TÉCNICO LISBOA

Natural Language Processing Applied to the Portuguese Consumer Law

Maria Ana de Sousa e Moura Duarte

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Pedro Alexandre Simões dos Santos
Prof. João Miguel de Sousa de Assis Dias

Examination Committee

Chairperson: Prof. Pedro Miguel dos Santos Alves Madeira Adão
Supervisor: Prof. Pedro Alexandre Simões dos Santos
Member of the Committee: Prof. Bruno Emanuel Da Graça Martins

November 2021

This work was created using LaTeX typesetting language
in the Overleaf environment (www.overleaf.com).

Acknowledgments

I am deeply thankful to both my supervisors, Professor Pedro Santos and Professor João Dias for their guidance and advice throughout the writing of this thesis. Not only did they help me with their expertise and knowledge, but they also always made me feel welcomed and part of a team, which motivated me and made me enjoy writing this thesis even more.

I would like to express my profound gratitude to Professor Jorge Baptista for his contributions to this work, as well as his patience whenever I needed his help understanding linguistic concepts that I was unfamiliar with.

In addition, I would like to thank Nuno Cordeiro and the annotation team, for their work, that made this dissertation possible. I would also like to thank Diogo Carvalho for his assistance with the virtual machine and server used for this work

Last but not least, I would like to thank my parents, my sisters, my little brother, and my closest friends, specially Inês Lacerda, Jin Xin, Dinis Araújo, and Alexandre Rocha, for believing in me and comforting me in my most stressful times. For listening to me when I burdened them with my worries and stress and for their patience and their support.

I appreciate every single one of you, for everything you have done to help me reach this point in my life. I will be forever grateful to have been given the opportunity to work on this project while being surrounded by amazing people who trusted me and were always by my side.

Abstract

Understanding our duties, obligations and other legal norms has become progressively more important, as societies have legal systems that impose rules based on these norms and expect all citizens to obey them. It is, therefore, crucial that citizens are able to interpret clearly what these norms entail and their meaning, instead of just simply having access to the laws. However, to be able to provide the legal information to citizens in a representation that is easy to interpret and understand, first it is necessary to extract the meaningful semantic information present in legal texts. Hence, this thesis aims to create a semantic information extraction system trained on domain-specific (legal) text corpora, in order to later be used to create and improve other related systems such as QA, inference systems, and Information Retrieval Systems, as well as develop rich visual representations of the extracted information.

To this end, we introduce the SNR (Semantic Norm Recognition) system, a domain-specific information extraction system that uses the Portuguese Bert (BERTimbau), and is based on the Named Entity Recognition Model of Yu et al. and trained on a large legislative Portuguese corpus, with 5.600 segments and an average of 60.96 tokens per segment. In this thesis, we describe the annotation procedure, the different information we are trying to extract, the architecture of our system, the different approaches that we implemented (Baseline, Hybrid, and Two phased), and the performance of the overall system regarding all the extracted information and each type of information as well. We demonstrate how our system achieved good results (81.44% f1 score) on the specific corpora, despite existing noise. Additionally, we showed how our SNR system predictions, for a new set of legal information, improved an information retrieval system using the extracted information. Finally, we describe our achievements, the

limitations of our system and make suggestions for future work.

Keywords

Legal Domain, Information Extraction, Semantic Extraction, Named Entity Recognition, Natural Language Processing, Neural Networks

Resumo

Compreender os nossos deveres, obrigações e outras normas legais tem se tornado cada vez mais importante, à medida que as sociedades possuem sistemas legais que impõem regras baseadas nestas normas e esperam que todos os cidadãos as cumpram. É então crucial que os cidadãos não só tenham acesso às leis, como sejam capazes de interpretar claramente o que as normas implicam e os seus significados. No entanto, de maneira a conseguir fornecer, aos cidadãos, a informação legal numa representação que seja de fácil interpretação e entendimento, primeiro é necessário extrair a informação semântica presente em textos legais. Com isto, o objetivo desta tese é desenvolver um sistema de extração de informação semântica treinado com um corpus de domínio legal, que depois poderá ser usado para ajudar outros sistemas relacionados, como sistema de perguntas e respostas, sistemas de inferência e sistemas de recuperação de informação, assim como ajudar na criação de representações da informação extraída.

Para este fim, introduzimos o sistema SNR (Semantic Norm Recognition), um sistema de extração de informação de domínio especifico, que usa o Bert português (BERTimbau), e é baseado no modelo de reconhecimento de entidades mencionadas de Yu et al. É treinado num grande corpus legislativo português, constituido por 5.600 segmentos com uma média de 60.96 tokens por segmento. Nesta tese descrevemos o processo de anotação, a diferente informação que estamos a tentar extrair, a arquitetura do nosso sistema, as diferentes variantes que implementámos (Baseline, Hybrid e Two phased) e os resultados do mesmo relativamente a toda a informação que tentamos extrair e a cada conceito também. Demonstramos como o nosso sistema antigiu bons resultados (81.44% f1 score) no corpus especifico, apesar de existência de barulho. Adicionalmente, mostramos como os resultados do nosso modelo, para um novo conjunto de informação legal, melhorou o desempenho de um sistema de extração de informação. Finalmente, descrevemos as nossas conquistas, limitações do nosso sistema

e trabalho futuro.

Palavras Chave

Domínio Legal, Extração de Informação, Extração Semântica, Reconhecimento de Enitdades Mencionadas, Processamento da Lingua Natural, Redes Neuronais

Contents

List of Figures

Acronyms

AI	Artificial Intelligence
ATA	Average Token Agreement
BPTT	Back Propagation Through Time
BERT	Bidirectional Encoder Representations from Transformers
BiLSTM	Bidirectional Long Short Term Memory Network
CRF	Conditional Random Field
DRE	Diário da República Electrónico
FFNN	Feed Forward Neural Network
ICT	Inverse Close Task
INCM	Imprensa Nacional Casa da Moeda
LSTM	Long Short Term Memory Network
MANN	Memory-Augmented Neural Network
MST	Maximum Spanning Tree
MLM	Masked Language Modeling
MLP	Multilayer Perceptron
NE	Named Entities
NER	Named Entity Recognition
NL	Natural Language
NLI	Natural Language Inference

NLP Natural Language Processing

NSP Next Sentence Prediction

QA Question-Answering

RNN Recurrent Neural Network

SNR Semantic Norm Recognition

SR Semantic Roles

SRL Semantic Role Labelling

1

Introduction

Contents

As societies grew they became more complex, and the need to formalize laws rose. For those reasons, legal systems were created. They define the rules of a society as well as the rights of the citizens who make up that society. Legal systems aim to create a sort of control in our societies, to try to give everyone a chance to live a fair and safe life. Citizens need to know their rights and duties in order to keep themselves safe, to have what they are entitled to, to make sure they know their responsibilities, respect codes of conduct, and behave in accordance with the rules, facing consequences when they break them.

All the norms and laws of the Portuguese Republic, are published by The Official Portuguese Gazette (Diário da República). Currently, the Imprensa Nacional Casa da Moeda (INCM) publishes it at DRE.pt. It works as a public service with universal and free access to its content and functionalities.

The Diário da República Electrónico (DRE) is composed of many articles, that contain definitions, rights, and obligations that citizens must obey. A portion of these articles is accessible at Direção Geral do Consumidor[1] and corresponds to the Portuguese Consumer Law. These articles are highly important, since citizens, being consumers, are affected by the laws included in these articles and so, they often find themselves searching for the information present in these articles. It is crucial for citizens to know what these rights and duties are, thus clearly it is important that the information of these legislative articles is not only easily accessible to all citizens but also easily understood. Publishing norms and laws online does not make them truly accessible.

1.1 Problem

Regular citizens are used to reading and writing in Natural Language (NL). However, when reading legal text, some citizens may have difficulties in understanding and interpreting the text, since legal text contains certain terminology and sentence construction, very specific to its legislative theme. Even though it is NL, it is in a much more formal format than the language regular citizens normally use. The legislative documents often contain juridical terms that are not easy to understand, which can cause citizens to misinterpret or simply not understand the documents, often leading them to reach for professional aid in issues that may not require it. This shows the strong need for a richer and more explicit representation for legislation. We believe that recent advances in Natural Language Processing (NLP) can help us move in this direction.

One example of an approach in NLP to automatically extract information from legal texts is the work of Humphreys et al. [6]. The authors extracted definitions and deontic concepts (such as rights and obligations) from legislation, and showed that capturing these legal concepts, and the relationships between the terms that represent such concepts, helps information retrieval.

[1] https://www.consumidor.gov.pt/

Similar to Humphrey et al., this work hypothesizes that the extraction of definitions, norms, and semantic relationships of legal articles would be beneficial to downstream tasks. By extracting all this information we are representing laws in a formalism that can then be used to infer knowledge and help build NLP-based systems (such as retrieval systems and question answering systems) that will help citizens to better understand and navigate the legal domain. Thus, the main research question this thesis aims to answer is the following:

How to create a system capable of automatically extracting and representing norms, semantic relationships, and entities from legal texts ?

1.2 Approach

In the context of this work, we wanted to extract several concepts from a specific corpus, which corresponds to legal articles of the Portuguese Consumer Law. The concepts we wanted to extract are legal norms (like definitions, rights, obligations, and so on), some entities (like legal references), and semantic roles (who is obligated, who obligates, among others) that are within those categories. As we can see, some of the information we wanted to extract is similar to the information given by Semantic Role Labelling (SRL), which gives us information regarding "who" did "what" to "whom", "when" and "where". Additionally, Named Entity Recognition (NER), consists of identifying named entities, which correspond to real-world objects, that can be classified with a specific name, such as Person or Location, among others. Just like NER extracts these named entities, we wanted to extract norms, semantic roles, and some named entities. Thus, we assumed that we were dealing with the NER classification task, regarding the extraction of information. More specifically, having in mind that some of these concepts can be nested within each other, we assumed we were dealing with a Nested NER task. Additionally, since the type of some of the information we wanted to extract corresponds to roles, like in SRL, and not exactly objects, like named entities, we denoted our task by Semantic Norm Recognition (SNR).

1.3 Thesis Outline

The rest of the thesis is organized as follows: Chapter 2, presents an overview of artificial intelligence in law. It also contains some concepts that served as inspiration for when choosing the relevant concepts that needed to be extracted from the legal domain; Chapter 3, includes the main concepts in NLP that are needed to understand the work described in this paper; Chapter 4, introduces current approaches for information extraction and the state-of-the-art for NER Models; Chapter 5, describes the dataset and how it was created; Chapter 6 describes the implemented SNR system approaches, training and optimization;

Chapter 7 describes the evaluation metrics used, analysis the results of all approaches, describes some noise that was found in the dataset, analyses the performance of our system in function of the existing noise, and shows how our system improves an existing information retrieval system; Finally, Chapter 8 summarizes the work explained throughout this thesis, describes the main achievements and discusses some future work that could be applied to the SNR system.

2

Artificial Intelligence in Law

Contents

The classic approaches to Artificial Intelligence (AI) applied to Law consist of Logic-based approaches and Data-centric approaches. Logic-based approaches application in the legal domain started in the eighties with the formalization of laws of the British National Act [9]. The application of data-centric approaches in the legal domain also started in the eighties. An example of this type of application is the HYPO system [10], whose goal was to model reasoning based on past cases and patterns to generate legal conclusions.

2.1 Logic-based Approaches

In Reasoning systems, it is possible to apply logic-based reasoning to infer legal conclusions from formalized versions of the laws. There are two main problems here, one regarding the formalization of the laws and another regarding the analysis of the legal predicates expressed in natural language.

Logic-based approaches presume that is plausible to create a legal formalization for a given text, that is loyal and authentic to the text's meaning [11]. These approaches defend that:

1. legal texts can be described by a specific logical structure;
2. the expressiveness of the logical formalism is enough to aid the complexity of legal reasoning;
3. a logical formalism is just as authoritative as the legal text it corresponds to.

It has not been decided what is the most suitable logic for specific goals. In recent years, many have worked on logics focused on deontic, defeasible, and temporal concepts of legal reasoning. Deontic logic, as the name indicates, reasons about deontic concepts like obligations, permissions, among others. Some of these concepts are also mentioned below, in Section 2.1.2. Defeasible logic is a non-monotonic logic, that can deal with rules that have exceptions to them. This concept is explained in more detail in Section 2.1.3. Finally, temporal logic includes any rules or symbols that represent any temporal terms.

Translating a legal text into a logical formalism is a heavy and difficult task. It requires deep legal and logical knowledge. [11] Luckily, recent advances in NLP can help with this task, as we will see in Section 2.2.

If we do convert a legal text into a logical formalism with perfect fidelity, we still have yet the second challenge that comes from the "natural language barrier" between the legal terminology and the natural language regular citizens use. Regardless if legal rules are formalized with perfect loyalty to the legal's text real meaning, the terms used are often impossible for a regular person to interpret. Several solutions have been put forward, combining data-centric approaches with recent advances like case-based reasoning, translation, question answering, and textual entailment.

2.1.1 Formal aspects of Legal norms

A legislative text can be viewed according to two different profiles [1]:

• a structural profile, that represents the traditional structure of legal texts (chapters, articles, paragraphs, and so on);

• a semantic profile, that represents a specific organization of legislative text substantial meaning; It can be described with normative prescriptions.

Normative prescriptions are divided into two groups: Rules, and Rules on Rules. Rules consist of constitutive rules (that present entities) and regulative rules (that convey deontic concepts). Rules on Rules, correspond to different types of amendments.

2.1.2 Hohfeldian relations and Deontic Logic

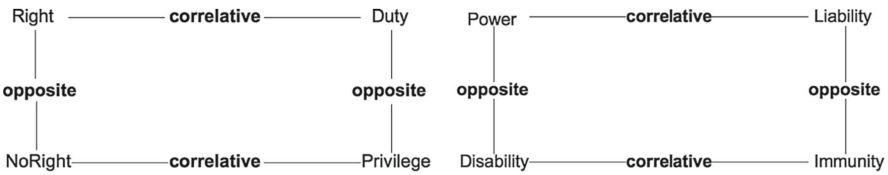

Figure 2.1: Hohfeldian relations [1]

Hohfeld pinned down two-quarters of very important relations regarding legal context, called Hohfeldian relations [12], which are represented in Figure 2.1. He identified deontic concepts in the first quarter, by showing the relations between pairs of concepts. He showed correlations between Right and Duty, and No-right and Privilege. He also showed the opposition between Right and No-right, as well as Duty and Privilege. For example, saying a person A has a right/privilege towards B, means the same as B having a duty/no-right towards A. For the second quarter, Hohfeld identified power relations. He showed correlations between Power and Liability, as well as Disability and Immunity. He also showed the opposition between Power and Disability, as well as Liability and Immunity. For example, saying a person A has power towards B, is the same as B having a liability regarding A. Equivalently, saying A has an immunity regarding B, it means that B is disabled regarding A.

Rule-based systems use Deontic Logics that regard obligations, rights, and linked terms, to model the relations mentioned above.

2.1.3 Defeasible reasoning, Non-monotonic Logic and hierarchies

The removal of information or the addition of new facts can alter previous laws. Thus, when reasoning about laws, we need to be able to change or infer new conclusions based on the information that we

know at the moment. For this reason, monotonic logic, as First Order Logic, is not enough, since after making deductions, these cannot be changed based on new information. For example, for the following sentences: “Any person will go to prison whenever they commit a crime” and ”Any person who is a minor will not go to prison”, following a monotonic logic if we know a person committed a crime, we would infer that the person will go to prison. If new information comes up stating that the person in question is a minor, that will not change the previous conclusion we made, which was that the person, who is a minor, will go to prison. To invalidate previous conclusions, due to new facts that result in such invalidation, defeasible reasoning is used. With this reasoning, we can have exceptions that can override existing rules [13]. For example, for the following sentence: “Any person will go to prison whenever they commit a crime, except if they are a minor.”, following defeasible reasoning, if someone was a minor they would not go to prison. If we received new information, stating that the person was now not a minor, the conclusion should change, since now they could go to prison. As we can see, Defeasible reasoning, in contrast to monotonic logic, allows us to represent such cases.

By using non-monotonic logic we are also able to represent hierarchies, by establishing priority relations over different rules, allowing us to resolve any conflicts that those rules might generate. [14]

2.2 Data-centric Approaches

Data-centric approaches have the advantage of creating automatic and flexible systems. These systems help reduce the amount of human annotations for legal texts. All that is needed is a dataset with examples, for example, pairs of legal sentences and logical representations for those texts, so that a machine learning algorithm (using deep neural networks, for example) can learn to create such representations for non-annotated legal texts. In spite of this, these approaches have some disadvantages. For example, when dealing with deep neural networks, we are faced with a very high complexity due to the high number of parameters often combine to determine a specific output. Anyone who is not an expert will face difficulties in interpreting how and why the model produces a specific output. Thus, we argue that these type of approaches should be applied to parts of reasoning systems instead of replacing the whole system, allowing any observer to better interpret the decisions made by the system. [11]

There are not many works that deal with the application of data-centric approaches in the legal domain. Actually, these types of approaches are usually applied to NLP. Thus, the following chapter includes the most relevant and important concepts when using these approaches for NLP tasks.

3

Background on NLP

Contents

In this chapter, we will tackle the main concepts in NLP needed to understand our SNR system as well as the related work mentioned in the following chapter.

3.1 Neural Networks for NLP

This section described several neural networks, which can be applied not only to Natural Language Processing Tasks but also in several other tasks like Image Recognition, Pattern Recognition, and so on.

3.1.1 Feed Forward Neural Network:

Feed Forward Neural Networks (FFNNs), also called Multilayer Perceptron (MLP), are one of the simplest neural networks. A FFNN is composed of an input layer, a number of hidden layers, and an output layer, as shown in Figure 3.1. The name feedforward comes from the fact that in these types of networks the information flows in one single direction, which is forward. It flows from the input layer, across the hidden layers, to the output layer.

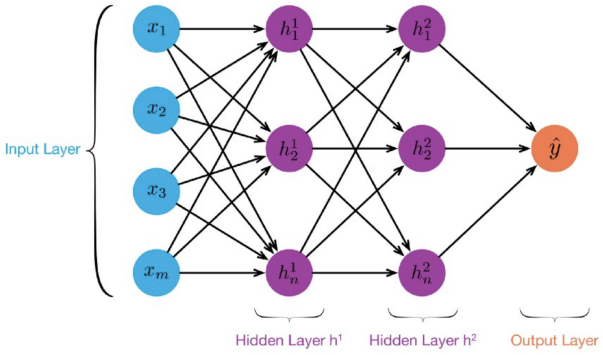

Figure 3.1: FNN Network with 2 hidden layers[1]

As we can see in Figure 3.1, every perceptron in each layer is connected to every perceptron in the previous layer, making the network fully connected. So, every perceptron is dependent on the outputs of the previous layer. Since the direction of the flow of information is forward, to achieve the final output, first all hidden units' outputs must be calculated. Every hidden unit corresponds to h_j^i, where i represents the layer and j the unit of that layer. The output of each unit comes from doing the two following operations:

$$z = \sum_{j=1}^{\infty} w_j^i x_j + bias$$

[1] https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f

$$h_j^i = f(z),$$

where w_j^i corresponds to the weight for the unit j, x_j corresponds to the input for the unit j, and the function f corresponds to an activation function. Often the sigmoid function is used. By using non-linear activation functions we can achieve non-linearity, allowing the network to separate data that is non-linearly separable.

After all hidden outputs are calculated, they are used as inputs for the output layer to calculate the final output. A similar procedure to the one described above occurs:

$$z = \sum_{j=1}^{\infty} w_j^i h_j^i + bias$$

$$y = f(z)$$

The difference here is that instead of multiplying the weights with the input features, we now multiply them with the hidden outputs previously calculated. Then, we apply another activation function to the output of the summation, which results in the final output of the network.

The training algorithm for these types of networks is called Back-propagation, where the weights and biases are updated considering the error (for example, the squared error function) between the predicted outputs and the real values. The derivative of the error with respect to a certain weight/bias is used to update that weight/bias, according to a learning rate, n.

Like it was mentioned above, this network is the most basic one and it has been extended to create more complex networks, for example, Convolutional Neural Networks.

3.1.2 Recurrent Neural Networks:

If we think about a human reading a text, we know that his understanding of each word comes from understanding the words before. A Recurrent Neural Network (RNN) is inserted in this context, its objective is to mimic this human behavior and keep a memory of the past. These networks are usually used for language translation, speech recognition, and image captioning. Formally, a RNN is a special type of neural network, that is best at processing sequences of data. Like the name “recurrent” suggests, a RNN consists of a single network, with multiple layers where each layer uses the output of the one before. These networks are mostly used in cases where elements in the data depend on each other,

like in pattern recognition tasks.

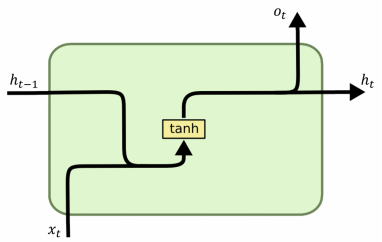

Figure 3.2: Representation of a RNN Cell[2]

As we can see in Figure 3.2, for each cell (H_t) of the network there are four main components: x_t, the input; h_{t-1}, the hidden state vector form the previous cell; h_t, the hidden state vector of the current cell; and finally, O_t, the output of the current cell, that relies on h_t. It can be, for example, what the network thinks should be the next word in a sentence. It is defined in the equation below, where W_O and b_O are the weights and bias for the output, respectively.

$$O_t = W_O \cdot h_t + b_O$$

As we can see from the image, the hidden state vector h_t, depends on the previous hidden state vector h_{t-1}, and on the current input x_t, which makes it responsible for holding the information of the previous states. It is defined as:

$$h_t = f(W_h \cdot h_{t-1} + b_h + W_x \cdot x_t + b_x),$$

where f is a non-linear activation function (usually corresponds to tahn, or sigmoid function). W_h and b_h, are the weights and bias for the hidden state vector, and W_x and b_x are the weights and bias for the input. The initial values of the weight matrices are all random. They are updated during training.

The training algorithm for a RNN is called Back Propagation Through Time (BPTT). It is based on the training algorithm for Feed Forward Networks, where weights and biases are updated accordingly to the error between the network's predicted output and the real value. BPTT works by unrolling all time steps. In each time step, we have an input and predicted output, so BPTT calculates the error between those two values. The errors and the gradient of the loss function are accumulated and the gradient is used to update the weights/biases, in accordance with a certain learning rate, n.

One big issue with RNNs is the Vanishing Gradient Problem. In back propagation, the errors tend to converge to zero or increase in an exponential way, which makes it impossible to update the weights.

[2] http://dprogrammer.org/rnn-lstm-gru

The problem of this is when we are faced with long-term dependencies (words very far from each other, that are dependent on each other), and when the vanishing gradient happens these dependencies get overestimated or just simply ignored.

3.1.3 Long Short Term Memory Networks:

Long Short Term Memory Networks (LSTMs) are recurrent neural networks capable of learning long-term dependencies, allowing information to flow between time steps. Different from typical RNNs, LSTMs repeating module does not have the simple structure of a RNN (like a single Tanh Layer), instead it uses gating mechanisms, as we can see in Figure 3.3. Another difference from RNNs, is the existence of an additional cell state vector, which can be read or written depending on the gate. These gates are simply vectors like the other structures, they are simply called gates since they control the amount of information that flows through each time step.

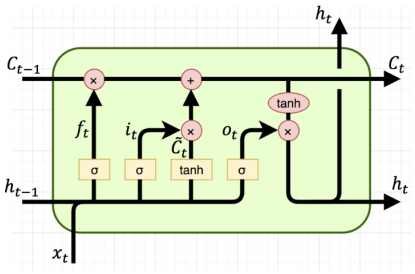

Figure 3.3: Representation of a LSTM Cell[3]

By looking at Figure 3.3, we can see in more detail what happens in each cell. First, f_t, the forget gate, is responsible for deciding which information is going to be thrown away from the cell state, C_{t-1}. The gate is defined as:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Then, the LSTM cell needs to decide which new information needs to be saved in the new cell state. To do this, first the input gate, f_t, chooses which values to update (which information of the candidate to use for the cell state). Next, the cell state candidate vector is created by a tanh layer. These two values are combined and used to update the cell state.

f_t, is the input gate, it's defined as:

[3] http://dprogrammer.org/rnn-lstm-gru

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

$\bar{C}_t$, is the new cell state candidate vector, it's defined as:

$$\bar{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Finally, C_t, is the calculated cell state, it's defined as:

$$C_t = f_t \odot C_{t-1} + i_t \odot \bar{C}_t$$

In order for an LSTM to control what it reads and writes, it uses sigmoid function in the gates, to make the range of values go from 0 to 1, tanh to calculate the hidden state vector, and the candidate to make the range of values go from -1 to 1. Finally, the LSTM provides an output, O_t, for that cell. It starts by choosing which parts of the cell state to output, using a sigmoid layer. Then, it applies a tanh layer to the cell state to make sure the values are between -1 and 1, and multiplies those values by the output to return only the chosen values (h_t), which corresponds to the hidden state vector that is going to be transmitted to the next cell. The output gate is defined as:

$$O_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

The hidden state vector is defined as:

$$h_t = O_t \odot tanh(C_t)$$

W_x and b_x with $x \in \{i, f, o, C\}$, are the weight matrices and bias vectors of the input, forget, output and cell state vectors, respectively. LSTMs demolish the Vanishing Gradient Problem by enabling the back-propagation of the errors unaltered, resulting in the update of long-term weights that need to exist in order to learn long-term dependencies.

3.1.4 Bidirectional Long Short Term Memory Networks:

LSTMs are unidirectional, there is an extended network called Bidirectional Long Short Term Memory Network (BiLSTM), which as the name indicates is bidirectional. The difference between the two is that in a BiLSTM the information flows in two directions, from left to right and from right to left, allowing the network to learn past and future information. When we described RNNs, we mentioned that when a human is reading a text, his understanding of each word comes from understanding the words before, this corresponding to past information. In a BiLSTM, not only do we have the past information from looking at the words before, but we also have future information from looking at the words that are after. A BiLSTM achieves this by having two LSTM hidden layers, with different directions, as shown in Figure 3.4.

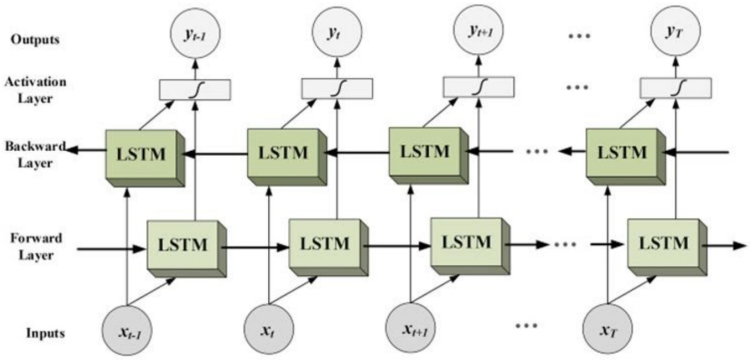

Figure 3.4: BiLSTM Architecture [4]

As we can see, it is as if we had 2 LSTMs where for one we use the original X as input (which allow us to determine a word's meaning from the words that are before that one) and for the other we use the reversed version of X(which allow us to determine a word's meaning from the words that are after that one). This network differs from a LSTM regarding the output, since now it needs to take into consideration the backward and forward information. So, the network merges the output of each backward LSTM cell with the forward LSTM cell. Normally this is done by concatenating the results since with concatenation there is no loss of information.

