM task is considerably more difficult and fine-grained than other classical NER tasks.

HAREM's golden collection [SC06] (described in more detail in Section 3.2), is a collection of portuguese textual documents of several genres, such as web pages and newspapers, in which NEs have been identified, semantically classified and morphologically tagged in context. There were 10 categories identified, namely Works of art (*Obra*), Event (*Acontecimento*), Organization (*Organização*), Misc (*Outro*), Person (*Pessoa*), Abstraction (*Abstração*), Time (*Tempo*), Value (*Valor*), Local (*Local*) and Thing (*Coisa*).

2.4.4.1 Participants overview

In both HAREM conferences, almost all participants resorted to hand-coded techniques. For the first conference, out of nine participants, only two (NERUA [FKT+07] and MALINCHE [Sol07]) used machine learning techniques, and, for the second conference [FMS+10], only one out of ten (R3M [Mot08]). For the NER task, the top scoring participants for the first HAREM conference were PALAVRAS [Bic07], using a rule-based approach and scored an F-measure of 58%, and SIEMÊS [Sar07], which used similarity rules and scored 53%. For the second HAREM, Priberam [AFM+08], with a rule-based approach, scored 57% and REMBRANDT [Car08], using also a rule-based approach, but using Wikipedia as a KB, scored 56%.

In the second HAREM, a Relation Extraction task was added. Out of ten participants, only three submitted for the RE task (REMBRANDT, SEI-Geo [Cha08] and SeRelEP [BDVR08]), where the others only participated in the NER task. Since they chose to cover different relation types, it is not possible to compare them directly. However, taking into account all relations, REMBRANDT takes the lead with an F-measure of 45%.

2.4.5 Other approaches

There are other attempts at evaluation apart from the shared-tasks conferences, mainly corresponding to specific cases. Furthermore, the above-mentioned conferences provide an evaluation only for participating members, being hard for non-participating programs to evaluate their own approaches. For instance, Marrero et al. [MSCMA09] evaluated multiple programs which had not participated in conferences, using mainly precision and recall, but also several features such as the typographical, lexical, semantic or heuristic factors used by each evaluated program. Other example is Konkol [Kon12], who states that attributing the correct span is hard, thus it gives more importance to the categorization.

For the Portuguese language, apart from the HAREM golden collection, there are other Portuguese datasets, such as CINTIL[1], although not freely available; or even Amazônia, which is a Portuguese (from Brazil) corpus and a subset of the Floresta Sintá(c)tica[2], available in tree form.

2.4.6 Conferences summary

Table 2.2 summarizes the information for each conference, covering relevant years, that is to say years in which the main focus was NER and/or RE, entity classes and language. All conferences have Person, Organization and Location as an entity class in common. English is the most used language when dealing with NER and RE, being HAREM the only conference which focused on the Portuguese language.

Table 2.2: Overview of named entity recognition conferences.

Conference	Relevant years	Entity classes	Languages
MUC	1996, 1998	Person, Organization, Location, Date, Time, Money, Percent	English
CoNLL	2002, 2003	Person, Location, Organization, Miscellaneous	Spanish, Dutch, English, German
ACE	2003	Person, Organization, Location, Facility, Weapon, Vehicle and Geo-Political Entity	English, Arabic, Chinese
HAREM	2005, 2008	Pessoa, Organização, Local, Tempo, Obra, Acontecimento, Abstração, Coisa, Valor, Outro	Portuguese

[1] http://cintil.ul.pt/. Accessed: 2017-06-27.
[2] http://www.linguateca.pt/floresta/corpus.html. Accessed: 2017-06-27.

2.5 Summary

Named Entity Recognition and Relation Extraction have been largely studied in recent years. The first approaches were mainly done through hand-coded techniques, either with patterns or with dictionary-based matching. However, throughout the years there was a shift in focus, where machine learning techniques started to gain more interest. This is due to the scalability of these techniques and the amount of work required, however data in specific domains remains scarce. Machine learning supervised approaches continue to be the most used techniques, but, recently, many semi-supervised approaches, involving bootstrapping, started to appear, because they require less annotation effort.

There were a lot of attempts to evaluate the current state of the art for NER and RE. Some examples are conferences such as MUC, CoNLL, ACE or HAREM. They had different views on which classes of entities would be considered and how to evaluate their recognition; for instance, while MUC allows partial NE recognition, CoNLL only considers exact-matches. HAREM provided a good state of the art for the Portuguese language but it is outdated, since there have been no more evaluation conferences focused on the Portuguese language in recent years. Nevertheless, the HAREM conferences showed that hand-coded techniques were still preferred over machine learning ones. This is probably due to HAREM being an initial effort regarding the Portuguese language, and there was still no opportunity to explore ML methods for this language.

Although there are some attempts at NER and RE for the Portuguese language they still perform worse than for other languages, such as English.

Chapter 3

Datasets and tools

This chapter presents the methodology used in this dissertation, and also the description of the HAREM golden collection and the SIGARRA News Corpus, including its main characteristics regarding the annotated entities. In addition, this chapter also describes the Named Entity Recognition (NER) tools used for this dissertation, namely Stanford CoreNLP, OpenNLP, spaCy and NLTK.

3.1 Methodology

As stated in the state of the art (Chapter 2), the NER tools focused on the Portuguese language still underperform in comparison to other languages, like English, with the best results achieved in the last HAREM with an F-measure of 57%. In order to try to improve this, I decided to test well-established generic NER tools with the Portuguese language. This tools would have to be completely free and able to be trained with a custom corpus, in Portuguese. I chose Stanford CoreNLP, OpenNLP, spaCy and NLTK, which are described in Section 3.4.

To my knowledge, the only freely available Portuguese dataset annotated with entity classes was the one developed in the HAREM conferences. Another Portuguese dataset is CINTIL[1], but it is not publicly available. Also, there is the Amazônia dataset, however it is only composed of Portuguese from Brazil textual content. So, the HAREM collection is the dataset that will be used to train the tools. This dataset is described in Section 3.2. Since the Portuguese language was altered, with the latest Portuguese orthographic reform (1990)[2], the HAREM dataset can be considered outdated. Because of this, and also because the main goal is to perform NER in SIGARRA (the information system of the University of Porto), I decided to manually annotate a subset of SIGARRA's news. The prepared corpus is described in Section 3.3.

[1] http://cintil.ul.pt/. Accessed: 2017-06-27.

[2] http://www.portaldalinguaportuguesa.org/?action=acordo&version=1990. Accessed: 2017-06-27.

First, I will assess the out-of-the-box performance of the selected tools with the HAREM dataset, to establish a baseline. This will be done using repeated 10-fold cross-validation (with four repeats), to ensure robustness. The dataset needed to be transformed to be used as an input to each tool. This required transformation is explained in Chapter 4 and the obtained baseline results are presented in Chapter 5.

Next, after establishing the baseline, I will perform a hyperparameter study for every tool, to see if I can improve the performance. This study will also be done using the HAREM dataset but with repeated holdout (with four repeats), in order to save time. Check Chapter 6 for more information on this hyperparameter study. Finally, after obtaining the best hyperparameter values, I will assess the performance of the tools, with these values, against the SIGARRA corpus, to see whether the prepared corpus has influence on the tools' performance. For the SIGARRA News Corpus, the evaluation was the same as with the HAREM baseline — repeated 10-fold cross validation, with four repeats. The results are presented in Chapter 7.

3.2 The HAREM golden collection

HAREM is an evaluation contest for NER in Portuguese. There were two main HAREM events, in 2005 [Car06, SSCV06] and 2008 [FMS+10]. Both provided golden collections and, for this dissertation, only the second one was used. HAREM's golden collection [SC06] is a collection of Portuguese textual documents of several genres, such as web pages and newspapers, in which named entities have been manually identified, semantically classified and morphologically tagged in context, together with identified relations between the named entities. It defines three levels of entity annotations, namely categories, types and subtypes, in which categories have types and types have subtypes. As already stated, there were 10 categories identified, specifically Works of art (*Obra*), Event (*Acontecimento*), Organisation (*Organizacao*), Misc (*Outro*), Person (*Pessoa*), Abstraction (*Abstracao*), Time (*Tempo*), Value (*Valor*), Local (*Local*) and Thing (*Coisa*). Apart from the categories, it has a total of 43 types and 21 subtypes. Table A.1 illustrates all categories, types and subtypes for HAREM.

The HAREM golden collection's annotated classes are not equally distributed. Table 3.1 shows the number of named entities in each category in the second HAREM golden collection. *Pessoa* is the most common category with 2,035 words, and *Outro* is the least common, with only 148 words. This golden collection has a total of 129 documents, divided into two Portuguese variants, as shown in Table 3.2, being Portuguese from Portugal the main variant. These documents were gathered from different sources, with multiple text genres, described in Table 3.3, with a third being news text.

3.3 SIGARRA's News Corpus

SIGARRA is the information system of the University of Porto (U.Porto), where every organic unit has its own domain. Organic Unit is the entity of the organizational model, endowed with

Table 3.1: Number of named entities per category. Values from do Amaral et al. [dAFLV14].

Categories	Number of entities	%
Pessoa	2,035	28%
Local	1,250	17%
Tempo	1,189	16%
Organizacao	960	13%
Obra	437	6%
Valor	352	5%
Coisa	304	4%
Acontecimento	302	4%
Abstracao	278	4%
Outro	148	2%
Total	7,255	100%

Table 3.2: Document distribution by Portuguese variant. Table from Mota et al. [MSC+08].

Portuguese variant	Number of documents	%
pt_PT	93	72.09%
pt_BR	36	27.91%
Total	129	100%

Table 3.3: Document distribution by text genre. Table from Mota et al. [MSC+08].

Text genre	Total	%
News	45	34.88%
Didactic	29.5	22.87%
Journalistic blog	13	10.08%
Personal blog	11.5	8.91%
Questions	6	4.65%
Rehearsal	5	3.88%
Opinion	5	3.88%
Humour blog	4	3.1%
Legislative	3	2.33%
Promotional	3	2.33%
Interview	2	1.55%
Private manuscript text	1	0.78%
FAQ questions	1	0.78%
Total	129	100%

Provas Doutoramento: "The Visitor Experience Using Augmented Reality on Mobile Devices in Museum Exhibitions"

Provas de Doutoramento em Media Digitais

Requeridas por:
Diana Cristina Valente Marques

Data, Hora e Local
Dia 7 de julho às 14h30, na Sala de Atos da Faculdade de Engenharia

Presidente do Juri
Doutor João Manuel Paiva Cardoso, Professor Catedrático da FEUP

Vogais
Doutor Pedro Júlio Enrech Casaleiro, Investigador Auxiliar da Universidade de Coimbra;
Doutora Ana Isabel Oliveira Delicado, Investigadora Auxiliar do Instituto de Ciências Sociais da Universidade de Lisboa;
Doutor Nuno Manuel Robalo Correia, Professor Catedrático do Departamento de Informática da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa;
Doutor António Fernando Vasconcelos Cunha Castro Coelho, Professor Auxiliar do Departamento de Engenharia Informática da Faculdade de Engenharia da Universidade do Porto;
Doutor Robert Kearins Costello, National Outreach Program Manager at Smithsonian's National Museum of Natural History, USA (Coorientador).

Figure 3.1: Screenshot of a SIGARRA news.

own personnel, that can be endowed with tributary personality and that has a direct hierarchical relation with the central government of the University of Porto. The University of Porto has 17 SIGARRA domains, namely *FADEUP* (Faculty of Sport), *FAUP* (Faculty of Architecture), *FBAUP* (Faculty of Fine Arts), *FCNAUP* (Faculty of Nutrition and Food Sciences), *FCUP* (Faculty of Sciences), *FDUP* (Faculty of Law), *FEP* (Faculty of Economy), *FEUP* (Faculty of Engineering), *FFUP* (Faculty of Pharmacy), *FLUP* (Faculty of Arts and Humanities), *FMDUP* (Faculty of Dental Medicine), *FMUP* (Faculty of Medicine), *FPCEUP* (Faculty of Psychology and Education Sciences), *ICBAS* (Abel Salazar Biomedic Sciences Institute), *Reitoria* (Rectory), *SPUP* (Shared Services of U.Porto) and *UP* (U.Porto) itself.

ANT periodically gathers SIGARRA news, which allowed me to get a sample of the latest 1000 SIGARRA news to the date of extraction (2017-03-02). This extraction was validated, to make sure that the extraction included news from all domains, as well as to make sure the number of news matched in accordance with the news flow of every organic unit. These news were concatenated into a *csv* file, with the attributes being: news id, title, subtitle, source url, content and published date. The gathered news were published between 2016-12-14 and 2017-03-01. I manually annotated some of those news using the Brat rapid annotation tool [SPT+12]. The reason for not annotating all 1000 news, was due to some of them not being in Portuguese. That being said, I annotated 905 news. Figure 3.1 shows an example of a news in SIGARRA.

I selected eight different classes for the entities, namely `Hora` (Hour), `Evento` (Event), `Organizacao` (Organization), `Curso` (Course), `Pessoa` (Person), `Localizacao` (Location), `Data` (Date) and `UnidadeOrganica` (Organic Unit). For this selection, I read some news from each organic unit, to understand the relevant entity classes present in each of them, which had the most value to the ANT project. Figure 3.2 shows the distribution of news per SIGARRA domain, in the SIGARRA News Corpus, showing that FLUP, FAUP, FEUP and FMUP have the highest number of news (over 100 news), in that order, each representing over 10% of the collection. Furthermore, UP and FMDUP have the least amount of news (less than 10 news), in that order, each representing less than 1% of the collection. I annotated a total of 905 news for SIGARRA News

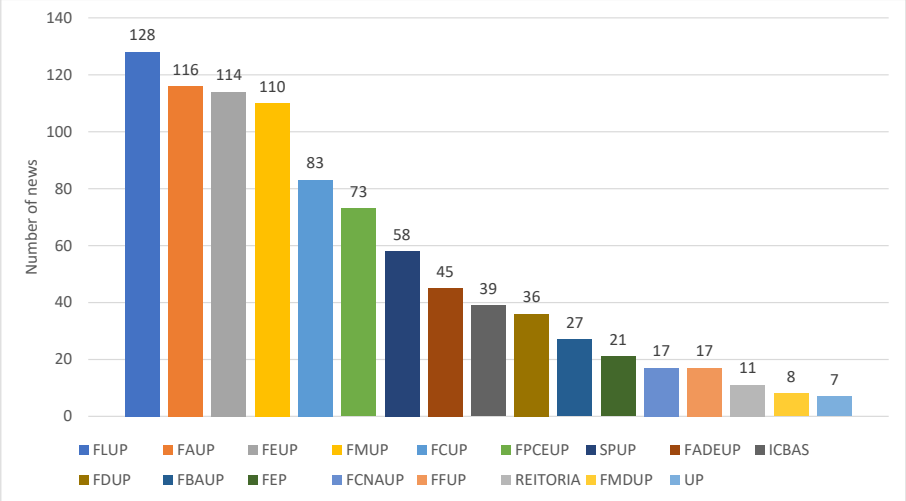

Figure 3.2: Number of news by SIGARRA domain.

