# NLP Applied To Portuguese Consumer Law

Applying NLP Techniques to the Search of Portuguese Consumer Law

**Nuno Cerqueira Pablo Cordeiro**

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. Diogo Manuel Ribeiro Ferreira
Supervisor: Prof. Pedro Alexandre Simões dos Santos
Member of the Committee: Prof. Bruno Emanuel Da Graça Martins

**June 2022**

# Acknowledgments

To say that this thesis was a challenge for me is not an understatement. I feel like I have grown a lot since I started it, not only from the time it took me to finish it, but also from the wonderful opportunities that I had before me during this adventure. And to that, I owe it, first and foremost, to my supervisors Prof. João Dias and Prof. Pedro Santos. They have supported me through a rough patch without ever making me feel like I owed them anything, but I do. They also gave me the opportunity of joining a project that a masters student can only dream of, giving me more of a purpose when the writing would not go so well and keeping me engaged in the work that I had to develop in order to complete the thesis. The knowledge and expertise that they passed on to me is something I do not underestimate — and, for that, I will forever be in debt. Obrigado, professores.

I would also like to thank my family for helping me, for taking care of me when even I could not, and for giving me all the tools necessary to navigate this path, when most of them did not have the opportunity when they were my age. I know the privilege I carry and I owe it to you. Obrigado pais, avós e mano.

In addition, I would also like to thank everyone that was involved in the development of this project and its thesis, starting with the team at INCM — Carlos Gonçalves, Bruno Patrão, Ricardo Matias, Diogo Proença and a special thanks to Helder Santos, without whom this project would not be possible. Thank you, Ana Catarina Fonseca and Ana Filipa Claro, from DGC, for your contribution and thank you Maria Duarte for providing me with resources that were very useful in the development process.

Last but not least, to all my friends that helped in keeping me sane with their company and friendship, you have all been irreplaceable throughout these years.

I cannot promise that I will never forget how important every single one of you was during this stage, because we all do, at some point in our lives — but know that you were. This thesis is the product of my work, but also yours.

Thank you.

# Abstract

As citizens, each and every one of us should know their rights and obligations, especially in a day to day context such as when we pose as a consumer. As of yet, the Portuguese Consumer law is not accessible to the point of being able to insert a sentence written in natural language in a search engine and getting a clear response without first having to scroll through multiple search results. This type of barrier is also what keeps the common citizen from consulting the legislation, especially given the amount of jargon used in legal documents and their structure, which can be difficult to navigate.

The issue lies in the way that the Diário da República Eletrónico (DRE) search engine works. Right now, it performs a word matching algorithm but the problem appears when the system cannot find the answer to a query when it has juridical jargon that the user does not understand and does not use to search.

To solve this issue, we introduce Legal Semantic Search Engine (LeSSE), an information retrieval system that uses a hybrid approach of semantic and syntactic information retrieval techniques, based on the Quin system created by Samarinas et al. LeSSE uses a traditional information retrieval algorithm (BM25) along with BERT to derive and use the semantic value of a query to search the corpus. By using the semantics of the query and the corpus, we are able to create a connection between synonymous words and expressions with different spelling. For this purpose the law corpus had to be fragmented into small segments of text, using a text parsing algorithm that accounted for the structure of each document. We explored the use of different training data for the language models used in the system, before arriving at a configuration that performed the best, with an accuracy of 89.0% on the legislation corpus where we had annotated queries for training, and 78.1% on another corpus with little annotated queries. LeSSE is currently in production, and is also available to the public for testing.

# Keywords

Information Retrieval; Natural Language Processing; Artificial Intelligence; Neural Network; Legal Domain; Portuguese Consumer Law; BERT

# Resumo

Enquanto cidadãos, todos nós temos o dever de estar informados acerca dos nossos direitos e obrigações, especialmente num contexto do quotidiano, em que passamos por consumidores. Neste momento, a lei portuguesa do consumidor não é acessível ao ponto de ser possível fazer uma pesquisa num motor de busca a partir de uma frase em língua natural e saber imediatamente a resposta sem ter primeiro de percorrer vários resultados de pesquisa. Este tipo de barreira é o que previne o cidadão comum de consultar a legislação que lhe diz respeito, especialmente tendo em conta a quantidade de jargão usada em documentos legais e a sua estrutura, por vezes difícil de navegar.