[4] https://analyticsindiamag.com/complete-guide-to-bidirectional-lstm-with-python-codes/

3.1.5 Transformer:

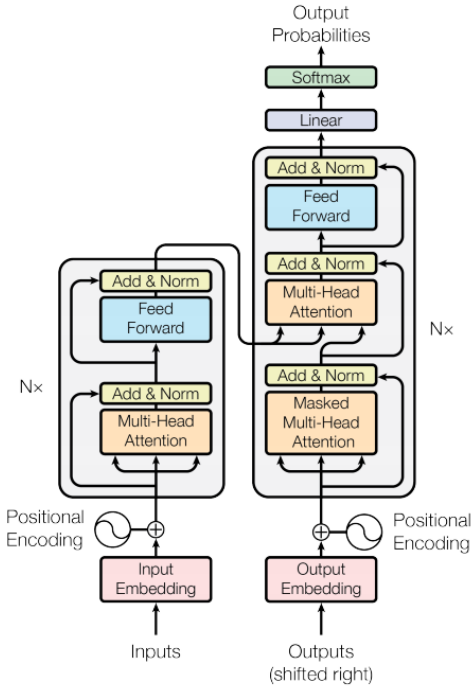

Figure 3.5: Transformer Architecture [2]

Even though LSTMs are good for learning long-term dependencies, they have some limitations. Due to the high complexity of the network, the training is slow. Also, since the process behind the network is sequential, the current state depends always on the previous state, so we cannot use parallel computation to accelerate training.

This is where Transformers [2] came in. In contrast to RNNs, Transformers do not use recurrence, but instead attention mechanisms [2]. By using such mechanisms, the network can take into consideration the relations between each element of the input sequence, independently of its position. There are no longer problems regarding long-term dependencies, and because the process is done for each element simultaneously, the vanishing gradient problem is lessened on the network.

As we can see in Figure 3.5, the Transformer has 2 embedding blocks, one for the encoder and another for the decoder. Each block is composed of two layers: an Embedding Layer and a Positional Encoding Layer. The first layer is responsible for transforming the input into embeddings. The second is responsible for adding the positional information of each element of the input sequence in its corresponding embedding. RNNs know where in the sentence the element is with recurrence, with Transformers, this information comes from the added positional encoding. The position embeddings are calculated by doing the following:

$$PE_{pos,2i} = sin(\frac{position}{10000^{\frac{2i}{d_{model}}}}) \tag{3.1}$$

$$PE_{pos,2i+1} = cos(\frac{position}{10000^{\frac{2i}{d_{model}}}}), \tag{3.2}$$

where i is the positional embedding entry. Each entry corresponds to a sinusoid, which enables the model to be able to extrapolate in longer sentences than the ones used during training.

Looking back at the figure, we can see that the encoder has two sub-layers: a Multi-Head Attention sub-layer and a Feed Forward sub-layer. To understand the first sub-layer, we need to understand the concept of Self-Attention first.

Let us consider: Query as Q, where $Q \in R^{w \times d_k}$, a matrix that contains the vector representations of the words in the query; keys as K, where $K \in R^{w \times d_k}$, a matrix that contains the keys of the words in the query; values as V, where $V \in R^{w \times d_v}$. d_k is the dimension of each query vector, d_v is the dimension of each value vector and w is the number of words in the query.

An attention function can be represented as the mapping of a query Q and a set of key-value (K,V) pairs, to an output [2]. It starts by calculating the similarity between the query Q, and each key (K), obtaining the weights (3.3). Next, after normalizing the weights, it calculates the attention with the weights and values (V) (3.4).

$$Compatibility(Q, K) = softmax(\frac{QK^T}{\sqrt{d_k}}) \tag{3.3}$$

$$Attention(Q, K, V) = Compatibility(Q, K)V \tag{3.4}$$

As we can see by looking at the second equation from above, the Transformer returns a weighted combination for the values, whose keys, have a higher compatibility with the query.

We can imagine an attention mechanism for a sentence, where each token is compared with all the other tokens, having in the end, a matrix with all the compatible scores for each pair of tokens. The self-attention mechanism, represented in Figure 3.6, was called Scaled Dot-Product Attention, since it uses the $softmax$ function.

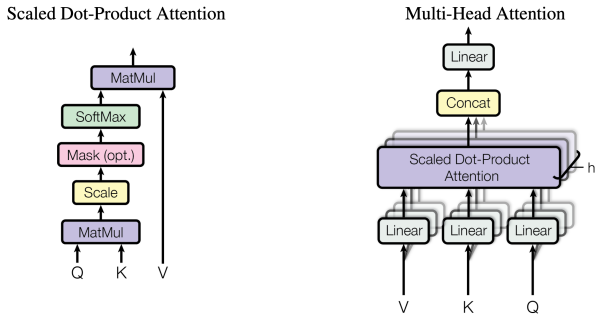

Figure 3.6: Scaled Dot-Product Attention (left) and Multi-Head Attention (right) [2]

From Figure 3.6, we can see that the Multi-head sub-layer is very similar to the self attention mechanism. The difference is that in the sub-layer each V,K and Q is linearly transformed h times into h different vectors, then for each, the self-attention mechanism is applied, in parallel, in order to get each output. Then all the h returned vectors are concatenated into a single vector before going to the final layer. The idea behind this behavior is to allow the network to learn different things (for each h), to increase the power of the model's representation. The final layer (linear layer) produces the final output.

The input of the encoder is added to the output of the Multi-head sub-layer. This addition is known as a residual connection [15]. The output of this connection is normalized and used as input for the second sub-layer, the feed forward sub-layer. The pointwise feed forward network uses $RELU$ activation function between its linear layers. Finally, the sub-layer's output will then be added to the sub-layer input (so there is another residual connection) followed by a normalization, which results in the encoder output.

If we look again at Figure 3.6, we can see that the encoder and decoder are very similar. They both apply the residual connections [15] followed by normalization, in each sub-layer. The difference is that the first sub-layer corresponds to a Masked Multi-Head Attention sub-layer, which differs from the Multi-Head Attention sub-layer by applying a masking method. For each time step t, the decoder generates a token. So at time step t, it should generate the token only taking into consideration the previous tokens. This is where the look ahead masks (masking method) come in. For example, for the sentence: "I like transformers", when generating the token "I", we need to set the values of the future tokens to infinity, in order to prevent the generated token from depending on future tokens. The following sub-layer works in the same way as the multi-head attention sub-layers of the encoder, the only difference being that the Q and K correspond to the encoder's output and V to the output of the previous sub-layer of the decoder, followed by the residual connection [15] and normalization. Afterward, the information goes through a feed forward network, followed by the residual connection [15] and normalization, just like in the encoder. Finally, the output goes through a linear classifier followed by a softmax layer to get the probabilities for each token.

The decoder's input corresponds to the encoder's output, except in training. In training, since we have

the target output, we assume the encoder generated the correct elements and use the target output as the decoder's input.

The residual connections [15] are used to help the training process, by allowing the gradient to flow through the networks directly. Normalization is done to settle the network, which results in a significant reduction of the training duration. The feed forward network is used to improve the attention output representation.

3.2 Word and Sentence Embeddings

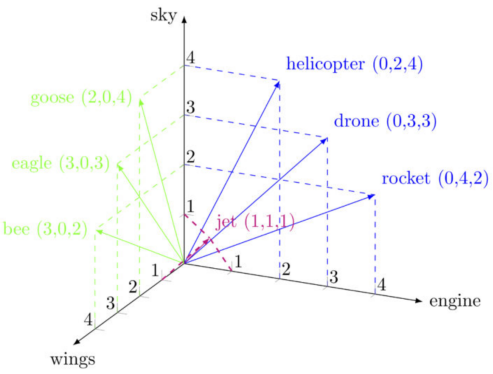

Figure 3.7: Word Embeddings in Vector Space[5]

Given that we need to represent semantic relationships and find important concepts of legal texts, it is pertinent to understand the concepts of Word and Sentence Embeddings, which correspond to the semantic representation of words and sentences, respectively.

In word embeddings, words are transformed into vectors, allowing us to compute the similarity of words by calculating distances between vectors. These embeddings can encapsulate the syntax and semantics of a word in a given context. An example is shown in Figure 3.7. In this example, we have a small corpus of seven words (the ones in green, blue, and red) which were represented as vectors whose coordinates correspond to the number of times they appear in a certain context (sky, wings and engine). The assumption behind word embeddings is that vectors of words that are semantically similar will be close to each other, as we can see by looking at rocket and drone, or bee and eagle. There are many algorithms responsible for generating these embeddings. Most of them use neural networks, such as Word2Vec [16], Glove [17] and Bidirectional Encoder Representations from Transformers (BERT). BERT has been recognized as the state-of-the-art model and a key difference between this model and

[5]https://corpling.hypotheses.org/495

the first two models, is that Word2vec and Glove are context-independent models. Thus, for each word they output one single vector, even if the word is used in different contexts. For example, for the word "book" in the sentence: "While waiting to book his motel room, he read his favorite book.", the model will output a single vector that combines all the different contexts of the word. This is a consequence of the fact that these two models do not take into consideration word order, while BERT does, since it uses a transformer. So, BERT would have a different representation for the first and second appearance of the word "book". Another key difference is regarding the generated embeddings. While the first two models generate word embeddings, BERT generates word pieces embeddings. BERT is described in more detail in the section below.

Sentence embeddings, like it was mentioned above, are similar to word embeddings except they correspond to vectors that represent sentences instead of single words. The initial techniques implemented to generate sentence embeddings used words embeddings. One of these techniques called average pooling, calculates the average of the word embeddings (of all the words in the sentence). There are several extensions to this method, like the average with weights (which depend on the word's frequency) for example.

3.3 Bidirectional Encoder Representations from Transformers

When reading a sentence, we understand each word of the sentence by looking at its surrounding words. In other words, our understanding of the context comes from a bidirectional extraction of information. So, if humans treat language understanding as a bidirectional task, computers should as well. This is where Bidirectional Encoder Representations from Transformers (BERT) comes in. BERT [3] is a language representation model, whose architecture corresponds to a multi-layer bidirectional Transformer encoder.

3.3.1 BERT Architecture

As previously stated, BERT's design is a multi-layer bidirectional Transformer encoder [2]. It is made up of L layers of identical Transformer encoders placed on top of one another. There are two different sorts of sub-layers in each encoder. The first is a multi-head self-attention mechanism that allows the model to glance at other words in the sequence while encoding one. The second is a basic position-wise fully connected FFNN, described in 3.1.1, which is applied to each position independently and identically. The $GELU$ activation [18] is used as the activation function in the feed-forward network, which is defined as:

$$GELU(x) = 0.5x(1 + tanh(\sqrt{\frac{2}{\pi}}(x + 0.044715x3))) \tag{3.5}$$

This activation function was found to operate better in a Transformer encoder, than the usual ReLU [19], which is defined as:

$$ReLU(x) = max(0, x) \tag{3.6}$$

Furthermore, an encoder layer uses a residual connection [15] surrounding each of the two sublayers, followed by layer normalization, so that each sublayer's output is equal to

$$LayerNorm(x + Sub\_layer(x)) \tag{3.7}$$

where $Sub\_layer(x)$ corresponds to the function that the sublayer implements. All sublayers in the model give outputs of the same dimension d_{model} to help with residual connections. Figure 3.8 corresponds to the architecture of a single encoder.

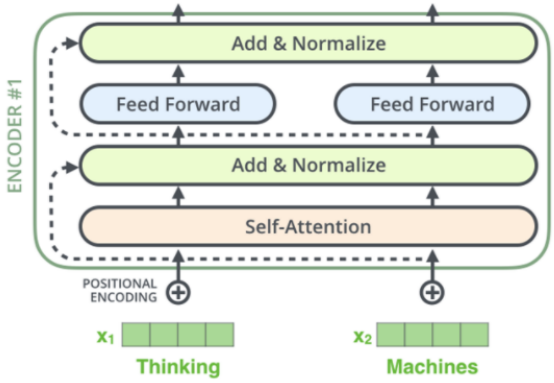

Figure 3.8: Encoder Architecture [6]

3.3.2 Input Representation

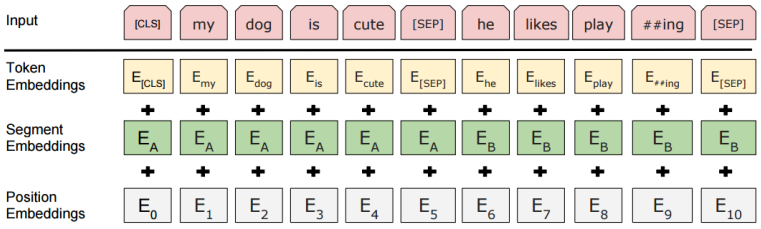

Figure 3.9: Bert Input Representation [3]

[6] https://davideliu.com/2020/12/30/how-genify-used-a-transformer-based-model-to-build-a-recommender-system-that-outperforms-industry-benchmarks/

BERT takes as input a sequence of words (limited to 512 tokens) and transforms them into numerical input representations to pass to the model. These embeddings have three components: a token embedding (value associated to the token based on a static vocabulary of the model); a segment embedding (representation of the segment that includes the word); and finally a positional embedding (just like the name indicates, it represents the position in the input sequence). The addition of these three components results in the word embeddings. We can see their structure in Figure 3.9.

BERT performs a transformation of the input, which corresponds to a sequence of words (limited to 512 tokens), in order to produce numerical input representations to transmit to the model. In practice, these input representations are built by adding three types of embeddings: token, segment, and positional embeddings [20]. Figure 3.9 shows a graphic representation of this structure.

3.3.2.A Token Embeddings:

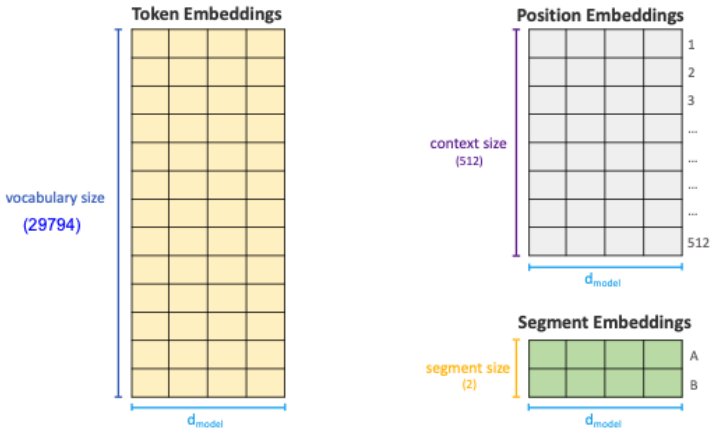

Figure 3.10: Parameters of the input embedding layer (of BERTimbau - Portuguese BERT)

BERT tokenizes a given word in the input sequence using WordPiece embeddings [21]. For a given language corpus, WordPiece constructs a fixed-size vocabulary of individual characters, subwords and words. The tokenizer begins the tokenization process by checking if the entire term is in the vocabulary. If this isn't the case, it tries to divide the word into as many subwords as possible. If it can't find the necessary subwords, it will split the word down into individual characters. After it has been broken down into one or more wordpieces, the vocabulary ids of these tokens are utilized to recover the appropriate embeddings included in its learned token embedding matrix, as shown in Figure 3.10, where d_{model} denotes the dimension of the model's input/output.

BERT's vocabulary includes the most frequent words and subwords in the language on which the model was trained(english for classic BERT, Portuguese for BERTimbau), as well as all of the language's

characters and three special tokens [20]:

- $[CLS]$, which is a unique classification token that appears at the start of each sequence. It is disregarded in non-classification tasks.

- $[SEP]$, a separator that is used when dealing with sentence pairs packed into a single sequence. It also always brings the sequence to a close.

- $[MASK]$, which is utilized for the Masked Language Modeling (MLM) training goal, explained in more detail later on.

3.3.2.B Segment Embeddings:

These embeddings correspond to learned embeddings that are used whenever working with sentence pairs. A segment embeddings is added to each token, to represent if it belongs to sentence A or B [20]. These embeddings are identical to token embeddings, except they have a vocabulary of size 2, as shown in Figure 3.10.

3.3.2.C Position Embeddings:

BERT, like the original Transformer, employs positional embeddings to insert information about the relative or absolute position of tokens in the input sequence. These share the same dimension as the token and segment embeddings, as shown in Figure 3.10, allowing them to be added easily [20]. Their computation is shown in equations 1 and 2 in Section 3.1.4.

3.3.3 BERT Pre-Training:

BERT is pre-trained on two tasks at the same time: MLM and Next Sentence Prediction (NSP). The training loss is equivalent to the sum of the mean MLM and mean NSP likelihoods [20].

3.3.3.A MLM

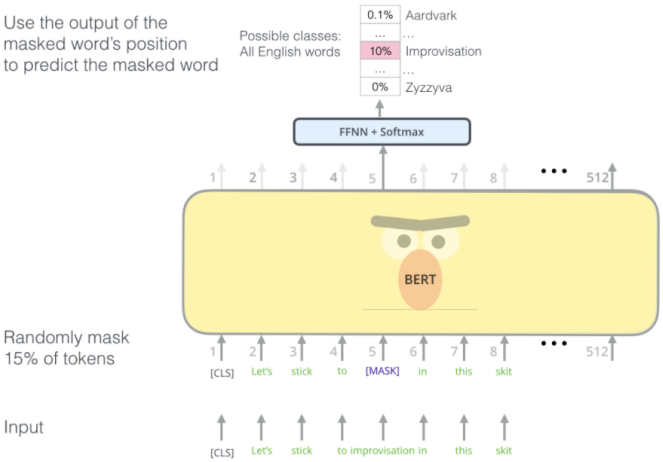

Figure 3.11: Masked Language Model [4]

In contrast to other language representation models, BERT pre-trains bidirectional representations by joining the context from left and right in its layers. Even though this process might look clean and simple there are problems that emerge from it. In unidirectional training, there is an unbiased prediction of the next word. With bidirectional training, since to predict a word we look at the surrounding words (left and right), each word indirectly sees itself, this results in words being able to predict themselves. To resolve this issue BERT employs the masked language modeling procedure, which corresponds to the task of predicting a percentage of input randomly masked tokens. It masks a set of k input tokens (replaces token with [MASK]) and then predicts those masked tokens. It learns to predict the original identities of the k masked-out tokens by computing an output word distribution $y\hat{}(h)(h = 1, ..., k)$ for each one of them. Then it calculates the loss using cross-entropy between the predicted probability distribution $y\hat{}(h)$ and the true word distribution. The overall loss of the sequence is simply the average loss for all the k masked-out tokens.

The drawback from only predicting this masked words is that it results in a discrepancy between pre-training and fine-tuning, since [MASK] is not included in the fine-tuning. To alleviate this, BERT does not always replace the mask words with [MASK]. For the fraction of chosen tokens (the ones to predict), it replaces those tokens with the [MASK] token 80% of the time and a random token 10% of the time. For the remaining 10% of the time it just leaves the original tokens (no replacement is done), in order for the model to use this bias to predict the correct word.

3.3.3.B NSP

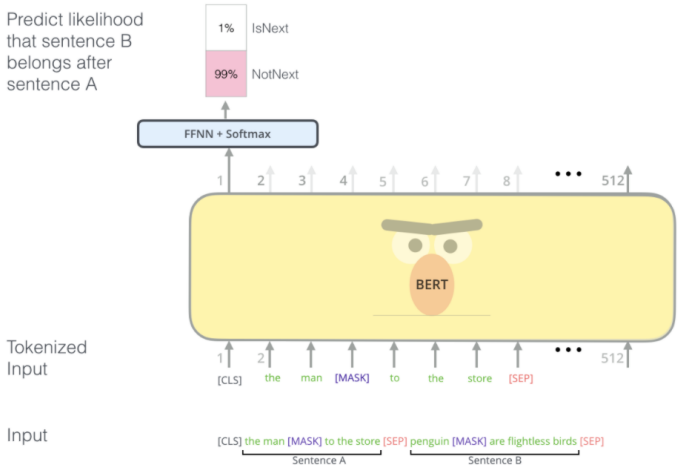

Figure 3.12: Nest Sentence Prediction [4]

The task of NSP is a binary classification problem, where the model is given pairs of phrases and is asked to predict whether the second sentence in the pair is the next sentence in the original corpus(label as IsNext or NotNext). This training goal aids in comprehending the link between pairs of sentences, which is relevant for many downstream tasks like Question-Answering (QA) and Natural Language Inference (NLI). The NSP task can be applied to any corpus that contains text in only one language [20]. An example of this task is shown in Figure 3.12.

3.3.4 Downstream Tasks

Fine-tuning and feature-based approaches are used to apply pre-trained language representations to downstream NLP tasks. The fine-tuning method brings in the minimal task-specific parameters and is trained on the downstream tasks by merely fine-tuning all pre-trained parameters. The feature-based method is more complex since it only uses the pre-trained representations as input features for a task-specific architectures [20].

3.3.4.A Fine-Tuning Strategy

BERT's two pre-training goals allow it to be utilized on any single sequence and sequence-pair task without requiring significant task-specific architecture changes. For each task, all that is required is to add the task-specific inputs and a new output layer that receives the BERT outputs, as well as fine-tune all the parameters for a few epochs [20].

3.3.4.B Feature-Based Strategy

In contrast to the fine-tuning approach, where a simple output layer is added to the pre-trained model and all parameters are fine-tuned together on a downstream task, in the feature-based approach word representations are taken from the pre-trained model and are used as input to other task-specific architectures. This approach offers some advantages over the fine-tuning one. Some tasks can not be represented by simply using a Transformer encoder architecture, needing instead the addition of a task-specific model architecture. Also, there are significant computational advantages of pre-computing an expensive representation of the training data once and then running many experiments with cheaper models on top of it [20].

There are many methods for extracting contextual word embeddings from BERT representations, and which one is preferable depends largely on the task at hand. Devlin et al. [3], for example, performed a study for the NER task, where they applied a feature-based approach. They fed the contextual embeddings, from the pre-trained model, to a randomly initialized BiLSTM before the classification layer. As demonstrated in Figure 3.13, concatenating the last four hidden layers as contextual word embeddings resulted in the highest F1 scores [20].

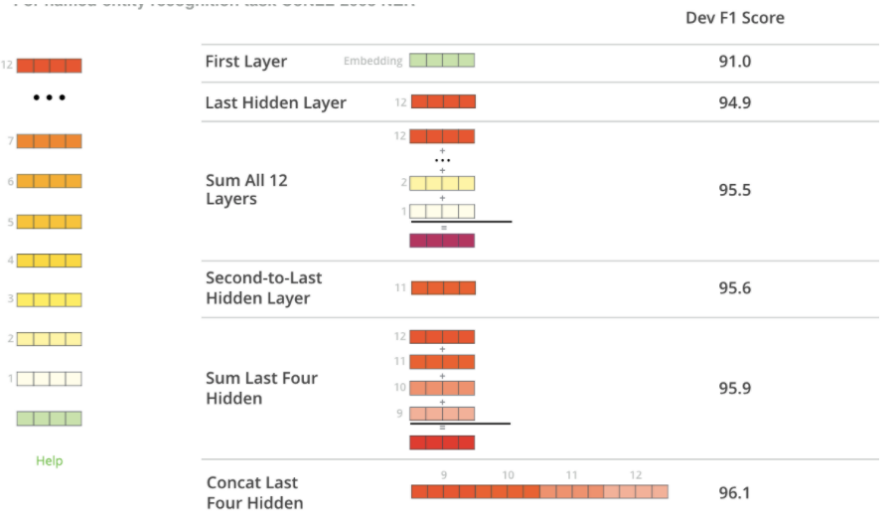

Figure 3.13: Results on NER using BERT embeddings with a feature-based approach [4]

From all the models we saw previously, the ones that are more relevant in the context of our problem are the FFNNs, described in Section 3.1.1, the BiLSTMs, described in Section 3.1.4, and BERT, described in Section 3.3. This decision was made after analyzing previous NER approaches, with are described in the following section.

4

Related Work on Semantic Parsing and information extraction

Contents

In this chapter, we will go through the related work and state-of-the-art models that were studied and analyzed, including current approaches and the state-of-the-art of information extraction systems and NER.

4.1 Dependency Parsers

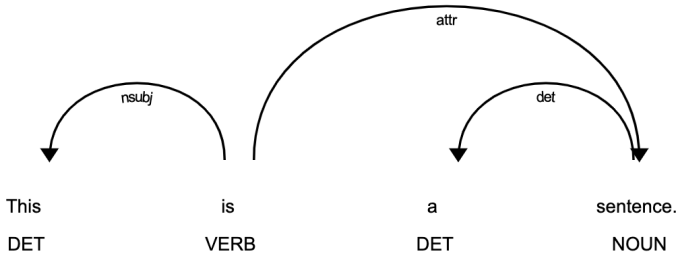

Figure 4.1: Visualization of the dependency relationships found by a Dependency Parser [1]

Dependency parsers [22] are used to extract relationships between words by analyzing the grammatical structure of a sentence. Each relationship is composed of a head and a dependent, that modifies the head, and is labeled according to the type of relationship that exists between the head and the dependent. They allow us to annotate sentences in a human understandable representation, and are useful for numerous NLP tasks. Dependency parsers also allow us to build dependency graphs, which are simply direct graphs that represent the dependencies found between the words, and dependency trees, which correspond to directed acyclic graphs where nodes represent words and edges represent the relationships between the nodes(words). Dependency parsers can be divided into two groups: transition-based and graph-based parsers.

Transition-based parsers were built based on the Shift-reduce parser [23]. This parser employs a context-free grammar[2], a stack, and a list of tokens to be parsed. Its objective is to reduce a string to the start symbol of a grammar. As the name indicates, it performs two actions: shift and reduce. The right-hand side of the rules of the grammar is matched against the input tokens as they are pushed into the stack. If there is a match then the matched elements are replaced on the stack (reduced) by the non-terminal symbol that they correspond to, according to the rules of the grammar. Transition-based parsers, like shift-reduce parsers, parse sentences from left to right. As the name suggests, a transition (action) is applied at each time step until the parsing is done. The transition (action) chosen at each step results from training the model on other sentences to predict the correct actions to perform in order to

[1] https://naturallanguage.pro/en/$dependency_parser.html$
[2] https://www.oxfordreference.com/view/10.1093/oi/authority.20110803095634847

obtain the corresponding dependency graphs. Consequently, these parsers do not use machine learning for directly predicting edges, instead they use it to predict the operations of the transition algorithm.

In contrast, Graph-based parsers, use machine learning to assign a weight or probability to each possible edge and then construct a Maximum Spanning Tree (MST)[3] from these weighted edges.

4.2 Semantic Role Labelling

Semantic (or thematic) roles [5] constitute one of the basic semantic information that can be extracted from a sentence. They correspond to the semantic relation between a predicative element (say, a verb) and its (both core and non-core) arguments, identifying the "who did what to whom, when and where" and thus, their representation helps systems to better "understand" events, their elements and the relations among them.