Corpus (see Table B.1 for more detail). Finally, we can see in Figure 3.3 the statistical data about the number of characters in each news by SIGARRA domain. SPUP shows the highest dispersion. In addition, SPUP has the highest average of characters (2340 characters) and FDUP the lowest (582). Also, on average, there are 1154 characters per news.

The entity classes and their distribution by SIGARRA domain in this annotated corpus can be seen in Figure 3.4 (see the distribution of entity annotations in Table B.2 and the distribution by SIGARRA domain in Table B.3). It is clear that Date (*Data*) has the highest number of annotations with 22.23% of the total entities, as expected because there were very few news which had no date (only 43 news). Since this is an academic domain, Organization (*Organizacao*) has a high percentage of mentions, mainly because it is common to name the organization a person works for as well as project funding organisms. Person (*Pessoa*) also represents a big fraction, mainly because of news related to presentations of master's and doctoral theses. On the other hand, Event (*Evento*) represents a lower percentage, because it is difficult to pinpoint the exact boundary for this entity tag, which can lead to sometimes skipping their annotation. This figure also shows that

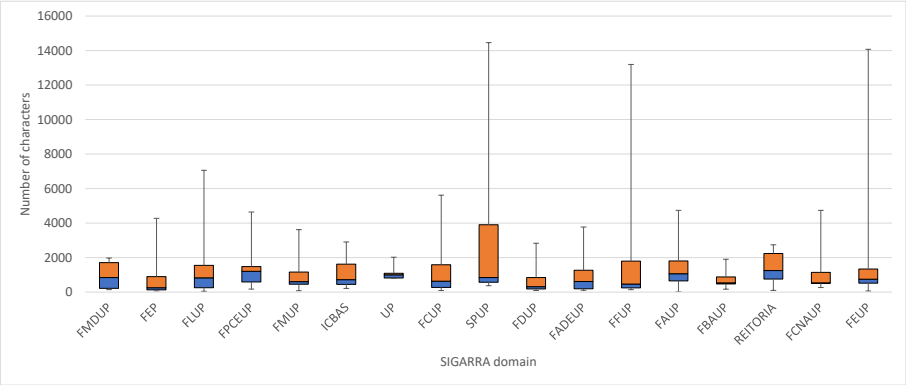

Figure 3.3: Number of characters per news by SIGARRA domain.

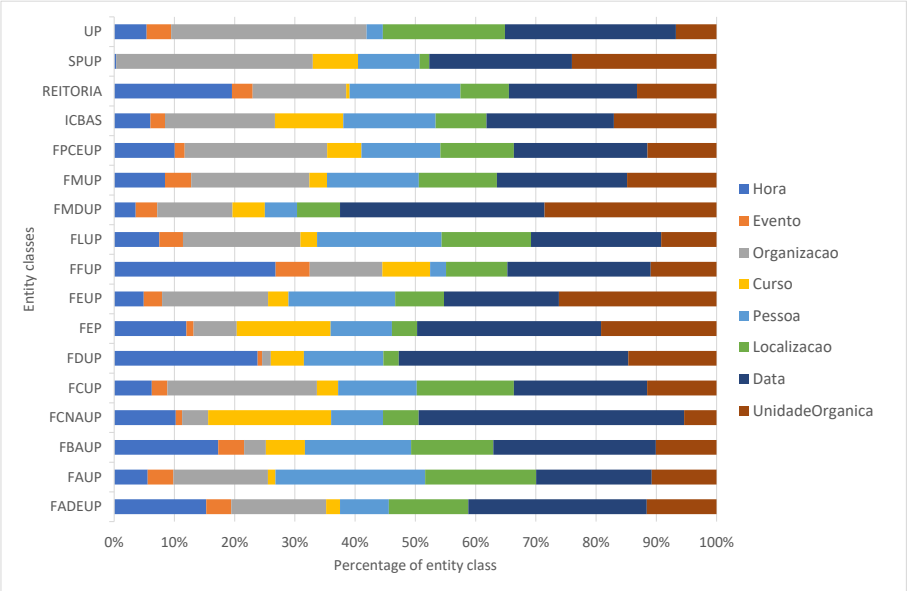

Figure 3.4: Number of annotated entities by SIGARRA domain.

FAUP has the highest number of annotations (2789), followed by FEUP (1711), and that FMDUP has the lowest number (56). Overall, there are 12644 entity annotations.

3.4 Natural Language Processing tools

In order to choose the best fitting tool to tackle the proposed challenge, I compared some of the main tools capable of performing NER in multiple languages, with particular focus on the Portuguese language. The main criteria for choosing the tools were:

- The tool has to be completely free.
- The tool has to be language and domain independent.
- The tool has to allow training a custom model for NER, with custom entity classes.

With these criteria in mind, I chose four different, well established tools: Stanford CoreNLP [FGM05], OpenNLP [Fou], spaCy [Hona] and NLTK [BKL09]. These tools are described in Table 3.4, which shows that all are freely available, with Apache, MIT or GNU General Public licenses, use either Java or Python as the main language and, apart from already having multiple default language models, all allow training with other languages. Also, I used the latest versions available at the beginning of this dissertation, namely 3.7.0 for Stanford CoreNLP, 1.7.2 for OpenNLP and spaCy and 3.2.2 for NLTK.

Neither of these tools have a Portuguese model for NER, so all of them require training with a Portuguese corpus, such as the HAREM collection. Apart from spaCy, all tools use well known algorithms to perform NER, for instance CRF and Maximum Entropy. spaCy uses its own algorithm,

Table 3.4: Overview of the assessed tools.

Tool	License	Version	Language	NER classifiers	Default language models
Stanford CoreNLP	GNU General Public License	3.7.0 (2016)	Java	CRF	Arabic, Chinese, English, French, German, Spanish
OpenNLP	Apache License	1.7.2 (2017)	Java	Maximum Entropy	Dutch, English, Spanish
spaCy	MIT License	1.7.2 (2017)	Python	Thinc linear model	English, German
NLTK	Apache License	3.2.2 (2016)	Python	Naive Bayes, Decision Tree, Maximum Entropy	English

called Thinc [Honb] linear model, which is a machine learning library, with an implementation of a structured average perceptron.

3.5 Summary

Although some years have passed since HAREM was built, it still remains the only publicly and freely available golden collection for the Portuguese language (from Portugal), with entity annotations, to my knowledge. The golden collection has 129 documents from different sources, with multiple text genres, such as news or legislative texts. I manually annotated a subset composed of 905 SIGARRA news, from the multiple domains, using the Brat rapid annotation tool. Resulting in a corpus with 12644 entity annotations, twice the size of the HAREM golden collection. Finally, I decided to work with four tools, all of them freely available and able to work with the Portuguese language. All of these tools have machine learning algorithms in order to train a NER model.

Chapter 4

HAREM corpus transformation

The HAREM golden collection cannot be directly used as input to the selected tools, so it had to be transformed for each tool. This transformation is described in this chapter. The golden collection is available in XML format, and contains information that was not used in this dissertation, so this information was stripped. Also, since the referred tools are not able to deal with entity subclasses, I flattened the types and subtypes levels. Since I will use a common script to evaluate all tools, it is required that all outputs have the same format, namely the CoNLL format with IOB tags — where `I` (Inside) means that the current token is inside an entity tag, `O` (Outside) means that the current token is outside an entity tag and `B` (Begin) means that the current token is the first one in an entity tag.

4.1 Initial transformation

This first transformation was applied for all tools. First of all, the tag `OMITIDO` was used to allow some control in the parts of the text without linguistic relevance to the evaluation. As stated in the HAREM book [OMF+08], it was ignored in the HAREM evaluation, so it was stripped from the golden collection, keeping the text without any annotation. Then, since HAREM provides alternatives to the annotation of entity tags, regarding the entity boundary, using the `ALT` tag, these tags were stripped. I decided to keep:

- The alternative which had the highest amount of entity tags OR
- The first alternative when there were no entity tags inside the `ALT` tag

This was required because the selected tools cannot handle alternative annotations. Apart from the alternatives using the `ALT` tag, there were alternatives in the annotation, regarding the category, type and subtype of an entity, separated by the `|` character. In this case, the first alternative for each case was selected. Also, there were tags which had no categories, and were identified only as an entity. In this case, the tag was stripped. A simplified example of this is presented in Listing 4.1.

```
<ALT><EM CATEG="ABSTRACCAO|ABSTRACCAO" TIPO="DISCIPLINA|IDEIA">Reforma de Henrique<
    /EM> | <EM CATEG="ABSTRACCAO" TIPO="DISCIPLINA">Reforma</EM> de <EM CATEG="
    PESSOA" TIPO="INDIVIDUAL">Henrique</EM></ALT>

<EM CATEG="ABSTRACCAO" TIPO="DISCIPLINA">Reforma</EM> de <EM CATEG="PESSOA" TIPO="
    INDIVIDUAL">Henrique</EM>
```

Listing 4.1: `ALT` tag transformation example.

Since the tools only allowed one level of entities, the levels were flattened, producing three different outputs: one with only the categories, one with only the types and another with only the subtypes. For scripting purposes, the types and subtypes were concatenated to keep the semantics and context inside the parent category (e.g. "LOCAL_HUMANO_PAIS"). Finally, apart from the category attribute, all other attributes were removed. For evaluation purposes, this dataset was divided into folds for repeated cross validation (see Chapter 5).

4.2 Tool-specific transformation and running steps

After the initial transformation, and in order to be able to use the dataset as input to each tool, the golden collection went through further transformations, described in this section. Each tool has a particular set of requirements in order to train a Named Entity Recognition (NER) model and then to perform NER. The steps for each tool are presented next, namely the required input format, the steps for converting HAREM into that format, how to train the NER model, how to perform NER, and finally how to convert it to the CoNLL evaluation format.

4.2.1 Stanford CoreNLP

CoreNLP requires a tokenized file as input, where each line contains a token and its entity class separated by a tab character. This entity class is the class of the entity for entity tokens, and an "O" class for other tokens. An example of the input format can be seen in Listing 4.2.

```
Desde TEMPO
1947  TEMPO
, O
o O
jovem O
William PESSOA
M.Gaines  PESSOA
```

Listing 4.2: Stanford CoreNLP example input format.

The first approach was to use *corpus-processor* [Dad13], whose purpose is to convert the HAREM golden collection into the Stanford CoreNLP format. However, since this tool used

a different tokenization scheme from the one in Stanford CoreNLP, the output from the entity recognizer was different, in terms of tokenization, from the test set. Because of this, I had to use the Stanford CoreNLP tokenizer `edu.stanford.nlp.process.PTBTokenizer`. The entity classes in XML were also tokenized in the process, so afterwards I "de-tokenized" them and then looped through the file adding the entity classes from the first match in an entity tag to a match in the closing tag. Every token outside this was tagged "O". It is important to note that I tokenized using the XML annotated file so I wouldn't loose the entity annotations. The classifier was trained using the command in Listing 4.3.

```
java -cp stanford-corenlp.jar edu.stanford.nlp.ie.crf.CRFClassifier -prop <file.
    prop>
```

Listing 4.3: Stanford CoreNLP training command.

where `file.prop` sets the hyperparameters and features to use and the path for the training file and output model.

To classify text documents, I used the command in Listing 4.3.

```
java -cp stanford-corenlp.jar edu.stanford.nlp.ie.crf.CRFClassifier -loadClassifier
     <ner-model.ser.gz> -testFile <file_test.txt>
```

Listing 4.4: Stanford CoreNLP command to perform NER.

where `file_test.txt` represents the input unannotated text to be classified.

Then, after performing the NER, I added IOB tags to the output for it to be comparable in the evaluation stage.

4.2.2 OpenNLP

This tool requires a sentence per line as input, where entities are annotated with a starting tag (`<START:tag-name>`) and an end tag (`<END>`). An example of this format can be seen in Listing 4.5.

```
<START:TEMPO> Desde 1947 <END> , o jovem <START:PESSOA> William M.Gaines <END>
```

Listing 4.5: OpenNLP example input format.

First, I converted the HAREM XML entity tags to the OpenNLP format. Then, using NLTK's sentence segmentation module, the dataset was segmented by sentences. Since this segmentation was not perfect, I had to join faulty segmentations by checking if every open entity tag had a corresponding closing tag and vice versa, in each sentence. When they had no corresponding tag, I joined the current sentence with the previous sentence. These faulty segmentations were mainly due to the segmentation being done with the presence of XML tags. Similar to Stanford

CoreNLP tokenizing issue, I decided to perform sentence segmentation with the XML tags, so that I wouldn't loose the entity annotations. This is not ideal, but it is a simpler method. Furthermore, I had to make sure that there was a space character before and after each tag, or else OpenNLP's interpreter would not work.

To train the model, I had to run the command in Listing 4.6.

```
opennlp TokenNameFinderTrainer -model <model.bin> -lang <pt> -data <training_data.
    txt> -encoding <UTF-8>
```

Listing 4.6: OpenNLP training command.

and to classify the text, the command was the one in Listing 4.7.

```
opennlp TokenNameFinder <model.bin> < <corpus_test.txt> > <output file>
```

Listing 4.7: OpenNLP command to perform NER.

Finally, after performing NER, I converted the output from the OpenNLP format to the CoNLL format, adding IOB tags.

4.2.3 spaCy

spaCy requires that the input dataset is in the standoff format, that is to say, there has to be two files: one with the plain text, and another with the entity annotations, containing a tab separated entry with the beginning and ending positions of the entity along with its class. The text had to be segmented into sentences. Listing 4.8 shows an example of the standoff format, first with a text file and next the *ann* format with annotations.

```
-- TXT --
Desde 1947, o jovem William M.Gaines
-- ANN --
0   10  TEMPO
20  36  PESSOA
```

Listing 4.8: SpaCy example input format.

Since I had already converted HAREM to the OpenNLP format, I used this data and transformed it to standoff. The transformation was done using the following steps:

1. Search until `<START:tag>` tag
2. Save starting position
3. Delete matched tag
4. Search for `<END>` tag

5. Save end position
6. Delete matched tag
7. Save `standoff = (beginPos, endPos, tag)`
8. Repeat from step 1 until no matches occur
9. Output to a tab separated file

The resulting files were then used to train a NER model in spaCy. The training script was based on an example script in spaCy's repository, with the additional preprocessing task of converting to the standoff format. The main NER classifier was changed from `EntityRecognizer` to `BeamEntityRecognizer`. This change was because this classifier, in direct conversations with the developer responsible for spaCy, was said to have better results. After the NER process, the result was converted to the CoNLL format with IOB tags for evaluation purposes.