O problema está no motor de busca do Diário da República Eletrónico (DRE). Neste momento, é usado um algoritmo de pesquisa de termos literais, mas o problema surge quando o sistema não consegue encontrar a resposta a uma query, porque a resposta inclui jargão jurídico que o utilizador não entende e não usa para fazer a pesquisa.

Por forma a resolver este problema, introduzimos o Legal Semantic Search Engine (LeSSE), um sistema de recuperação de informação que usa uma abordagem híbrida de técnicas de recuperação de informação semântica e sintática, baseado no sistema Quin criado por Samarinas et al. O LeSSE usa um algoritmo de recuperação de informação tradicional (BM25) em conjunto com o BERT de forma a auxiliar na pesquisa tradicional ao derivar e usar o valor semântico de uma query para fazer uma procura no corpus. Ao usar a semântica de uma query e do corpus, conseguimos criar uma ligação entre palavras e expressões sinónimas com diferentes grafias. Para atingir este propósito, o corpus de legislação teve que ser fragmentado em pequenos segmentos de texto, usando um algoritmo de parsing de texto que teve em conta a estrutura de cada documento. Explorámos o uso de diferentes conjuntos de dados de treino para os modelos de linguagem usados no sistema, até chegar a uma configuração que atingiu o melhor resultado, com uma precisão (accuracy) de 89% no corpus de legislação em que tínhamos anotações para treino, e 78.1% num outro corpus com poucas anotações de treino. O LeSSE está neste momento em produção, e está disponível ao público para efeitos de teste.

## Palavras Chave

Recuperação de Informação; Processamento de Língua Natural; Inteligência Artificial; Rede Neuronal; Domínio Legal; Lei do Consumidor em Portugal; BERT

# Contents

# List of Figures

# List of Tables

# Acronyms

| | |
|---|---|
| **AI** | Artificial Intelligence |
| **API** | Application Programming Interface |
| **BERT** | Bidirectional Encoder Representations from Transformers |
| **BiLSTM** | Bidirectional Long Short-Term Memory |
| **BM25** | Best Matching 25 |
| **BPTT** | Back-Propagation Through Time |
| **BrWaC** | Brazilian Web as Corpus |
| **CNN** | Convolutional Neural Network |
| **DAN** | Deep Averaging Network |
| **DNN** | Deep Neural Network |
| **DGC** | Direção-Geral do Consumidor |
| **DRE** | Diário da República Eletrónico |
| **Faiss** | Facebook AI Similarity Search |
| **FNN** | Feed-Forward Neural Network |
| **GPU** | Graphics Processing Unit |
| **ICT** | Inverse Cloze Task |
| **INCM** | Imprensa Nacional-Casa da Moeda |
| **INESC-ID** | Instituto de Engenharia de Sistemas e Computadores: Investigação e Desenvolvimento em Lisboa |
| **LeSSE** | Legal Semantic Search Engine |
| **LIR** | Legal Information Retrieval |
| **LSTM** | Long Short-Term Memory |
| **NER** | Named Entity Recognition |

| | |
|---|---|
| **NL** | Natural Language |
| **NLP** | Natural Language Processing |
| **NLTK** | Natural Language Toolkit |
| **PBT** | Population Based Training |
| **QA** | Question-Answer |
| **RNN** | Recurrent Neural Network |
| **SNLI** | Stanford Natural Language Inference |
| **STS** | Semantic Textual Similarity |
| **TPU** | Tensor Processing Unit |
| **USE** | Universal Sentence Encoder |
| **WMD** | Word Mover’s Distance |

# 1 Introduction

## Contents

The Official Portuguese Gazette (Diário da República) is tasked with the publication of all laws and norms of the Portuguese Republic. Currently, it is exclusively published electronically at Diário da República Eletrónico (DRE)[1] by the Imprensa Nacional-Casa da Moeda (INCM) as a public service that offers universal and free access to all of its content and functionalities. The DRE is composed of a vast set of publications, from which procedures, norms, applications and rules are derived. This online resource currently provides access to all of the Portuguese legislation, as well as services that allow citizens to find the norms and procedures that are inherent to their search.

A portion of these legislative articles belong to the Portuguese Consumer Law, also available at Direção-Geral do Consumidor (DGC)[2]. The articles in this section of the Gazette span across topics such as Insurances, Commercial Activities, Public Services, Health and Safety, Retirement, to name a few. It is, therefore, important that this type of information is easily accessible to all of the citizens given that there are rights they have and obligations they must abide.