Thematic Role	Definition
Agent	The volitional causer of an event
Experiencer	The experiencer of an event
Force	The non-volitional causer of the event
Theme	The participant most directly affected by an event
Result	The end product of an event
Content	The proposition or content of a propositional event
Instrument	An instrument used in an event
Beneficiary	The beneficiary of an event
Source	The origin of the object of a transfer event
Goal	The destination of an object of a transfer event

Figure 4.2: Traditional Thematic Roles [5]

No consensual list of basic semantic roles exists, and the topic is controversial. Figure 4.2 shows a list of arguably the most consensual, classic semantic roles. A different list can also be found in [24]. Some have larger sets than others, some have roles that are more abstract to certain situations (like Proposition Bank [25]), while others are more specific. Semantic roles are used as shallow semantic representations to make basic conclusions that cannot be made from ordered, rooted trees that represent the syntactic structure of sentences according to some context-free grammar, like a parse tree [26] for example. Still, it has its own limitations, for example, it is almost impossible to create a universally consensual set of roles and provide a definition for each, since some thematic roles may correspond to words whose position varies in a sentence and may not always have the same dependency relation for every situation. For example, in the sentence "John broke the glass", there is a relation between the event and the causer of the event ("Agent"). In this case, the event is dependent on "John", which is a subject. In contrast, in the sentence "The glass was broken by John", we have again a dependency relation, with John being the causer of the event, but in this case he is not the subject.

[3] https://automaticaddison.com/what-is-a-maximum-spanning-tree/

4.3 Information Extraction with SRL

Not long ago, Humphreys et al. [6], tackled the problem of how to automate knowledge extraction from legal text, and how to use such knowledge to populate legal ontologies. They built a system based on NLP and post-processing rules based upon domain knowledge. They argue that they were the first ever to make use of the SRL Proposition Bank to extract norms and definitions from legal text.

Type	Element	
Obligation	Norm Type	Exception
Power	Definiendum	Reason
Legal effect	Definiens	Situation
Definition	Includes	Result
Permission	Excludes	Object
Scope	Action	ExcludesObject
Rationale	Active Role	HigherPriority
Right	Passive Role	LowerPriority
Exception	Condition	
Hierarchy	Timeframe	

Figure 4.3: Norms and Elements [6]

Their system can be divided into two main components. The first is a Mate Tools semantic role labeler [27]. This component is responsible for extracting an abstract semantic representation, along with dependency parse trees. The second is a group of rules, responsible for recognizing norms and definitions, classifying their type and mapping arguments in the semantic role tree to domain-specific slots in the legal ontology. In Figure 4.3 we can see the type of Norms and which elements they extracted, such as “Definiendum”, which corresponds to the subject of a definition, and “Definiens”, which corresponds to a word or a sequence of words used to define the subject “Definiendum”. By using SRL, the rules needed to extract legal information, are simplified, since the parse tree is enhanced with pertinent semantic information. In figure 4.4, we can see this two-stage extraction process.

Legal text	Extracted SRL Roles	Extracted Norm
A lawyer registered in a host Member State under his home-country professional title may practise as a salaried lawyer in the employ of another lawyer, an association or firm of lawyers, or a public or private enterprise to the extent that the host Member State so permits for lawyers registered under the professional title used in that State.	<SRL> <PREDICATE>practice</PREDICATE> <A0:SBJ> A lawyer registered in a host Member State under his home-country professional title </A0:SBJ> <A2:ADV> practise as a salaried lawyer in the employ of another lawyer, an association or firm of lawyers, or a public or private enterprise, to the extent that the host Member State so permits for lawyers registered under the professional title used in that State </A2:ADV> </SRL>	<Norm> <NormType>Permission</NormType> <ActiveRole> A lawyer registered in a host Member State under his home-country professional title <ActiveRole> <Action> practise as a salaried lawyer in the employ of another lawyer, an association or firm of lawyers, or a public or private enterprise </Action> <Condition> to the extent that the host Member State so permits for lawyers registered under the professional title used in that State </Condition> </Norm>

Figure 4.4: Two-stage Extraction Process Example [6]

In terms of the evaluation of the system, the authors started by calculating the accuracy of the SRL on legal text. They saw that after testing it on 58 definitions and 166 norms, 78.5% of definitions had all the arguments for all the verbs correct, only 52% of norms had fully correct arguments. In spite of this, the overall performance of the system was relatively good, reaching an f-score of 76.33 in detecting Obligation type norms and 85.15 in power type norms. Additionally, they evaluated the system in unseen legal texts, and saw that there are some norms and some of their elements (such as Active Role) that are extracted with high accuracy while for some other elements (such as Passive Role and Conditions), the system seems to face problems detecting them.

4.4 Deep Biaffine Classifier

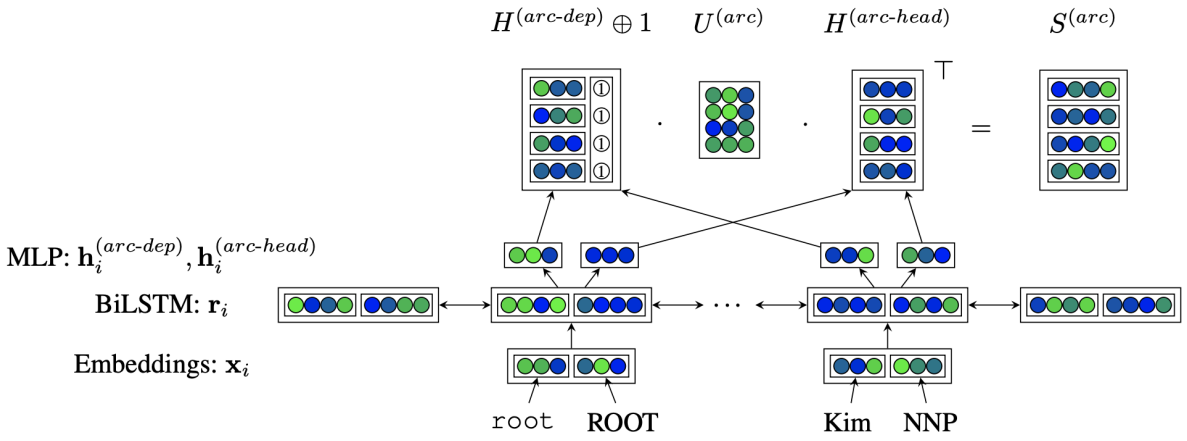

Figure 4.5: BiLSTM with deep biaffine attention to score each possible head for each dependent, applied to the sentence “Casey hugged Kim” [7]

The Deep Biaffine Classifier [7] is a graph-based dependency parser, which uses neural attention mechanisms. It was built by applying certain modifications to previous graph-based parsers including Kiperwasser and Goldberg's Dependency Parser [28]. Figure 4.5 corresponds to the model architecture for the sentence: "Casey Hugged Kim". From the image, we can see that this model applies a BiLSTM to the input sequence, generating a representation for each token (4.1), that takes into account both the entire history(previous tokens) and the entire future(following tokens).

$$r_i = BiLSTM(x_{1:n}, i) \tag{4.1}$$

Then, it applies two Multi Layer Perceptron networks (4.2 and 4.3) to reduce the dimension of each output vector (of the BiLSTM), using ReLu as the activation function. This is done with the purpose of dropping irrelevant information.

$$h_i{}^{(arc-head)} = MLP^{(arc-head)}(r_i) \tag{4.2}$$

$$h_i{}^{(arc-dep)} = MLP^{(arc-dep)}(r_i) \tag{4.3}$$

After the dimensionality reduction is done, it applies a biaffine attention mechanism, which the authors denoted by deep bilinear attention mechanism. A biaffine attention mechanism, can be thought as a simple affine classifier except with some small changes. An affine classifier corresponds to affine transformations, which is a geometric transformation that maintains lines and parallelism. For example, if X is a set of points, an affine transformation of X can be represented by a linear transformation on X and a translation on X. These type of transformations can include translations, scaling, rotation, among others [4]. The name Biaffine comes from applying an affine transformation twice (4.4). The authors argued that the mechanism had the advantage of directly modeling both the prior probability of a word receiving any dependents and the likelihood of a word receiving a specific dependent i. This mechanism can be seen in equation 4.4.

$$s_i{}^{(arc)} = H^{(arc-head)}(U^{(1)} h_i{}^{(arc-dep)} + u^{(2)}), \tag{4.4}$$

where $s_i{}^{(arc)}$ corresponds to the probabilities of a word i being dependent of the other words and $U^{(1)}$ and $u^{(2)}$ are simply weight matrices, that correspond to the transformations and are learned by the model. They calculate the predicted head to an word i by choosing the highest scoring head:

$$y_i = argmax s_i{}^{(arc)} \tag{4.5}$$

Equivalently, the model [7] also uses a biaffine classifier to predict the labels given a predicted head

[4] https://en.wikipedia.org/wiki/Affine_transformation#Examples

y_i:

$$s_i{}^{(label)} = r_{y_i}^T U^{(1)} r_i + (r_{y_i} \oplus r_i)^T U^{(2)} + b,$$

where $s_i{}^{(label)}$ corresponds to the probability of a relationship between the predicted head y_i and i having that label, $U^{(1)}$ and $U^{(2)}$ are simply weight matrices and b corresponds to the bias.

4.5 Named Entity Recognition

Named Entity Recognition (NER), or Entity extraction, is an information extraction task. This task consists of the identification and classification of named entities in text. For example, for the sentence "James Bonds lives in London" we could use the NER task to identify the entities Person ("James Bond") and Location ("London"). We could also be dealing with more complex situations where there could be nested or overlapping entities, Nested Named Entity Recognition. For example, in the sentence, "James Bond was born on 16th December of 1920." we could identify the entity Date ("16th December of 1920") and also the entities Day ("16th"), Month ("December") and Year ("1920").

4.5.1 Named Entity Recognition Approaches

In recent years, many works regarding NER have been put forward. Most of the NER models that have been created focused only on flat NER. Some use Conditional Random Fields (CRFs), which are discriminative classifiers, that make predictions taking into account the context, by modeling the predictions as graphical models that create dependencies among each prediction. Some of these works include applying linear-chain conditional random fields [29] and semi-Markov conditional random fields [30].

Still, there are a few models that deal with extracting nested entities along with flat entities. Lu and Roth (2015) [31] proposed a hypergraph model to detect nested entities, their model had a linear time complexity. Muis and Lu (2017) [32] improved the model by proposing a multigraph representation based on mention delimiters. In order to not allow the model to learn fraudulent structures, the authors assigned tags between each pair of sequenced words. These structures correspond to entities that overlap each other that are grammatically impossible.

In the last years, many more neural models have been put forward targeting nested NER as well. Katiyar and Cardie (2018) [33] expanded Muis and Lu (2017) model. They created a model that learns the hypergraph representation directly for nested entities by utilizing features extracted from adapting an LSTM. Katiyar and Cardie use two different training methods, they tried using softmax and sparsemax to

calculate the predicted distribution for the labels. In contrast to softmax that always results in a non zero probability, sparsemax usually output zero probability when the scenario is unlikely. During evaluation they saw that using the latter provided better results. They also saw that their model, using sparsemax, had some limitations regarding the prediction of labels when dependent on context, and when predicting the entity “it”.

Fisher and Vlachos (2019) [34] solution breaks nested NER into two stages. First, it recognizes the limits of the named entities at all degrees of nesting. They move the embeddings of each pair of subsequent words through a FFNN, in order to predict if the tokens should be part of the same entity. Whenever the network returns values close to 0, this indicated that the pair of adjacent tokens are part of the same entity. Second, it generates embeddings for each entity based on the resulting structure by combining the embeddings of smaller entities/tokens from prior levels. Tokens are combined into entities, which are combined with other tokens or entities in higher levels. These merges are encrypted as real-valued decisions, allowing for a parameterized mixture of word embeddings into entity embeddings at various levels. These are utilized to classify the predicted entities. The model then uses FFNN layers to make the label predictions for each entities. Training with back-proppagation is simple when utilizing this kind of network. In contrast to other approaches like Katiyar and Cardie, its entity predictions are not 0-1 labels, allowing the model to flexibly accumulate information across layers. During evaluation, they saw that using contextual embeddings such as ELMO and BERT increased the performance of their system, the latter being the one that resulted in the best performance.

Additionally, Yu, Bohnet, and Poesio [8] proposed a model based on the dependency parsing model of Dozat and Manning [7] to provide a full perspective on the input using a biaffine model. Since our system is based on this model, we will explain it in more detail in the following section.

Finally, recently this year, Neda and Mark [35] proposed an information extraction process in order to build a legislation network. Due to the structure of their legal corpus, they were able to apply NER by defining clear rules to extract the relevant entities. An example of one of their rules is “the [keyword] of this act [keyword] the [act name] [year]”, where the keywords correspond to “short title” and “shall be”. This rule applies to any act that was commenced between the years 1850 and 1900.

4.5.2 Different Representations for Start and End of Spans

In this section we will describe in more detail Yu, Bohnet, and Poesio’s model [8]. Since this model is relevant to our SNR System, we decided to place it in its own section. In 2020, the authors [8] proposed a model based on the dependency parsing model of Dozat and Manning [7] to provide full perspective on the input using a biaffine model. The overview of their model is shown in Figure 4.6.

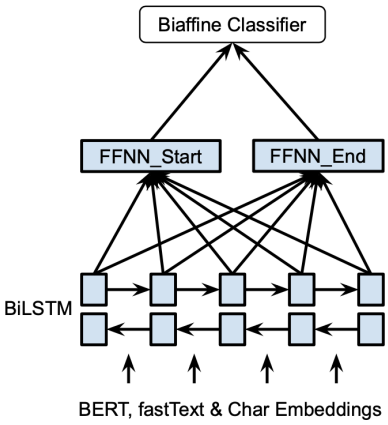

Figure 4.6: Overview of Yu, Bohnet, and Poesio's model [8]

They used word (BERT and fastText) and character (charCNN) embeddings and feed them towards a BiLSTM, to learn the words' representation, and lastly to a biaffine classifier. Before, applying the biafiine classifier, they applied two FFNNs to the word representations generated by the BiLSTM, in order to learn different representations for the start and end of the spans. Each FFNN computed representations h_s, h_e, for the start and end of the entities, respectively. Yu, Bohnet, and Poesio argumented that using different representations for the start and end of spans, allowing the model to learn these representations separately, resulted in an increase in accuracy when compared to previous systems that did not follow this idea. They stated that the increase in accuracy, comes from the fact that the start and end of spans have different contexts, thus their learning should be done separately.

The biaffine classifier [7] allowed the model to have a global view on the input by generating a scoring tensor r_m, of size l x l x c, over the input sentence. l corresponds to the sentence length and c to the number of entities, plus one to represent non-entities. The score for each span i, corresponded to:

$$r_m(i) = h_s(i)^T U_m h_e(i) + W_m(h_s(i) \bigoplus h_e(i)) + b_m$$

where U_m was a d$\times$c$\times$d tensor, W_m a 2d$\times$c matrix and b_m the bias. All the valid spans (spans whose end is after its start) were scored by the tensor. Then, they assigned for each span the entity label with the highest score:

$$y'(i) = argmax\ r_m(i)$$

After having all spans and their possible labels they also did a final post-processing step to make

sure there were not any nested entities whose boundaries clash. For example, the entities: “the article number 5 defines an” and “article number 5 defines an obligation to consumers” clash with each other (boundaries overlap), and so in this situation, they would select the one with a higher score. The model’s learning goal was to assign the right category (including non-entity) to each valid span. Being a multi-class classification problem the authors optimized the model with softmax cross-entropy:

$$p_m(i_c) = \frac{exp(r_m(i_c))}{\sum_{c'=1}^{C} exp(r_m(i_{c'}))} \tag{4.6}$$

$$loss = \sum_{I=1}^{N} \sum_{c=1}^{C} y_{i_c} log p_m(i_c) \tag{4.7}$$

By looking at Figure 4.7, we can see that the model outperformed the previous ones mentioned above. Yu, Bohnet, and Poesio also evaluated their model on seven other datasets, including ACE2004, GENIA, ONTONOTES, among others. For all the datasets, their model was the one that achieved the best results.

ACE 2005			
Katiyar and Cardie (2018)	70.6	70.4	70.5
Wang et al. (2018)	-	-	73.0
Wang and Lu (2018)	76.8	72.3	74.5
Lin et al. (2019)	76.2	73.6	74.9
Fisher and Vlachos (2019)	82.7	82.1	82.4
Luan et al. (2019)	-	-	82.9
Straková et al. (2019)	-	-	84.3
Our model	85.2	85.6	**85.4**

Figure 4.7: Precision, Recall and F1 scores on ACE2005 dataset [8]

4.6 Other Related Work regarding Legal Contracts for Consumers

Recently this year, Ruggeri et at. [36] proposed a data-driven approach for detecting and explaining fairness in legal contracts for consumers. They treated the classification of unfairness clauses as a classification problem, but in order to be able to explain the fairness and not simply classify it, they incorporated external knowledge into their data-driven approach. They did this by using the Memory-Augmented Neural Network (MANN). MANNs [37] are neural networks that can save and get data from an external memory. Usually, typical machine learning algorithms (such as neural networks) accept input and process it in order to make a prediction, MANNs can use explicit memory to store relevant information and retrieve it when needed. So the author’s system works by feeding a clause q to the network and the system then retrieves from the memory (set of legal explanations) the most similar information regarding the clause q, in order to generate a final representation for the original clause that

will be utilized to classify the clause itself.

There have been many different works regarding data approaches applied to the legal domain, but all mostly focus on making legal information more understandable and explainable, which shows the importance of making these types of texts interpretable for regular citizens.

5

Dataset Creation

Contents

Like we have mentioned, our goal is to create an automatic information extraction system, which we denoted by SNR, capable of extracting relevant concepts (such as norms, semantic roles, and named entities), from a specific corpora that contains the Portuguese Consumer Law. Since we are dealing with a classification problem, we need to have a dataset with the laws and corresponding gold labels. For this reason, the corpus was annotated so we could generate the needed dataset for our task. This chapter covers the creation of our dataset, with a brief description of the corpus, of the annotation process and the tools used.

5.1 Corpus

The SNR system uses a subset of legal articles from the Portuguese Consumer Law. These legal texts were made available by INCM and were segmented at the level of the article number, containing in total 5.600 segments, and 341.392 tokens. This segmentation was done at the level of the article number, due to it being considered the level that gave the necessary context to extract and annotate the relevant semantic information, since it corresponds to a full sentence.

Thus, each segment, roughly speaking, corresponds to a number of an article of a legal act. For example, segment ID 2720, shown in Figure 5.1, corresponds to the number 1 of article 24º of the Decree-Law no. 72/2008 of 16th April.

> *1 - O tomador do seguro ou o segurado está obrigado, antes da celebração do contrato, a declarar com exatidão todas as circunstâncias que conheça e razoavelmente deva ter por significativas para a apreciação do risco pelo segurador.*

Figure 5.1: Example of a segment from the corpus[1]

Each segment is associated with a unique identifier (segment ID) and other metadata regarding the legal text from which the segment was extracted. This information includes the number of the legal act, its publishing date, chapter, paragraph number, and so on.

5.2 Annotation Procedure

It is not enough to provide large amounts of data to machine learning models and expect them to learn important information. When dealing with a classification (supervised learning) problem, annotating a corpus is crucial for a model to be able to find patterns and make inferences. Also, annotations have to be precise and pertinent in regards to the task at hand, in order for models to make accurate predictions.

[1] The segment translates to: "1 - The policyholder or the insured is obliged, before celebrating the contract, to accurately state all the circumstances that he/she knows and reasonably should consider as significant for the insurer's assessment of the risk."

In the context of this work, the SNR system's goal is to extract norms, semantic roles, and named entities present in legal text. Thus, we needed to have our corpus annotated with these concepts.

5.2.1 Labels Choice

Before starting the annotation process, the type of information (which norms, semantic roles, and named entities) to be annotated, needed to be defined. Thus, a set of directives, based on Humphreys *et al.* [6], but drawing on examples from Portuguese legal texts, was produced to guide the annotation task.

> A lawyer registered in a host Member State under his home-country professional title may practise as a salaried lawyer in the employ of another lawyer, an association or firm of lawyers, or a public or private enterprise to the extent that the host Member State so permits for lawyers registered under the professional title used in that State.

Figure 5.2: Example of "Permission" category provided in Humphrey et al. [6]

The norms, shown in Figure 5.7, closely follow those of Humphrey *et al.* [6], v.g. "Definition", "Right", "Obligation", "Legal effect", and "Intro". Some concepts defined in [6] were not included, either because they were considered insufficiently defined for practical application in the task, or because of having been considered to involve distinctions too thin or too hard to signal from the textual evidence. For example, the concept of "Power" was integrated in "RIGHT", being difficult to differentiate them on formal (i.e. lexical) grounds. The concept of "Permission" can not clearly be derived from the definition and the example, provided by Humphrey *et al.* [6], shown in Figure 5.2. In our view, this concept necessarily requires as arguments two entities and a propositional object (an action or a process), with a facultative "medium" complement. Other concepts, such as "Rationale" were renamed as "INTRO" for the segments included in the legal text that would usually appear in a preamble to justify its need or context of application. The legal preambles, since they are not part of the law itself, were excluded from the corpus. The concept of "Legal effect" was kept but redefined for cases where the legal text refers to itself as a whole concerning its coming into effect, revoking some previous legal text or changing the content of some legal reference.

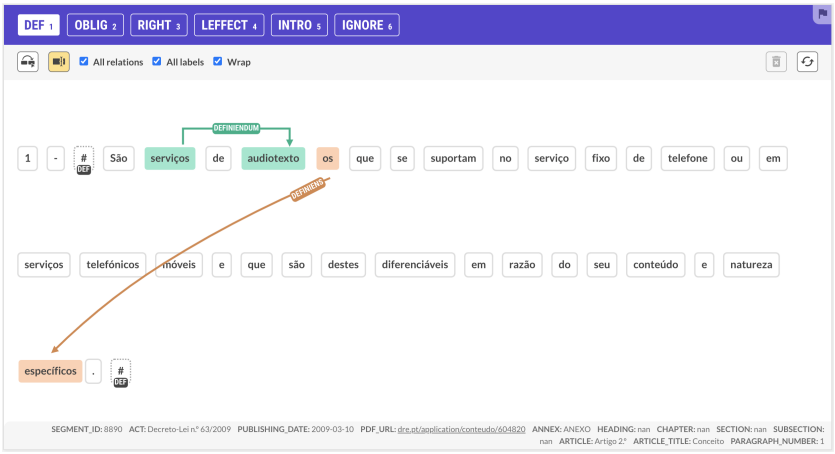

Figure 5.3: Example of sentence with definition and corresponding semantic roles

Regarding the semantic roles, the concepts were chosen not only due to being considered essential, regarding our legal domain, and their frequency in our corpus but also because part of these concepts is related to a type of norm. For example, a definition includes semantic roles that are specific to the definition itself, as we can see in Figure 5.3. For this norm , our related semantic roles (“DEFINIENDUM”, “DEFINIENS”, “DEF-INCLSUION”) closely follow those of Humphrey *et al.* [6], v.g. “definiendum”, “definiens”, “inclusion”. For the norm “LEFFECT”, the annotators saw that this norm implies the semantic role “EFFECT”, which represents the legal effect. For the norms “RIGHT” and “OBLIG”, the annotators saw that these norms frequently imply two main semantic roles: “ACTION”, which represents the content of a right or obligation, similarly to Humphrey *et al.* [6], and “EXPERIENCER”, which represents the entity that has the right or obligation. An example of this is shown in Figure 5.4.

Figure 5.4: Example of sentence with “ACTION” and “EXPERIENCER”

From the concept of “ACTION”, came the semantic role “THEME”. This concept indicates an object of nuclear predictive element of an “ACTION”. It was found to be associated with “ACTION” especially when “EXPERIENCER” is omitted. This difference regarding which concept (“EXPERIENCER” or “THEME”) is paired with “ACTION” can be seen in Figure 5.5.

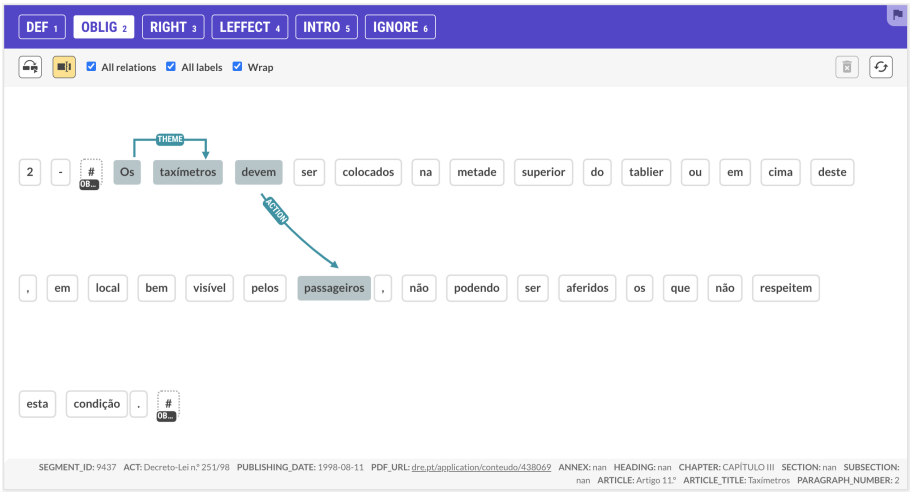

Figure 5.5: Example of sentence with “ACTION” and “THEME”

Humphrey *et al.* [6], also included the semantic roles “Exception” and “Condition”, which the annotators decided to include since they represent concepts that were very frequent in our corpus and were considered to be essential in the legal domain. The roles “CONCESSION” and “PURPOSE” were added by the annotators from also being considered frequent and essential in our corpus. Humphrey *et al.* [6] included a concept denoted by “Scope” as a type of category, just like the other norms. The annotators changed this, and decided to include “SCOPE” as a semantic role that represents the context in which the norms are being applied, since this is a better representation of the information in our corpus. Finally, the annotators also decided to include the semantic role ”NEG”. The idea behind this concept was to represent situations in our corpus where norms were being negated. For example, a segment that has the expression “does not have the right to”, would be marked as a “RIGHT” with the semantic role “NEG”.

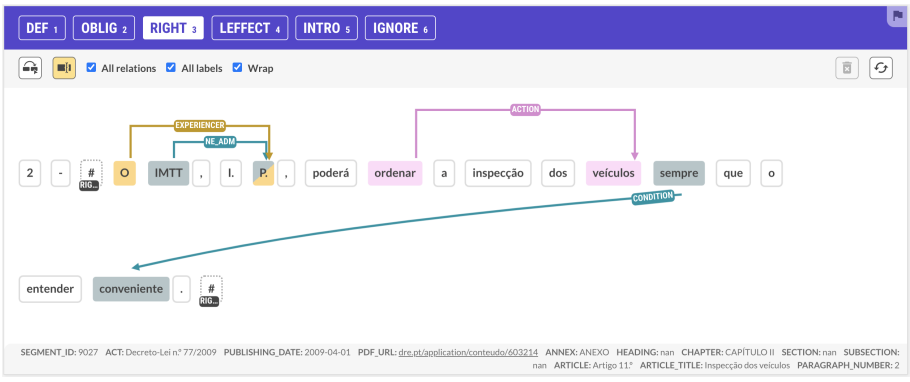

Figure 5.6: Example of a right with “NE_ADM”

Finally, for the named entities five main concepts, shown in Figure 5.7, were chosen to represent legal references and temporal expressions that were found after analyzing the legal texts of the Portuguese

Consumer Law. The temporal references were based on the temporal concepts of HAREM [38]. The named entity "NE_ADM" was chosen to represent any entity that references organs of the public administration of a state or states association, an example is shown in Figure 5.6. The entity "LREF" was created to represent expressions that allow identifying a legal act, and the entity "TREF" was created to capture the anaphoric expressions that refer to (part of) legal acts.