4.2.4 NLTK

This toolkit not only requires that the input dataset is in the CoNLL format with IOB tags, but also that it has the associated POS tag. In other words, the input file must be a tab separated file, where each line has the token, the POS tag and the entity tag in IOB format. Listing 4.9 shows an example of this format.

```
Desde   n B-TEMPO
1947    num I-TEMPO
, ,   O
o art O
jovem adj O
William n   B-PESSOA
M.Gaines  n   I-PESSOA
```

Listing 4.9: NLTK example input format.

The requirement for having both the IOB and the POS tags led to the need for three major steps, namely tokenizing and performing POS tagging, tokenizing while keeping the entity tags, and joining both files. The steps for this were the following:

1. Tokenize and POS tag
   - Remove all tags from the HAREM dataset, in order to tokenize the text
   - Tokenize the dataset using `nltk.tokenize.word_tokenizer` (*TreebankWordTokenizer*)
   - Using the resulting tokenized text, perform POS tagging
     - Train POS model using *floresta* corpus from `nltk.corpus`

  - POS tag resulting file from tokenizer step

2. Tokenize while keeping the entities

   - Tokenize the dataset using `nltk.tokenize.word_tokenizer` (*TreebankWordTokenizer*)
   - Join consecutive tokenized entity tags
   - Transform dataset, matching the entity tags using regular expressions, assigning tags to each token
     - For each token, after an entity tag, assign a *B-tag*
     - For each token after the first entity token (previous step), or after an Inside (I) token, assign an *I-tag*
     - Assign an *O* tag for other tokens

3. Join POS tagged file with entity tagged file

   - Iterate through both files simultaneously
   - For each line, set *token POS-tag IOB-entity-tag*

To train the NLTK's classifiers, I used NLTK trainer[1]. This allowed me to run the command in Listing 4.10.

```
python train_chunker.py <path-to-training-file> [--fileids <fileids>] [--reader \
    <reader>] [--classifier <classifier>]
```

Listing 4.10: NLTK training command.

I had to choose a different document reader, `nltk.corpus.reader.conll-.ConllChunkCorpusReader`, since the training files were in the CoNLL format. For this reader, I had to specify the entity categories in the NLTK trainer init file. Running the command resulted in a serialized model saved in the pickle format.

In order to classify text, the model had to be loaded, the dataset to be classified was required to contain POS tag annotations and then classified with `chunker.parse(tagged)`. The parser returned the result in a tree format, which was converted to the CoNLL format using `nltk.chunk.util.tree2conlltags(ner_result)`.

4.3 Summary

Since the HAREM golden collection cannot be used directly as an input to the selected tools, this chapter provided the steps to transform it to the right input type. The HAREM golden collection was first subjected to an initial transformation, common to all tools, and then to a tool specific

[1] https://github.com/japerk/nltk-trainer. Accessed: 2017-06-27.

transformation. Stanford CoreNLP required, as input, a file in the CoNLL format. OpenNLP required a sentence segmented file with specific tags. SpaCy required the input file be in the standoff format. And NLTK required not only a entity tagged text in the CoNLL format, but also POS tags.

This chapter provided important guidelines to transform datasets, specifically HAREM, into well-known formats. These guidelines can be used by the scientific community to transform between input formats required by different tools. Furthermore, sometimes tools' documentation can be scarce or have too much detail, so it can be difficult to understand the basic configurations needed to run each tool.

Chapter 5

Evaluation

This chapter describes the evaluation method and the results obtained for Named Entity Recognition (NER) for the discussed tools in Chapter 3. The method used to assess the performance of each tool is described next. Each tool has its own evaluation scheme, hence they are not directly comparable. To make it comparable, a common evaluation scheme was used. The chosen evaluation method was the one used in the CoNLL [TD03] conference, meaning credit was only given to exact-matches or, in other words, both entity tags and boundaries had to be correct for it to count as a correct match.

5.1 Evaluation method

Evaluation was done using the *conlleval*[1] script, taken directly from the website for the conference's occurrence in 2000. The script requires a file in the CoNLL format, with both the output of the NER tool and the golden standard. To be more precise, it is a space separated file, where each line contains a token, the golden standard tag and the predicted tag. It accepts files with or without IOB tags, being that, for the latter, each identification is treated as a single token entity. For this dissertation, the results were evaluated with IOB tags. Since not all the tools outputted the results with IOB tags, I added them whenever they were missing. After each run, I had to merge the outputs with the golden standard, so that it could be used as input for the evaluation script. For this merge to be correct, I had to use each tool's tokenizer for the testing set, or else the merge would not be successful as the tokens would not match correctly to the golden standard set.

In order to ensure the robustness of the evaluation metrics, I used repeated 10-fold cross validation, with four repeats, and calculated the average of the precision and recall, and the macro-average for the F-measure, from every run. In other words, in each repeat, I split the dataset into 10 equal sized folds (in terms of documents), where one fold was used for testing and the rest for

[1] See http://www.cnts.ua.ac.be/conll2000/chunking/conlleval.txt - version: 2004-01-26.

training. Then I ran each tool for each fold and for each repeat, and also for each level (categories, types and subtypes). Resulting in a total of 4 $repeats \times 10\ folds \times 3\ levels \times 4\ tools = 480\ runs$.

Finally, after running all the folds and repeats for each tool, that is to say training the models, and performing NER on the testing set, I evaluated the outputs using the CoNLL script and then computed the average for each level. This resulted in an average for each tool, each level, and each entity tag. The results are presented in Section 5.2.

5.2 Results

The results from the repeated 10-fold cross validation with the HAREM dataset, for all tools, along with a comparison between them, are presented in this section. There is a clear distinction between the tools, and also between the different entity class levels.

5.2.1 OpenNLP

Table 5.1 presents the results obtained for the Maximum Entropy classifier from OpenNLP. Categories have the best result, with an F-measure of 53.63%, followed by subtypes (50.74%). This is probably due to the large amount of categories when compared to the types and subtypes. Subtypes have the best precision, probably due to being very specific.

Table 5.1: Results for OpenNLP, for the HAREM collection.

Level	Precision	Recall	F-measure
Categories	55.43%	51.94%	53.63%
Types	52.13%	45.40%	48.53%
Subtypes	72.60%	39.00%	50.74%

5.2.2 spaCy

Table 5.2 presents the results obtained for the Beam Entity Recognizer from spaCy. Categories have the best results, with an F-measure of 46.81%, followed by types (44.04%). Similar to OpenNLP, this is probably due to the large amount of categories when compared to the types and subtypes. Subtypes have the best precision, probably due to being very specific.

5.2.3 Stanford CoreNLP

Table 5.3 presents the results obtained for the CRF classifier from Stanford CoreNLP. It is important to note that Stanford CoreNLP is highly computationally demanding, so I was not able to run it with the default configuration for the remaining levels, i.e. types and subtypes. Because of this, comparison between the levels was not possible, but I obtained an F-measure of 56.10% for the categories.

Table 5.2: Results for spaCy, for the HAREM collection.

Level	Precision	Recall	F-measure
Categories	51.21%	43.10%	46.81%
Types	49.76%	39.50%	44.04%
Subtypes	79.21%	24.88%	37.86%

5.2.4 NLTK

Since NLTK provides different classifiers to perform NER, I present the results for each of them in Table 5.4. Table 5.4a presents the results obtained for the Naive Bayes classifier from NLTK, Table 5.4b for the Maximum Entropy classifier and Table 5.4c for the Decision Tree classifier. Looking at Table 5.4, we can see the average results for each classifier, for the highest level (categories). The `NaiveBayes` and `DecisionTree` classifiers have similar results, although `NaiveBayes` has better results. `Maxent` (Maximum Entropy), however, performed way worse than the other classifiers, with almost zero recall and F-measure. One possible explanation for these results is the default number of iterations (ten) that the algorithm goes through, or even the features used for training. OpenNLP's classifier also uses Maximum Entropy and it performed much better, but it was configured to use 100 iterations in the default configuration.

5.2.5 Comparison

Table 5.5 represents the evaluation metrics for all tools (precision, recall and F-measure), for the highest entity class level (categories). Stanford CoreNLP takes the lead with an F-measure of 56.10%, followed by OpenNLP with 53.63% and then by spaCy (46.81%) and NLTK (30.97%). The results for NLTK are the ones obtained using the Naive Bayes classifier. It was NLTK's best result but it was the lowest score among all tools, way below the other tools performance. Table 5.6 shows the average F-measure for all the levels. The ranking was similar to the categories result. In other words, Stanford CoreNLP remains on top in the category level, followed by OpenNLP, then spaCy and finally NLTK, in all levels. Again, NLTK's results remain far below the values for the other tools. Naive Bayes remained the best scoring algorithm amongst NLTK's algorithms.

Table 5.3: Results for Stanford CoreNLP, for the HAREM collection.

Level	Precision	Recall	F-measure
Categories	58.84%	53.60%	56.10%
Types	-	-	-
Subtypes	-	-	-

Table 5.4: Results for NLTK, for the HAREM collection.

(a) `NaiveBayes` classifier.

Level	Precision	Recall	F-measure
Categories	30.58%	31.38%	30.97%
Types	29.66%	28.01%	28.82%
Subtypes	21.15%	22.72%	21.91%

(b) `Maxent` classifier.

Level	Precision	Recall	F-measure
Categories	18.19%	0.58%	1.13%
Types	9.84%	1.07%	1.93%
Subtypes	0.19%	0.28%	0.23%

(c) `DecisionTree` classifier.

Level	Precision	Recall	F-measure
Categories	21.84%	25.72%	23.62%
Types	25.37%	24.34%	24.84%
Subtypes	27.71%	35.81%	31.25%

Table 5.5: Results for the category level, for all tools.

Tool	Precision	Recall	F-measure
Stanford CoreNLP	58.84%	53.60%	56.10%
OpenNLP	55.43%	51.94%	53.63%
SpaCy	51.21%	43.10%	46.81%
NLTK	30.58%	31.38%	30.97%

Table 5.6: F-measure for all levels.

Tool	Categories	Types	Subtypes
Stanford CoreNLP	56.10%	-	-
OpenNLP	53.63%	48.53%	50.74%
SpaCy	46.81%	44.04%	37.86%
NLTK	30.97%	28.82%	21.91%

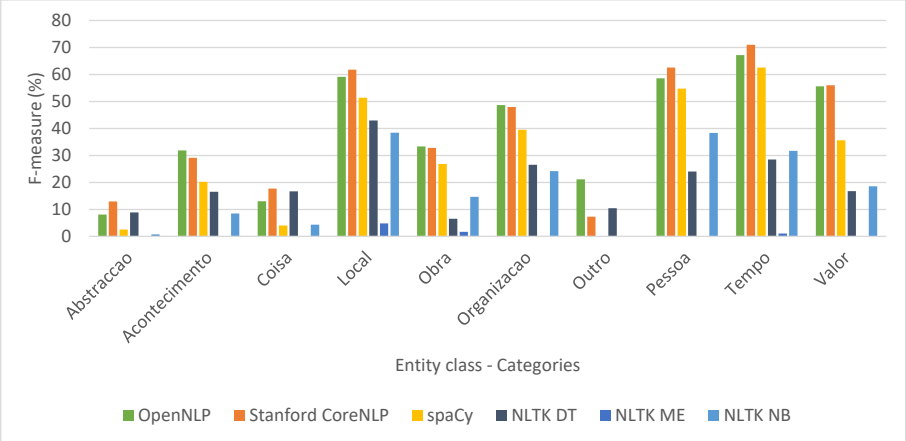

Figure 5.1: F-measure by entity class, by tool.

Figure 5.1 shows the F-measures for every tool, regarding the entity classes, for the category level (detailed results can be seen in Table D.1). Needless to say, NLTK's Maximum Entropy classifier is the lowest scoring algorithm, with zero F-measure in almost all entity classes. Stanford CoreNLP has the highest score in almost all entity classes, followed by OpenNLP. Overall, the best results belong to the *Tempo* entity class, and the worse to *Outro*. In addition, details for the types and subtypes entity classes can be seen in Table D.2 and Table D.3, respectively. Stanford CoreNLP is missing due to it not being ran with these entity levels. In the types level, the ones related to time and people got the best scores. Additionally, NLTK's Decision Tree classifier got interesting results in *Coisa_Substancia* and *Pessoa_GrupoCargo*, being the only classifier with decent results in these classes (70.32% and 63.67%, respectively). Finally, in the subtypes level, location and time related entity classes got the best F-measure; all others were close to zero.

Table 5.7 shows the comparison between the average training time for all tools. As I said before, it is important to note that Stanford CoreNLP did not run for the types and subtypes levels. This table shows that NLTK's `NaiveBayes` is the fastest to run (needing only 2s per level), followed by OpenNLP (with less than a minute for each levels). NLTK's `DecisionTree` is the slowest (with 11h47m to run all repeats and folds), however, I estimate that if I were able to run all the levels for Stanford CoreNLP, this tool would be the slowest despite having the best results. Although, Stanford CoreNLP got the best results, according to the obtained F-measures, OpenNLP could be considered the best tool to use since it was the second best, and it is much faster. That being said, if the NER process is time sensitive, OpenNLP should be the one to use.

5.3 Summary

This experiment only provided a comparison for the baseline configuration of each tool, so, in order to improve the results, one has to tune the hyperparameters as well as experiment with feature engineering for each algorithm implementation. While it is important to note that these results are not directly comparable to the ones in HAREM, since different methods of evaluation

Table 5.7: Average training time for all tools per fold.

Tool		Categories	Types	Subtypes	All
Stanford CoreNLP		11m40s	-	-	7h47m
OpenNLP		22s	52s	44s	1h19m
spaCy		3m17s	5m19s	5m20s	9h18m
NLTK	`NaiveBayes`	2s	2s	2s	4m19s
	`Maxent`	1m56s	5m23s	4m24	7h50m
	`DecisionTree`	5m55s	5m54s	5m52s	11h47m

were used, there is a strong indication that these tools perform similar to the results achieved in the HAREM conferences, with the best F-measure of 56.10% by Stanford CoreNLP, with a difference of only 1% in the highest scoring participant in the last HAREM, about 10 years ago.

Results have shown that the tools perform worse when there are more entity classes. Also, these results are still less precise when compared to other languages, such as English. This might be because the HAREM dataset is too much specific in its annotation scheme. In other words, while in a MUC conference, for example, only generic entity classes were considered, in HAREM there were a lot more entity classes, with more specific meanings.