## 1.1 Current Challenge

The current search methodology used in the search engine created for the Portuguese Consumer Law allows a search for legislation that is based on literal keyword search (articles are chosen according to a comparison between the literal keywords in their text and the ones that the user inputs as a search query) which poses some limitations on the accuracy of the results.

Citizens normally use Natural Language (NL) to introduce a search query. The problem is that sometimes the legislative articles contain language that — despite being technically NL — is not easily understood by regular users, due to its formality. The legislative documents often contain legislative jargon, which are technical terms of juridical nature that are not easily understood, and that can cause citizens to reach for professional help in issues that may not require it.

An apparent solution for this type of problem would involve manual classification of legal articles such as the creation of relevant keywords for each article. That way, the search query could be compared to those keywords in order to evaluate their similarity. However, the problem behind manual classification is that it would need to be done by legal professionals, and this process could take a lot of time — which obviously poses as a great obstacle in the classification of those texts and the subsequent extraction of meaning and knowledge from them — thus, slowing down the process of including them in a search result, something that is simply not possible given the urgency of including newly passed laws in the gazette.

There are currently more than a million and a half digital juridical documents and, every month, more than a thousand new ones are published. Only a small portion of these are considered to be a part of

[1] https://dre.pt/dre/home
[2] https://www.consumidor.gov.pt/

Consumer Law but the intention of applying a new search methodology to the Portuguese Consumer Law is then to expand it to the whole Portuguese gazette.

Another way to look at the problem would be to search for a connection between legal text (in formal language) and the user's query (in informal language), which would exclude the need of text classification and the inconveniences that accompany it. From this, we are left with an important question that we aim to answer in this thesis:

**How to create an information retrieval system capable of searching legislative documents using only natural language?**

## 1.2 Procedure

With this challenge in mind, our main goal was to engineer a system capable of searching through the Portuguese Consumer Law by providing a query in NL and returning a set of results, in the form of segments of text, with their corresponding information such as the title of the act and its article.

The first step would be organizing the law corpus in such a way that each document is fragmented into small segments of text, so that each of these can serve as a possible answer to a query. If more than one segment is needed to answer a query, then they could be combined into the search result. Once fragmented, the segments would need to be identified with the information pertaining to their location on an act, i.e., each segment needs to be accompanied by the divisions in which it is inserted, such as Chapter II, Section VIII, for instance.

This task could be viewed as a Legal Information Retrieval (LIR) problem, but more specifically as a Semantic Textual Similarity (STS) problem, since we are trying to find the semantic relation between queries and segments of the corpus. For this task, we receive a query in natural language and then we compare it to the existing segments. Those that best resemble the query in semantic value become the set of relevant results.

The system developed in this thesis was the product of a partnership between Instituto de Engenharia de Sistemas e Computadores: Investigação e Desenvolvimento em Lisboa (INESC-ID) and INCM in the context of the project *Descodificar a Legislação*.

## 1.3 Thesis Structure

The remainder of this thesis is organized in 6 chapters, starting with Chapter 2, in which we present all of the concepts that were essential during the development of our system, and which are in the understanding of our proposal. Next, we have Chapter 3, which will introduce current approaches to problems similar to ours and state-of-the-art concepts that were used or compared to the system we implemented. In Chapter 4 we describe how the law documents were fragmented into smaller segments and we also describe how the annotations dataset came to be. Chapter 5 presents the architecture of the developed system and goes into finer detail in each step of the pipeline. In Chapter 6, we analyse the performance of the system according to different factors and we compare it to a baseline information retrieval system. Finally, in Chapter 7, we have the conclusion in which we provide a brief summary of the work conducted, the milestones achieved and the improvements that could be made, in the future, to achieve a better performance of the system.

2

# Background

**Contents**

In this chapter, we will go through the essential concepts that will be needed in order to fully understand the related work and state-of-the-art to be described in the next chapter and the solution to our problem.

## 2.1 Semantic Textual Similarity

STS is a concept that deals with determining how similar two pieces of text are, on a semantic level — that is, if they mean the same or not. STS techniques are widely used nowadays in information retrieval, machine translation and sentiment analysis, among others. There are ways of representing the semantic value of a word or an expression and, in this section, we will present them, as well as a metric used to compare text based on their semantics.

### 2.1.1 Word and Sentence Embeddings

Given our problem of STS, it is especially relevant to address Word Embedding and Sentence Embedding, which are two very relevant techniques involved in the representation of words and sentences, respectively, in a semantic way.