The document containing the set of directives, which contained the fundamental concepts and whose purpose was to guide the annotators in the annotation task, was updated throughout the annotation, especially with the inclusion of more examples for the most problematic cases. The list of all 23 final concepts is shown in Figure 5.7. Initially, there was a larger number of concepts, but some had to be removed due to not appearing enough in the corpus or due to being considered too complex to represent and/or inconsistently annotated.

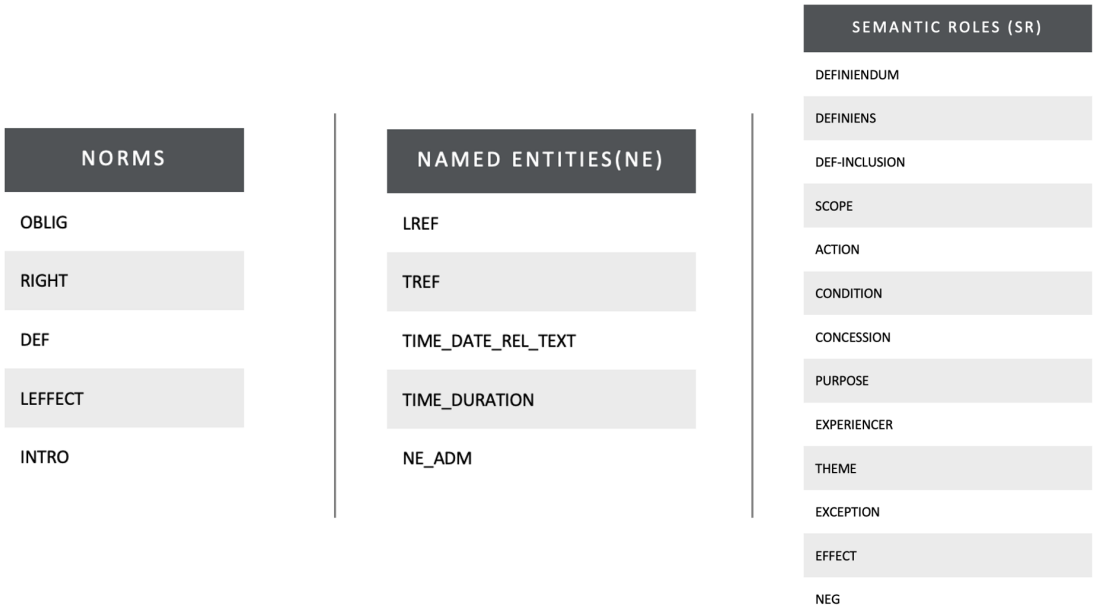

Figure 5.7: All 23 chosen concepts

5.2.2 Annotation Tool

Our problem corresponds to a classification task, for this reason, the annotation that needed to be done consisted of labeling spans with the correct label. This annotation was performed using the Prodigy tool[2]. Figures 5.8 and 5.9 show the tool's interface. This tool was chosen due to its intuitiveness and simplicity, which allows for a very fast and efficient annotation process.

[2]https://prodi.gy/

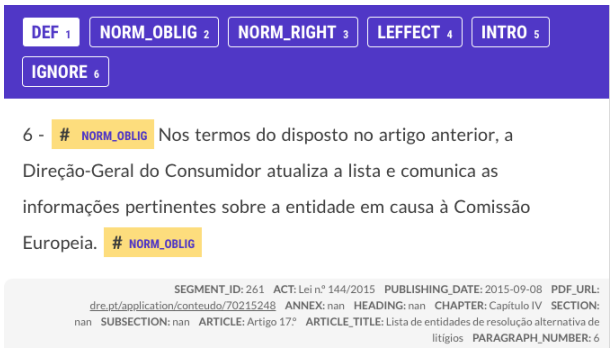

Figure 5.8: Interface of the Prodigy annotation tool used for the SNR Model: high level annotation

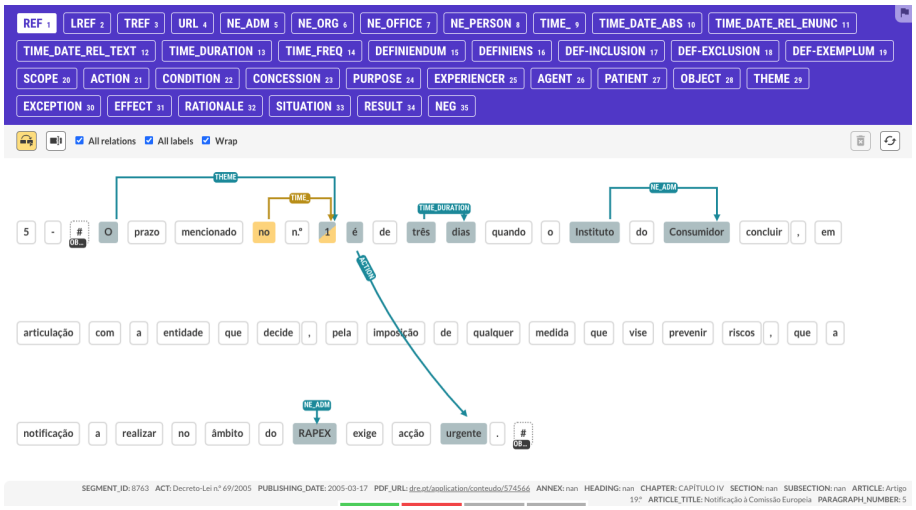

Figure 5.9: Interface of the Prodigy annotation tool used for the SNR Model: low level annotation

5.2.3 Annotation Phases

The SNR annotation task was done by three Portuguese experts and was divided into two main phases, corresponding to two levels of annotation: (i) a high level annotation, which consists of identifying the types of norms attributed to each segment; (ii) a low level, where within each segment, annotators looked for the named entities and semantic roles. All named entities and semantic roles were delimited and tagged accordingly. Both levels of annotation are shown in Figures 5.8 and 5.9. Also, throughout the annotation, the agreement between annotators was calculated, in order to make sure there was consistency between annotators in their annotations.

Due to some limitations of the Prodigy tool regarding the annotation of nested spans, placeholders ("#") for high level tags, were added at the start and end of each segment, which can be seen in Figures 5.8 and 5.9. Also, the high level tags were annotated by only tagging the start and end tokens. For the low level, the "relationship" feature was used, allowing annotators to connect the start and end of each

low level tag.

Once the two phases were completed, a posterior validation of the annotations, regarding the named entities and semantic roles, was done. This was done in order to make a verification of the corpus, label by label, checking not only the delimitation of different labels but the classification as well.

Having in mind the ambiguity and complexity of some of these labels and of this type of task, and also considering that it involved four different annotators which can cause inconsistencies and human errors, the annotation that was done is not perfect. Throughout the development of our system, we found many noisy labels in many segments which were then solved by the main annotator. This process of how our system helped find and reduce noise in our corpus, and the impact of the corrections on the system will be explained in more detail in Chapter 7.

5.2.4 Annotation Output

The output of the Prodigy tool, is a jsonl file, where each line corresponds to a single segment, with all the annotated information, as it can be seen in Figure 5.10.

{"text": "2 - # Quando convencionado, pode o segurador entregar a apólice ao tomador do seguro em suporte electrónico duradouro.
#", "meta": {"segment_id": "2762", "act": "Decreto-Lei n.\u00ba 72/2008", "publishing_date": "2008-04-16", "pdf_url":
"https://dre.pt/application/conteudo/249804", "annex": "Anexo", "heading": "T\u00edtulo I", "chapter": "Cap\u00edtulo II",
"section": "Sec\u00e7\u00e3o V", "subsection": "nan", "article": "Artigo 34.\u00ba", "article_title": "Entrega da apólice",
"paragraph_number": "2"}, "_input_hash": -213213513, "_task_hash": -840007753, "tokens": [{"text": "2", "start": 0, "end": 1, "id": 0,
"ws": true, "disabled": false}, {"text": "-", "start": 2, "end": 3, "id": 1, "ws": true, "disabled": false}, {"text": "#", "start": 4, "end": 5,
"id": 2, "ws": true, "disabled": false}, {"text": "Quando", "start": 6, "end": 12, "id": 3, "ws": true, "disabled": false}, {"text":
"convencionado", "start": 13, "end": 26, "id": 4, "ws": false, "disabled": false}, {"text": ",", "start": 26, "end": 27, "id": 5, "ws": true,
"disabled": false}, {"text": "pode", "start": 28, "end": 32, "id": 6, "ws": true, "disabled": false}, {"text": "o", "start": 33, "end": 34, "id":
7, "ws": true, "disabled": false}, {"text": "segurador", "start": 35, "end": 44, "id": 8, "ws": true, "disabled": false}, {"text": "entregar",
"start": 45, "end": 53, "id": 9, "ws": true, "disabled": false}, {"text": "a", "start": 54, "end": 55, "id": 10, "ws": true, "disabled": false},
{"text": "ap\u00f3lice", "start": 56, "end": 63, "id": 11, "ws": true, "disabled": false}, {"text": "ao", "start": 64, "end": 66, "id": 12, "ws":
true, "disabled": false}, {"text": "tomador", "start": 67, "end": 74, "id": 13, "ws": true, "disabled": false}, {"text": "do", "start": 75,
"end": 77, "id": 14, "ws": true, "disabled": false}, {"text": "seguro", "start": 78, "end": 84, "id": 15, "ws": true, "disabled": false},
{"text": "em", "start": 85, "end": 87, "id": 16, "ws": true, "disabled": false}, {"text": "suporte", "start": 88, "end": 95, "id": 17, "ws":
true, "disabled": false}, {"text": "electrónico", "start": 96, "end": 107, "id": 18, "ws": true, "disabled": false}, {"text": "duradouro",
"start": 108, "end": 117, "id": 19, "ws": false, "disabled": false}, {"text": ".", "start": 117, "end": 118, "id": 20, "ws": true, "disabled":
false}, {"text": "#", "start": 119, "end": 120, "id": 21, "ws": false, "disabled": false}], "_session_id": null, "_view_id": "relations",
"spans": [{"text": "#", "start": 4, "token_start": 2, "token_end": 2, "end": 5, "type": "span", "label": "RIGHT"}, {"text": "#", "start": 119,
"token_start": 21, "token_end": 21, "end": 120, "type": "span", "label": "RIGHT"}], "answer": "accept", "relations": [{"head": 3,
"child": 4, "head_span": {"start": 6, "end": 12, "token_start": 3, "token_end": 3, "label": null}, "child_span": {"start": 13, "end": 26,
"token_start": 4, "token_end": 4, "label": null}, "color": "#96e8ce", "label": "CONDITION"}, {"head": 7, "child": 8, "head_span":
{"start": 33, "end": 34, "token_start": 7, "token_end": 7, "label": null}, "child_span": {"start": 35, "end": 44, "token_start": 8,
"token_end": 8, "label": null}, "color": "#ffd882", "label": "EXPERIENCER"}, {"head": 9, "child": 19, "head_span": {"start": 45, "end":
53, "token_start": 9, "token_end": 9, "label": null}, "child_span": {"start": 108, "end": 117, "token_start": 19, "token_end": 19, "label":
null}, "color": "#b5c6c9", "label": "ACTION"}]}

Figure 5.10: Line of the prodigy tool output

Each line contains: the property “text”, which corresponds to the segment text; the property “meta”, which corresponds to segment id and metadata information of the segment; the“tokens”; a “spans” property describing the norms that were highlighted; and a “relations” property that contains the semantic roles and named entities that were highlighted as well.

After the annotation was done, the annotations from the prodigy output were converted to a more compact and simple format to be used as input for our SNR System. This is described in more detail in Section 6.5.1.

6

Semantic Norm Recognition System

Contents

This chapter covers the description of the implemented SNR system. Sections 6.1, 6.2 and 6.3 describe different versions of the SNR system that were implemented, Section 6.4 explains the training and optimization of the models, and finally Section 6.5 describes some details of the implementation that was done.

6.1 SNR System Overview

Like it has been mentioned, the purpose of this work is to create a system capable of identifying norms, semantic roles and named entities present in legal texts. These concepts can also be nested, which means for example there can be an “EXPERIENCER” inside an “ACTION” .

Once the annotation procedure was done, we saw that we could have spans with more than one label, with each belonging to a different group (either a norm, or a semantic role, or a named entity). For example, a certain span could not only be a “EXPERIENCER”(semantic role), but also a “NE_ADM”(named entity). These, and only these cases, correspond to a multi-label problem. A span that is labeled with a concept of a certain group (either norms, semantic roles, or named entities) will never be labeled with another concept of the same group. For example, a certain span that is labeled as “EXPERIENCER”(semantic role), will not have any other semantic role associated with it. Thus, for each group, we have a multi-class problem.

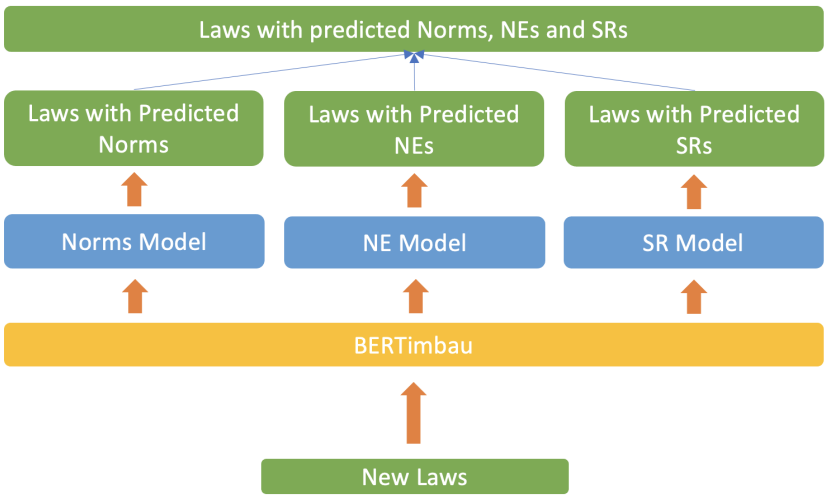

Figure 6.1: Baseline SNR System

To make sure we were only dealing with a multi-class problem for all concepts and not a multi-label problem, we created an information extraction system composed of three models: (i) the Norms Model, responsible for predicting the norms; (ii) the Named Entities Model, responsible for predicting the named entities; (iii) the Semantic Roles Model, responsible for predicting the semantic roles. All models have

the same architecture, but are trained to learn different types of labels (norms, named entities, and semantic roles, respectively). The overview of the SNR system is shown in Figure 6.1. A more detailed representation including inputs and outputs of each model and the Baseline system, for a given example, is shown in A.3.

The architecture of the models is based on the dependency parsing model of Yu, Bohnet, and Poesioo [8], with a small difference regarding the embeddings used, as we can see in Figure 6.2. We chose to use this model, since not only it is a nested NER model but also due to its state-of-the-art results compared to previous models [8]. Like we mentioned in Section 4.5.2, the authors used BERT, fasText and charCNN embeddings. Due to the nature of our corpus, which corresponds to Portuguese legal text, we used as word embeddings Portuguese BERT (BERTimbau) [39]. BERTimbau is the Portuguese version of BERT. Since BERT is the state-of-the-art for word embeddings, we decided there was no need to use fastText. Regarding charCNN, these character embeddings are mostly used to deal with misspelling words and emoticons. Since we are dealing with legislative text, which is subject to intense scrutiny and careful edition prior to its publication, the existence of misspelled words will be rare if any. Therefore this type of embeddings was not included in our system.

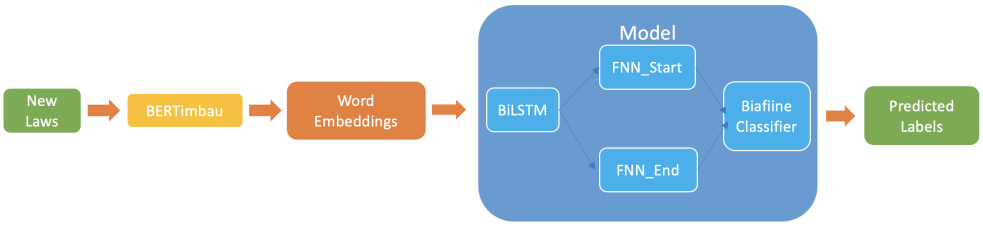

Figure 6.2: Architecture of each SNR System Model

The SNR system starts by using BERTimbau to generate the word embeddings for the dataset. This process is explained in more detail in Section 6.5.2. After generating the embeddings, it uses them as input for each of the three models. In each, the embeddings are feed to a BiLSTM, responsible for learning the word representations, that will be used by two feed forward networks, "FNN_Start" and "FNN_End", to learn all possible start and end representations, respectively. These start/end representations are then classified by a biaffine classifier, for each possible label (norms for the Norms Model, semantic roles for the Semantic Roles (SR) Model, and named entities for the Named Entities (NE) Model). The predictions of each model are concatenated together in order to obtain the final classification.

6.2 Two Phased SNR System

The system we previously described, let us denote it by Baseline SNR system, was built in order to learn to predict all the information that was annotated. Looking back at the annotation process, we saw that it was divided into two phases, with the annotators first deciding which norms were present in the segments - high level, and then deciding which named entities and semantic roles were present in those norms - low level. This lead us to create another version of our system, which we denoted by Two Phased, shown in Figure 6.3, to try to replicate the annotation process and human logic behind it. A more detailed representation including inputs and outputs of each model and the Two Phased system, for a given example, is shown in A.4.

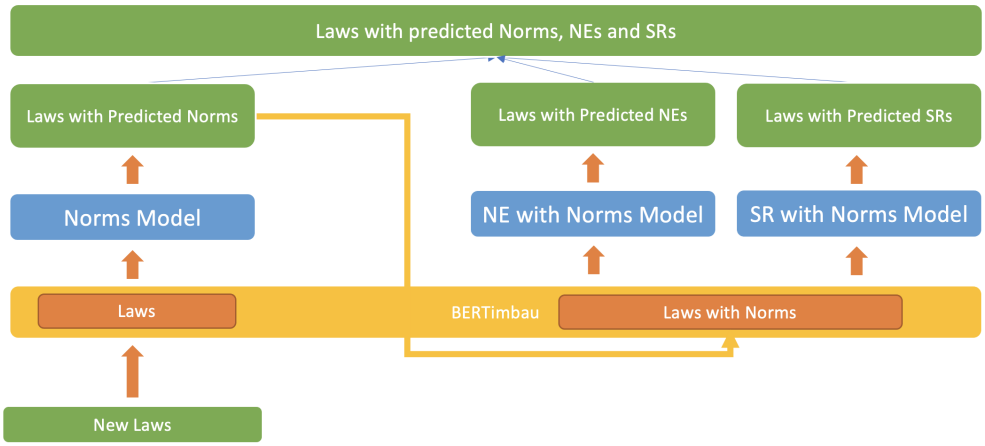

Figure 6.3: Two Phased SNR System

The Two Phased system is very similar to the Baseline system, the key difference being the fact that the system tries to learn to identify the named entities and semantic roles, knowing the norms that are present in the segments. Our idea was that by predicting the norms first (first phase), it would allow the system to focus on predicting the semantic roles and named entities for the norm in question only, making this second phase easier, just like it helped the annotation process.

This system starts by generating the word embeddings using BERTimbau, just like the Baseline System. Again, this process is explained in more detail in section 6.5.2. After generating these embeddings, it uses them as input for the Norms Model. The model will then make its predictions regarding the norms, returning all the predicted norms. These predicted norms will be added into the segments and fed into our feature extractor (BERTimbau) to generate another set of embeddings. For example, for the segment "1 - É revogada a lei ... no dia anterior" [1], if the Norms Model predicted the norm "LEFFECT" from token 2 ("É") to token 22 ("anterior"), we instead of feeding BERTimbau with the original segment, we feed it with "1 - ILEFFECT É revogada a lei ... no dia anterior FLEFFECT". In short, we

[1] This sentence translation for English is "1- The law is revoked ... on the previous day"

added ”lLabel” to represent the start of the norm and ”FLabel” to represent the end of the norm, with ”Label” being the label of the corresponding norm. After generating these embeddings, we feed them to two models responsible for predicting the named entities and the semantic roles, respectively. These two new models differ from the Named Entities and Semantic Roles Model of the baseline approach, since they are trained with more information (the norms) than those two, which are only trained with the original segments. Let us denote these two new models by NE with Norms Model, and SSR with Norms Model, respectively. Finally, like the Baseline system the predictions of each model are concatenated together in order to obtain the final classification.

As we mentioned before, the system uses the predicted norms, as input for the other two models. Regarding the training of the two models though, we did not feed the models with the predicted norms in the training and validation segments. Instead, we used the gold labels in order to have each model learn the relationships between the norms and other labels (named entities for the NE with Norms Model, and semantic roles for the SR with Norms Model) correctly. Since we did not want our model to learn wrong relationships. For example: Let us consider that we have a segment which has a “DEF”(norm) and also has “DEFINIENDUM”(semantic role). If we were training the SR with Norms Model, and if the Norms Model had predicted (assuming it was trained already) incorrectly that the segment had an “OBLIG” instead of a “DEF”, by using this incorrect label, the model could learn to associate ”OBLIG” with ”DEFINIENDUM”. So, to make sure it only learned the correct relationships we only used the gold labels. This also allows us to train all three models simultaneously, after generating the two sets of embeddings (one for the original segments, and another for the segments with the norms gold labels) .

6.3 Hybrid SNR System

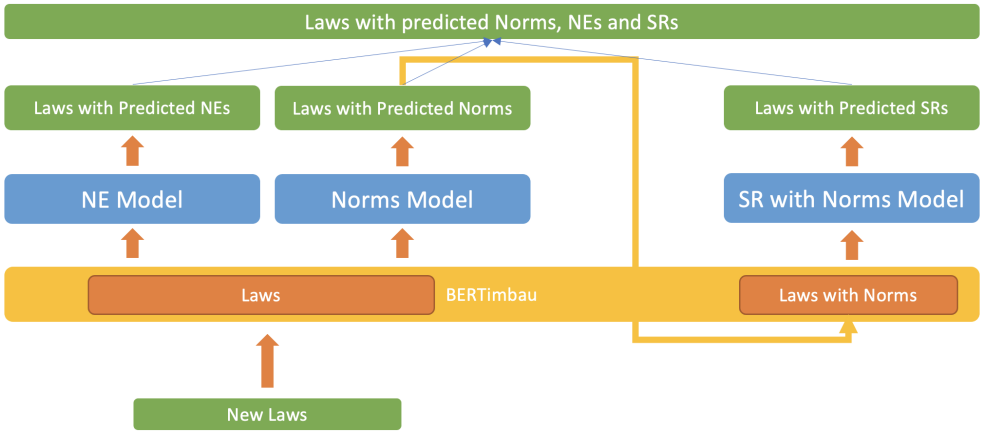

Figure 6.4: Hybrid SNR System

After doing some experimental training and evaluating with both of our previous systems, we saw that some named entities seem to have worse results when using the Two Phased system. After looking at each named entity and semantic role, we saw there was a stronger relationship between norms and semantic roles than there was between norms and entities. For example, in our corpus we can have a "LREF" or "TREF", which are named entities, in any type of norm, but we can only have a "DEFINIDEUM" inside a "DEF", or a "EFFECT" inside a "LEFFECT". Having this in mind, we decided to create a third version of our system, which we denoted as Hybrid, shown in Figure 6.4. A more detailed representation, including inputs and outputs of each model and the Hybrid system, for a given example, is shown in A.5.

As we can see, the Hybrid approach is very similar to the Two Phased. In the previous approach, we had the NE with Norms Model and SR with Norms Model responsible for predicting the corresponding labels, knowing the norms that were present in the segments. In this approach though, we only use the norms information for the model responsible for predicting the semantic roles (SR with Norms Model). Thus, we use the Named Entities Model, instead of the NE with Norms model to predict the named entities.

6.4 Model Training and Optimization

As we mentioned before, we created three different versions of our system. In total there are five different models: the Norms Model, the Named Entities Model, the Semantic Roles Model, the Named Entities with Norms Model, and finally the Semantic Roles with Norms Model.

To train the models we shuffled the dataset and applied a division of 80%/10%/10% to it. This means that 80% of the data is used to train the models, 10% of the data is used to validate the training and for parameter optimization, and the remaining 10% of the data is used to evaluate the models. The number of occurrences of each label in each set can be found in A.1.

6.4.1 Number of Epochs

Parameter	Value
BiLSTM size	200
BiLSTM layer	3
BiLSTM dropout	0.4
FFNN size	150
FFNN dropout	0.2
BERT size	1024
BERT layer	last 4
Embeddings dropout	0.5
Optimiser	Adam
learning rate	1e-3
Decay Rate	0.999

Figure 6.5: Model's Hyper Parameters [8]

In order to optimize our models and find the best parameters, we started by looking at the most appropriate number of maximum epochs for each model, by training the models using a batch size of 32 (which means 1 epoch corresponds to 140 iterations, since we have 4.480 segments in the train set) and the default parameters (the ones used by Yu, Bohnet, and Poesioo [8], which can be seen in Figure 6.5). For this thesis, the computational power at our disposal was limited to one machine of 2 $\times$ 26GB NVIDIA GeForce Version 460.91.03 GPUs. Due to the memory available of the machine at our disposal, we could not use a batch size bigger than 32. Also with the available computational power, training a single model during 1 single epoch, in one of the GPUs, takes about 2.6 minutes.

During training Yu, Bohnet, and Poesioo [8], fed the batches (created prior to training) to their model and shuffle them at each epoch. In contrast, we feed the train set to each model and at each epoch, we shuffle the data and only after do we create the batches. We did this since, machine learning models can learn complex relationships between the input and output, and these can include things such as always using the same order, structure or even data (static batches). For example considering an input $x = [2, 5, 1, 6, 8, 9]$, let us say we create batches of size 2, resulting in $b1 = [2, 5], b2 = [1, 6], b3 = [8, 9]$ and then train a model with them. In every single epoch, the model will be trying to find relationships by looking at the same batches that it has already seen in previous epochs. This might lead to the model finding patterns that we don't want it to learn, which could also lead the model to overfit. On the other hand, if we shuffle the data every epoch and then create the batches we could have $b1 = [2, 5], b2 = [1, 6], b3 = [8, 9]$ on the first epoch, $b1 = [1, 8], b2 = [5, 6], b3 = [2, 9]$, on the second epoch and so on. Thus, by shuffling the data at every epoch and then creating the batches we do not risk learning any relationships that might come from having those static batches.

During training, the authors [8] also used exponential decay learning rate, which we kept, since it is usually suggested to lower the learning rate as the training progresses, and by using an exponential decay the learning rate will decay exponentially over time but never reach zero. This exponential decay allows for training to have a higher learning rate, which is useful in the initial "exploration" phase of training, and a lower rate in later epochs, which allows a slow settling down to a local minimum.

For each model, during each epoch, we use the validation set to validate the training. We do this by training the model for a maximum number of epochs, and at each epoch, we evaluate the model with the validation set and save the epoch that has the highest f1 validation score. We choose to use the validation f1 score and not the validation loss, because since our models use softmax cross entropy to calculate the probability error between the true labels and the predicted probabilities, the loss does not represent the accuracy of the predictions but it shows how certain the model is in the predictions it makes. Also, we choose F1 Score and not Precision or Recall, since it combines both of these metrics. The training results of all models can be found in A.3.