Chapter 6

Hyperparameter study

This chapter presents the effects of hyperparameters, measuring the effect of individually changing each individual hyperparameter for every available tool, using the HAREM dataset. Furthermore, it also provides a small description of each hyperparameter. I tested the effects of a total of nine hyperparameters. The number of iterations is the most common of all, being present in almost all classifiers.

6.1 Description

In the previous chapter, I presented the results for the baseline of every tool, meaning that all tools were run with the default configuration (default hyperparameters). In order to improve those results, I re-ran the tools with different configurations. Table 6.1 shows the available hyperparameters for each tool and their default value. I also present a short description of each hyperparameter, focusing on its goal within the named entity recognition algorithm. The most common hyperparameter is the number of iterations the classifier goes through, being present in three out of five classifiers. OpenNLP and NLTK both share a Maximum Entropy classifier, having both the number of iterations as a hyperparameter. There are also multiple cut-off values for classifiers, such as entropy or depth cut-off.

- **OpenNLP**
  - *Iterations* - Number of iterations in the Maximum Entropy algorithm.
  - *Cutoff* - Minimum number of instances that are required so that the feature is chosen.
- **NLTK** [Per14]
  - `MaxEnt`
    - *max iterations* - "The max_iter variable specifies the maximum number of iterations to go through and update the weights" [Per14].

Table 6.1: Hyperparameters for each tool.

Tool	Parameter	Default value
OpenNLP	iterations	100
	cutoff	5
NLTK	[ME] max iterations	10
	[ME] min_lldelta	0.1
	[DT] entropy cutoff	0.05
	[DT] depth cutoff	100
	[DT] support cutoff	10
Stanford CoreNLP	tolerance	1e-4
spaCy	iterations	10

    - *min_lldelta* - This value specifies the minimum change in the log likelihood, in each iteration. The algorithm stops training when the log likelihood delta between two consecutive iterations is below this value.
  - `DecisionTree`
    - *entropy cutoff* - Entropy value of the tree. When entropy is greater than this value, the tree must be refined with further decisions to reduce the uncertainty.
    - *depth cutoff* - This value controls the depth of the tree, specifying the that the tree is no deeper than this value.
    - *support cutoff* - This value sets the number of labeled feature sets that are required to refine the tree. Labeled feature sets that have a number below this value, are eliminated since they no longer provide value to the training process.

- **Stanford CoreNLP**
  - *Tolerance* - Convergence tolerance in optimization. In other words, the optimization function attempts to find an unconstrained minimum of the objective function starting at the initial value, accurate to within the tolerance value.

- **SpaCy**
  - *Iterations* - Number of iterations in the training algorithm.

6.2 Results

To perform this study, I used repeated holdout, with four repeats. In other words, I split the HAREM dataset into two distinct sets: 70% for training and 30% for testing, at the document level, meaning 98 documents were used for training and 31 for testing. And then, I repeated this process four times. Since training the classifiers is very time consuming, I decided to do repeated holdout as opposed to the repeated 10-fold cross validation, which was done in the previous task

— obtaining baseline results for the aforementioned tools. That being said, if I only tested a single value on each hyperparameter, I would have done 1080 runs (4 $repeats \times 10$ $folds \times 3$ $levels \times 9$ $hyperparameters = 1080$ $runs$) with the repeated 10-fold cross-validation, and with only repeated holdout I would have only performed 108 runs (4 $repeats \times 3$ $levels \times 9$ $hyperparameters =$ 108 $runs$). So, taking into account all hyperparameter values tested, I would have performed 13560 runs, but instead I only performed 455 runs, which allowed me to save an enormous amount of time. That being said, taking into account the training times, in Table 5.7, and the previously stated hyperparameters and the tested values, and ignoring the hyperparameters' effects on training time, I estimate a total running time of approximately 35 days (837h31m19s)[1] if I were to run repeated 10-fold cross validation for this hyperparameter, which is not feasible. With the repeated holdout, and taking into account all hyperparameter values tested, and estimating similar times to one fold for a single repeat, I estimate a total training time of approximately 10 and a half days (4 $repeats \times 3$ $levels \times 20h55m25s = 251h05m \approx 10.5days$). For the same reason, this hyperparameter study was not performed using a grid search, because it would mean I would have to perform 30696 runs, instead of the 455 I mentioned previously. Another option would be to use a sample of the dataset instead, however this was not done because the dataset can already be considered small, and some algorithms do not gain much time difference from changing the training size. This section presents the results for all four tools, showing the effects of their hyperparameters.

6.2.1 OpenNLP

OpenNLP allows tuning the iterations and the cut-off values. Table 6.2 shows the results for the relevant tested cut-off values, showing that the value 4 was the best scoring for the categories level (52.38%), and the value 3 was the best for the types (48.90%) and subtypes (52.52%). Figure 6.1 clearly shows that the best values are 3 and 4. Check Table C.1 for further detail.

Table 6.2: Results for cut-off values of OpenNLP.

Value	Categories	Types	Subtypes
3	52.05%	**48.90%**	**52.52%**
4	**52.38%**	48.12%	52.35%
5 (default)	50.90%	47.59%	50.76%

Table C.2 also shows the details for the default value and the best F-measures for the different iterations. Looking at Figure 6.2, the behaviour indicates that the higher the number of iterations, the higher the resulting F-measure, however, it seems to reach a plateau at around 170 iterations. That being said, categories and subtypes had the best result at 170 iterations (51.52% and 52.61%, respectively) and types at 120 (47.81%). See Table C.2 for the details on the remaining results.

[1] $837h31m19s = 7h47 \times 6$ $stanfordcorenlp\_hyper\_values + 1h19 \times 21$ $opennlp\_hyper\_values + 9h18 \times 12$ $spacy\_hyper\_values + 4m19 \times nltk\_naivebayes + 7h50 \times 23$ $nltk\_me\_hyper\_values + 11h47 \times 40$ $nltk\_dt\_hyper\_values$

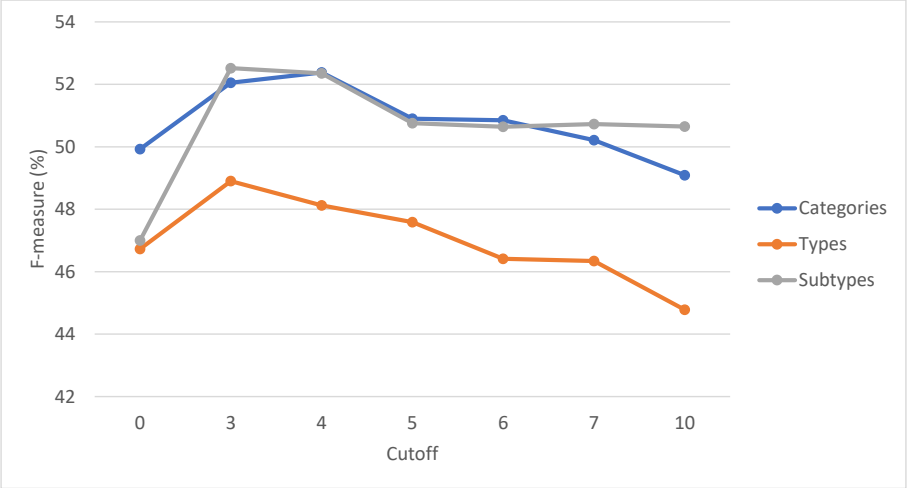

Figure 6.1: Results for cut-off values of OpenNLP.

Table 6.3: Results for iteration values of OpenNLP.

Value	Categories	Types	Subtypes
100 (default)	50.90%	47.59%	50.76%
120	51.19%	**47.81%**	51.81%
170	**51.52%**	47.43%	**52.61%**

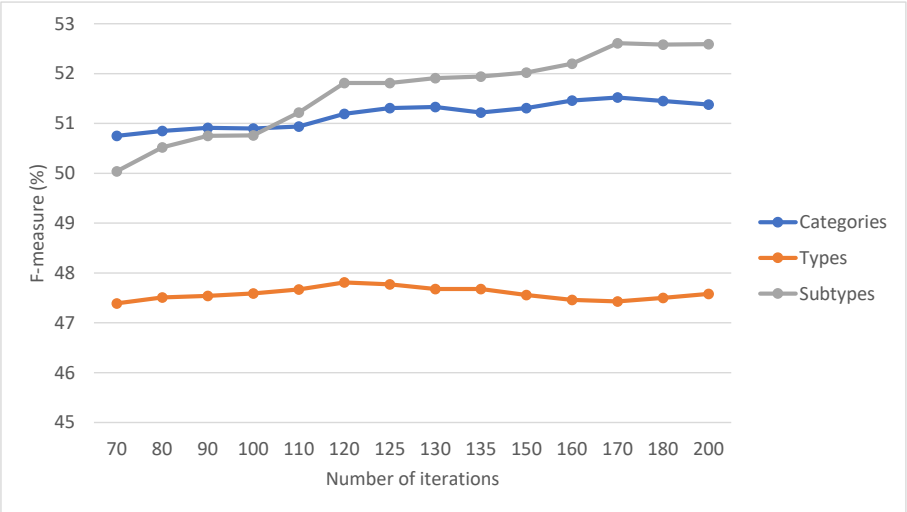

Figure 6.2: Results for iteration values of OpenNLP.

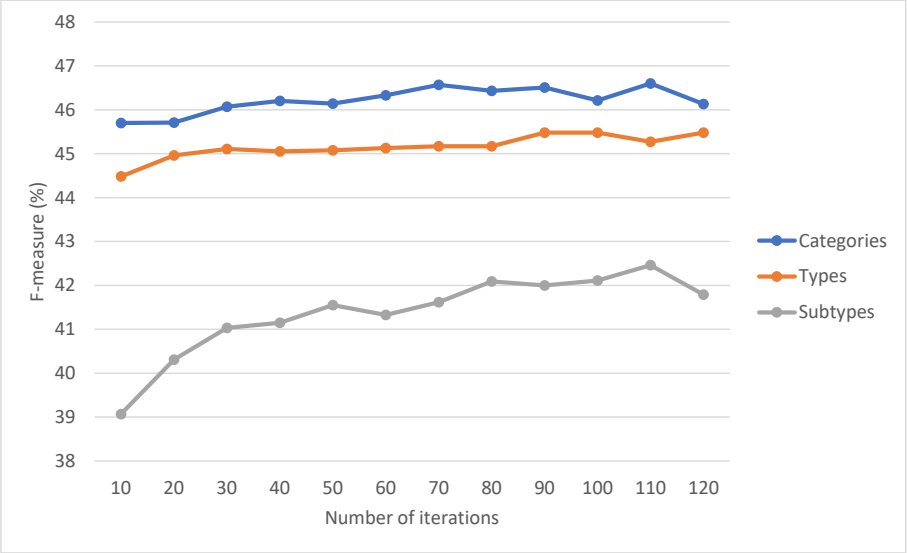

Figure 6.3: Results for iteration values of spaCy.

6.2.2 spaCy

To my knowledge, the only hyperparameter available for tuning is the number of iterations. I used the default value of 10 iterations, and tested for all values between 10 and 120 iterations (with a 10 iterations span). Results, in Figure 6.3, show that the higher the iterations, the better the results. Table 6.4 shows that the best obtained result was for 110 iterations, where the categories level got an F-measure of 46.60% and subtypes an F-measure of 42.46%. On the other hand, types seem to get their highest score with higher than 90 iterations, with an F-measure of 45.48%. All results can be seen in Table C.3.

Table 6.4: Results for iteration values of spaCy.

Value	Categories	Types	Subtypes
10 (default)	45.70%	44.48%	39.07%
90	46.51%	**45.48%**	42.00%
110	**46.60%**	45.27%	**42.46%**

6.2.3 Stanford CoreNLP

Stanford CoreNLP allows specifying some of the parameters in a properties file. I decided to check the influence of the tolerance. The results for the tested tolerance values can be seen in Figure 6.4 (and in Table C.4), with the most relevant results displayed in Table 6.5. The default value (1e-4) got an F-measure of 54.15%, and the best obtained result an F-measure of 54.31%, for the 1e-3 tolerance value. As I have previously stated, the *types* and *subtypes* levels were not tested due to the amount of computational effort required. I also checked the influence of the *epsilon* parameter, however it did not have any effect at all. I believe this parameter is not used

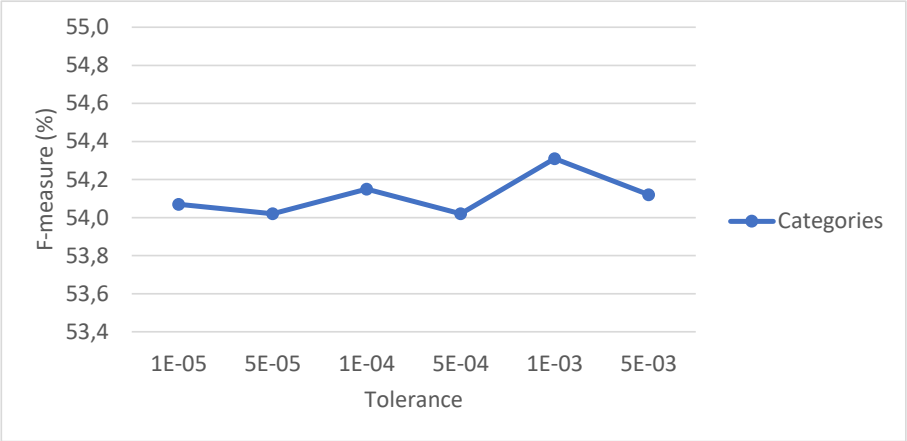

Figure 6.4: Results for tolerance values of Stanford CoreNLP.

with this configuration. On a side-note, despite not being a hyperparameter, I also tested changing the value of *maxNGramLeng*. Results for this last parameter can be seen in Table C.5, showing that the performance increases with higher values. However, this was not used in the SIGARRA dataset (Chapter 7), leaving the default value of 6.

Table 6.5: Results for tolerance values of Stanford CoreNLP.

Value	Categories	Types	Subtypes
1e-4 (default)	54.15%	-	-
1e-3	**54.31%**	-	-

6.2.4 NLTK

NLTK was trained using three different classifiers - `NaiveBayes`, `MaxEnt` and `DecisionTree`. Only two of them allow changing hyperparameters (`MaxEnt` and `DecisionTree`). The results for the hyperparameter study for these classifiers are presented next.

6.2.4.1 MaxEnt

The Maximum Entropy classifier allows tuning three different hyperparameters, namely *max_iter* (maximum iterations) and *min_lldelta* (minimum log-likelihood delta). Table 6.6 shows the results for the Maximum Entropy classifier. Table 6.6a shows the results from the default number (10) to the highest tested value (120), with a delta of 10. It shows that the number iterations does not change the outcome of the classifier, this is probably due to the default *min_lldelta* hyperparameter value being so high that it stops the algorithm before reaching any maximum iteration value.