Word Embedding is the process of transforming a written word into a feature vector. It can also be seen as the mapping of a word to a point in multi-dimensional space. These points can then be used to calculate their distance, which would be their similarity. Word embeddings are able to capture not only the syntax of the word but also its semantic value, given the context in which it is used. Word embedding is available in algorithms such as word2vec [6] and GloVe [7]. A simple example is present in Figure 2.1.

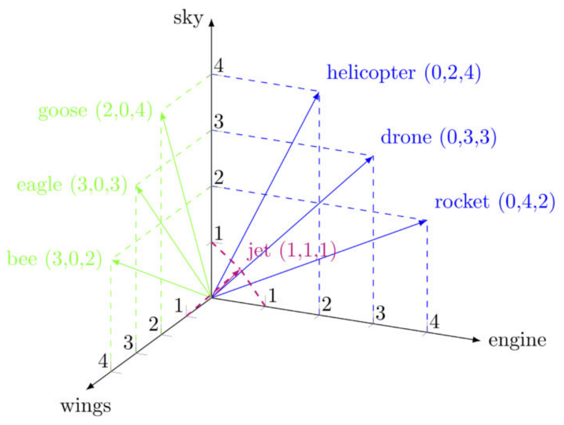


**Figure 2.1:** Simple Word Embeddings Example.[1]

[1] From https://corpling.hypotheses.org/495

Sentence embedding is, much like word embedding, the mapping of a written sentence to a numeric vector. There are multiple ways of embedding a sentence. The first that were implemented were the ones re-purposing word embedding methods, like the ones mentioned above.

Despite the fact that those techniques were not based to embed whole sentences, there are ways in which a sentence embedding can be deduced from the individual embeddings of the words in a sentence. The most common way is to average the word embeddings of all the words in a sentence — average pooling. When both sentences that we wish to semantically compare are embedded, we calculate a similarity metric to evaluate how semantically similar they are.

There are, however, other ways of improving this baseline approach. Some include averaging the word embeddings in a weighted form, according to their frequency in a document, for instance. Some exclude the embeddings of stop-words. And then there are other techniques that have shown better results — Word Mover's Distance (WMD), for instance, goes beyond the usual comparison and allows the semantic comparison of sentences that have no words in common, and yet, share the same meaning. It works by calculating a distance between two documents (or sentences acting as documents), that in other words is their degree of similarity. This distance is the minimum cumulative distance between the word embeddings of both documents. What also sets WMD apart from the previous algorithms is the fact that it accounts for not only the semantics of the sentences that are compared but also their syntax.

These great steps in the STS field all seem to fall into a crucial dilemma in the task of sentence embedding — they do not take into account the word order in a sentence. In this chapter we will explore an encoder, Bidirectional Encoder Representations from Transformers (BERT), that has state-of-the-art word embedding mechanisms and in Chapter 3, we will explore some other encoders, including sentence encoders, that have excelled at creating semantically meaningful embeddings.

### 2.1.2 Similarity Metric for Semantic Textual Similarity

In order to infer how semantically similar two pieces of text are, we need to define a metric by that we can use to quantify the similarity. One of the most commonly used metrics for determining similarity between embeddings is the Cosine Similarity. It is therefore able to serve as a semantic similarity metric, given that textual embeddings are able to encode the semantic value of its text.

Cosine Similarity is a metric that is used to compare two vectors on the basis of the cosine of their angular distance. It is derived from the formula of the euclidean inner product between two vectors $A$ and $B$, described in Equation (2.1).

$$A.B = ||A|| \cdot ||B|| \cdot \cos\theta \tag{2.1}$$

When we resolve by isolating $\cos\theta$, we have the cosine similarity, defined in Equation (2.2).

$$CosineSimilarity = \cos\theta = \frac{A.B}{||A|| \cdot ||B||} = \frac{\sum_{i=1}^{n} A_i \cdot B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2.2}$$

## 2.2 Neural Networks for Natural Language Processing

Neural Networks have shown to be an indispensable tool in various Natural Language Processing (NLP) tasks, such as the one highlighted in this thesis, but these have also evolved pretty quickly, and this evolution has followed an iteration progress, where each new and more sophisticated network builds upon the older ones. In this section, we intend to start with the most simple, older network types that build on top of each other to create the ultimate result that is what we used in our system — BERT.