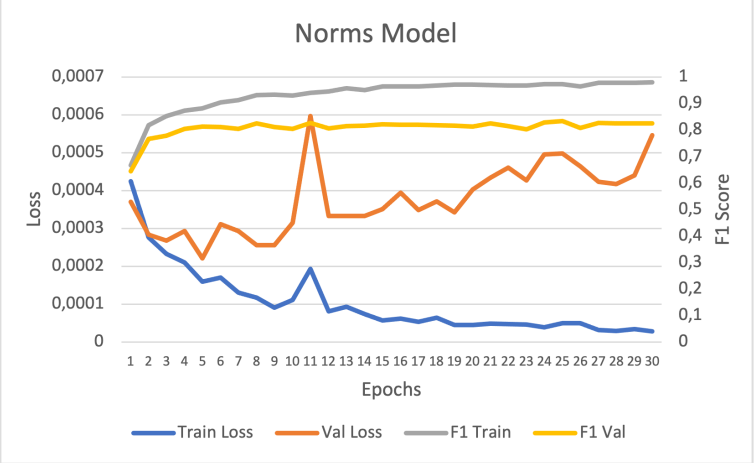

Figure 6.6: Norms Model using default parameters

In Figure 6.6, we can see the training results for the Norms Model at each epoch. we can see that from epoch 1 to epoch 11, the f1 validation score keeps increasing (it sometimes reduces but after 2 or 3 epochs it increases again). From epoch 11, it only increases again at 24, which is after a long time. So it seems that 15 epochs should be enough, since more than that might lead to overfitting, as we can see that from epoch 15 the model's validation f1 score seems to stabilize and while the validation loss (model's uncertainty) seems to be increasing and the training loss keeps decreasing.

Every time we train a model the training will always be different, since all the weights and biases are randomly initialized. To make sure that the behavior we saw previously is not a one time thing and actually represents the training of our model, we decided to repeat the training 9 more times (having a total of 10 times). We calculated the average for the loss and f1 score, of the 10 trainings, and the standard deviation as well. These results are shown in Figure 6.7.

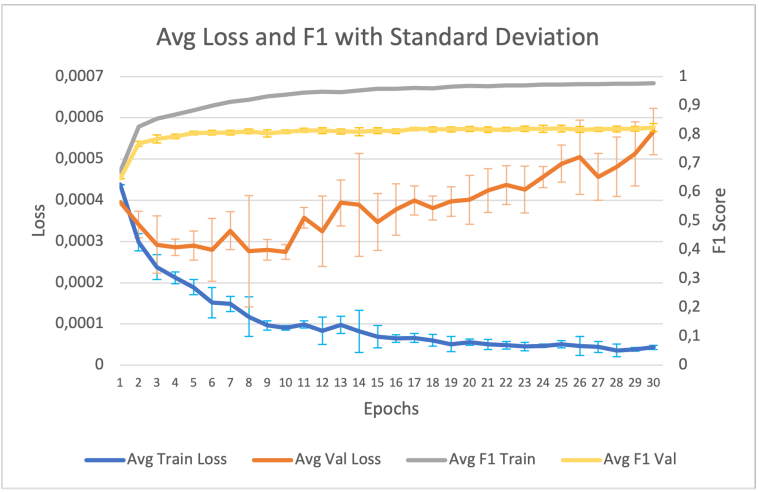

Figure 6.7: Average of training the Norms Model 10 times using default hyper parameters

As we can see, the same behavior we saw in Figure 6.6 is reflected on the average results as well. We can see that again from epoch 1 to 11, the avg f1 validation score keeps increasing and at epoch 11 is when it reaches the highest value. After that epoch, it seems to stabilize only increasing again much later. By looking at the standard deviation we can see that there were not any major deviations from this behavior. Thus, using a maximum of 15 epochs still seems to be the most appropriate choice.

Still, to make sure we made a robust decision, and that when optimizing our model and trying different parameters, 15 epochs would still be the most suited number for maximum epochs, we decided to train the model 9 more times again, but this time with different parameters each time. We only changed the parameters that we were going to focus on during optimization, which were: the learning rate, the decay rate, the FFNN size (units) and the BiLSTM size (units). After doing this, we calculated the average for the loss and f1 score, for the 10 different trainings, and the standard deviation as well. The results are shown in Figure 6.8.

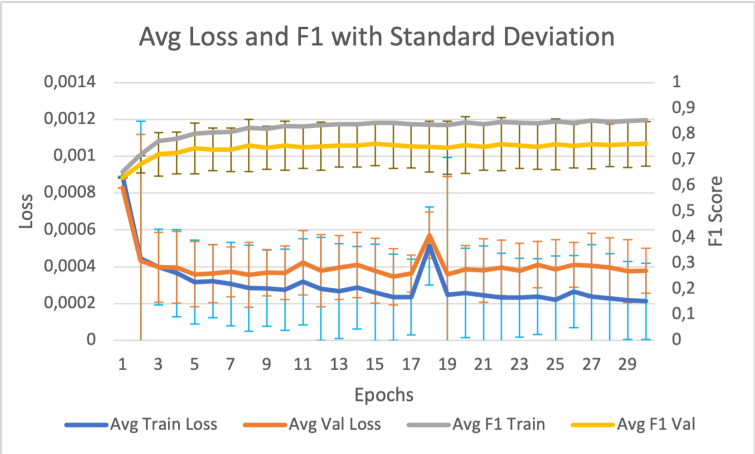

Figure 6.8: Average training results with standard deviation for the Norms Model

By looking at the average validation f1 score, we can see that the same behavior we saw before repeats itself once more. The average f1 validation score keeps increasing (sometimes reducing but soon after 1 or 2 epochs it increases again) until epoch 15 where it reaches the highest value. Only, after a long time, at epoch 30, does the average f1 validation score slightly gets higher than the one at epoch 15. Again it seems that after 15 epochs the model's validation f1 score seems to stabilize and while the average validation loss (uncertainty) seems to start slowly increasing, the average training loss keeps decreasing slowly. By looking at the standard deviation of these three metrics (average validation f1 score, average training and validation loss) we can see that there does not seem to be any major deviation from the average results. We looked into each training and after analyzing it just like we did with the training in Figure 6.6, we confirmed that in most trainings (from the total of 10) using a maximum of 15 epochs was enough.

Thus, we can see that 15 is the most appropriate choice for the maximum number of epochs for the Norms Model. Having also trained the NE and SR models as well, with the default parameters, we saw that the same logic that was applied to the Norms Model with the default parameters, could be applied for them too. If we look at the graphs of both these models, presented in A.3, we can see that in the NE Model it reaches its highest f1 validation score at epoch 11 as well (only increasing again much later at epoch 18), and in the SR Model it reached its highest at 14 (only increasing again much later at epoch 28). Since these three models only differ on the information (labels) they are trying to predict, and knowing that 15 epochs were enough for all three with the default parameters, we assumed that 15 epochs would be enough for those two models as well, without training them as many times as we did for the Norms Model.

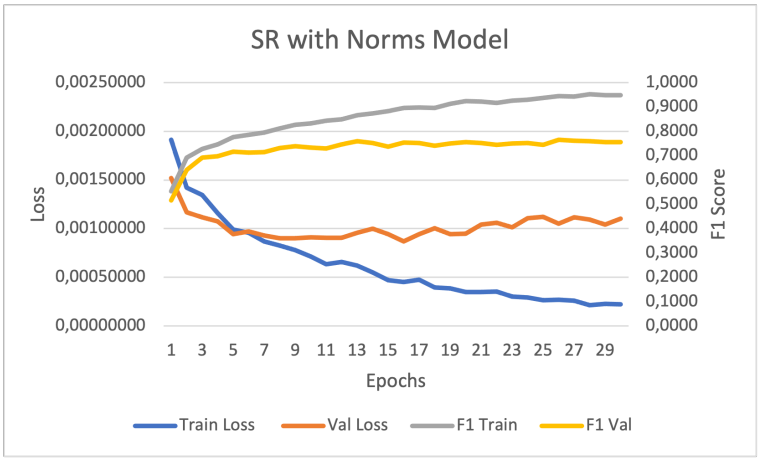

Figure 6.9: SR with Norms Model using default hyper parameters

Regarding the SR with Norms Model, even though after training the model with the default parameters it seemed that 15 was an appropriate number of maximum epochs, we decided to not assume that it behaved in the same manner as the previous models. By looking at Figure 6.9, we can see that during training the F1 validation Score keeps increasing, until epoch 13 where it reaches its highest score. Only at epoch 26 does it reach a higher value. Looking at the loss curves, while the training loss keeps decreasing, the validation loss from about epoch 16 starts increasing, which could mean overfitting. So even though 15 seems to be a good number to stop the training, since this model receives more information in the input (the segments have the norms information) it might need more time to learn. For this reason, we ran the SR with Norms Model 10 different times with the default parameters and 10 different times with different parameters, just like we did for the Norms Model.

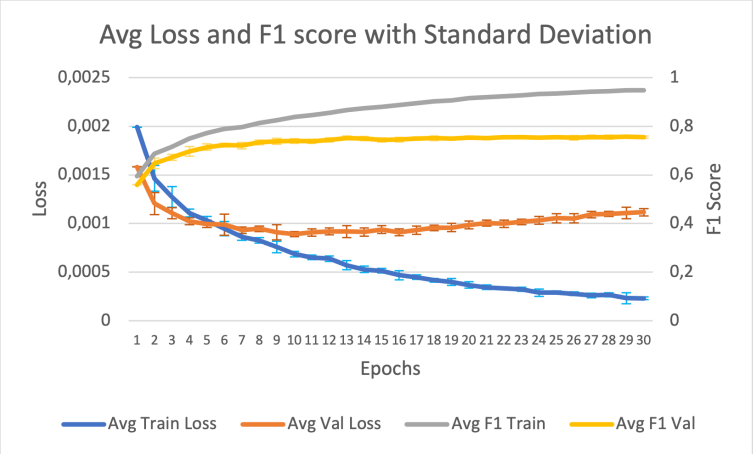

Figure 6.10: Average of training the SR with Norms Model 10 times using default hyper parameters

After running the model a total of 10 times, with the default parameters, we can see by looking at Figure 6.10 that there is no point of training the model for more than 15 epochs. Almost the exact behavior we saw before repeats itself, which makes sense considering the small deviation values that we see. Once more, at epoch 13 the average f1 validation score reaches its highest value. Only at epoch 20 does it reach a higher one, but from epoch 15 the model seems to overfit as the f1 validation score is almost constant while the validation loss starts increasing and the training loss keeps on decreasing.

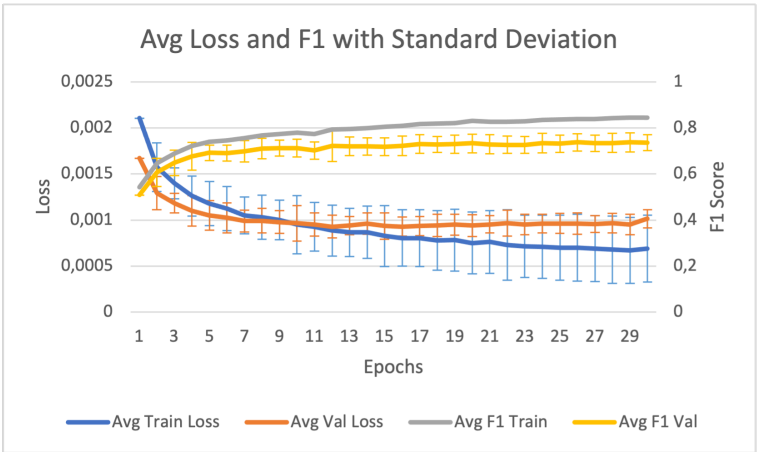

Figure 6.11: Average training results with standard deviation for the SR with Norms Model

Regarding the model's training when using different parameters, we can see in Figure 6.11 that the F1 validation score keeps increasing. Looking closely though, we see that at epoch 13 the increase is much more significant, and that from that epoch further, the increase is very small almost looking like the validation f1 score becomes constant. Also, while the training loss keeps decreasing, at epoch 15 we can see that the validation loss has stopped decreasing. By looking at the standard deviation, we can see that the 10 different trainings were very similar to each other, since there is not any big deviation

results. Thus, training the model for more than 15 epochs seems to be unnecessary.

Having all this in mind, the default behavior of the model and its behavior when using different parameters, we chose to use 15 as the maximum number of epochs for the SR with Norms Model. Seeing that, we also concluded that 15 epochs were enough to train the NE with Norms Model (after looking at Figure in A.3), we assumed 15 would be an appropriate number for this model as well.

6.4.2 Hyperparameters Optimization

Any machine learning model must go through hyperparameters optimization in order to identify the best combination of hyperparameters that minimizes a preset loss function generating better outcomes. A very common technique used in machine learning is Bayesian Optimization, which is an effective method for locating the extrema of expensive to evaluate objective functions [40].

6.4.2.A Bayesian Optimization

Bayesian Optimization guides the search with the purpose of finding the extrema(minimum or maximum) of an objective function by utilizing Bayes Theorem[2]. Let us recall Bayes Theorem, which defines how to calculate the conditional probability:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Since we are concern with optimizing a value, this definition can be simplified by removing the normalization value $P(B)$. Now, if we collect a set of samples $(x_1, ..., x_i)$ and evaluate each with the objective function, $f(x)$, we can then define our data D as $x_1, f(x_1), ..., xi, f(x_i)$. This allows us to define the posterior probability as:

$$P(f|x) = P(x|f)P(f)$$

which is an approximation of the objective function and can be used to estimate the cost of different samples that we may want to evaluate. With this surrogate function, we can direct our search when looking for more samples, by using the posterior within a function, known as acquisition function (function responsible for acquiring more samples), that uses the current beliefs to find the sample within the search space that is more likely to provide better results. Whenever new samples are collected and evaluated, the posterior probability is updated, and the process keeps repeating itself until the extrema of the objective function is found.

[2]https://machinelearningmastery.com/what-is-bayesian-optimization/

6.4.2.B Optimization Results

Having in mind all the different created versions of our SNR system (Baseline, Two Phased, Hybrid) and all the different models they include, we had to optimize each one. To apply the Bayesian Optimization we used the function gp_minimize[3], from the python library skopt. We applied it to find the minimum of our training function, which receives as input the set of parameters of our model, that we want to optimize. The training function trains a model, with the given parameters for a determined number of epochs, and returns the highest validation f1 score that it reached. Since we want to find the minimum of this function, once training is complete, instead of returning the actual validation f1 score we return its negative.

From all the parameters of our models (which can be seen in Figure 6.5), we decided to optimize only the learning rate, the decay rate, the FFNN size (units) and the BiLSTM size (units), since optimizing all parameters would be computationally costly and these were the ones that we believe to be the most important. We did not change the optimizer, since Adam [41] is known to be an effective gradient descent optimizer algorithm, well suited for problems that are large in terms of data, parameters and are known to be noisy. When it is said that a problem is noisy, this means that data may be corrupted or cannot be understood correctly by machines, such as in case of unstructured text. The concept of noise in our dataset is discussed later on Section 7.3.

We choose to find for each of the chosen parameters the best value included in the interval of values shown in Figure 6.12. For the learning rate, we initially had set the minimum value to 1^{-6}, but after doing optimization with that minimum value we saw that when using a learning rate smaller than 0.001 the models always achieved worse results, and it would take them a lot of epochs to learn. This makes sense since we are using a decay learning rate method, and so, if we start training with a very small learning rate, that learning rate will progressively become smaller and the model won't be able to learn as much. For the remaining parameters, the maximum and minimum values were chosen based on the default ones, making sure we gave the optimizer a good enough search window for each parameter.

		Minimum Value	Maximum Value
Parameters to optimize	Learning rate	0.001	0.1
	FFNN_Size	50	250
	BILSTM_Size	50	300
	Decay Rate	0.5	1

Figure 6.12: Interval of the values for the parameters to optimize

We decided to optimize our models by calling the training function 50 times. We choose 50 and not more, due to the optimization of each model taking a long time, and because we believed 50 different

[3] https://scikit-optimize.github.io/stable/modules/generated/skopt.gp_minimize.html

combinations to be enough for the optimizer to find the minimum. This can be seen by looking at the convergence trace plot of the optimizations. The plot shows the minimum negative validation f1 score that was found after doing n trainings with different parameters values. For example, in Figure 6.13, we can see that the minimum for the Semantic Roles model was found at iteration 34. In all models, the minimum was always found before 35 calls were made, as it can be seen in A.4.

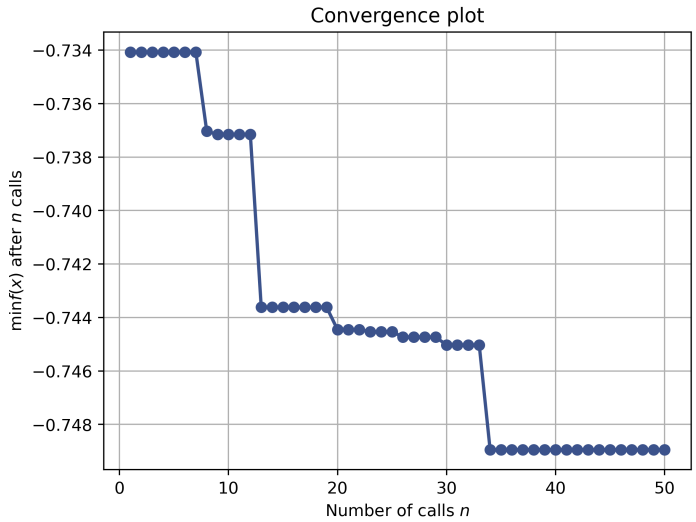

Figure 6.13: Convergence trace of the optimization for the Semantic Roles Model

After running the optimization for all five models, we obtained the following optimal parameters:

Model	LEARNING RATE	DECAY_RATE	FFNN_SIZE	BILSTM_SIZE	F1 SCORE
Norms	0.00677857167840111	0.8656666356174549	50	300	**0.83921569**
NE	0.0042441413312466125	0.8581906831865003	75	252	**0.8675535**
SR	0.008375732919899442	0.9346949324635897	152	218	**0.74894292**
NE with Norms	0.004531165342543305	0.8718741293739042	54	284	**0.86171429**
SR with Norms	0.0029915955553583219	0.8829473172470457	134	273	**0.77345775**

Figure 6.14: Optimal parameters for each model

As we can see in Figure 6.14, the Norms Model that seems to have resulted in the best performance, has 50 units for the FFNNs and 300 for the BiLSTM, which were the limits that we defined. To make sure there was not any other combination that was outside the search window we defined, and would result in a better performance, we decided to optimize once more the Norms Model, using [30,49] for the $FFNN\_SIZE$ and [301,500] for the $BILSTM\_SIZE$. After doing the optimization, which included 10 different combinations, we found a better parameter combination for the Norms Model, which is shown in Figure 6.15. All combinations and convergence trace graphs of each model can be found in A.4.

Model	LEARNING RATE	DECAY_RATE	FFNN_SIZE	BILSTM_SIZE	F1 SCORE
Norms	0.004168096898787611	0.9194413614582968	39	410	**0.84377358**
NE	0.0042441413312466125	0.8581906831865003	75	252	**0.8675535**
SR	0.008375732919899442	0.9346949324635897	152	218	**0.74894292**
NE with Norms	0.004531165342543305	0.8718741293739042	54	284	**0.86171429**
SR with Norms	0.002991595553583219	0.8829473172470457	134	273	**0.77345775**

Figure 6.15: Final Optimal parameters for each model

6.5 Implementation

In this section, we describe some more modifications that were done regarding the model of Yu, Bohnet, and Poesioo [8].

6.5.1 Input Transformation

In our system, each model has as input a JSONL[4] file, which is an abbreviation for JSON Lines, used for structured data, where each line corresponds to a JSON object (single segment). In Figure 6.16 you can see an example of one segment, the same of the previous example shown in figure 5.10.

{"id": "2762", "ners": [[2, 18, "RIGHT"]], "sentence": ["2", "-", "Quando", "convencionado", ",", "pode", "o", "segurador", "entregar", "a", "apólice", "ao", "tomador", "do", "seguro", "em", "suporte", "eletrónico", "duradouro", "."]}

Figure 6.16: Line of Norms model's Input

We implemented a python script, that is responsible for converting the prodigy annotation information to the input format of the models of our system (Figure 6.16). This script also removes the additional tokens "#", and then remaps the start and end tokens of each span after the removal. It also deals with situations where the start or end of spans corresponds to punctuation tokens. For example, if a span ends in the token x, which corresponds to ".", it will change the end token to x - 1. Looking at Figures 6.16 and 5.10, we can see this transformation by looking at how the text lost the "#" tokens, and how the start and end tokens indices, of the spans, were updated.

6.5.2 Extract Features

Instead of using the feature extractor created by Yu, Bohnet, and Poesioo[5], we decided to implement our own feature extractor, since the authors' version seemed to have a dependency problem regarding the Tensorflow Library. We decided to use the huggingface[6] library in order to apply BERTimbau (Portuguese BERT) to generate word embeddings for each segment of the dataset.

[4] https://jsonlines.org/
[5] https://github.com/juntaoy/biaffine-ner
[6] https://huggingface.co/neuralmind/bert-large-portuguese-cased

We started by, for each sentence, tokenizing each token in order to get the tokenized ids of the text. The ids of each token are saved in a list. In Figure 6.17, there is an example of this wordpiece tokenization and used list format: $[[t1\_1], [t2\_1], [t3\_1], ...[t512\_1, t512\_2], ...[t521\_1], [t522\_1]]$, where ti corresponds to token i of the sentence, and ti_j corresponds to word piece j of the token i. We can see that each sublist contains the ids of the word pieces of the corresponding token. For example, $[t512\_1, t512\_2]$ corresponds to the ids of the two word pieces that make up the token "esquecidos" of the example. The other tokens of the example, all have one single element in their corresponding sublist, since they are not divided into word pieces.

Figure 6.17: Example of tokenized ids for a single sentence

This list format was used, so that, when feeding the (word piece) ids of a sentence to the model, we used less or equal to 512 ids at a time, since this is the maximum number of input tokens that BERTimbau is able to process. In the case of the previous example, we would feed the model After generating the embeddings, we chose to only use the last four layers and represent each token by averaging its word pieces. The reason behind choosing the last 4 layers is explained in Section 3.3.4. As to why we decided to average the word pieces of each token, this was in order to generate an approximate compressed representation of the word.

7

SNR System Evaluation

Contents

7.1 Evaluation of the SNR approaches

This section describes the evaluation metrics used and the results of all the approaches we implemented.

7.1.1 Evaluation Metrics

After finding our optimal parameters, we evaluated all three approaches (Baseline, Two Phased and Hybrid) by calculating the F1 Score, Precision, and Recall. These metrics are commonly used for binary classification and are defined in A.7.

To compute these metrics for our multi-classification task we choose to do micro-averaging since even though our classes are imbalanced, there is no priority between them, and what interests us is maximizing our classifier numbers of hits (number of correct predictions the classifier makes) and minimizing its misses. Using micro-averaging, we calculated our system precision, recall, and f1 score, by counting the total true positives, false negatives, and false positives of all classes:

$$Micro - Precision = \frac{\sum_c TP_c}{\sum_c TP_c + \sum_c FP_c}$$

$$Micro - Recall = \frac{\sum_c TP_c}{\sum_c TP_c + \sum_c FN_c}$$

$$Micro - F1 = \frac{2 * Micro - Precision * Micro - Recall}{Micro - Precision + Micro - Recall}$$

Not only does our system calculate the Micro-F1, Micro-Precision, and Micro-Recall, it also calculates the F1, Precision, and Recall for each label.

A slight problem with these metrics is that they only count exact matches as correct. For example, if we have the gold label [2,22,“OBLIG”] (with 2 being the start token of the span and 22 the end token), and the model predicts [2,23,“OBLIG”], the model will not consider this as a correct prediction, since the start and end token are not the same. In short, these metrics don’t account for predictions which only differ from the gold labels by a certain number of tokens. For that reason, we created a metric, which we denoted by “Average Token Agreement (ATA)”, which we applied to the overall model and each label(“ATA by label”) as well.

The ATA can be calculated by doing the following:

$$agreement(t) = \frac{|goldLabels(t) \cap predLabels(t)|}{|goldLabels(t) \cup predLabels(t)|}$$

$$segmentAgreement(s) = \frac{\sum_t agreement(t)}{s_{size}}$$

$$ATA(set) = \frac{\sum_s segmentAgreement(set)}{set_{size}}, \tag{7.1}$$

where set corresponds to a set of segments, s is a segment of the set, set_size corresponds to the number of segments in the set, set_s corresponds to the number of tokens in the segment, t corresponds to a token of the segment s, and $goldLabels(t)$ corresponds to the gold labels and $predLabels$ to the predicted labels, of the token t.

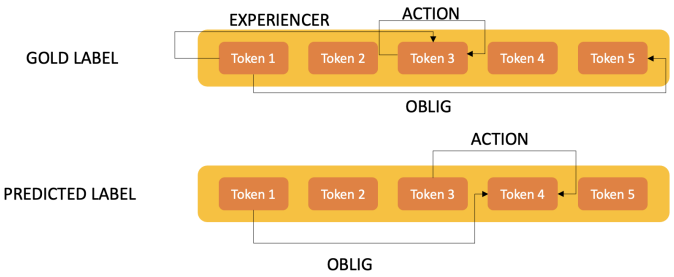

Figure 7.1: Example for agreement metric

Figure 7.1 shows a segment and its corresponding gold and predicted labels. Each arrow represents a span and its legend corresponds to the name of its predicted/gold label. The ATA of the system for that single segment, is (1/2 + 1/2 + 2/3 + 1/2 + 0/1)/5 = 0.43. For example, in token 1 the gold and predicted labels agree in 1 label("OBLIG") from 2("OBLIG" and "EXPERIENCER"). In this case, 0.43 means that the system overall and the gold labels agree with each other 43%, regarding that sentence. For x sentences, the agreement would be the sum of the agreement of each sentence divided by the number of sentences, as we can see in the equation 7.1.

If we consider a specific model, and not the full system, for example, the SR Model, the agreement for that sentence would be (0/1 + 0/1 + 1/2 + 0/1 + 1)/5 = 0.3, which means the gold and predicted labels have an agreement of 12.5%, regarding the semantic roles, which makes sense, since from 5 tokens in total, only in token 3 and in token 5 do the predicted and gold labels match(in case of token 5 only in 1 of 2 of its semantic roles). Also as we can see tokens that have no labels (in this case the token 5 has no semantic role) will not be considered.

This metric still does not show us if there were certain labels that were mostly corrected and were only missing a few tokens. For that reason, we also created an agreement by label. For a label x, the ATA of that label corresponds to the percentage of agreement between the gold and predicted labels. For example, by looking at Figure 7.1, the agreement of "OBLIG" for that segment would be (1+1+1+1+0)/5 = 0.8, since from the 5 tokens, there are 4 that were predicted correctly to be an "OBLIG". So, 80% of the tokens were correctly classified as an "OBLIG". The agreement for "EXPERIENCER" would be 0, since no tokens are predicted to have the label. For "ACTION" it would be (1+0)/2 = 0.5, since from the

2 tokens that are associated with the label, only token 3 has “ACTION” as a gold and predicted label. The total agreement for a label x (for multiple segments), corresponds to the average of the agreement of each segment. For the agreement by labels, we decided not to consider the tokens that did not have the label either in the gold and predicted labels. We did this since for large sentences that only had a few tokens marked either as gold labels or predicted labels, the agreement would always be a higher value. For example, if we consider all tokens when calculating the agreement for “EXPERIENCER” it would be (0 + 0 + 0 + 1 + 1)/5=0.4, when in reality no token was predicted to be an “EXPERIENCER”.