I experimented with different *min_lldelta* values, with two different iteration values. The most relevant results can be seen in Table 6.6b, where the first three columns have 10 iterations (default) and the next three columns have 100 iterations (See Table C.6 for detailed results). The

results show that the performance stays the same above the 0.01 value, starting at the 0.05 value. However, when *min_lldelta* is below that threshold, it starts to get better results. This means that this hyperparameter is the main reason that the algorithm performs so poorly. Also, it is important to note that this classifier overflows when training in the *types* and *subtypes* levels, leading to the poor results shown this section.

Table 6.6: Results for NLTK's Maximum Entropy classifier.

(a) Results for maximum iteration (*max_iter*) values for MaxEnt.

Value	Categories	Types	Subtypes
10 (default)	1.11%	1.68%	0.29%
10 - 120 (All)	1.11%	1.68%	0.29%

(b) Results for minimum log-likelihood delta (*min_lldelta*) values for MaxEnt.

Value	Categories	Types	Subtypes	Categories	Types	Subtypes
0	**22.28%**	1.68%	0.29%	**35.24%**	1.68%	0.29%
0.01	22.28%	1.68%	0.29%	24.40%	1.68%	0.29%
0.05	1.11%	1.68%	0.29%	1.11%	1.68%	0.29%
0.1 (default)	1.11%	1.68%	0.29%	1.11%	1.68%	0.29%

6.2.4.2 DecisionTree

The Decision Tree classifier allows changing three distinct hyperparameters, namely *support_cutoff*, *depth_cutoff* and *entropy_cutoff*. The results for these hyperparameters are presented in Table 6.7, where Table 6.7a shows the most relevant results for the multiple values for the *support_cutoff* hyperparameter, having a default value of 10. It shows that this hyperparameter has little effect in the outcome of the classifier. That being said, in the category level the best achieved result was with a support cut-off of 16 (with an F-measure of 26.18%), the type level had the highest value at 14 (with an F-measure of 24.31%) and the subtypes level had the best value at 11 (with an F-measure of 32.63%). Check Table C.7 for the detailed results.

Table 6.7b shows the results for the multiple values for the *depth_cutoff* hyperparameter, having a default value of 100. It shows that this hyperparameter has no effect whatsoever in the outcome of the classifier. So, all results were the same, with categories having an F-measure of 26.14%, types with 24.24% and subtypes with 32.61%. Table 6.7c shows the best results for the *entropy_cutoff* hyperparameter, having a default value of 0.05. As we can see, this hyperparameter had the highest impact of the three available. Overall, it shows that the higher the entropy cut-off value, the higher the outcome, reaching the best results at around 0.08 and 0.09, namely categories at 26.36% F-measure, types at 24.29% and subtypes at 32.70%. See Table C.8 for all of the results.

Table 6.7: Results for NLTK's Decision Tree classifier.

(a) Results for support cut-off in Decision Tree classifier.

Value	Categories	Types	Subtypes
10 (default)	26.14%	24.24%	32.61%
11	26.14%	24.25%	**32.63%**
14	26.13%	**24.31%**	32.63%
16	**26.18%**	24.27%	32.50%

(b) Results for depth cut-off in Decision Tree classifier.

Value	Categories	Types	Subtypes
100 (default)	26.14%	24.24%	32.61%
70-120 (All)	26.14%	24.24%	32.61%

(c) Best results for entropy cut-off in Decision Tree classifier.

Value	Categories	Types	Subtypes
0.05 (default)	26.14%	24.24%	32.61%
0.08	**26.36%**	**24.29%**	32.69%
0.09	26.36%	24.29%	**32.70%**

6.3 Summary

While the best solution for this study would be a grid search for hyperparameter tuning, I was not able to do it because of time issues (a grid search would mean having to perform 30696 runs instead of the performed 455). However, this study proved to be valuable nonetheless. Table 6.8 presents a summary of the results obtained, showing that some hyperparameters indeed have effects on the tools performance. In OpenNLP, I managed to get almost a 2% improvement by using a cut-off of 4 instead of 5. I also noticed improvements by changing the iterations, but only improving up to 51.52%, with 170 iterations. In addition, spaCy's performance also improved by almost 1% by changing the number of iterations, with the best result for 110 iterations. Furthermore, for Stanford CoreNLP, it proved to be more difficult to find the available hyperparameters. That being said, I obtained a better F-measure by changing the tolerance hyperparamenter to *1e-3*, resulting in an F-measure of 54.31%. Finally, NLTK's Naive Bayes classifier did not have any hyperparameters, so only the remaining two classifiers were tested. The Decision Tree classifier got an improvement of 0.22%, with an entropy_cutoff of 0.08, and also there was a smaller improvement by changing the support_cutoff to 16. The best improvement was, without a doubt, from NLTK's Maximum Entropy classifier, which increased more than 34%, which leads to the conclusion that the default parameters are not even close to acceptable. It is important to note that the default results obtained in this chapter may differ from the ones in Chapter 5 because this time repeated holdout was used

instead of repeated 10-fold cross-validation.

Table 6.8: Summary results for the category level, for all tools.

Tool	Default F-measure	Best configurations	Best F-measure
OpenNLP	50.90%	cutoff=4	52.38%
		iterations = 170	51.52%
SpaCy	45.70%	iterations=110	46.60%
Stanford CoreNLP	54.14%	tolerance=1e-3	54.31%
NLTK DT	26.14%	entropy_cutoff=0.08	26.36%
		support_cutoff=16	26.18%
NLTK ME	1.11%	min_lldelta=0, iterations=100	35.24%

Chapter 7

SIGARRA News Corpus

This chapter presents the results obtained when training the selected tools with the SIGARRA News Corpus, which I manually annotated. Furthermore, it also explains how the corpus was built and the necessary transformations for it to be used as a training set with each tool.

I decided to prepare a new annotated corpus — SIGARRA News Corpus — due to the following reasons:

- Get better results when performing Named Entity Recognition (NER) in SIGARRA news.
- The HAREM collection is outdated, primarily because of the latest Portuguese orthographic reform (1990).
- The HAREM collection does not have the same entity classes required for the ANT project.

For the stated reasons, I prepared this new annotated corpus from a subset of SIGARRA news from its multiple domains. I used the Brat rapid annotation tool for this task. For more information on this dataset check Section 3.3.

7.1 Transformation

Similarly to the HAREM dataset, this dataset also went through the transformations described in Chapter 4. However, it did not go through the same initial transformation. After manually annotating the Named Entities (NE) in all 905 news, I converted the annotations from the standoff format to a XML format similar to the HAREM format. This was done with the purpose of reusing the pipeline already created for the HAREM dataset. After that, I joined the different files of news, into a single XML file, with each news being a `DOC` element, with an *id* attribute being the news id. Afterwards, as I previously stated, this dataset was transformed in the required input formats for all the four tools. An example of a news document excerpt can be seen in Listing 7.1.

```
<DOC id="feup-12345">
    Os estudantes estudam na <EM CATEG="Organizacao">Universidade do Porto</EM>.
</DOC>
```

Listing 7.1: SIGARRA News Corpus example format.

7.2 Results

I evaluated the same four tools — OpenNLP, spaCy, Stanford CoreNLP and NLTK —, using repeated 10-fold cross validation, with four repeats. The tested hyperparameters were the default ones, and the best ones found in the hyperparameter study (Section 6.2).

7.2.1 OpenNLP

For OpenNLP, the default configuration has 100 iterations and a cut-off value of 5, which resulted in an F-measure of 83.52%. With the best configuration found in the previous chapter — 170 iterations and cut-off value of 4 — I obtained an F-measure of 83.74%. The results can be seen in Table 7.1.

Table 7.1: Results for OpenNLP with SIGARRA News Corpus.

Precision	Recall	F-measure	Hyperparameters
88.43%	79.12%	83.52%	*default*
88.03%	79.85%	83.74%	Iterations=170, Cutoff=4

7.2.2 spaCy

For spaCy, the default configuration has 10 iterations, which resulted in an F-measure of 79.95%. With the best configuration found in the previous chapter — 110 iterations — I obtained an F-measure of 81.27%. The results can be seen in Table 7.2.

Table 7.2: Results for spaCy with SIGARRA News Corpus.

Precision	Recall	F-measure	Hyperparameters
84.85%	75.58%	79.95%	*default*
83.95%	78.76%	81.27%	Iterations=110

7.2.3 Stanford CoreNLP

Table 7.3 shows the results obtained, while using the SIGARRA News Corpus, for Stanford CoreNLP. I obtained an F-measure of 86.78% with the best configuration (tolerance=1e-3), with only a slight decrease from the default configuration (86.86%).

Table 7.3: Results for Stanford CoreNLP with SIGARRA News Corpus.

Precision	Recall	F-measure	Hyperparameters
89.80%	84.10%	86.86%	*default*
89.81%	83.95%	86.78%	tolerance=1e-3

7.2.4 NLTK

As for NLTK, the results for the three classifiers can be seen in Table 7.4. Since `NaiveBayes` does not have hyperparameters it only presents the default result, with an F-measure of 58.36%. The `DecisionTree` classifier has `entropy_cutoff=0.05`, `support_cutoff=10` and `depth_cutoff=100` as the default configuration, resulting in an F-measure of 62.37%. On the other hand, the best configuration has `entropy_cutoff=0.08`, `support_cutoff=16` (and `depth_cutoff=100` equal to the default configuration), obtaining a smaller F-measure of 62.26%. Finally, the `MaxEnt` classifier got an F-measure of 5.11% with the default configuration (`iterations=10`, `min_lldelta=0.1`), and 38.49% with the best configuration (`iterations=100`, `min_lldelta=0`).

It is interesting to note that the `MaxEnt` classifier got a similar problem as the *types* and *subtypes* levels in the HAREM dataset, overflowing when training, in some folds. If those folds hadn't overflowed, it would have gotten an F-measure close to 75% with the best configuration.

Table 7.4: Results for NLTK with SIGARRA News Corpus.

Classifier	Precision	Recall	F-measure	Hyperparameters
NaiveBayes	54.47%	62.86%	58.36%	*default*
DecisionTree	56.03%	70.32%	62.37%	*default*
	55.93%	70.21%	62.37%	entropy_cutoff=0.08, support_cutoff=16
MaxEnt	16.29%	3.03%	5.11%	*default*
	45.30%	33.47%	38.49%	Iterations=100, min_lldelta=0

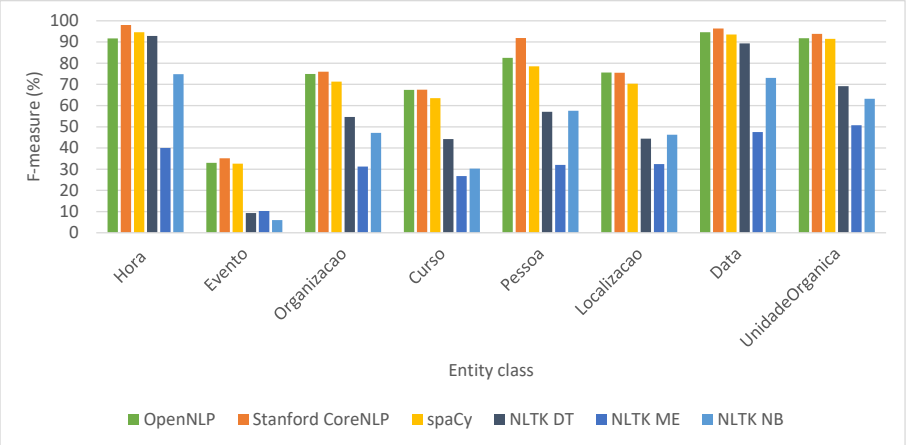

Figure 7.1: F-measure by entity class, by tool.

7.3 Comparison by entity class

Figure 7.1 shows the comparison between the F-measures for every tool, regarding the entity classes. All results can be seen in Table D.4. It is clear that Stanford CoreNLP has the best score over all entity classes, except for *Localizacao*, where OpenNLP scores higher, with an F-measure of 75.61% (and Stanford CoreNLP with 75.47%). Overall, *Data* and *Hora* are the highest scoring entity classes, and *Evento* the lowest. The lowest scoring algorithm is NLTK's Maximum Entropy in all entity classes, except for *Evento*, where the lowest ranking classifier was NLTK's Naive Bayes.

7.4 Summary

The results were significantly better with the SIGARRA News Corpus than with the HAREM golden collection. This could be due to the SIGARRA corpus having a lot of documents with the same structure, making it easier to learn. Also the SIGARRA News Corpus is bigger than the HAREM collection, which could improve the training process. The tools' ranking remained the same as the one in the baseline ranking (Section 5.2), namely Stanford CoreNLP obtained the best F-measure, with 86.64%, followed by OpenNLP with 83.19%, then spaCy with 80.30%, and finally NLTK with 64.70% with the Decision Tree classifier.

Overall the hyperparameter study proved to be beneficial, because almost all "best configurations" had better results, apart from the NLTK's Decision Tree classifier and Stanford CoreNLP, which had better results with the default configuration. It would be interesting, in the future, to make the same hyperparameter study for the SIGARRA News Corpus, because the hyperparameters could have a different effect on different corpora.

Chapter 8

Conclusions and future work

This chapter presents the conclusions of this dissertation regarding the results obtained in the baseline and best performances of the selected tools against the HAREM dataset and the SIGARRA corpus, and the achieved objectives and major contributions. In addition, some possible future work in this area is described, namely improve the hyperparameter study, or even continuing improving the SIGARRA corpus.

8.1 Contributions

This dissertation allowed me to make several contributions to the scientific community, in particular to the Named Entity Recognition (NER) field. Here are the most significant ones:

- Provided an out-of-the-box assessment of several popular tools, based on the HAREM dataset.
- Made a hyperparameter study, improving over the default configuration and describing the impact of individual hyperparameters for each tool.
- Developed a number of scripts to transform datasets, more specifically the HAREM dataset, into various formats. Including the dataset already transformed[1].
- Provided documents with guidelines for each tool, to make it easier to know how to run, train and test each tool. (See Appendix E)
- Prepared an annotated Portuguese dataset (SIGARRA News Corpus [PND17]), twice as big as HAREM.
- Created, and published, trained NER models for the HAREM dataset [Pir17a] and the SIGARRA News Corpus [Pir17b], for every tool.

[1] Check https://github.com/arop/ner-re-pt/tree/master/datasets/harem. Accessed: 2017-06-27.

- Proposed the best tool for the context of SIGARRA.