### 2.2.1 Recurrent Neural Network

A Recurrent Neural Network (RNN) is a type of neural network that excels at processing sequences of data with its connective structure that allows information to be passed from the previous input to the current. RNNs can also be seen as multiple Feed-Forward Neural Network (FNN)s that share weights and flow information from one network to the other, when in truth they are simply one network in which the cells (nodes) loop over themselves for every input element that it receives.

These networks are commonly used for tasks that require pattern recognition in which elements in a sequence depend on each other's value.

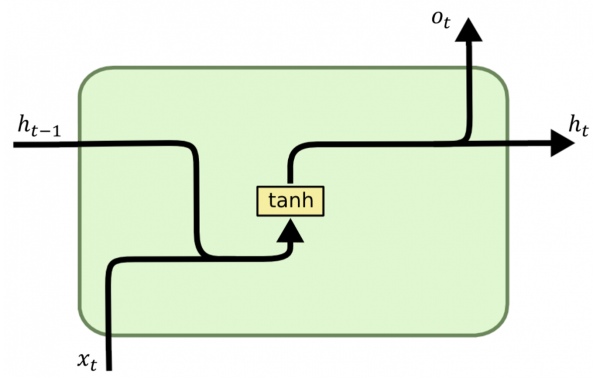


**Figure 2.2:** RNN cell.[2]

RNNs use a hidden state vector to pass information from the previous state to the next. The hidden state vector, $h_t$, depends on the previous hidden state, $h_{t-1}$, and the current input, $x_t$, and therefore holds information about the previous state (see Equation (2.3)).

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

$$h_t = f(W_{hh} \cdot h_{t-1} + W_{hx} \cdot x_t + b_h) \tag{2.3}$$

where $f$ is a non-linear activation function that is usually a hyperbolic tangent function, tanh, or a sigmoid function, $\sigma$.

The output vector (prediction), $o_t$, depends on the hidden state vector, $h_t$ (see Equation (2.4)).

$$o_t = g(W_{oh} \cdot h_t + b_o) \tag{2.4}$$

where $g$ is another activation function, normally $softmax$.

The weight matrices, $W$, are initialised randomly and adjusted using the error from the loss function. The added $b$ in each formula are biases.

In the training of a RNN, the weights and biases are adjusted through a process called Back-Propagation Through Time (BPTT) that is heavily based on the standard back-propagation method, in which the weight matrices are updated according to the loss function — the error between the predicted and actual observations.

In a summarized version of the algorithm, for each time-step we calculate the error. Those errors are then accumulated through the time-steps and a gradient of the loss function is calculated for each weight matrix (a partial derivative of the loss function with respect to the weight matrix). The gradient of each weight matrix is used to update the values of the matrix, according to a learning rate. When the error between prediction and observation is satisfied, we stop.

However useful these networks may be for some tasks, they pose an issue for others in which the values in a sequence have long-term dependencies (values far apart in a sequence depend on each other). This issues arises when the back-propagated errors start to diminish along the layers (tend to zero) or increase in an exponential way, which inhibits the weights from updating — Vanishing/Exploding Gradient Problem. This, in turn, renders the training useless and long-term dependencies get ignored or over-estimated in the process of prediction. In next word prediction, a word is predicted based on all of the previous words in a sentence, e.g., in the sentence, *Yesterday, I walked my dog*, when trying to predict the word *dog*, a network would have to consider all of the words that came before it. Nonetheless, the closest words tend to be more relevant for the prediction but that is not always the case. In some cases, words that are far apart in a sentence have a high dependency, such as the words *dog* and *bones* in the sentence *My dog is very smart and he likes bones*. These long-term dependencies are not easily learnt by RNNs.

This example was demonstrative only and does not represent the real challenge of long-term dependencies. In a real setup, long-term dependencies are between thousands of elements in a sequence.

### 2.2.2 Long Short-Term Memory

Long Short-Term Memory (LSTM) [8] is a special type of a RNN that employs a gating mechanism when flowing information between time-steps. Apart from the hidden state vector present in RNN cells, the LSTM cell is characterized by having an additional cell state vector that, at each time-step, can be read, written or reset — depending on the gate. (see Figure 2.3).

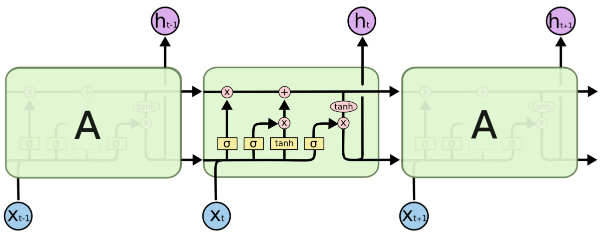


**Figure 2.3:** LSTM Structure.[3]

LSTMs overcome the Vanishing/Exploding Gradient Problem by allowing the errors to back-propagate unchanged, resulting in the update of long-term weights that need to exist in order to learn long-term dependencies.