In spite of ATA being a good metric, regarding how much the predictions and gold labels agree with each other in terms of tokens, it has some limitations compared with the F1 Score. These limitations come from the fact that ATA is not a balanced reflection of precision and recall, it does not take into consideration true positives, false positives, or false negatives. It only sees exact matches (gold labels equal to predicted labels) for a specific token. So for example, let us consider a segment that has the span [3,5,“DEF-INCLUSION”], where 3 represents the token index of the start of the span, and 5 represents the index of the end of the span. If our model instead of predicting one single “DEF-INCLUSION”, predicted [3,3,“DEF-INCLUSION”] and [5,5,“DEF-INCLUSION”], the ATA would be high, since from 3 tokens, 2 match their gold labels with predicted labels. On the other hand, the f1 score would be zero, since there is no TP. This is the reason why the ATA measure was not used to train/validate the network, since we did not want our model to learn to label tokens correctly, but predict the wrong spans. This measure was created to give some more insight to how the predictions were being made at the token level, in order to see if the system was learning the correct start or/and end limits of the spans and if there was inconsistency in the delimitation of spans during annotation. In case of the latter, this would correspond to a label having a very small f1 score but a high ATA for the corresponding label.

7.1.2 Baseline VS Two Phased VS Hybrid

Before analyzing our system’s final results, we need to compare all approaches with each other in order to decide which one performs the best for our dataset. Looking back at Figures 6.1, 6.3 and 6.4 we know that these three approaches use in total five different models: Norms Model, NE Model, SR Model, NE with Norms Model and SR with Norms Model. The Norms model is common on all approaches, so the difference in performance among the different SNR approaches comes from using the NE Model instead of the NE with Norms Model, and using the SR Model instead of the SR with Norms Model. In Figure 7.2 we can see the global performance of each model followed by the global performance of each SNR system approach.

Model	F1 Score	Recall	Precision	ATA
Norms	84.74%	83.01%	86.54%	0.9132516737021712
NE	88.67%	90.45%	86.96%	0.9756663032866839
SR	76.06%	73.12%	79.25%	0.8070073252578439
NE with Norms	86.94%	89.50%	84.52%	0.9780493232839289
SR with Norms	76.86%	75.08%	78.72%	0.8329559068560555

SNR System	F1 Score	Recall	Precision	ATA
Baseline SNR	81.05%	79.41%	82.75%	0.8509155186007729
Two Phased SNR	81.00%	80.26%	81.75%	0.8430064632297725
Hybrid SNR	81.44%	80.50%	82.40%	0.845501888436459

Figure 7.2: Performance of each model and each SNR System approach

Immediately by looking at the results, we can see that all approaches have very similar results. Considering only the models, it seems that the NE Model outperforms the NE with Norms Model in all metrics (F1 Score, Precision, and Recall) except the Agreement, which is very close. We were expecting the NE Model to outperform the NE with Norms Model, since we know that the named entities the model was trying to predict do not seem to be dependent on the norms. Thus, from the results, we conclude that the Two Phased approach is not the best choice since using the NE with Norms Model instead of just the NE Model leads to worse results. The SR with Norms Model has a higher F1 Score than the SR Model. This comes from the fact that, even though it has a smaller precision than the SR Model, the SR with Norms Model has a higher recall (being a bigger difference when compared with the increase in precision) than the SR Model. This is reflected in the results when we compare the Baseline Approach with the Hybrid approach and see that the Hybrid approach has a higher F1 score and recall than the Hybrid approach. Based on these results and since our priority is the model making the highest number of corrected span predictions (F1 score), we concluded that the best approach is the Hybrid approach. Having in mind the strong relationships that exist between some SR and some of the norms (such as between “DEF” and “DEFINIDEUM”), we analyzed the results and discovered limitations of using Hybrid approach that could be causing the system to have a worse precision than the Baseline Model.

First of all, the difference between the Baseline approach and the other two is that it does not have any model that is dependent of the results of another. For example, in the hybrid approach, the SR with Norms Model is dependent on the predictions the Norms Model did. The consequence of this is that when the Norms Model makes wrong predictions, it can lead the SR with Norms model to associate the wrong information with the one it was provided. By looking at Figure 7.3, we can see the confusion matrix for the Norms model (the confusion matrix for the Ne Model and the SR with Norms Model can be found in Section A.5). We can see that the model incorrectly predicted 11 “DEFs” to be “OBLIGs”. This means that for those segments, the SR with Norms received the segments with the start and end tokens of the predicted “OBLIGs”, which could have caused the model to associate labels such as “ACTION” and “THEME”, instead of “DEFINIENDUM” or “DEFINIENS”, leading to wrong predictions.

		PREDICTED LABELS					
		NoLabel	DEF	OBLIG	RIGHT	LEFFECT	INTRO
GOLD LABELS	NoLabel	0	7	20	8	0	0
	DEF	8	30	10	0	0	0
	OBLIG	36	2	376	16	0	0
	RIGHT	16	0	14	79	0	0
	LEFFECT	0	0	0	0	19	0
	INTRO	0	0	1	1	0	4

Figure 7.3: Confusion Matrix for Norms Model

From this first scenario, comes our second hypothesis. As we have already mentioned, we trained and validated the models that receive the norms information (SR with Norms Model, NE with Norms Model) with the gold labels. This means that these models were trained with no errors from the Norm recognition model, and thus they might have learned to blindly trust the information coming from the Norms Model. Thus, when later on, during the testing phase (and the normal functioning of the SNR system) there is an error with the predictions of the Norms Model, the models that use that information will also very likely fail. This, unfortunately, might have caused those models to not be robust to deal with the errors that the Norms Model makes. Our decision to make sure it learned to associate the labels with the corrected norms information, might have caused the SR with Norms Model, and the NE with Norms Model, to not be able to deal with situations where they are given the wrong norms information.

7.2 Hybrid SNR System

Now that we know the hybrid to be the best approach we need to evaluate our system based on its results regarding each label that it is trying to predict. Figures 7.4, 7.5 and 7.6, show the results of each model of the Hybrid SNR system.

Norms Model				
Labels	**F1 Score**	**Recall**	**Precision**	**ATA**
DEF	68.97%	62.50%	76.92%	0.6392479782737972
OBLIG	88.37%	87.44%	89.31%	0.8774654069151692
RIGHT	74.18%	72.48%	75.96%	0.6692444930394013
LEFFECT	100.00%	100.00%	100.00%	1.0
INTRO	80.00%	66.67%	100.00%	0.6666666666666666

Figure 7.4: Hybrid SNR System: Norms Model Results

From looking at Figure 7.4, the Norms Model seems to have good results for almost every norm (74% to 100% f1 score), except for “DEF”, which has satisfactory results (68.97% f1 score). “LEFFECT” was the norm that had the best results having a 100% f1 score, precision and recall, as well as an ATA of 1. This type of norm is probably the one that we found to be more consistently annotated since it was probably the most simpler concept to identify. “INTRO” was also another norm that was less complex compared with the remaining ones. Still from the 6 spans present in the test set, the model incorrectly classified one as an “OBLIG” and another as a “RIGHT”, as shown in Figure 7.3. “DEF” on the other hand, was a norm that caused some ambiguity, especially in some cases where segments were too long and included many definitions which might have caused indecision of where to start and end each definition, which in turn could explain why the ATA for that norms is so low. After looking at the predictions, we saw that the Norms Model predicted 30 definitions correctly, incorrectly predicted 10 definitions as “OBLIGs” and it failed to predict the other 8 definitions (were marked as ”NoLabel”). Similarly, the “RIGHT” spans were either predicted correctly or marked as a “OBLIG” or not predicted at all (“NoLabel”). The model seems to sometimes confuse definitions with obligations, and rights with obligations as well, which is also seen in its predictions regarding the label “OBLIG”. From the spans that corresponded to “OBLIG”, two were predicted as “DEF” and 16 as “RIGHT”.

NE Model				
Labels	**F1 Score**	**Recall**	**Precision**	**ATA**
LREF	90.77%	96.72%	85.51%	0.8336994229502419
TREF	91.76%	93.62%	89.98%	0.9212101919737369
NE_ADM	89.37%	90.61%	88.17%	0.8184070849563373
TIME_DATE_REL_TEXT	62.63%	58.49%	67.39%	0.517487639698593
TIME_DURATION	62.22%	63.64%	60.87%	0.43428571428571433

Figure 7.5: Hybrid SNR System: NE Model Results

The NE Model, has great results for every named entity (80% to 92% f1 score) except “TIME_DURATION” and “TIME_DATE_REL_TEXT”, which had satisfactory results. Based on the NE Model predictions we saw that sometimes the model confused these two labels, it could be that since both are temporal expressions the model had some difficulty identifying each one separately.

For the SR with Norms Model, only “DEFINIENS”, “ACTION”, and “PURPOSE” had satisfactory results, the remaining semantic roles all had pretty good results. “ACTION” was the semantic role most common in our dataset, which would make us think it should have better results, since it had more samples to train on. Yet, we can see that it had a good ATA score (0.76), which may indicate that some of the incorrect predictions were only off a few tokens. From all the spans that corresponded to an “ACTION”, 182 were predicted to be ”NoLabel” by the system, also the model predicted incorrectly 132 spans as “ACTION”. This could contain cases where the predicted span was incorrect by only lacking certain tokens. Regarding the labels “DEFINIENS” and “PURPOSE” incorrect predictions, the

model mostly seemed to either incorrectly predict the labels when it was not the case (false positive), and incorrectly predict a span to be ”NoLabel” when it actually was a “DEFINIENS”/“PURPOSE” (false negative).

SR with Norms Model				
Labels	**F1 Score**	**Recall**	**Precision**	**ATA**
DEFINIENDUM	77.27%	66.67%	91.89%	0.6445578231292517
DEFINIENS	64.71%	56.41%	75.86%	0.46234501209034323
DEF-INCLUSION	83.24%	81.91%	84.62%	0.6080430039393254
SCOPE	72.96%	69.05%	77.33%	0.6321754550670213
ACTION	69.35%	65.60%	73.56%	0.7578923887633061
CONDITION	81.45%	80.72%	82.19%	0.7842337139125956
CONCESSION	92.00%	93.88%	90.20%	0.8874094202898551
PURPOSE	65.85%	69.23%	62.79%	0.5140650854936569
EXPERIENCER	81.28%	75.34%	88.24%	0.782821082347928
THEME	79.95%	83.02%	77.09%	0.8031079299351725
EXCEPTION	80.00%	82.76%	77.42%	0.7538132440476191
EFFECT	82.54%	86.67%	78.79%	0.8650793650793651
NEG	86.36%	95.00%	79.17%	0.76

Figure 7.6: Hybrid SNR System: SR with Norms Model Results

These mistakes the models made (such as confusing certain norms with each other as well as confusing the two temporal expressions) might come from noise still in the dataset, as we had already found some situations like this, which we talk about in more detail in the following section.

Hybrid System Performance			
F1 Score	**Recall**	**Precision**	**ATA**
81.44%	80.50%	82.40%	0.845501888436459

Model	**F1 Score**	**Recall**	**Precision**	**ATA**
Norms	84.74%	83.01%	86.54%	0.9132516737021712
NE	88.67%	90.45%	86.96%	0.9756663032866839
SR with Norms	76.86%	75.08%	78.72%	0.8329559068560555

Figure 7.7: Hybrid SNR System Results

Overall, our Hybrid SNR system performed pretty well. The NE Model achieved the best results, with a 88.67% f1 score, followed by the Norms Model, with a 84.74% f1 score, and finally the SR with Norms Model, with a 76.86% f1 score. Thus, resulting in a 81.44% f1 score for the system, as we can see in Figure 7.7.

7.3 Noise

Noise is a very important concept when dealing with machine learning problems. In classification having noisy labels, which correspond to samples that were labeled incorrectly (in our case by the annotators) can cause a model to either learn the incorrect information (even if its predictions are correct according to the annotations that were done), and have inconsistency, which leads to wrong predictions.

An annotation task is a very complex task, especially when dealing with text. NL can be very complex and ambiguous, making its annotation prone to error. For this reason, it is important to look at our dataset to make sure it is clean.

In this section, we will talk about how we found noise present in the dataset throughout the development of our system and how we corrected it, and finally how we checked the presence of noise after the development of our system and the impact it had on the system results.

7.3.1 Noisy Labels

As we mentioned previously, noisy labels correspond to spans that were labeled incorrectly during the annotation process. In the dataset, if there are expressions that are incorrectly missing a label, these are also considered noisy labels. Since the models in our systems consider ”NoLabel” as a label as well.

During the system implementation, when using the Baseline system, with the default parameters, we saw that there were labels that had much worse results when compared to others. We decided to check the models' results and based on those, check in the dataset for inconsistencies and noise that could be causing such results. We found many situations where there were indeed noisy labels. Due to this, the segments that had annotations errors had to be corrected. This correction step was crucial since we did not want our model to be learning the noise present in the dataset and we also feared that the noise was impacting some of our results. The performance of the Baseline System using the unrevised dataset(the version before any corrections were done) can be found in A.6. The number of occurrences of each label in the unrevised dataset can also be found in A.1.

For example, regarding the label “DEF-INCLUSION”, we saw that in the test set there were two segments that had 7 spans marked as “DEF-INCLUSION”. The baseline system (more precisely the Semantic Roles model) resulted in 0% in F1 score, Precision, and Recall for that label. We saw that those segments, whose spans were incorrectly classified, corresponded to “DEFs”, where for some “DEFINIENDUM”(the object being defined), there were multiple definitions (“DEF-INCLUSION”). After acknowledging the context of those segments we decided to look into the dataset to find other similar segments and compare the annotations. More than 80 segments were found where instead of being marked “DEF-INLCUSION”, the label “DEFINIENS” was used incorrectly. This lead us to discuss with the main annotator and filter these types of segments, that were wrongly annotated, so that they could

be corrected. In the end, our dataset went from having 119 spans marked as “DEF-INLCUSION” and 798 marked as “DEFINIENS”, to 576 marked as“DEF-INLCUSION” and 431 marked as “DEFINIENS”.

We also found certain expressions that in most cases should have been tagged with a specific label but were not, which created more noisy labels in our dataset. For example, “desde que” (which means “as long as”), is an expression that is usually associated with a conditional value. In the test set, there were 6 segments that did not have that expression marked as a “CONDITION”. There were also another 6 that had the expression marked as “CONDITION”. We saw that most times the system would say that the expression was not a “CONDITION”. We believed that the existence of inconsistency and noisy labels was causing the Semantic Roles Model to fail at predicting these conditions. We checked the whole dataset and saw that from 146 segments that had the expression, only about 67 segments had the expression marked as a “CONDITION”, which means only about 50% of those cases were labeled correctly. The segments which had the expression and did not have the label “CONDITION” were reviewed by the main annotator. From the correction that was done, the dataset went from having 1517 spans marked with “CONDITION” to 2311.

Other examples include the following expressions: (i)“sempre que” (which means “anytime that”), having only 76 segments with the expression tagged as a “CONDITION”, from 173 segments. Again, only about 50% of those cases were labeled correctly; (ii)“para o(s) efeitos ” (which means “to the effect(s) of”), which when inside a “DEF” should have been tagged as a “SCOPE”, otherwise a “PURPOSE”. From 68 segments with the expression in a “DEF”, 5 weren’t tagged as “SCOPE” but with “PURPOSE” instead, and from 82 segments with the expression not inside a “DEF”, 18 were not tagged with “PURPOSE” but with “SCOPE” instead. Also, there were 117 segments where the expression was not tagged with either label. So from a total of 150 segments with the expressions there was 140 segments that were incorrectly annotated, which is more than 90% ; (iii) “duração” (which means “duration”), for which we found 9 segments not tagged with “TIME_DURATION”. Not all segments with the expression are supposed to be marked with “TIME_DURATION”, but considering that about 30 segments had the expression tagged with the label, if we consider that those 39 were all the segments in which the context asked for a “TIME_DURATION”, then about 23% of those cases were not correctly annotated.

We also found the existence of ambiguity between the labels “SCOPE” and “CONDITION”. The label “SCOPE” can be used to represent the context in which an action is applied, and since for certain segments, it could be interpreted as the conditional context that made an action possible, some segments were incorrectly annotated. When checking the model predictions, we saw that the Semantic Roles model sometimes predicted a span to be a “SCOPE” when the gold label was “CONDITION”, and in most of these situations the label was noisy. For example, for the segment 3018, which corresponds to: “No contrato de seguro celebrado com um período de vigência inicial inferior a cinco anos e prorrogação automática , a liberdade de denúncia não é afectada pelas limitações indicadas no artigo

seguinte.", the span "No contrato de seguro celebrado com um período de vigência inicial inferior a cinco anos e prorrogação automática", which represents the context, was initially marked as a "CONDITION", when in reality it corresponds to a "SCOPE".

The examples mentioned above were just a few of the ones we found. By comparing the values in A.1 and A.2, we can see how much more information was added into the dataset (from a total of 34178 marked spans to 36711) and the changes that were done to each label. By comparing Figures 7.4, 7.5 and 7.6 with Figure A.19, we can see the difference regarding the performance of our optimal Hybrid SNR System, when using the unrevised dataset (before doing the previously described corrections) versus the our final dataset (revised).

The process of annotating a corpus can be quite complex and having in mind the type of corpus, the number of people that annotated the corpus, and the complexity of the information that was intended to be captured, we can see that this task is neither simple nor direct. Even though we kept correcting the noise we came across, throughout the development of this project, the dataset still contains some noise, which is shown in the following section.

7.3.2 Current Noise and Hybrid SNR System Results

In the previous section, we mentioned how we found noisy labels during the system implementation, which we then corrected. Still, we did not review the whole dataset since that would take a lot of time. After having finished developing our system, we decided to check if there was still noise in the dataset. We decided to collect randomly, for each label, 30 segments and ask the main annotator to verify how many errors were present. This was done, in order to estimate the presence of noise concerning a specific label, so that when evaluating our system we have that information in mind. Of course, none of the following percentage values are an accurate representation of the real amount of noise in the dataset. Only 30 segments were chosen for each label and they were all chosen randomly, which means that it could happen that all 30 were segments that were correctly annotated or that all 30 segments were incorrectly annotated.

Nevertheless, we started by doing this sampling for the Norms labels, whose results can be seen in Figure 7.8. For this group of labels, since a segment always has some norm, and it is relatively simple to see the norm that should be used instead, while doing this verification, whenever the main annotator found a noisy label, not only did he marked the noisy label but also pointed out the real label that should have been used instead.

		Correct Label					
		OBLIG	DEF	RIGHT	INTRO	LEFFECT	ERROR
Annotation	OBLIG	22	2	2	0	0	13%
	DEF	0	30	0	0	0	0%
	RIGHT	0	0	30	0	0	0%
	INTRO	2	0	0	28	0	7%
	LEFFECT	0	0	0	0	30	0%

Figure 7.8: Errors found for Norms Labels

If we compare the errors present in the norms and the predictions the Norms Model made (we had already mentioned in Section 7.2), we can find many relationships. For example, looking at the confusion matrix of the Norms Model predictions, shown in Figure 7.3, we can see that it predicted one "INTRO" as a "OBLIG". In our sample of 30 segments, for the label "INTRO", we found an error of 7% related with the label "OBLIG". This could mean that the incorrect predictions the Norms Model made came from noise in the dataset. For the label "OBLIG" we found an error of 13%, from which half comes from the label "DEF". This also reflects the incorrect predictions the model made, when from the 48 spans that corresponded to "DEFs", 10 were predicted to be "OBLIGs" (about 2%).

Regarding the named entities, the two named entities ("TIME_DURATION" and "TIME_DATE_REL_TEXT") for which the NE Model had worse results, were also the two named entities for which we found a higher percentage of error.

		ERROR
NE	LREF	3%
	TREF	0%
	NE_ADM	8%
	TIME_DURATION	12%
	TIME_DATE_REL_TEXT	9%

		ERROR
SR	DEF-INCLUSION	2%
	ACTION	11%
	CONDITION	3%
	CONCESSION	3%
	THEME	16%

Figure 7.9: Errors found for NE and SR Labels

Finally, for the semantic roles, we did not find any errors associated with the labels "DEFINIENS" and "PURPOSE", still that does not mean it does not exist with certainty. We did found errors in other labels for which our model performed well, which could mean that if there was not any noise the SR with Norms Model could perform even better regarding those labels, or that it simply learned to predict the wrong information as well. The last situation is a possible scenario in cases like this, when we are giving a machine learning model information to learn, except that information is incorrect. It can still learn the information and have a good performance, but what is it learning is not what is pretended. Thus, the annotation process is highly important and needs to be taken seriously.

7.4 Downstream Task: Information Retrieval

At the start of this thesis, when we described our problem, we mentioned how we hypothesized that the extraction of these relevant concepts could help downstream tasks. For this reason, we decided to see if our system improved the performance of the retrieval information system denoted by Legal Semantic Search Engine [42], that was being implemented for the legal texts of the Portuguese Consumer Law. The system [42] works by returning the 100 legal acts that are deemed relevant for a given query and is evaluated by calculating the accuracy of the top x results.

We used a new set of legal texts that was made available by INCM and were later segmented. Due to this set having different structured texts, the segmentation was done differently, sometimes by points, other times by paragraph (whenever there are no numbers, or points), among others. After using the Hybrid SNR system to predict the relevant concepts for the corresponding set, we used the system predictions to generate a set of questions and answers, using simple rules that were given by the main annotator. These represented the main rules that were found to apply to our corpus. For example, if a "OBLIG" had the expression "must", the concept "EXPERIENCER", and "ACTION", but not "THEME", the question and answer generated would be "Q: What must <EXPERIENCER> do? A:<ACTION> ". If the 'OBLIG" also had a "CONDITION" or a "SCOPE", then the QA generated would be "Q: What must <EXPERIENCER> do <CONDITION or SCOPE>? A:<ACTION> ".

This way, we could use our set of QA to fine-tune the information retrieval system. The system, actually only uses the questions and the segments themselves to create pairs question-segment to fine-tune the system. We created question and answer pairs, so that we could use them for a QA system as well. Still, using the question-segment pairs it allows us to make conclusions about whether or not using our system predictions aids the retrieval system. The following figure contains all the results:

Accuracy	No fine-tuning	Fine-tuning (ICT)	Fine-tuning(Pred Labels)
TOP 1	44.0% (44/100)	45.0% (45/100)	51.0% (51/100)
TOP 3	74.0% (74/100)	76.0% (76/100)	77.0% (77/100)
TOP 5	88.0% (88/100)	84.0% (84/100)	90.0% (90/100)
TOP 12	95.0% (95/100)	95.0% (95/100)	95.0% (95/100)

Figure 7.10: Accuracy Results for the information retrieval system: Combine Search

Figures **??** and 7.10 show the accuracy results for the top 1, 3, 5, and 12, first returned legal act.

As we can see we have the results from not doing fine-tuning, and from doing fine-tuning using Inverse Close Task (ICT) and using the question-segment pairs generated from the predicted labels. ICT [43] was the method used to construct the corpus, which the model was trained with. This method basically consists of the division of a segment into parts, and the creation of pairs between each part and the segment itself, to represent question and answer pairs.

Now if we compare the results, we can clearly see that not only did the retrieval system's performance when fine-tuning with our predictions, outperformed the system when no fine-tuning was done, but also it outperformed when fine-tuning using the ICT method. Thus, this shows that the information that our system captures does indeed help generate relevant questions and answers that improve the overall performance of an information retrieval system.

8

Conclusion

Contents

Understanding our duties, obligations and other legal norms is crucial for all citizens. Currently, citizens have online access to the legal texts of Portuguese Consumer Law, but publishing these laws online does not make them truly accessible, since these types of texts include terminology that is not easily understood by regular citizens. For this reason, we created an automatic information extraction system that we denoted by SNR[1]. This system is responsible for extracting all relevant semantic concepts present in these types of texts.

The SNR system was trained using a dataset created from applying an annotation process to the articles of the Portuguese Consumer Law. The annotation process was done in two levels, first identifying the norms(high level) and then the remaining concepts(low level).

The SNR system is composed of three nested NER models, each one responsible for predicting a certain group of labels (norms, semantic roles, and named entities). During the implementation of our system, three different approaches were created: Baseline, Two Phased, and Hybrid. These approaches differ in regards to the existent levels of learning: the baseline being the most simple one, where each one of the three modes simply learns how to predict their corresponding labels; the Two Phased, where the models responsible for predicting the named entities and semantic roles, make their predictions knowing the norms that are present in the laws, by adding this information into the segments; and the Hybrid, where only the model responsible for predicting the semantic roles knows the norms information.

After training and optimizing all the models of our different approaches, we saw that the Hybrid approach had a better performance than the remaining approaches. Thus, we choose the Hybrid SNR system as our solution and evaluated its performance. After evaluating our hybrid SNR system, we saw that all its models achieved a good performance, achieving a 84.74% for the Norms Model, 88.67% for the NE Model, and 76.86% for the SR with Norms Model. Regarding the norms, "DEF" was the norm with worse results, with 68.97% f1 score, and "LEFFECT" was the norm with the highest f1 score (100%). For the named entities, the time-related concepts ("TIME_DURATION" and "TIME_DATE_REL_TEXT") were the ones with lower f1 score (62.63% and 62.22%, respectfully). "TREF" was the concept that had the highest F1 score (91.76%). In regards to the semantic roles, most concepts had a good f1 score (between 70% and 95%). Only three ("DEFINIENS", "ACTION" and "PURPOSE") had satisfactory results (f1 score between 64% and 70%). Based on all the results and the existing noise, we concluded the Hybrid SNR system had a good performance, resulting in 81.44% f1 score. Finally, we also show how our system improved an existing information retrieval system.

[1] The code for our system can be found in https://github.com/mariaDuarte98/Dissertation_Project_LawNLP

8.1 Achievements

For the first time, a corpus of the Portuguese Consumer Law was created and annotated, which is a complex process due to the ambiguity and language used in this type of texts.

Also, to the best of our knowledge, we are the first to create a system of this kind (classification) for Portuguese legal text. With this thesis, we were able to answer the research question it focused on. We showed how the Hybrid SNR system was implemented, and how it is able to capture the relevant semantic concepts (norms, semantic roles, and named entities), allowing laws to have a more informative representation. We demonstrated how the Hybrid SNR system achieved a pretty great performance, reaching a 81.44% f1 score. With our system, now a law is not represented only by its text. Now it is represented by the concepts it includes. For example, the system turns the segment "O presente decreto-lei entra em vigor 30 dias após a sua publicação" (which translates to,"The current decree-law takes effect 30 days after its publication), into [LEFFECT [TREF O presente decreto-lei] [EFFECT entra em vigor] [TIME_DATE_REL_TEXT 30 dias após a sua publicação]].