8.2 Conclusions

The main goal of this dissertation was achieved, as I was able to find a tool for entity extraction that works in the SIGARRA news domain. Not only it works in this domain, but results show that it can also work on other domains in the Portuguese language, which by itself is a major contribution since this area — Named Entity Extraction for the Portuguese language — is underdeveloped in comparison to other languages.

First, I established a baseline performance of well known tools, with the HAREM dataset, with the best result, by Stanford CoreNLP, being the same as the latest work with this dataset, namely an F-measure of 56.10%. This was followed by OpenNLP (53.63%), spaCy (48.81%) and NLTK (30.97%). This shows that machine learning classifiers can perform as well as a pattern-based algorithm, in the Portuguese language. However, results are still below the state of the art for other languages, such as English, which already show F-measures higher than 90%. Comparison between entity classes (see Appendix D) leads me to conclude that the HAREM dataset is too fine-grained, it has too many entity classes in comparison to other datasets, for example the ones used in the MUC conference.

Another contribution was a set of documents with guidelines (see Appendix E) for how to run the selected tools (Stanford CoreNLP, OpenNLP, spaCy and NLTK) with a custom dataset. More specifically, I laid out the requirements for every tool, with regards to the input dataset format and how to transform the datasets to that format; how to train a custom model with a custom dataset and custom entity classes; and finally how to use the model to perform NER.

After establishing a baseline performance, I performed a hyperparameter study, which allowed me to select the best configurations for each tool, with the HAREM dataset. Again, the best results showed that the ranking of the tools' performance remained the same, with Stanford CoreNLP taking the lead. This experience allowed me to improve the baseline results by an average of 2% for every tool, except for NLTK's Maximum Entropy classifier, for which I managed to get an improvement of almost 35%, leading me to conclude that the default parameters were not well chosen, at all. The best performing models are published in INESC TEC's CKAN research data repository [Pir17a]. These models were trained with the HAREM dataset, producing three different models, one per entity level, for each tool, and can be directly used for Portuguese NER.

The development of a new annotated dataset for the Portuguese language — SIGARRA News Corpus —, using Brat, is a contribution to the scientific community and in particular to the ANT project. Apart from the HAREM collection, this is a novel Portuguese (from Portugal) annotated corpus. In addition, this dataset has entity classes not present in other datasets, which are specific to the SIGARRA academic domain, such as Course or Organic Unit. The developed corpus is twice the size of the HAREM collection (HAREM has approximately 86k tokens, and SIGARRA has 185k tokens), with twice the number of entity annotations (HAREM has 7255 annotations, and SIGARRA has 12644 annotations). Due to legal copyright processes, in the time of this

dissertation, I was not able to publish this dataset, however it is planned to be published in the near future in INESC TEC's CKAN [PND17].

Finally, using the best configurations found in the hyperparameter study, I trained a NER model, with the SIGARRA News Corpus, for all tools. As with the results obtained with the HAREM dataset, the tools ranking remained the same, with Stanford CoreNLP having the best result with an F-measure of 86.86%. The best performing models are published in INESC TEC's CKAN [Pir17b]. These models were trained with the SIGARRA News Corpus for each tool, and can be directly used for Portuguese NER.

8.3 Future work

This experience presented promising results, showing that it is possible to train well-established tools with a Portuguese corpus and obtain acceptable results. However, it is important to note that the training dataset is central and, in particular, expanding the dataset can improve results. In addition, the hyperparameter study was not ideal but feasible, because of time constraints. Instead, I should have done hyperparameter tuning with a grid search in order to find the best configuration, and not only the best single value for each standalone hyperparameter. So future work could be performing a more thorough study in this matter.

Apart from Named Entity Recognition, it would be interesting to have these or other tools perform entity linking and Relation Extraction, as these tasks would improve greatly the ANT search engine. Finally, in this dissertation I created trained models with the SIGARRA News Corpus, which should be used, together with the tools, as an API integrated in the ANT search engine in the future.

References

[AFM+08] Carlos Amaral, Helena Figueira, Afonso Mendes, Pedro Mendes, Cláudia Pinto, and Tiago Veiga. Adaptação do sistema de reconhecimento de entidades mencionadas da Priberam ao HAREM. In Cristina Mota and Diana Santos, editors, *Desafios na avaliação conjunta do reconhecimento de entidades mencionadas: O Segundo HAREM*, chapter 9, pages 171–179. Linguateca, 2008.

[AM03] Masayuki Asahara and Yuji Matsumoto. Japanese Named Entity extraction with redundant morphological analysis. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - NAACL '03*, volume 1, pages 8–15, Edmonton, Canada, 2003. Association for Computational Linguistics.

[Bar16] Caroline Barrière. Pattern-Based Relation Extraction. In *Natural Language Understanding in a Semantic Web Context*, chapter IV, pages 205–229. Springer International Publishing, Cham, 2016.

[BC99] Matthew Berland and Eugene Charniak. Finding parts in very large corpora. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 57–64, Morristown, NJ, USA, 1999. Association for Computational Linguistics.

[BDVR08] Mírian Bruckschen, José Guilherme Camargo De Souza, Renata Vieira, and Sandro Rigo. Sistema SeRELeP para o reconhecimento de relações entre entidades mencionadas. In Cristina Mota and Diana Santos, editors, *Desafios na avaliação conjunta do reconhecimento de entidades mencionadas: O Segundo HAREM*, chapter 14, pages 247–260. Linguateca, 2008.

[Bic07] Eckhard Bick. Functional aspects on Portuguese NER. In Diana Santos and Nuno Cardoso, editors, *Reconhecimento de entidades mencionadas em português: Documentação e actas do HAREM, a primeira avaliação conjunta na área*, chapter 12, pages 145–155. Linguateca, 2007.

[BKL09] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media Inc, 2009.

[BMSW97] Daniel M. Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. Nymble: a High-Performance Learning Name-finder. In *5th International Conference on Applied Natural Language Processing*, pages 194–201, Washington, DC, 1997. Association for Computational Linguistics.

REFERENCES

[Bri99] Sergey Brin. Extracting Patterns and Relations from the World Wide Web. In *Selected Papers from the International Workshop on The World Wide Web and Databases*, pages 172–183, London, UK, 1999. Springer-Verlag.

[BSAG98] Andrew Borthwick, John Sterling, Eugene Agichtein, and Ralph Grishman. Exploiting Diverse Knowledge Sources via Maximum Entropy in Named Entity Recognition. In *Proceedings of the Sixth Workshop on Very Large Corpora*, pages 152–160, 1998.

[Car06] Nuno Francisco Pereira Freire Cardoso. Avaliação de Sistemas de Reconhecimento de Entidades Mencionadas. Master's thesis, University of Porto, 2006.

[Car07] Wesley Seidel Carvalho. Reconhecimento de entidades mencionadas em português. Master's thesis, Universidade de São Paulo, São Paulo, February 2007.

[Car08] Nuno Cardoso. REMBRANDT -Reconhecimento de Entidades Mencionadas Baseado em Relações e ANálise Detalhada do Texto. In Cristina Mota and Diana Santos, editors, *Desafios na avaliação conjunta do reconhecimento de entidades mencionadas: O Segundo HAREM*, chapter 11, pages 195–211. Linguateca, 2008.

[Cha08] Marcirio Silveira Chaves. Geo-ontologias e padrões para reconhecimento de locais e de suas relações em textos: o SEI-Geo no Segundo HAREM. In Cristina Mota and Diana Santos, editors, *Desafios na avaliação conjunta do reconhecimento de entidades mencionadas: O Segundo HAREM*, chapter 13, pages 231–245. Linguateca, 2008.

[CM98] Nancy Chinchor and Elaine Marsh. MUC-7 Information Extraction Task Definition. In *Proceedings of a 7th Message Understanding Conference (MUC-7)*, pages 359–367, 1998.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[Dad13] Das Dad. Corpus processor. https://github.com/dasdad/corpus-processor, 2013. Accessed: 2017-06-27.

[dAFLV14] Daniela O. F. do Amaral, Evandro Fonseca, Lucelene Lopes, and Renata Vieira. Comparative Analysis of Portuguese Named Entities Recognition Tools. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 244–249. European Language Resources Association (ELRA), 2014.

[dAV13] Daniela O. F. do Amaral and Renata Vieira. O Reconhecimento de Entidades Nomeadas por meio de Conditional Random Fields para a Língua Portuguesa. In *Proceedings of the 9th Brazilian Symposium in Information and Human Language Technology*, pages 59–68, 2013.

[DB96] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2(1):263–286, 1996.

[DMP+04] George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. The Automatic Content Extraction (ACE) Program Tasks, Data, and Evaluation. In *4th International Conference on Language Resources and Evaluation*, pages 24–30, Lisbon, Portugal, 2004.

[EB10] Asif Ekbal and Sivaji Bandyopadhyay. Named Entity Recognition using Support Vector Machine : A Language Independent Approach. *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 4(September):155–170, 2010.

[FGM05] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pages 363–370, 2005.

[FKT+07] Óscar Ferrández, Zornitsa Kozareva, Antonio Toral, Rafael Muñoz, and Andrés Montoyo. Tackling HAREM's Portuguese Named Entity Recognition task with Spanish resources. In Diana Santos and Nuno Cardoso, editors, *Reconhecimento de entidades mencionadas em português: Documentação e actas do HAREM, a primeira avaliação conjunta na área*, chapter 11, pages 137–144. Linguateca, 2007.

[FMS+10] 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)*, number 3, pages 3630–3637, 2010.

[Fou] The Apache Software Foundation. Apache OpenNLP. https://opennlp.apache.org/. Accessed: 2017-06-27.

[FSM+09] Cláudia Freitas, Diana Santos, Cristina Mota, Hugo Gonçalo Oliveira, and Paula Carvalho. Relation detection between named entities: report of a shared task. In *Proceedings of the NAACL HLT Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 129–137, Boulder, Colorado, 2009. Association for Computational Linguistics.

[GDL+13] Abhishek Gattani, AnHai Doan, Digvijay S. Lamba, Nikesh Garera, Mitul Tiwari, Xiaoyong Chai, Sanjib Das, Sri Subramaniam, Anand Rajaraman, and Venky Harinarayan. Entity extraction, linking, classification, and tagging for social media. In *Proceedings of the VLDB Endowment*, volume 6, pages 1126–1137. VLDB Endowment, August 2013.

[GS96] Ralph Grishman and Beth Sundheim. Message Understanding Conference-6. In *Proceedings of the 16th conference on Computational linguistics*, volume 1, pages 466–471, Morristown, NJ, USA, 1996. Association for Computational Linguistics.

[Hea92] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics*, volume 2, pages 539–545, Morristown, NJ, USA, 1992. Association for Computational Linguistics.

[Hona] Matthew Honnibal. spaCy. https://spacy.io/. Accessed: 2017-06-27.

REFERENCES

[Honb] Matthew Honnibal. Thinc: Practical Machine Learning for NLP in Python. https://github.com/explosion/thinc. Accessed: 2017-06-27.

[Kon12] Michal Konkol. Named Entity Recognition PhD Study Report. Technical report, University of West Bohemia in Pilsen, Pilsen, Czech Republic, 2012.

[KT07] Jun'ichi Kazama and Kentaro Torisawa. Exploiting Wikipedia as External Knowledge for Named Entity Recognition. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 698–707, Prague, Czech Republic, 2007. Association for Computational Linguistics.

[KT08] Jun'ichi Kazama and Kentaro Torisawa. Inducing Gazetteers for Named Entity Recognition by Large-scale Clustering of Dependency Relations. In *Proceedings of ACL-08: HLT*, pages 407–415, Columbus, Ohio, USA, 2008. Association for Computational Linguistics.

[LMP01] John Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the eighteenth international conference on machine learning*, pages 282–289, 2001.

[MD07] Ruy Luiz Milidiú and Julio Cesar Duarte. Machine Learning Algorithms for Portuguese Named Entity Recognition. *Intel. Artif.*, 11(36), 2007.

[ML03] Andrew McCallum and Wei Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, volume 4, pages 188–191, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[MMG99] Andrei Mikheev, Marc Moens, and Claire Grover. Named Entity recognition without gazetteers. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*, pages 1–8, Morristown, NJ, USA, 1999. Association for Computational Linguistics.

[Mot08] Cristina Mota. R3M, uma participação minimalista no Segundo HAREM. In Cristina Mota and Diana Santos, editors, *Desafios na avaliação conjunta do reconhecimento de entidades mencionadas: O Segundo HAREM*, chapter 10, pages 181–193. Linguateca, 2008.

[MSC+08] Cristina Mota, Diana Santos, Paula Carvalho, Cláudia Freitas, and Hugo Gonçalo Oliveira. Apresentação detalhada das coleções do Segundo HAREM. In Cristina Mota and Diana Santos, editors, *Desafios na avaliação conjunta do reconhecimento de entidades mencionadas: O Segundo HAREM*, chapter Apêndice H, pages 355–377. Linguateca, 2008.

[MSCMA09] Mónica Marrero, Sonia Sánchez-Cuadrado, Jorge Morato, and Yorgos Andreadakis. Evaluation of Named Entity Extraction Systems. *Research In Computer Science*, 41:47–58, 2009.

REFERENCES

[OMF+08] Hugo Gonçalo Oliveira, Cristina Mota, Cláudia Freitas, Diana Santos, and Paula Carvalho. Avaliação à medida do Segundo HAREM. In Cristina Mota and Diana Santos, editors, *Desafios na avaliação conjunta do reconhecimento de entidades mencionadas: O Segundo HAREM*, chapter 5, pages 97–129. Linguateca, 2008.

[Per14] Jacob Perkins. *Python 3 Text Processing with NLTK 3 Cookbook*. Packt Publishing, 2014.

[Pir17a] André Pires. HAREM NER Models for OpenNLP, Stanford CoreNLP, spaCy, NLTK. https://rdm.inesctec.pt/dataset/cs-2017-005, June 2017.

[Pir17b] André Pires. SIGARRA News Corpus NER Models for OpenNLP, Stanford CoreNLP, spaCy, NLTK. https://rdm.inesctec.pt/dataset/cs-2017-006, June 2017.

[PND17] André Pires, Sérgio Nunes, and José Devezas. SIGARRA News Corpus. https://rdm.inesctec.pt/dataset/cs-2017-004, June 2017.

[PPA16] Laura Pandolfo, Luca Pulina, and Giovanni Adorni. A Framework for Automatic Population of Ontology-Based Digital Libraries. In *AI*IA 2016 Advances in Artificial Intelligence*, pages 406–417. Springer International Publishing, 2016.

[PRPM07] Natalia Ponomareva, Paolo Rosso, Ferran Pla, and Antonio Molina. Conditional Random Fields vs. Hidden Markov Models in a biomedical Named Entity Recognition task. In *Proc. of Int. Conf. Recent Advances in Natural Language Processing, RANLP*, pages 479–483, 2007.