LSTM networks work pretty similarly to standard RNNs with the exception of the structure of the cells. In a LSTM cell, there are three major gates — the forget gate, the input gate and the output gate (See Figure 2.4).

The gates are merely vectors, like the others in the structure, but they are called gates given that they control the amount of information that flows between cells at different time-steps.

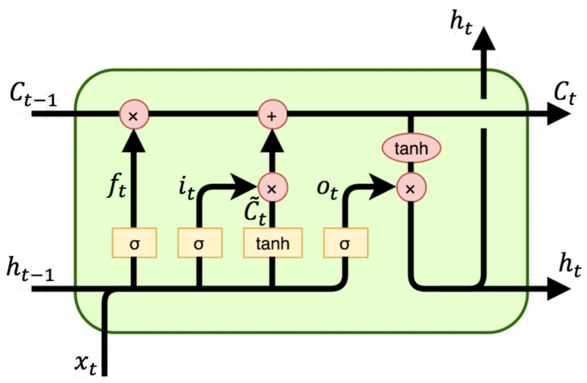


**Figure 2.4:** LSTM cell.[4]

The forget gate, $f_t$, is defined as:

[3] From http://colah.github.io/posts/2015-08-Understanding-LSTMs/
[4] From http://dprogrammer.org/rnn-lstm-gru

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.5}$$

The input gate, $i_t$, is given by:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.6}$$

The new cell state candidate vector, $\tilde{C}_t$, is given by:

$$\tilde{C}_t = \text{tanh}(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.7}$$

With these values, we are then able to calculate the cell state, $C_t$.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{2.8}$$

The forget gate, $f_t$, defines how much of the previous cell state, $C_{t-1}$, goes into the current one, and the input gate, $i_t$, determines how much the cell state candidate, $\tilde{C}_t$, should contribute to the cell state.

Ultimately, we have the output gate, $o_t$ which is responsible for controlling what parts of the cell state get to be transmitted to the next cell.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.9}$$

This output vector is then used to calculate the hidden state vector, $h_t$, which is transmitted to the next cell.

$$h_t = o_t \cdot \text{tanh}(C_t) \tag{2.10}$$

$W_x$ and $b_x$ with $x \in \{i, f, o, C\}$, are the weight matrix and bias vector of the input, forget, output and cell state vectors, respectively. The weights and biases will get adjusted during training. $h_{t-1}$ is the hidden state of the previous cell and $x_t$ is the input vector of the current cell.

All of the three major gates use a sigmoid function, $\sigma$, to push the values into the range of 0 to 1, in order to control what gets read, written and reset (0 nothing, 1 everything). The same reason applies to the use of the hyperbolic tangent function, tanh, in the computation of the hidden state vector and the cell state candidate vector, but in this case it is so that the values range between -1 and 1. The distribution of the gradients prevents the vanishing gradient problem.

The dimensions of the vectors are as follows:

$i_t, h_t, b_x, \tilde{C}_t, f_t, C_t, o_t \in \mathbb{R}^h$

$x_t \in \mathbb{R}^d$

$W_x \in \mathbb{R}^{h \times d}$

where $h$ is the number of input features (the size of the input vector $x_t$) and $d$ is the number of hidden units (size of the vector $h_t$).

### 2.2.3 Transformer

The main problem with LSTM networks is that they train very slowly — even slower than standard RNNs, given their complexity. The problem stems from the very long gradient paths in the BPTT process — the gradients have to be propagated all the way from the end to the start of the network and, when the input sequences are very long, this translates into a very deep network that is very slow to train.

Another problem is that the input of an LSTM network is serialized, i.e., the state of an LSTM cell depends on the state of the previous one, so the previous input has to be introduced first.

Considering that current computer processors thrive on parallel computation, the training process of an LSTM cannot make use of it given the restraint that was mentioned before. To overcome this necessity of training models on sequential data that make use of parallel computation, a team of researchers at Google came up with Transformer.

The Transformer [1] is a neural network architecture that relies on an Encoder-Decoder [9] structure that can process the input sequence in parallel, i.e., the input elements are processed at the same time. It can be structured into several components that are displayed in Figure 2.5.