Regarding previous work done on legal text, even though our system results and Humphreys et al. [6] are not directly comparable (different number of samples and different labels), just by looking at their results and ours, it seems that our system outperforms theirs. Their system achieved 81.60 f1 score in detecting the norm type, while our norms model achieved a 84.74%. Their worse norm was "Legal Effect" with 34.78% f1 score, while our worse was "DEF" with 68.97%. Regarding their norms elements, from those that are similar to ours ("Definiendum", "Definiens", "Includes", "Action", "Condition" and "Exception"), only "Includes" had a higher f1 score (90.91%) than our concept "DEF-INCLUSION". Their worse result was 22.22% f1 score for the concept "Exception", which for our "EXCEPTION" concept, we achieved a 80% f1 score.

Finally, our system was shown to improve an information retrieval system [42] by using generated QAs from the predictions of the system, just as we hypothesized it would. We saw that using QAs from the annotations, resulted in a higher performance than using the QAs generated from the system predictions, but considering the complexity of an annotation task, using our system is a good solution.

Despite our achievements, there is some future research that could be made to try to improve our SNR system. This future work is described in the next section.

8.2 Future Work

Even though the system obtained good results, there is still some work that could be done to improve the system:

- Review the dataset and current annotations, in order to reduce the existent noise in the dataset;

- We used the characters “ILABEL” and “ELABEL” when adding the norms information into the segment. Future work regarding experimenting using different characters could lead to SNR system improvement;
- Train and validate the Two Phased and Hybrid approaches using the predicted labels instead of the gold labels. In order to see, if by doing this, the second level models (SR with Norms Model and NE with Norms Model) will be more robust to errors regarding the norms labels;
- Improve the ATA metric to make sure that it works, not only at the token level, but at the span level as well. This means having the metric not only tells us the agreement at the token level, but doing this for a corresponding pair of predicted and gold spans. By making such improvement, the metric could be used to validate training and to compare the different approaches, instead of using the f1 score.
- Try a different architecture for the models, by simply using the BERT model and fine tuning for our SNR task, so using the fine-tuning strategy instead of the feature-based strategy.
- Finally, we used our predictions to generate simple questions and answers to improve an existing information retrieval system [42]. Future work, not directly regarding our system but its application, could include generating more complex rules to create more informative questions and answers, which could increase the improvement of the information retrieval system.

Bibliography

[1] E. Francesconi, "A description logic framework for advanced accessing and reasoning over normative provisions," *Artificial intelligence and law*, vol. 22, no. 3, pp. 291–311, 2014.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

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

[4] J. Alammar, "The illustrated bert, elmo, and co," *How NLP Cracked Transfer Learning*, 2019. [Online]. Available: http://jalammar.github.io/illustrated-bert

[5] D. Jurafsky and J. H. Martin, "Speech and language processing (third edition draft)," *chapter 20: Semantic role labeling*, 2019.

[6] L. Humphreys, G. Boella, L. van der Torre, L. Robaldo, L. Di Caro, S. Ghanavati, and R. Muthuri, "Populating legal ontologies using semantic role labeling," *Artificial Intelligence and Law*, pp. 1–41, 2020.

[7] T. Dozat and C. D. Manning, "Deep biaffine attention for neural dependency parsing," *arXiv preprint arXiv:1611.01734*, 2016.

[8] J. Yu, B. Bohnet, and M. Poesio, "Named entity recognition as dependency parsing," *arXiv preprint arXiv:2005.07150*, 2020.

[9] M. Sergot, F. Sadri, R. Kowalski, F. Kriwaczek, P. Hammond, and H. Cory, "The british nationality act as a logic program." *Communications of the ACM 29*, pp. 370–386, 1986.

[10] K. Ashley, "Reasoning with cases and hypotheticals in hypo," *International Journal of Man-Machine Studies 34*, pp. 753–796, 1991.

[11] L. K. Branting, “Data-centric and logic-based models for automated legal problem solving,” *Artificial Intelligence and Law*, vol. 25, no. 1, pp. 5–27, 2017.

[12] W. N. Hohfeld, “Fundamental legal conceptions as applied in judicial reasoning,” *The Yale Law Journal*, vol. 26, no. 8, pp. 710–770, 1917.

[13] J. L. Pollock, “Defeasible reasoning,” *Cognitive science*, vol. 11, no. 4, pp. 481–518, 1987.

[14] H. Prakken and G. Sartor, “Law and logic: A review from an argumentation perspective,” *Artificial Intelligence*, vol. 227, pp. 214–245, 2015.

[15] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” 2015.

[16] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.

[17] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[18] D. Hendrycks and K. Gimpel, “Gaussian error linear units (gelus),” 2020.

[19] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Icml*, 2010.

[20] A. Louis, “Netbert: A pre-trained language representation model for computer networking.” Ph.D. dissertation, 06 2020.

[21] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Łukasz Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, “Google’s neural machine translation system: Bridging the gap between human and machine translation,” 2016.

[22] D. Jurafsky and J. H. Martin, “Speech and language processing (third edition draft),” *chapter 14: Dependency Parsing*, 2019.

[23] A. V. Aho and J. D. Ullman, *The theory of parsing, translation, and compiling*. Prentice-Hall Englewood Cliffs, NJ, 1973, vol. 1.

[24] D. Gildea and D. Jurafsky, “Automatic labeling of semantic roles.” *Computational Linguistics*, pp. 245–288, 2002.

[25] M. Palmer, D. Gildea, and P. Kingsbury, “The proposition bank: An annotated corpus of semantic roles,” *Computational linguistics*, vol. 31, no. 1, pp. 71–106, 2005.

[26] K. D. Cooper and L. Torczon, “Chapter 3 - parsers,” in *Engineering a Compiler (Second Edition)*, second edition ed., K. D. Cooper and L. Torczon, Eds. Boston: Morgan Kaufmann, 2012, pp. 83–159. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780120884780000037

[27] A. Björkelund, L. Hafdell, and P. Nugues, “Multilingual semantic role labeling,” in *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*. Boulder, Colorado: Association for Computational Linguistics, Jun. 2009, pp. 43–48. [Online]. Available: https://aclanthology.org/W09-1206

[28] E. Kiperwasser and Y. Goldberg, “Simple and accurate dependency parsing using bidirectional lstm feature representations,” *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 313–327, 2016.

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

[30] S. Sarawagi and W. W. Cohen, “Semi-markov conditional random fields for information extraction,” *Advances in neural information processing systems*, vol. 17, pp. 1185–1192, 2004.

[31] W. Lu and D. Roth, “Joint mention extraction and classification with mention hypergraphs,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 857–867.

[32] A. O. Muis and W. Lu, “Labeling gaps between words: Recognizing overlapping mentions with mention separators,” *arXiv preprint arXiv:1810.09073*, 2018.

[33] A. Katiyar and C. Cardie, “Nested named entity recognition revisited,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 861–871.

[34] J. Fisher and A. Vlachos, “Merge and label: A novel neural network architecture for nested ner,” *arXiv preprint arXiv:1907.00464*, 2019.

[35] N. Sakhaee and M. C. Wilson, “Information extraction framework to build legislation network,” *CoRR*, vol. abs/1812.01567, 2018. [Online]. Available: http://arxiv.org/abs/1812.01567

[36] F. Ruggeri, F. Lagioia, M. Lippi, and P. Torroni, “Detecting and explaining unfairness in consumer contracts through memory networks,” *Artificial Intelligence and Law*, 05 2021.

[37] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus, “End-to-end memory networks,” *Arxiv*, pp. 1–11, 03 2015.

[38] C. Freitas, P. Carvalho, H. Gonçalo Oliveira, C. Mota, and D. Santos, “Second harem: advancing the state of the art of named entity recognition in portuguese,” in *quot; In Nicoletta Calzolari; Khalid Choukri; Bente Maegaard; Joseph Mariani; Jan Odijk; Stelios Piperidis; Mike Rosner; Daniel Tapias (ed) Proceedings of the International Conference on Language Resources and Evaluation (LREC 2010)(Valletta 17-23 May de 2010) European Language Resources Association*. European Language Resources Association, 2010.

[39] F. Souza, R. Nogueira, and R. Lotufo, “BERTimbau: pretrained BERT models for Brazilian Portuguese,” in *9th Brazilian Conference on Intelligent Systems, BRACIS, Rio Grande do Sul, Brazil, October 20-23 (to appear)*, 2020.

[40] E. Brochu, V. M. Cora, and N. de Freitas, “A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning,” 2010.

[41] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[42] N. Cordeiro, “NLP applied to the portuguese consumer law,” Master’s thesis, Instituto Superior Técnico, 2021 (to be published).

[43] W. L. Taylor, ““cloze procedure”: A new tool for measuring readability,” *Journalism & Mass Communication Quarterly*, vol. 30, pp. 415 – 433, 1953.

A

Appendix

A.1 Dataset Categorization

		Number of occurrences			
		Dataset	Train	Val	Test
Label	OBLIG	4332	3460	442	430
	DEF	487	364	75	48
	RIGHT	1139	916	114	109
	INTRO	79	58	15	6
	LEFFECT	188	150	19	19
	LREF	1232	978	132	122
	TREF	3945	3037	438	470
	NE_ADM	1828	1447	200	181
	TIME_DURATION	560	446	61	53
	TIME_DATE_REL_TEXT	219	175	22	22
	DEFINIENDUM	528	402	75	51
	DEFINIENS	431	320	72	39
	DEF-INCLUSION	576	416	66	94
	SCOPE	693	536	73	84
	ACTION	5733	4577	592	564
	CONDITION	2311	1834	254	223
	CONCESSION	479	364	66	49
	PURPOSE	439	363	37	39
	EXPERIENCER	2201	1735	247	219
	OBJECT	3984	3200	407	377
	THEME	311	249	33	29
	EXCEPTION	278	219	29	30
	EFFECT	406	324	42	40
	NEG	4332	3460	442	430

Figure A.1: Labels statistics for the dataset and train/val/test sets

		Number of occurrences			
		Dataset	Train	Val	Test
Label	OBLIG	4235	3390	419	426
	DEF	527	390	88	49
	RIGHT	1122	905	109	108
	INTRO	84	62	15	7
	LEFFECT	188	150	19	19
	LREF	1216	966	129	121
	TREF	3863	2968	433	462
	NE_ADM	1836	1448	205	183
	TIME_DURATION	496	394	53	49
	TIME_DATE_REL_TEXT	235	189	23	23
	DEFINIENDUM	501	375	76	50
	DEFINIENS	798	548	126	124
	DEF-INCLUSION	119	100	12	7
	SCOPE	384	295	50	39
	ACTION	5643	4514	567	562
	CONDITION	1517	1192	174	151
	CONCESSION	247	195	28	24
	PURPOSE	195	157	19	19
	EXPERIENCER	2104	1663	231	210
	OBJECT	3682	2948	369	365
	THEME	259	208	26	25
	EXCEPTION	291	231	29	31
	EFFECT	401	320	41	40
	NEG	4235	3390	419	426

Figure A.2: Labels statistics for the unrevised dataset and train/val/test sets

A.2 Approaches with Example

The following figures show the different SNR approaches and their inputs and outputs for the following sentence: “O presente decreto-lei entra em vigor 30 dias após a sua publicação.” which translates to, “The current decree-law takes effect 30 days after its publication”.

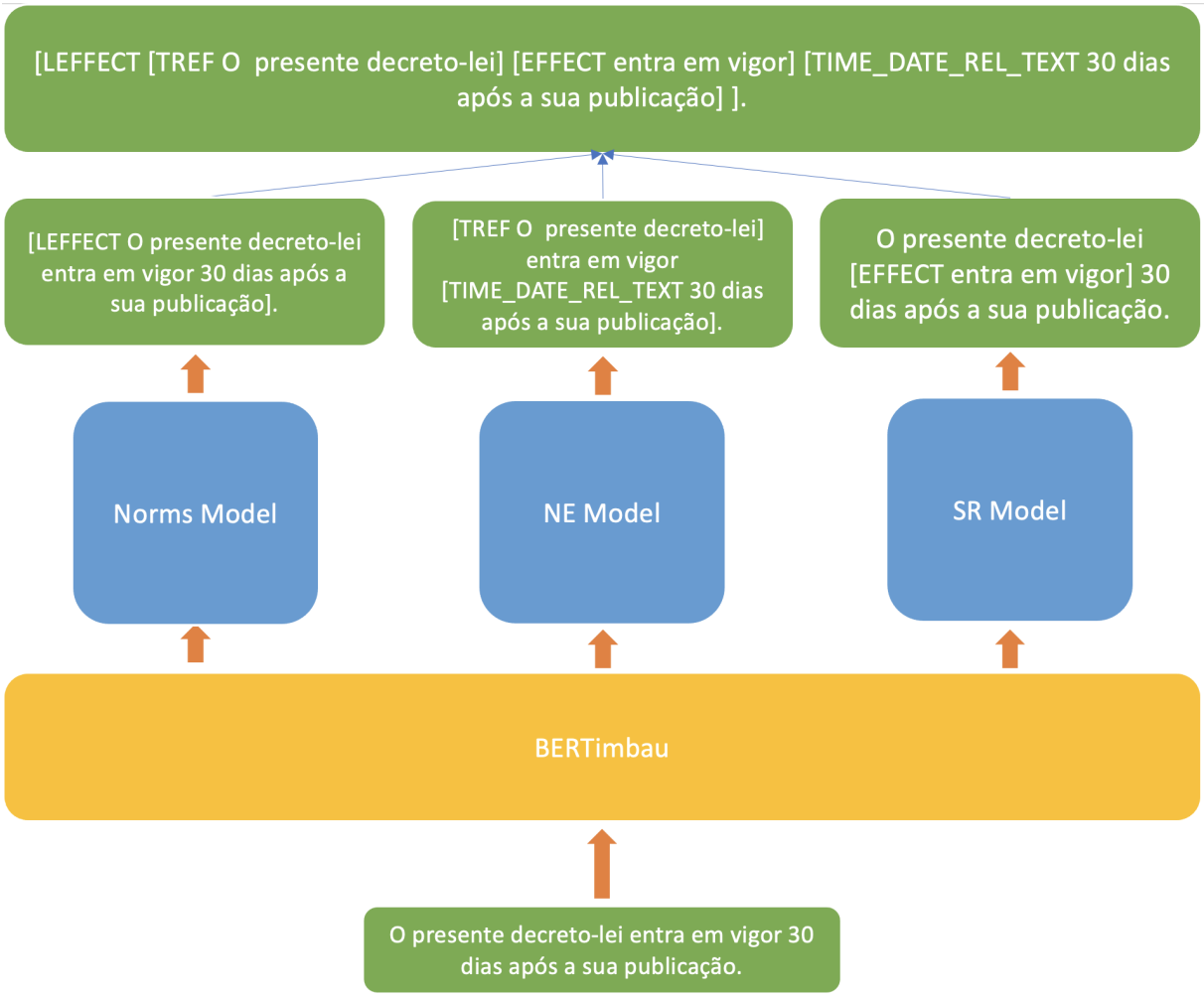

Figure A.3: Baseline SNR System for a single sentence

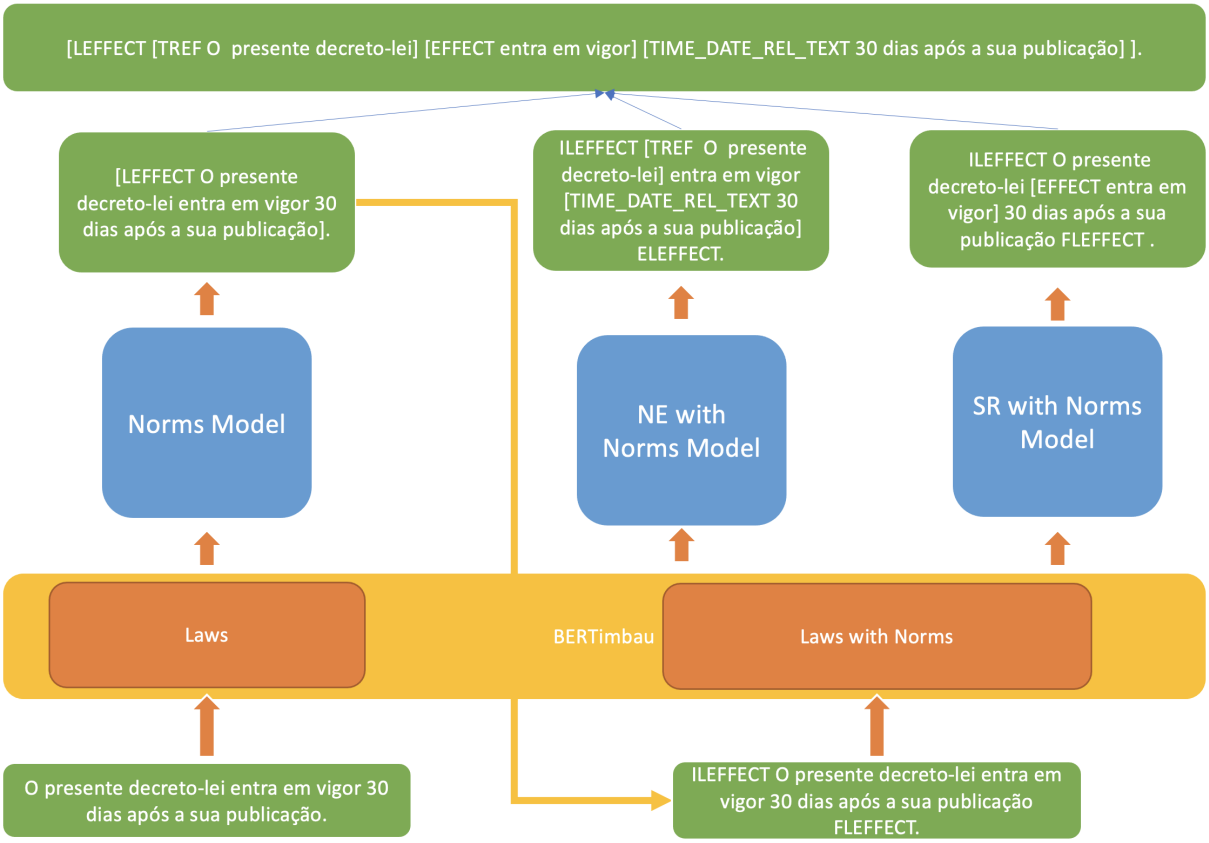

Figure A.4: Two Phased SNR System for a single sentence

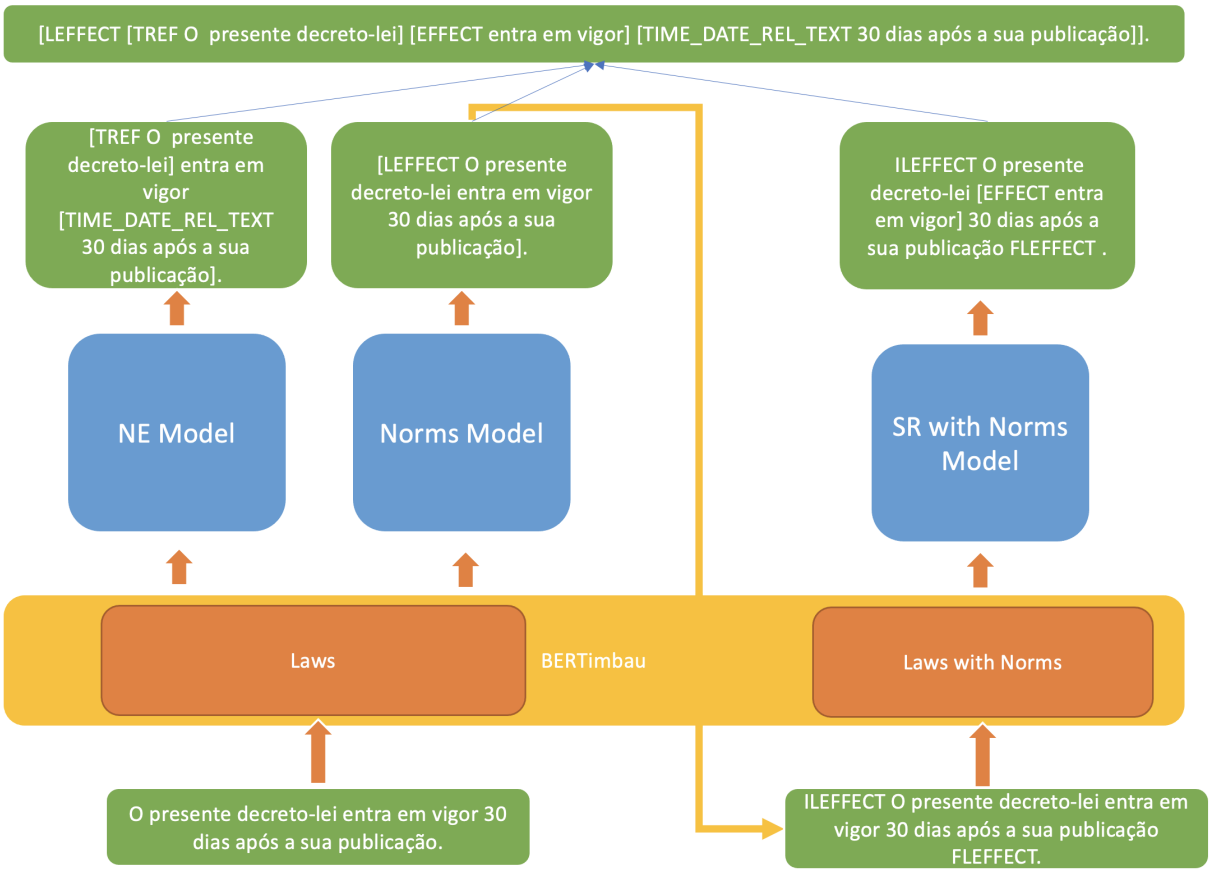

Figure A.5: Hybrid SNR System for a single sentence

A.3 Training Models with Default Parameters

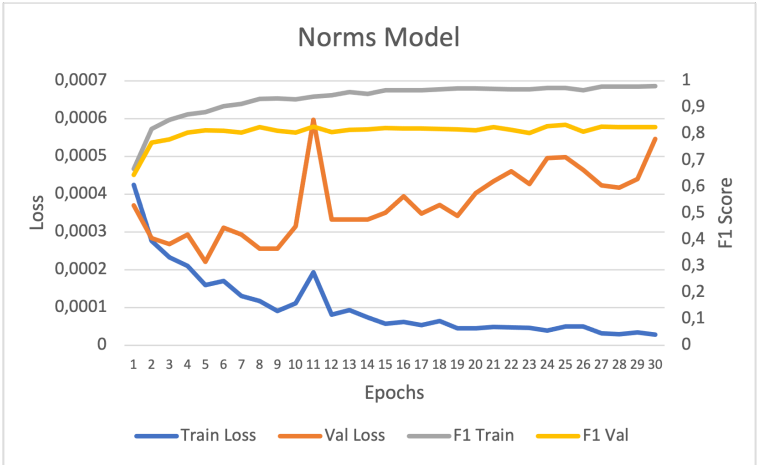

Figure A.6: Norms model using default parameters

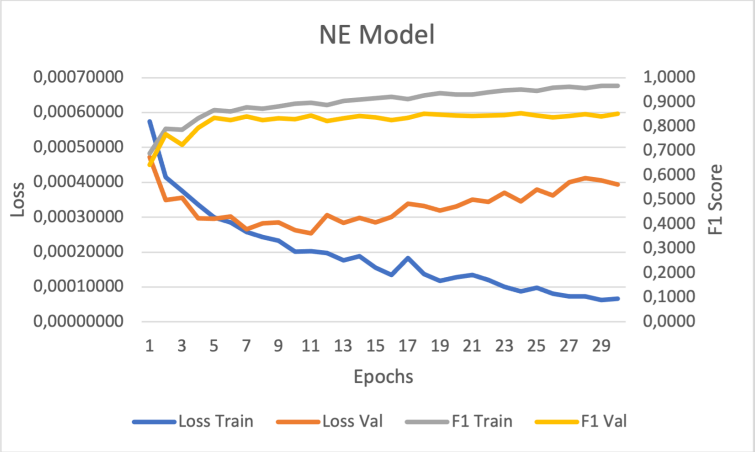

Figure A.7: Named Entities model using default parameters

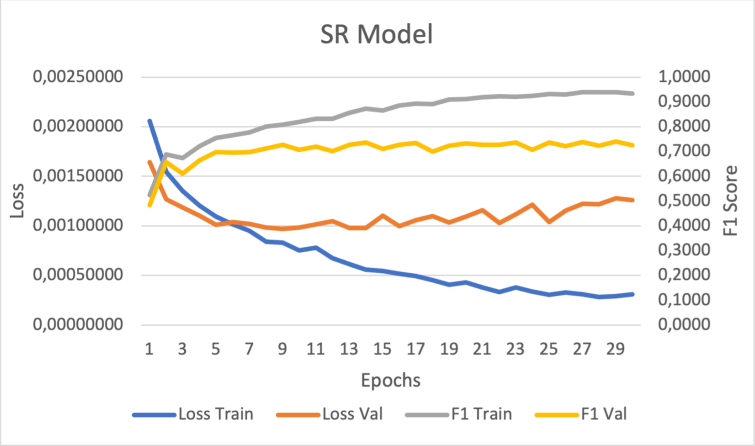

Figure A.8: Semantic Roles model using default parameters

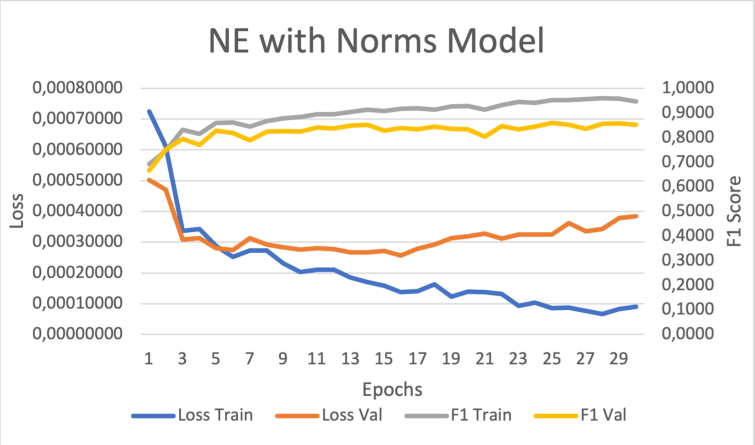

Figure A.9: Named Entities model with Norms using default parameters

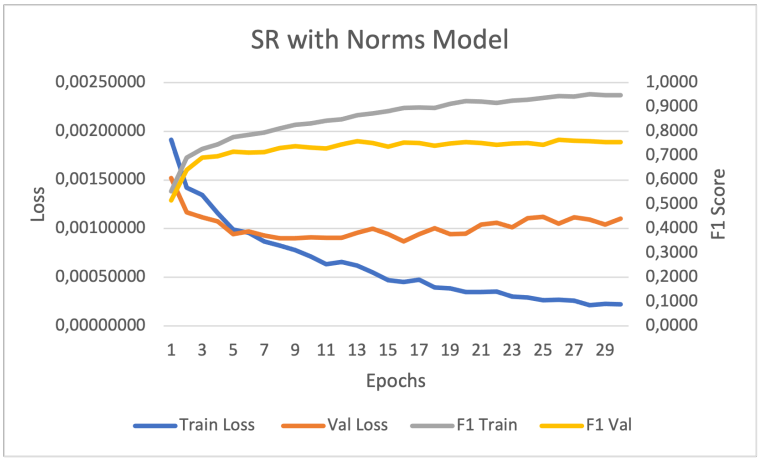

Figure A.10: Semantic Roles model with Norms using default parameters

A.4 Bayesian Optimization Results

LEARNING RATE	FFNN_SIZE	BILSTM_SIZE	DECAY RATE	F1 VAL SCORE
0.001	150	200	0.999	0.8271237
0.02423900653612488	138	205	0.8034692393895433	0.68081633
0.07267652752883945	219	237	0.8931722606410599	0.6263287
0.002633499164162156	104	56	0.8768565277497855	0.81667978
0.001448307275332236	104	195	0.501810916397751	0.79069767
0.0028629611602629377	93	121	0.9229602381157702	0.82957198
0.009225743999705759	175	79	0.5351703335431729	0.77221325
0.005553099298213195	165	228	0.6293495514683931	0.80754124
0.04429400494132821	115	139	0.6421194226770716	0.64600326
0.023159522200063096	164	80	0.9482397510400009	0.80621423
0.01909860892357006	129	152	0.5807164522356729	0.69657423
0.00924937723910393	233	50	1.0	0.80820837
0.005474930290028525	181	104	0.7248737157568105	0.80895284
0.095411150133247 55	59	66	0.8297054375906233	0.56284153
0.0702871790231733	179	185	0.9323146256298612	0.62908497
0.007943010006907081	144	196	0.9525988169642057	0.82753287
0.001186428759827467	107	300	0.6491899049505832	0.80414013
0.004215144328560204	195	131	0.7984912255878125	0.83102919
0.0010991431393342784	169	141	0.5942209242718379	0.78406375
0.011641257866464392	76	284	0.8355066605460645	0.82867925
0.002080004215086348	171	178	0.6821377716948549	0.81175536
0.06644971530000038	232	170	0.6727846877387273	0.62857143
0.006778571678401113	50	300	0.8656666356174549	**0.83921569**
0.001	50	300	0.8560619735876909	0.83218391
0.015112205154280545	50	300	1.0	0.79149632
0.009199271385496357	204	300	1.0	0.80390561
0.008809999695794416	250	50	0.878219310318779	0.8153967
0.0019942057180530634	183	253	1.0	0.81638847
0.001	243	65	0.9098055701068262	0.81068342
0.005311537922315577	50	300	0.864182470495047	0.83830455
0.08139763614055 85	224	89	0.7128167080958314	0.62871287
0.022633034146088478	225	64	0.6583904379994244	0.80668258
0.07103105784309212	229	99	0.6348940948574485	0
0.006577632981987029	221	96	0.9531661065076971	0.83007519
0.04839253470559522	159	234	0.8380779001467892	0.63882064
0.022143894348139213	105	150	0.7084371168960986	0.7745098
0.003410143567564213	66	229	0.6956612560270943	0.81115538
0.07725306957015363	62	145	0.5077561734108453	0.59010601
0.019177864670821904	85	249	0.9550758507001241	0.68613139
0.0014989543352374028	227	59	0.574194321025888	0.78646253
0.0037199642376855935	67	251	1.0	0.83523159
0.003728180913317692	250	300	1.0	0.83157895
0.0037449902330210704	50	300	1.0	0.83675019
0.0015624168059303045	153	251	0.6266931617302005	0.7961935
0.0018071055572788455	151	72	0.950437283227592	0.82893745
0.0021404862184295206	62	162	0.8352926188858476	0.82795699
0.004014495946723046	50	50	1.0	0.82371295
0.0010007706637533502	67	198	0.7930632667136621	0.81463803
0.0012311438284942561	219	156	0.917099506244182	0.8322884
0.0021968034178164258	179	248	0.9946113364296097	0.83126935