[Rau91] Lisa F. Rau. Extracting company names from text. In *Proceedings of the Seventh IEEE Conference on Artificial Intelligence Application*, volume i, pages 29–32. IEEE Comput. Soc. Press, 1991.

[RJS+16] Conceição Rocha, Alípio Jorge, Roberta Sionara, Paula Brito, Carlos Pimenta, and Solange Rezende. PAMPO: using pattern matching and pos-tagging for effective Named Entities recognition in Portuguese. *CoRR*, abs/1612.0, December 2016.

[Sar06] Luís Sarmento. SIEMÊS - A named-entity recognizer for Portuguese relying on similarity rules. 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: 7th International Workshop, PROPOR 2006, Itatiaia, Brazil, May 13-17, 2006. Proceedings*, volume 3960 LNAI, pages 90–99. Springer Berlin Heidelberg, 2006.

[Sar07] Luís Sarmento. O SIEMÊS e a sua participação no HAREM e no Mini-HAREM. In Cristina Mota, editor, *Reconhecimento de entidades mencionadas em português: Documentação e actas do HAREM, a primeira avaliação conjunta na área*, chapter 14, pages 173–189. Linguateca, 2007.

[SC06] Diana Santos and Nuno Cardoso. A Golden Resource for Named Entity Recognition in Portuguese. In *Proceedings of the 7th International Workshop, PROPOR 2006*, pages 69–79. Springer Berlin Heidelberg, 2006.

[SFMB07] Horacio Saggion, Adam Funk, Diana Maynard, and Kalina Bontcheva. Ontology-based Information Extraction for Business Intelligence. In *Proceedings of the 6th International The Semantic Web and 2Nd Asian Conference on Asian Semantic Web Conference*, pages 843–856, Berlin, Heidelberg, 2007. Springer-Verlag.

REFERENCES

[Sin12] Amit Singhal. Introducing the Knowledge Graph: things, not strings. https://googleblog.blogspot.pt/2012/05/introducing-knowledge-graph-things-not.html, May 2012.

[SO06] Andrew Smith and Miles Osborne. Using gazetteers in discriminative information extraction. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 133–140, New York City, New York, 2006. Association for Computational Linguistics.

[Sol07] Thamar Solorio. MALINCHE: A NER system for Portuguese that reuses knowledge from Spanish. In Diana Santos and Nuno Cardoso, editors, *Reconhecimento de entidades mencionadas em português: Documentação e actas do HAREM, a primeira avaliação conjunta na área,*, chapter 10, pages 123–136. Linguateca, 2007.

[SPC06] Luís Sarmento, Ana Sofia Pinto, and Luís Cabral. REPENTINO – A Wide-Scope Gazetteer for Entity Recognition in Portuguese. *Lecture Notes in Computer Science*, 3960:31–40, 2006.

[SPT+12] Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. brat: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations Session at EACL 2012*, Avignon, France, April 2012. Association for Computational Linguistics.

[SSCV06] Diana Santos, Nuno Seco, Nuno Cardoso, and Rui Vilela. HAREM: An Advanced NER Evaluation Contest for Portuguese. In *Proceedings of the 5th International Conference on Language Resources and Evaluation, LREC'2006*, pages 1986–1991, 2006.

[TD03] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, volume 4, pages 142–147, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[TSO11] Jorge Teixeira, Luís Sarmento, and Eugénio Oliveira. A Bootstrapping Approach for Training a NER with Conditional Random Fields. In *Progress in Artificial Intelligence*, pages 664–678. Springer Berlin Heidelberg, 2011.

[WD10] Daya C. Wimalasuriya and Dejing Dou. Ontology-based information extraction: An introduction and a survey of current approaches. *Journal of Information Science*, 36(3):306–323, 2010.

[YALR13] Ugan Yasavur, Reza Amini, Christine Lisetti, and Naphtali Rishe. Ontology-based Named Entity Recognizer for Behavioral Health. *In Proceedings of the 26th International Florida Artificial Intelligence Research Society Conference*, (3):249–254, 2013.

[ZS02] Guodong Zhou and Jian Su. Named Entity Recognition using an HMM-based Chunk Tagger. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 473–480, Philadelphia, 2002. Association for Computational Linguistics.

Appendix A

HAREM classes

Table A.1: Categories, types and subtypes for HAREM's golden collection.

Categories	Types	Subtypes
Abstraccao	Disciplina	
	Estado	
	Ideia	
	Nome	
Acontecimento	Efemeride	
	Evento	
	Organizado	
Coisa	Classe	
	MembroClasse	
	Objecto	
	Substancia	
Local	Fisico	Ilha, AguaCurso, Planeta, Regiao, Relevo, AguaMassa, Outro
	Humano	Rua, Pais, Divisao, Regiao, Construcao, Outro
	Virtual	ComSocial, Sitio, Obra, Outro
Obra	Arte	
	Plano	
	Reproduzida	
Organizacao	Administracao	
	Empresa	
	Instituicao	
Pessoa	Cargo	
	GrupoCargo	
	GrupoInd	
	GrupoMembro	

Continued on next page

Table A.1 – continued from previous page

Categories	Types	Subtypes
	Individual	
	Membro	
	Povo	
Tempo	Duracao	
	Frequencia	
	Generico	
	Tempo_Calend	Hora, Intervalo, Data, Outro
Valor	Classificacao	
	Moeda	
	Quantidade	
Outro		

Appendix B

SIGARRA News Corpus entity distribution

Table B.1: Distribution of SIGARRA news and average number of characters by domain.

SIGARRA domain	News	%	Average number of characters
FLUP	128	14.14%	1151
FAUP	116	12.82%	1343
FEUP	114	12.60%	1384
FMUP	110	12.15%	831
FCUP	83	9.17%	1156
FPCEUP	73	8.07%	1324
SPUP	58	6.41%	2340
FADEUP	45	4.97%	901
ICBAS	39	4.31%	1053
FDUP	36	3.98%	582
FBAUP	27	2.98%	667
FEP	21	2.32%	759
FCNAUP	17	1.88%	986
FFUP	17	1.88%	1635
REITORIA	11	1.21%	1441
FMDUP	8	0.88%	959
UP	7	0.77%	1111
Total	905	100%	(Average) 1154

Table B.2: Distribution of entity annotations in SIGARRA news.

Entity tag	**Number of annotated classes**	**%**
Data	2811	22.23%
Organizacao	2320	18.35%
Pessoa	2159	17.08%
UnidadeOrganica	1814	14.35%
Localizacao	1593	12.60%
Hora	1015	8.03%
Curso	521	4.12%
Evento	411	3.25%
Total	12644	100%

Table B.3: Distribution of entity annotations in SIGARRA news, per domain.

SIGARRA domain	**Hora**	**Evento**	**Organizacao**	**Curso**	**Pessoa**	**Localizacao**	**Data**	**UnidadeOrganica**
FADEUP	66 (15.3%)	18 (4.2%)	68 (15.7%)	10 (2.3%)	35 (8.1%)	57 (13.2%)	128 (29.6%)	50 (11.6%)
FAUP	155 (5.6%)	119 (4.3%)	438 (15.7%)	35 (1.3%)	693 (24.8%)	514 (18.4%)	535 (19.2%)	300 (10.8%)
FBAUP	48 (17.3%)	12 (4.3%)	10 (3.6%)	18 (6.5%)	49 (17.6%)	38 (13.7%)	75 (27%)	28 (10.1%)
FCNAUP	19 (10.2%)	2 (1.1%)	8 (4.3%)	38 (20.4%)	16 (8.6%)	11 (5.9%)	82 (44.1%)	10 (5.4%)
FCUP	53 (6.2%)	22 (2.6%)	211 (24.8%)	30 (3.5%)	111 (13.1%)	137 (16.1%)	188 (22.1%)	98 (11.5%)
FDUP	65 (23.8%)	2 (0.7%)	4 (1.5%)	15 (5.5%)	36 (13.2%)	7 (2.6%)	104 (38.1%)	40 (14.7%)
FEP	20 (12%)	2 (1.2%)	12 (7.2%)	26 (15.6%)	17 (10.2%)	7 (4.2%)	51 (30.5%)	32 (19.2%)
FEUP	84 (4.9%)	52 (3%)	301 (17.6%)	58 (3.4%)	303 (17.7%)	139 (8.1%)	326 (19.1%)	448 (26.2%)
FFUP	71 (26.8%)	15 (5.7%)	32 (12.1%)	21 (7.9%)	7 (2.6%)	27 (10.2%)	63 (23.8%)	29 (10.9%)
FLUP	125 (7.5%)	65 (3.9%)	325 (19.5%)	46 (2.8%)	344 (20.7%)	247 (14.8%)	360 (21.6%)	153 (9.2%)
FMDUP	2 (3.6%)	2 (3.6%)	7 (12.5%)	3 (5.4%)	3 (5.4%)	4 (7.1%)	19 (33.9%)	16 (28.6%)
FMUP	110 (8.4%)	57 (4.4%)	256 (19.6%)	38 (2.9%)	199 (15.2%)	169 (13%)	282 (21.6%)	194 (14.9%)
FPCEUP	122 (10.1%)	20 (1.6%)	287 (23.7%)	69 (5.7%)	159 (13.1%)	148 (12.2%)	269 (22.2%)	139 (11.5%)
ICBAS	35 (6%)	14 (2.4%)	106 (18.2%)	66 (11.4%)	89 (15.3%)	49 (8.4%)	123 (21.2%)	99 (17%)
REITORIA	34 (19.5%)	6 (3.4%)	27 (15.5%)	1 (0.6%)	32 (18.4%)	14 (8%)	37 (21.3%)	23 (13.2%)
SPUP	2 (0.3%)	0 (0%)	204 (32.6%)	47 (7.5%)	64 (10.2%)	10 (1.6%)	148 (23.7%)	150 (24%)
UP	4 (5.4%)	3 (4.1%)	24 (32.4%)	0 (0%)	2 (2.7%)	15 (20.3%)	21 (28.4%)	5 (6.8%)
Total	1015 (8%)	411 (3.3%)	2320 (18.3%)	521 (4.1%)	2159 (17.1%)	1593 (12.6%)	2811 (22.2%)	1814 (14.3%)

SIGARRA News Corpus entity distribution

Appendix C

Hyperparameter study results

C.1 OpenNLP

Table C.1: Results for cutoff values in OpenNLP.

Value	Categories	Types	Subtypes
0	49.93%	46.73%	47.00%
3	52.05%	**48.90%**	**52.52%**
4	**52.38%**	48.12%	52.35%
5 (default)	50.90%	47.59%	50.76%
6	50.85%	46.41%	50.64%
7	50.21%	46.34%	50.73%
10	49.09%	44.78%	50.65%

Table C.2: Results for iteration values in OpenNLP.

Value	Categories	Types	Subtypes
70	50.75%	47.39%	50.04%
80	50.85%	47.51%	50.52%
90	50.91%	47.54%	50.75%
100 (default)	50.90%	47.59%	50.76%
110	50.94%	47.67%	51.22%
120	51.19%	**47.81%**	51.81%
125	51.31%	47.77%	51.81%
130	51.33%	47.68%	51.91%
135	51.22%	47.68%	51.94%
150	51.31%	47.56%	52.02%
160	51.46%	47.46%	52.20%
170	**51.52%**	47.43%	**52.61%**
180	51.45%	47.50%	52.58%
200	51.38%	47.58%	52.59%

C.2 SpaCy

Table C.3: Results for number of iterations in spaCy.

Value	Categories	Types	Subtypes
10 (default)	45.70%	44.48%	39.07%
20	45.71%	44.96%	40.31%
30	46.07%	45.11%	41.03%
40	46.20%	45.05%	41.15%
50	46.14%	45.08%	41.55%
60	46.33%	45.13%	41.33%
70	46.57%	45.17%	41.62%
80	46.43%	45.17%	42.09%
90	46.51%	**45.48%**	42.00%
100	46.21%	45.48%	42.11%
110	**46.60%**	45.27%	**42.46%**
120	46.13%	45.48%	41.79%

C.3 Stanford CoreNLP

Table C.4: Results for tolerance values in Stanford CoreNLP.

Value	Categories	Types	Subtypes
1e-5	54.07%	-	-
5e-5	54.02%	-	-
1e-4 (default)	54.15%	-	-
5e-4	54.02%	-	-
1e-3	**54.31%**	-	-
5e-3	54.12%	-	-

Table C.5: Results for maxNGramLeng values in Stanford CoreNLP.

Value	Categories	Types	Subtypes
4	53.47%	-	-
5	53.77%	-	-
6 (default)	54.15%	-	-
7	**54.37%**	-	-

C.4 NLTK

Table C.6: Results for min_lldelta with 10 iterations to the left, and 100 iterations to the right, for Maximum Entropy classifier.

Value	Categories	Types	Subtypes	Categories	Types	Subtypes
0	**22.28%**	1.68%	0.29%	**35.24%**	1.68%	0.29%
0.0000001	22.28%	1.68%	0.29%	35.24%	1.68%	0.29%
0.000001	22.28%	1.68%	0.29%	35.24%	1.68%	0.29%
0.00001	22.28%	1.68%	0.29%	35.24%	1.68%	0.29%
0.0001	22.28%	1.68%	0.29%	35.24%	1.68%	0.29%
0.001	22.28%	1.68%	0.29%	32.69%	1.68%	0.29%
0.01	22.28%	1.68%	0.29%	24.40%	1.68%	0.29%
0.05	1.11%	1.68%	0.29%	1.11%	1.68%	0.29%
0.1 (default)	1.11%	1.68%	0.29%	1.11%	1.68%	0.29%
0.15	1.11%	1.68%	0.29%	1.11%	1.68%	0.29%
0.2	1.11%	1.68%	0.29%	1.11%	1.68%	0.29%

Table C.7: Results for support cutoff in Decision Tree classifier.

Value	Categories	Types	Subtypes
3	26.12%	24.25%	32.59%
7	26.14%	24.25%	32.62%
8	26.14%	24.25%	32.61%
9	26.14%	24.24%	32.61%
10 (default)	26.14%	24.24%	32.61%
11	26.14%	24.25%	**32.63%**
12	26.14%	24.28%	32.60%
13	26.13%	24.30%	32.63%
14	26.13%	**24.31%**	32.63%
15	26.17%	24.28%	32.50%
16	**26.18%**	24.27%	32.50%
17	26.18%	24.27%	32.46%
18	26.16%	24.29%	32.46%
19	26.16%	24.27%	32.47%
20	26.14%	24.28%	32.47%

Table C.8: Results for entropy cutoff in Decision Tree classifier.