The first component is the Input Embedder. This component takes in a sequence of words (query) and converts those words into scalar vectors. These vectors, however, do not have any context embedded into them. And, for that reason, an additional encoding component, Positional Encoder, is in place to add to the word embedding a context based encoding according to the position of the word in a sentence. After this process, the resulting word embedding with context is passed on to another component, the Encoder. This component has two layers — a Multi-Head Attention Layer and a Feed Forward Layer.

The Multi-Head Attention Layer (see Figure 2.6) uses an attention mechanism to assign different weights of importance to certain words in a sentence. To do this, it takes in:

- $Q \in \mathbb{R}^{w \times d_k}$, a matrix that contains the vector representations of the words in the query
- $K \in \mathbb{R}^{w \times d_k}$, a matrix that has the keys of the words in the query
- $V \in \mathbb{R}^{w \times d_v}$, a matrix that holds the values of the query words.

where $d_k$ is the dimension of each query vector, $d_v$ is the dimension of each value vector, $w$ is the number of words in the query and the attention scores of the query are calculated by the following function.

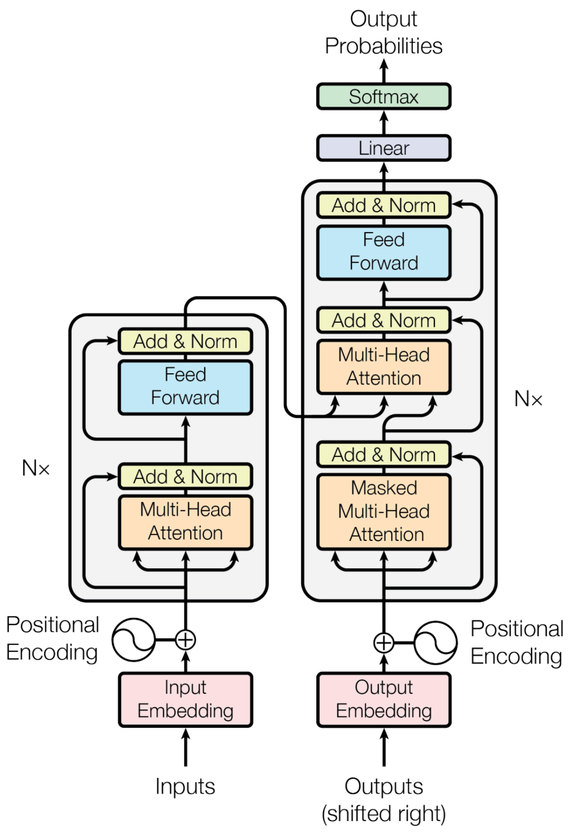


**Figure 2.5:** The Transformer Model Architecture. [1]

$$Attention(Q,K,V) = softmax(\frac{Q \cdot K^T}{\sqrt{d_k}}) \cdot V \tag{2.11}$$

where $K^T$ is the matrix $K$ transposed. The $softmax$ function is used to obtain a distribution of average weights ranging from 0 to 1 where the low scores get lower and the high scores get higher. This helps the model to increase the confidence in the words that have a higher score.

Then we concatenate all of the attention scores of all the words in the sequence and we multiply that by a weight matrix, $W^O$, to linearize the result.

$$MultiHead(Q,K,V) = Concat(head_1, ..., head_h) \cdot W^O \tag{2.12}$$

where:

$$head_i = Attention(Q \cdot W_i^Q, K \cdot W_i^K, V \cdot W_i^V) \tag{2.13}$$

$W_i^Q, W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{d_model}}$ are all linear projection weight matrices.

In the paper [1], $h$ is the number of parallel attention layer (heads) and it is 8, $d_{model}$ is the dimension of the output (512) and $d_k = d_v = \frac{d_{model}}{h} = 64$ .

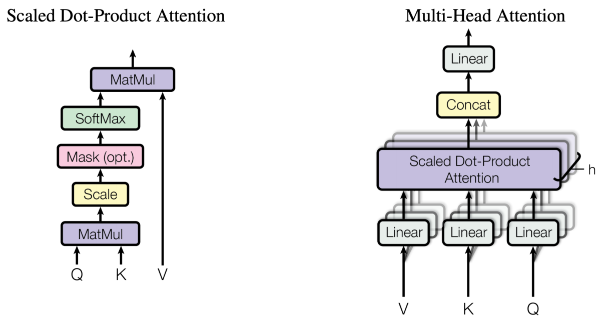


**Figure 2.6:** Scaled Dot-Product Attention (on the left) and Multi-Head Attention (on the right), decomposed. [1]

The reason why multiple attention vectors are used to determine the relevance of a word is because each word tends to overvalue itself among other words in the sequence, and so, to prevent that, those contributions get averaged out.