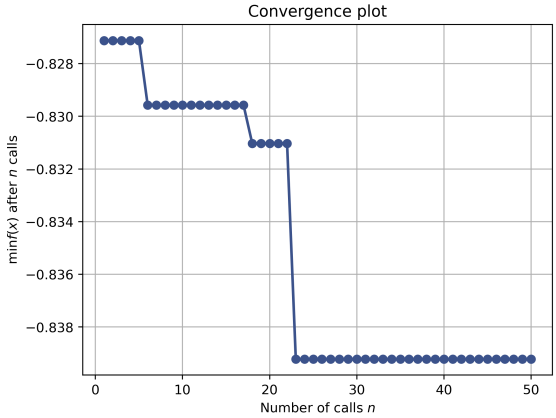

Figure A.11: Optimization Results of the Norms Model: First Limits

LEARNING RATE	FFNN_SIZE	BILSTM_SIZE	DECAY RATE	F1 VAL SCORE
0.010714700412183541	48	302	0.7302767385832211	0.8321513
0.0027624682824698363	37	479	0.5300119397577977	0.80321285
0.0010738020154281158	30	404	0.6136813933736048	0.7926045
0.007899749323035601	39	397	0.6392251228306547	0.7960688
0.02213211748434263	46	425	0.9108303983576711	0.65415987
0.008591684449790951	36	483	0.568817567696064	0.70597871
0.004168096898787611	39	410	0.9194413614582968	0.84377358
0.06477041272431117	36	373	0.8949973986279487	0.62233169
0.01523820102679427	33	470	0.5223416352901493	0.64326531
0.021926264974133376	43	305	0.9006722499011631	0.68515498

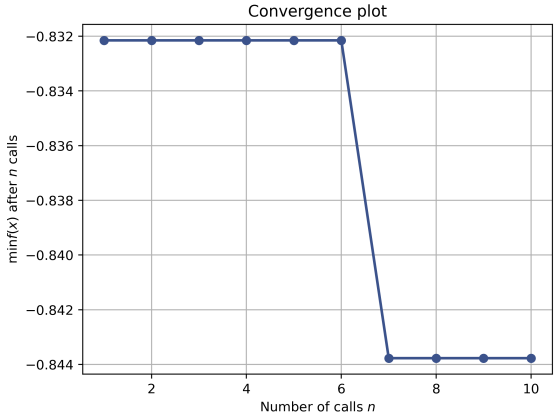

Figure A.12: Optimization Results of the Norms Model: Second Limits

LEARNING RATE	FFNN_SIZE	BILSTM_SIZE	DECAY RATE	F1 VAL SCORE
0.001	150	200	0.999	0.84511981
0.002112038582321693	152	54	0.9338416473582855	0.84820921
0.0034717670336597306	214	80	0.814365976847168	0.84897959
0.0021347819380605135	228	178	0.6147652393131846	0.83419392
0.0042441413312466125	75	252	0.8581906831865003	**0.8675535**
0.0020215780679044937	149	299	0.9880080408863902	0.84216524
0.010183937779554956	60	149	0.954154748368373	0.84465446
0.026308175165268708	225	263	0.7574808733038481	0.78084715
0.0015576512702164696	167	277	0.8524278813875635	0.85909869
0.07859344025476182	79	212	0.6686633847326235	0.35314685
0.03544029847044905	90	280	0.5486092614318355	0.30261882
0.001	50	300	0.9419279184162905	0.86042504
0.06529828104170128	250	300	1.0	0.37937872
0.001	50	300	0.5	0.81102814
0.018455877872014366	250	254	1.0	0.72682927
0.001	50	300	0.6831838674383617	0.84014002
0.010473608831536729	50	300	0.8308863800864856	0.86398132
0.00595727390138861	219	287	0.5	0.82842288
0.001245495357704435	104	289	1.0	0.84597433
0.0029611893511501085	50	300	0.5	0.83969466
0.006148594748070925	50	300	0.6506153709943996	0.84420929
0.005997669294242812	250	300	0.9064475312431364	0.84943011
0.012256337165453901	103	186	0.5	0.82183563
0.007737438456869853	50	300	1.0	0.84270953
0.004840196312983251	50	50	1.0	0.83451537
0.002546702135894514	71	279	1.0	0.85159011
0.003951947189611815	63	300	1.0	0.84241359
0.009210389894972196	250	300	0.5	0.79089791
0.006800478370967059	250	50	1.0	0.83833718
0.004173449094844068	250	50	0.5	0.81512109
0.0014876293411780815	95	130	0.9676240791734103	0.85155351
0.0024480967023427485	169	257	0.9282701804579871	0.86446469
0.0023874292038302066	240	58	0.5500375090397787	0.81147541
0.005109521517913037	53	214	0.6312123567815945	0.84761905
0.0195451035547539	114	260	0.8640679914263102	0.75782748
0.0046509972292693895	129	125	0.8236908600075483	0.86244922
0.0014292203172118555	203	172	0.6924829352829522	0.83025404
0.008283692533785489	195	204	0.8694743207414446	0.86508396
0.09249517092083599	97	79	0.8891614782615725	0.48351648
0.0029971234485304367	123	179	0.8952254282128916	0.85969084
0.0025437747753260967	50	300	0.8015436657883543	0.86016451
0.026235430663726005	156	50	1.0	0.80600462
0.019004918057838712	227	63	0.5	0.82239615
0.009437727278128596	250	50	0.8030538998692229	0.8496732
0.008902219176738758	50	300	0.786296691799504	0.85596222
0.012333030529188879	250	300	0.7979984427260061	0.83235639
0.007366997137110374	248	300	0.7966562998611786	0.85898942
0.0018015370814330176	50	300	0.7845228781238026	0.85982405
0.002969212211775273	250	300	0.7407383574957009	0.8524971
0.0011199296765818675	51	50	0.849500490517612	0.82380127

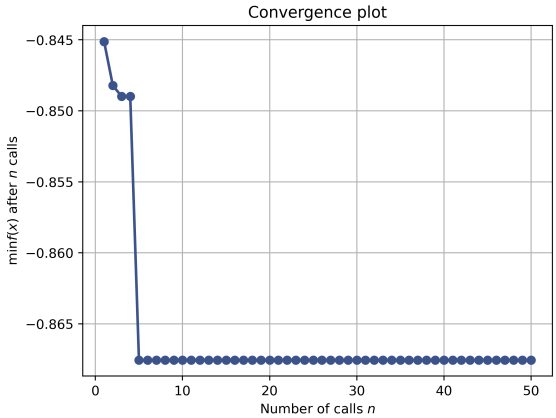

Figure A.13: Optimization Results of the Named Entities Model

LEARNING RATE	FFNN_SIZE	BILSTM_SIZE	DECAY RATE	F1 VAL SCORE
0.001	150	200	0.999	0.73407056
0.016628108661634593	144	139	0.5489350966562777	0.68797737
0.0021609988892547245	93	54	0.9438800121344812	0.7321334
0.017322951589416283	244	82	0.9974801524857386	0.68806584
0.005706502180437914	190	99	0.674147818975756	0.71928879
0.004672016219943546	135	63	0.796592616872885	0.71972318
0.028759725370779615	245	165	0.560949751003618	0.62957453
0.005777645381858156	163	92	0.8512671168352388	0.73703226
0.005332968728548248	225	99	0.9279587644445474	0.73714891
0.01096007270664905	164	146	0.7958542830731916	0.73374038
0.0068897545485747365	102	173	0.5542378031703983	0.70448912
0.007897744601006297	250	50	1.0	0.70878113
0.0059449823524667265	164	287	1.0	0.74361647
0.001	235	86	1.0	0.72901921
0.04279693561536906	50	60	1.0	0.66355909
0.001	73	300	0.6113695929121833	0.68588362
0.0014917800955853958	174	230	1.0	0.73219448
0.0034903067856396964	81	300	1.0	0.73533246,
0.001	50	93	1.0	0.72616847
0.007104142864242049	195	300	0.837087429221989	0.74445618
0.1	117	300	0.5	0
0.0021083914550706897	250	300	0.5	0.70129171
0.001925576961622797	250	300	0.8104983132673114	0.7445294
0.0022061290655594743	250	300	1.0	0.72779221,
0.021813547678492903	50	300	0.8019089908383021	0.57773512
0.004812618402801099	250	235	0.8387730857155411	0.74473068
0.0014515719262783173	250	156	0.8038380673809724	0.73493024
0.035787731265611064	50	50	0.5	0.62958509
0.004217119075734874	250	300	0.811930725541437	0.74074074
0.0070517040418782975	250	202	0.8718646196282249	0.74502521,
0.009535659426755464	190	229	0.9997070867851339	0.71498054
0.003767170916819986	172	156	0.7688429825409131	0.73226062
0.07465015942437978	168	262	0.8624693058337674	0.21083744
0.008375732919899442	152	218	0.9346949324635897	**0.74894292**
0.030415573893458694	188	234	0.953548574137709	0.52675159
0.002548606558995194	231	56	0.5762205569572618	0.67283431,
0.0022174137391686374	63	290	0.9593936116429427	0.73361747
0.0015290444625810677	191	179	0.5609447071484932	0.69185105
0.007431755048165889	210	276	0.8018465652295439	0.74115456
0.02057367654798743	214	125	0.6672574200306983	0.70074606
0.006488616010990514	50	219	0.9181099264397787	0.7362607
0.001	250	300	0.8230844921745296	0.73754839
0.014099531474524665	50	50	1.0	0.70610481
0.001	60	300	1.0	0.72984517
0.0029227373238444893	250	153	1.0	0.7224501
0.005896757507570082	159	232	0.8544216326671468	0.74834099
0.001	250	50	0.7854824369843629	0.67207157
0.0022762500067598952	53	250	0.7806788351146239	0.73086286
0.0055749756241774175	250	300	0.9168780472015543	0.73705898
0.007169500629158238	157	300	0.5963534809286439	0.7250202

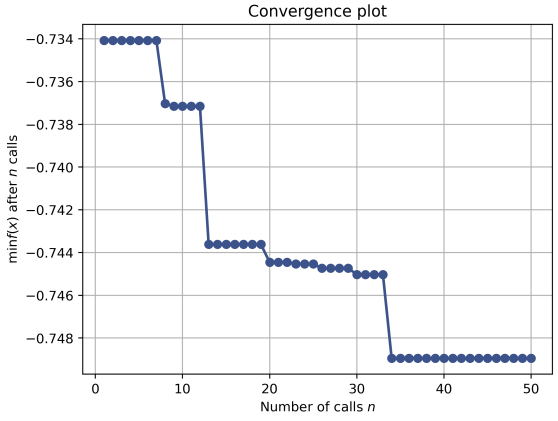

Figure A.14: Optimization Results of the Semantic Roles Model

LEARNING RATE	FFNN_SIZE	BILSTM_SIZE	DECAY RATE	F1 VAL SCORE
0.001	150	200	0.999	0.8519174
0.028897727660629573	245	59	0.7579279774598522	0.84521739
0.042326844296646626	228	129	0.8849754533803929	0.75064599
0.07342096880421152	162	146	0.5656083766684391	0
0.0018493663625436898	82	282	0.8855826664409727	0.86083382
0.004531165342543305	54	284	0.8718741293739042	**0.86171429**
0.017354037308295385	101	88	0.5882372105003779	0.83920188
0.009879689419542483	74	277	0.5837959877120615	0.83548766
0.0010145902067357354	181	250	0.5799438155595796	0.81667619
0.012693595652811085	97	146	0.5784875348377975	0.83976608
0.011688034221682722	221	195	0.970218434804526	0.81468732
0.0028668452420429974	50	50	1.0	0.84518828
0.001	242	256	0.9519076870111983	0.84619849
0.0061728834434303164	50	50	0.5	0.79976233
0.0013758627569471237	50	50	1.0	0.84502262
0.003856472182009422	50	300	0.5	0.82862297
0.0021193006183416467	50	50	1.0	0.83704149
0.0036673156319169016	50	300	1.0	0.84852002
0.02923316928534802	50	300	1.0	0.52211127
0.005429081379500235	250	275	1.0	0.83352736
0.001	118	186	1.0	0.84773663
0.003518199593516861	156	300	1.0	0.85259434
0.010557653019306517	50	50	0.5	0.82276995
0.001954921500809089	250	300	1.0	0.84994401
0.014192355084474721	250	300	0.80554436136662	0.82242991
0.0013647902148945673	50	50	0.5	0.72875
0.002346325909340456	250	300	0.5963991567206675	0.83680556
0.027260626388321385	250	300	1.0	0.34066986
0.03384646340317416	50	50	0.5399283146417562	0.81253696
0.08096624729733061	50	280	0.5003466719886219	0.31309298
0.0011342629800121192	241	156	0.6181150029933002	0.81342435
0.0038924609726325252	183	84	0.6691122737387628	0.82691202
0.03449093079955794	192	241	0.8361895925129945	0.68079952
0.039264751276792895	127	93	0.7472362012941354	0.81200245
0.019134947230531316	65	93	0.7135475751054459	0.84570082
0.09404752065505917	111	247	0.6346947574249782	0.36268527
0.0846985077350712	124	286	0.9263909516655807	0.48888889
0.003784005428948044	128	208	0.9136349541350111	0.86080373
0.006946611691978616	182	96	0.8602650315743006	0.85530922
0.05209321995579226	59	284	0.6688830924026239	0.30198915
0.0013626281122920117	50	241	1.0	0.85
0.0323149247746147	50	50	1.0	0.78950736
0.00243012594162251	50	198	1.0	0.84554679
0.0016615746392997666	250	154	1.0	0.83828775
0.022176044031986034	50	50	0.5	0.7990373
0.006965999406940399	50	195	1.0	0.84778863
0.012907043238151564	50	300	0.5	0.7905854
0.007909246280964033	250	189	0.5	0.83313886
0.015733564036663438	50	50	1.0	0.84637029
0.001	250	50	1.0	0.83744974

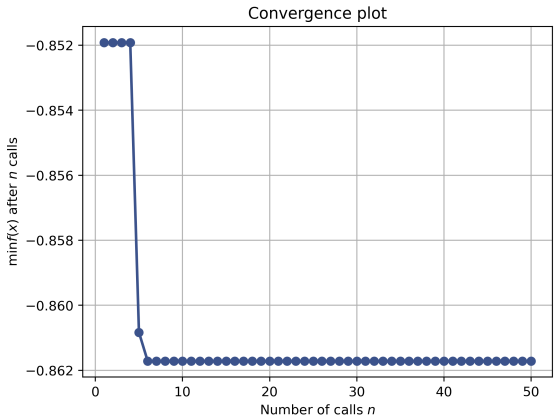

Figure A.15: Optimization Results of the Named Entities with Norms Model

LEARNING RATE	FFNN_SIZE	BILSTM_SIZE	DECAY RATE	F1 VAL SCORE
0.001	150	200	0.999	0.75986758
0.002550488705013783	191	182	0.6486524272052533	0.73499734
0.0467214364350893	95	267	0.6159306098046401	0.00100301
0.0048886931885861774	90	289	0.5103451764597475	0.73199783
0.00508125636281769	134	202	0.5834921674828942	0.73633441
0.012491765030010444	141	163	0.9121565234400455	0.75940825
0.029948936665740406	224	114	0.5145641500439269	0.60866948
0.02260660228486574	211	292	0.8743601256108473	0.64769323
0.021202745143058536	220	88	0.9597325828319551	0.73375474
0.0012708354727404057	236	96	0.9556117849264412	0.75221463
0.0018106857228353933	227	148	0.6462117490477607	0.72152907
0.011832650617002317	250	50	0.5576202106795308	0.65985816
0.001971295966933988	50	50	1.0	0.74800515
0.007699891421322025	103	50	1.0	0.75104603
0.001	50	50	0.5	0.49285228
0.002321727863944083	117	300	1.0	0.7535545
0.0032767397771356495	174	64	0.9241435543827969	0.75854813
0.006149757601914804	195	217	0.9398691066063583	0.76287554
0.010523818856267013	174	50	1.0	0.71604938
0.0036343635396982087	50	50	1.0	0.73192191
0.001	115	281	0.9855143335618357	0.75435005
0.015922352868666483	50	50	1.0	0.72694945
0.010903785833911835	50	300	1.0	0.72653503
0.001	250	300	1.0	0.74926254
0.001	50	300	0.8018422097004836	0.74441153
0.1	50	300	1.0	0
0.0014029718116045818	248	215	1.0	0.75514019
0.019556980427433843	250	50	0.6930639802376473	0.71145804
0.0037468182219759795	250	300	1.0	0.7467584
0.013830790625960447	213	300	0.5	0.65521258
0.012373980720511747	164	231	0.9987319205216273	0.70559345
0.021286795812612995	246	114	0.7424778639361598	0.73478261
0.0014368535581601107	161	223	0.8938340398675262	0.76441103
0.002991595553583219	134	273	0.8829473172470457	**0.77345775**
0.009239960393087119	194	217	0.6581901160240116	0.72657952
0.03261839143056099	85	90	0.7256852731647792	0.70538956
0.0789891017505432	241	123	0.8541929975436469	0
0.01190787125542986	70	139	0.9687560525627139	0.74945652
0.004172096290914826	216	62	0.8404217481509813	0.74594307
0.018053427850933487	186	64	0.9395788839028842	0.7398374
0.026587064215886358	50	50	0.8159361536445273	0.7223114
0.00257482470364164	50	162	0.9278095556362411	0.76275177
0.00609373198072829	50	300	0.8031625055076228	0.76165113
0.0019608828052015306	50	300	0.683847233593727	0.7383025
0.007073801218111093	50	121	0.8560689737666183	0.76752768
0.0037085005457513707	50	300	0.7385860047773929	0.75764831
0.0036963970060859912	50	183	0.8494955752520812	0.76759283
0.0025737780556584298	50	50	0.7666399061338366	0.69831631
0.005307771002998097	250	128	0.9936263829943903	0.75024826
0.0017736409307171617	250	144	1.0	0.75454313

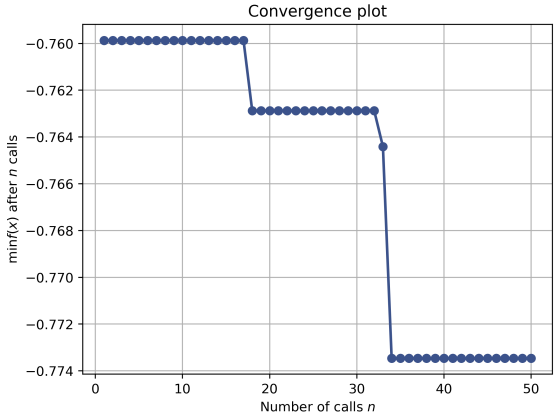

Figure A.16: Optimization Results of the Semantic Roles with Norms Model

A.5 Hybrid SNR System Confusion Matrix Results

PREDICTED LABELS

GOLD LABELS	NoLabel	LREF	TREF	NE_ADM	TIME_DATE_REL_TEXT	TIME_DURATION
NoLabel	0	19	48	22	14	6
LREF	4	118	0	0	0	0
TREF	28	1	440	0	1	0
NE_ADM	17	0	0	164	0	0
TIME_DATE_REL_TEXT	18	0	1	0	31	3
TIME_DURATION	8	0	0	0	0	14

Figure A.17: Hybrid SNR System: NE Model confusion matrix

PREDICTED LABELS

GOLD LABELS	NoLabel	DEFINIENDUM	DEFINIENS	DEF-INCLUSION	SCOPE	ACTION	CONDITION	CONCESSION	PURPOSE	EXPERIENCER	THEME	EXCEPTION	EFFECT	NEG
NoLabel	0	2	5	14	16	132	36	5	11	12	68	7	7	10
DEFINIENDUM	13	34	0	0	0	0	0	0	0	2	2	0	0	0
DEFINIENS	13	0	22	0	0	1	1	0	0	0	2	0	0	0
DEF-INCLUSION	14	0	0	77	0	0	0	0	0	0	3	0	0	0
SCOPE	20	0	0	0	58	0	2	0	4	0	0	0	0	0
ACTION	182	0	1	0	0	370	0	0	0	0	11	0	0	0
CONDITION	43	0	0	0	0	0	180	0	0	0	0	0	0	0
CONCESSION	3	0	0	0	0	0	0	46	0	0	0	0	0	0
PURPOSE	12	0	0	0	0	0	0	0	27	0	0	0	0	0
EXPERIENCER	47	0	0	0	0	0	0	0	0	165	7	0	0	0
THEME	52	1	1	0	1	0	0	0	1	8	313	0	0	0
EXCEPTION	5	0	0	0	0	0	0	0	0	0	0	24	0	0
EFFECT	4	0	0	0	0	0	0	0	0	0	0	0	26	0
NEG	2	0	0	0	0	0	0	0	0	0	0	0	0	38

Figure A.18: Hybrid SNR System: SR with Norms Model confusion matrix

A.6 Hybrid SNR System Results with Unrevised Dataset

Norms Model (Unrevised Dataset)				
Labels	**F1 Score**	**Recall**	**Precision**	**ATA**
DEF	57.43%	59.18%	55.77%	0.6275160717742821
OBLIG	89.05%	86.85%	91.36%	0.8800986357045613
RIGHT	75.12%	74.07%	76.19%	0.6957944896087531
LEFFECT	100.00%	100.00%	100.00%	1.0
INTRO	72.73%	57.14%	100.00%	0.5714285714285714

NE Model (Unrevised Dataset)				
Labels	**F1 Score**	**Recall**	**Precision**	**ATA**
LREF	88.45%	91.74%	85.38%	0.855694256991707
TREF	89.11%	88.53%	89.69%	0.914479767532399
NE_ADM	85.04%	88.52%	81.82%	0.7810628624836923
TIME_DATE_REL_TEXT	60.22%	57.14%	63.64%	0.49126527067703546
TIME_DURATION	58.54%	52.17%	66.67%	0.41987577639751555

SR with Norms Model (Unrevised Dataset)				
Labels	**F1 score**	**Recall**	**Precision**	**ATA**
DEFINIENDUM	74.73%	68.00%	82.93%	0.6993691493691494
DEFINIENS	72.87%	72.58%	73.17%	0.6996657583480156
DEF-INCLUSION	0.00%	0.00%	0.00%	0.0
SCOPE	50.75%	43.59%	60.71%	0.3775146422205245
ACTION	74.49%	71.17%	78.12%	0.8063670432072907
CONDITION	65.43%	58.28%	74.58%	0.573069634342892
CONCESSION	59.09%	54.17%	65.00%	0.41935483870967744
PURPOSE	32.26%	26.32%	41.67%	0.21294021294021295
EXPERIENCER	83.00%	79.05%	87.37%	0.7986878557104938
THEME	83.45%	80.82%	86.26%	0.8258586003919628
EXCEPTION	70.83%	68.00%	73.91%	0.6216545551222971
EFFECT	90.62%	93.55%	87.88%	0.8951388888888889
NEG	88.89%	100.00%	80.00%	0.8

Hybrid System Performance (Unrevised Dataset)			
F1 Score	**Recall**	**Precision**	**ATA**
80.37%	78.15%	82.72%	0.831888833409051

Model	**F1 Score**	**Recall**	**Precision**	**ATA**
Norms	84.09%	82.43%	85.81%	0.9141356330338617
NE	85.75%	86.16%	85.34%	0.9743342403962338
SR with Norms	76.08%	72.50%	80.03%	0.8308883779603922

Figure A.19: Hybrid SNR System: All Model Results using old corpus

A.7 Classification Metrics

In order to understand the classification metrics described in this thesis, let us define the concept of True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN):

- TP correspond to samples that were classified as positive and really are positive.
- TN correspond to samples that were classified negative and really are negative.
- FP correspond to samples that were classified positive when in reality they are negative.
- FN correspond to samples that were classified negative when in reality they are positive.

Knowing these concepts, we can know define the binary classification metrics:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN}$$