Value	Categories	Types	Subtypes
0.03	26.14%	24.24%	32.60%
0.04	26.14%	24.24%	32.61%
0.05 (default)	26.14%	24.24%	32.61%
0.06	26.19%	24.24%	32.61%
0.07	26.19%	24.25%	32.62%
0.08	**26.36%**	**24.29%**	32.69%
0.09	26.36%	24.29%	**32.70%**
0.10	26.36%	24.29%	32.70%
0.11	26.36%	24.28%	32.70%
0.12	26.36%	24.28%	32.65%
0.13	26.36%	24.28%	32.65%

Appendix D

Baseline results by entity class

D.1 HAREM

Table D.1: F-measure by entity class (categories), by tool.

Entity class	OpenNLP	Stanford CoreNLP	spaCy	NLTK DT	NLTK ME	NLTK NB
Abstraccao	8.05%	12.95%	2.57%	8.90%	0.00%	0.76%
Acontecimento	31.81%	29.14%	20.17%	16.49%	0.00%	8.46%
Coisa	13.00%	17.67%	4.06%	16.72%	0.00%	4.32%
Local	59.12%	61.74%	51.38%	42.92%	4.81%	38.41%
Obra	33.32%	32.81%	26.82%	6.49%	1.65%	14.67%
Organizacao	48.61%	47.96%	39.47%	26.57%	0.00%	24.16%
Outro	21.14%	7.28%	0.00%	10.40%	0.00%	0.00%
Pessoa	58.55%	62.53%	54.78%	24.05%	0.00%	38.35%
Tempo	67.18%	71.01%	62.54%	28.51%	1.09%	31.65%
Valor	55.63%	55.97%	35.58%	16.74%	0.00%	18.54%

Table D.2: F-measure by entity class (types), by tool.

Entity class	OpenNLP	spaCy	NLTK DT	NLTK ME	NLTK NB
Abstraccao_Disciplina	7.27%	3.93%	12.43%	0.00%	0.29%
Abstraccao_Estado	29.16%	0.00%	0.00%	0.00%	0.00%
Abstraccao_Ideia	0.00%	0.00%	0.00%	0.00%	0.00%
Abstraccao_Nome	7.73%	0.56%	0.00%	0.00%	0.00%
Acontecimento_Efemeride	25.12%	18.74%	15.98%	0.00%	0.00%
Acontecimento_Evento	3.34%	0.00%	0.00%	0.00%	0.00%
Acontecimento_Organizado	24.62%	22.61%	26.21%	0.00%	9.58%
Coisa_Classe	11.94%	7.72%	18.78%	0.00%	5.64%

Continued on next page

Table D.2 – continued from previous page

Entity class	OpenNLP	spaCy	NLTK DT	NLTK ME	NLTK NB
Coisa_MembroClasse	0.00%	0.00%	0.00%	0.00%	0.00%
Coisa_Objecto	4.17%	0.00%	0.00%	0.00%	0.00%
Coisa_Outro	0.00%	0.00%	0.00%	0.00%	0.00%
Coisa_Substancia	0.00%	0.00%	70.32%	6.46%	0.00%
Local_Fisico	13.70%	0.76%	16.14%	0.00%	0.00%
Local_Humano	57.32%	53.77%	47.07%	7.43%	35.68%
Local_Outro	0.00%	0.00%	16.46%	0.00%	0.00%
Local_Virtual	11.87%	3.80%	6.43%	0.00%	0.00%
Obra_Arte	8.64%	0.00%	8.41%	0.00%	0.00%
Obra_Plano	23.59%	12.12%	13.77%	2.08%	5.26%
Obra_Reproduzida	31.41%	30.10%	4.11%	0.44%	17.43%
Organizacao_Administracao	39.81%	33.12%	32.96%	0.00%	14.60%
Organizacao_Empresa	14.90%	2.94%	8.93%	0.00%	1.29%
Organizacao_Instituicao	37.09%	35.35%	20.63%	0.00%	14.83%
Pessoa_Cargo	31.92%	33.98%	13.36%	0.00%	1.71%
Pessoa_GrupoCargo	1.67%	0.00%	63.67%	1.31%	0.00%
Pessoa_GrupoInd	0.00%	0.00%	0.00%	0.00%	0.00%
Pessoa_GrupoMembro	21.75%	9.22%	21.31%	0.00%	6.95%
Pessoa_Individual	62.67%	56.59%	23.87%	1.72%	42.85%
Pessoa_Membro	0.00%	0.00%	0.00%	0.00%	0.00%
Pessoa_Povo	0.00%	0.00%	0.00%	0.00%	0.00%
Tempo_Duracao	7.45%	7.89%	1.83%	0.00%	0.00%
Tempo_Frequencia	30.23%	12.19%	19.08%	0.00%	0.00%
Tempo_Generico	7.36%	2.53%	12.62%	0.00%	0.00%
Tempo_Tempo_Calend	68.59%	67.43%	28.51%	1.71%	32.89%
Valor_Classificacao	11.62%	30.82%	13.35%	0.00%	0.00%
Valor_Moeda	52.09%	29.63%	5.09%	0.00%	16.99%
Valor_Quantidade	45.76%	31.78%	17.77%	0.00%	20.61%

Table D.3: F-measure by entity class (subtypes), by tool.

Entity class	OpenNLP	spaCy	NLTK DT	NLTK ME	NLTK NB
Local_Fisico_AguaCurso	25.01%	0.00%	0.00%	0.00%	0.00%
Local_Fisico_AguaMassa	8.80%	0.00%	33.63%	1.19%	0.00%
Local_Fisico_Ilha	0.00%	0.00%	7.86%	3.10%	0.00%

Continued on next page

Table D.3 – continued from previous page

Entity class	OpenNLP	spaCy	NLTK DT	NLTK ME	NLTK NB
Local_Fisico_Outro	0.00%	0.00%	0.00%	0.00%	0.00%
Local_Fisico_Planeta	0.00%	0.00%	16.25%	0.00%	0.00%
Local_Fisico_Regiao	0.00%	0.00%	6.28%	0.00%	0.00%
Local_Fisico_Relevo	14.68%	0.00%	12.75%	6.86%	0.00%
Local_Humano_Construcao	17.91%	3.09%	12.39%	0.00%	5.14%
Local_Humano_Divisao	52.20%	27.60%	50.75%	0.09%	24.89%
Local_Humano_Outro	23.76%	0.00%	0.00%	0.00%	0.00%
Local_Humano_Pais	58.87%	38.28%	64.62%	0.00%	22.70%
Local_Humano_Regiao	13.45%	2.03%	16.47%	0.00%	0.00%
Local_Humano_Rua	22.17%	5.16%	1.39%	0.00%	0.00%
Local_Virtual_ComSocial	3.98%	0.00%	8.80%	0.00%	0.00%
Local_Virtual_Obra	0.00%	0.00%	0.00%	0.00%	0.00%
Local_Virtual_Outro	0.00%	0.00%	0.00%	0.00%	0.00%
Local_Virtual_Sitio	0.00%	0.00%	0.00%	0.00%	0.00%
Obra_Arte_Construcao	0.00%	0.00%	0.00%	0.00%	0.00%
Obra_Arte_Edificio	0.00%	0.00%	0.00%	0.00%	0.00%
Obra_Arte_Outro	0.00%	0.00%	0.00%	0.00%	0.00%
Obra_Arte_Pintura	0.00%	0.00%	0.00%	0.00%	0.00%
Obra_Reproduzida_Filme	0.00%	0.00%	0.00%	0.00%	0.00%
Obra_Reproduzida_Livro	0.00%	0.00%	0.00%	0.00%	0.00%
Obra_Reproduzida_Musica	8.60%	0.00%	0.00%	0.00%	0.00%
Obra_Reproduzida_Outro	0.00%	0.00%	0.00%	0.00%	0.00%
Obra_Reproduzida_Programa	0.00%	0.00%	0.00%	0.00%	0.00%
Obra_Reproduzida_Teatro	0.00%	0.00%	0.00%	0.00%	0.00%
Organizacao_Administracao_Sub	0.00%	0.00%	0.00%	2.27%	0.00%
Organizacao_Empresa_Sub	0.00%	0.00%	0.00%	0.00%	0.00%
Organizacao_Instituicao_Sub	2.62%	0.00%	11.12%	0.00%	0.00%
Tempo_Tempo_Calend_Data	64.16%	59.47%	30.22%	0.99%	27.51%
Tempo_Tempo_Calend_Hora	16.63%	5.34%	0.00%	0.00%	0.00%
Tempo_Tempo_Calend_Intervalo	48.70%	46.28%	0.00%	0.00%	0.00%

D.2 SIGARRA News Corpus

Table D.4: F-measure by entity class, by tool.

Entity class	OpenNLP	Stanford CoreNLP	spaCy	NLTK DT	NLTK ME	NLTK NB
Hora	89.69%	96.70%	93.49%	90.16%	63.40%	71.42%
Evento	37.10%	33.42%	32.67%	11.29%	14.62%	3.24%
Organizacao	75.59%	77.43%	72.06%	53.30%	51.02%	44.07%
Curso	66.64%	66.99%	60.73%	43.97%	43.30%	22.48%
Pessoa	81.88%	91.22%	77.30%	54.62%	51.82%	58.05%
Localizacao	74.97%	77.36%	69.16%	56.87%	55.95%	43.50%
Data	94.79%	96.48%	93.37%	88.24%	68.40%	72.47%
UnidadeOrganica	92.08%	93.22%	90.75%	77.59%	65.80%	91.94%

Appendix E

Manuals for Portuguese

These are some guidelines to run the tools used in this dissertation. For more information check the wiki page at https://github.com/arop/ner-re-pt/wiki.

E.1 Stanford CoreNLP

Main steps to run Stanford CoreNLP with HAREM dataset. Version: 3.7.0

1. First download Stanford CoreNLP tool jar from its webpage[1].
2. Navigate to the path of the *stanford-corenlp.jar*.
3. Run command:

```
java -cp stanford-corenlp.jar edu.stanford.nlp.ie.crf.CRFClassifier -prop <file.
    prop>
```

   (a) This command trains and generates a CRF model according to `file.prop`.

   (b) `file.prop` specifies the training file and the features to be used in the training process.

4. Run command:

```
java -cp stanford-corenlp.jar edu.stanford.nlp.ie.crf.CRFClassifier -loadClassifier
    <ner-model.ser.gz> -testFile <file_test.txt>
```

   (a) This command classifies the file `file_test.txt` using the CRF model generated in Step 3.

   (b) It also presents the evaluation regarding the Precision, Recall and F1 for the multiple classes.

[1] http://stanfordnlp.github.io/CoreNLP/download.html - Accessed: 2017-06-27.

Input format: File with each line having a token and the respective entity type, being `O` the tokens which are not an entity.

E.2 OpenNLP

Main steps to run OpenNLP with HAREM dataset. Version: 1.7.2

1. Download OpenNLP from its webpage [2].
2. Run command to train model:

```
opennlp TokenNameFinderTrainer -model <model.bin> -lang <pt> -data <training_data.
    txt> -encoding <UTF-8>
```

   (a) `model.bin` - Output model name

   (b) `pt` - Language of the model

   (c) `training_data.txt` - Input dataset, in the right format, for training the NER model

   (d) `UTF-8` - Encoding of the model

3. Run command to perform NER:

```
opennlp TokenNameFinder <model.bin> < <corpus_test.txt> > <output file>
```

   (a) `model.bin` - Input model name

   (b) `corpus_test.txt` - Input dataset, in the right format, for evaluating the NER model (**Note**: it has to be in UTF-8)

   (c) `output file` - Output file for the tagged text

4. Run command to evaluate NER:

```
opennlp TokenNameFinderEvaluator -encoding <UTF-8> -model <model.bin> -data <
    corpus_test.txt>
```

   (a) `model.bin` - Input model name

   (b) `corpus_test.txt` - Input dataset, in the right format, for evaluating the NER model

   (c) `UTF-8` - Encoding of the model

[2] https://opennlp.apache.org/ - Accessed: 2017-06-27.

Input format: File with sentences separated with a new line character. Entities separated with a `<Start:tag-name>` and `<END>` tags.

E.3 SpaCy

Main steps to run spaCy with HAREM dataset. Version: 1.7.2

Install spaCy: run command:

```
pip install -U spacy
```

1. Training a NER model (check script[3])
   (a) Choose main algorithm
   (b) Change language from EN to PT
   (c) Gather standoff files to a list
2. Perform NER (check script[4])
   (a) Load NER model
   (b) Get testing data from files to a list
   (c) Get the *same tokenization*
   (d) Tag the untagged text
   (e) Output both the golden data and the tagged text to CoNLL format

Input format: File in the standoff format.

E.4 NLTK

Main steps to run NLTK with HAREM dataset. Version: 3.2.2

Install NLTK run command:

```
sudo pip install -U nltk
```

Install Numpy (optional) run command:

[3] https://github.com/arop/ner-re-pt/blob/master/tools/spacy/src/ner-train-spacy.py - Accessed: 2017-06-27.

[4] https://github.com/arop/ner-re-pt/blob/master/tools/spacy/src/ner-test-spacy.py - Accessed: 2017-06-27.

```
sudo pip install -U numpy
```

Test installation run python then type `import nltk`.

Training model

1. Download NLTK trainer[5]
2. (*Optional*) Put training data in `nltk-data` folder
3. Run command for training using `train_chunker.py`:

```
python train_chunker.py <path-to-training-file> [--fileids <fileids>] [--reader <
    reader>] [--classifier <classifier]
```

   (a) `path-to-training-file`: specifies path to training file (or files) relative to `nltk-data` folder or current path. (file has to be in UTF-8 encoding)

   (b) `fileids`: regex expression to match the files inside the `path-to-training-file`, (if no expression is given, all files will be used)

   (c) `reader`: specify the reader for the corpus. In my case, since the corpus was in the Conll2002 IOB format I chose `nltk.corpus.reader.conll.ConllChunkCorpusReader`. Note: For this reader I had to specify the entities classes used in the `__init__.py` file from `nltk-trainer`

   (d) `classifier`: specify the classifier to use, main options: `Maxent`, `DecisionTree`, `NaiveBayes`

Perform NER

1. Load chunker model using `pickle`: `pickle.load(open(model_path))`
2. Load input dataset (already tokenized and POS-tagged, done in training step)
3. Perform NER; `chunker.parse(tagged)`
4. The parser returns the result in a tree format, which was converted to the CoNLL format using `nltk.chunk.util.tree2conlltags(ner_result)`
5. Output to file

Input format: File in the CoNLL format, with POS tag and NER tag with IOB tagging scheme.

[5] https://github.com/japerk/nltk-trainer - Acessed: 2017-06-27.