Each attention vector then gets sent to the Feed Forward Network Layer, where essentially they transform each vector so that they can be received by the next encoder/decoder. Since the attention layers are independent from each other, the vectors are able to transverse the Feed Forward Network in parallel, which means it is possible to pass a sequence of words through the network, all at once. This allows for parallel computation which, in turn, improves the speed of the model.

The output embedding and position encoding is identical to the input ones, which leaves a final component, the Decoder. This block is composed by three layers — a Masked Multi-Head Attention layer, a Multi-Head Attention layer and a Feed Forward layer. The last two are similar to their homologous in the Encoder Block, with a slight difference in the input of the Multi-Head Attention Layer, which will be discuss ahead. The real difference, however, is in the first layer, the Masked Multi-Head Attention. This layer is characterized by the use of a masking technique that prevents the existence of a bias towards a specific use-case of a sequence, i.e., we want to learn a possible output for a sequence of words, but not generalize that output for all similar sequences. This is done by resetting to zero all of the relation weights between words that are ahead of the prediction. By doing this, essentially it is creating a mask for the words that will not be used to predict a word in the sequence, by assigning them a null weight (no relevance).

After this step, the output (a masked attention vector) goes into another Multi-Head Attention Layer, similar to the one in the Encoder, and is used as the vector of values, whereas the query and keys are provided by the Encoder.

With every layer, an additional normalization step is executed in the following manner.

$$LayerNorm(x + SubLayer(x)) \tag{2.14}$$

where $SubLayer$ is the sub-layer function, such as the $MultiHead$, addressed above.

#### 2.2.3.A Pre-training, Transfer Learning and Fine-tuning

In order to achieve the best results in a task using a neural network, one must first train the neural network. This process can be done in multiple ways, but essentially it requires using a set of data that is passed to the neural network and, according to the training strategy, the weights in the layers of the network are adjusted to make it so that the network can perform better on the task that it is being trained on and with data similar or equal to the training data.

That said, there are essentially two types of training methods. The first, Pre-training, is the most basic type of training and it consists in training the network on a (typically) large dataset, starting with randomly initialized weights in all of the layers. This type of training usually takes the longest to execute and it requires a large learning rate to be able to make the most changes to a network that has never been trained.

In addition to pre-training, it is also possible to train a network that has already been pre-trained. This process is called Transfer Learning and it is simpler than the first since it involves freezing some layers of a pre-trained network and training the remaining with data adjusted to a different task than the one from the original dataset. In this type of training the learning rate used has to be smaller than the one used in the pre-training stage, since it could lead to a significant impact on the weights that are already adjusted if the learning rate selected is too high.

Fine-tuning is a type of Transfer Learning in which only the weights in the last layer of the network are changed, and this is usually a very fast procedure compared to pre-training, given that the amount of data used is usually much smaller and so is the learning rate.

#### 2.2.3.B BERT

Many systems have been engineered for the sole purpose of retrieving information from unstructured data. This retrieval might be applied to various types of information, including legal. BERT [2], is a language representation model that was built upon the transformer architecture and developed to help with such tasks.

BERT is different from other language representation models in the sense that it uses a pre-training technique that relies on bidirectional modeling, as opposed to a unidirectional modeling (left to right or right to left). Until BERT appeared, language representation models would usually train this way since the existence of a direction was what allowed the system to generate a probability distribution of the words.

When we're given the task of understanding textual data, such as a sentence, we subconsciously understand each word by looking at its surrounding words. That is the ultimate way that we have to know in which context it is being applied. This type of understanding is bidirectional. And so, given that language understanding is a bidirectional task for humans, it only makes sense that it should too for computers. However, it is not as straight-forward to put in motion.

The problem that emerges when applying bidirectional unsupervised training is the trivial prediction of words. In a unidirectional training, words can only see their left or right context. It is the existence of direction in the training that allows for an unbiased prediction of each next word. When that does not exist, as is the case in a bidirectional training, the prediction of each word — based on both its left and right words — becomes biased, and words are then able to predict themselves. (See Figure 2.7).

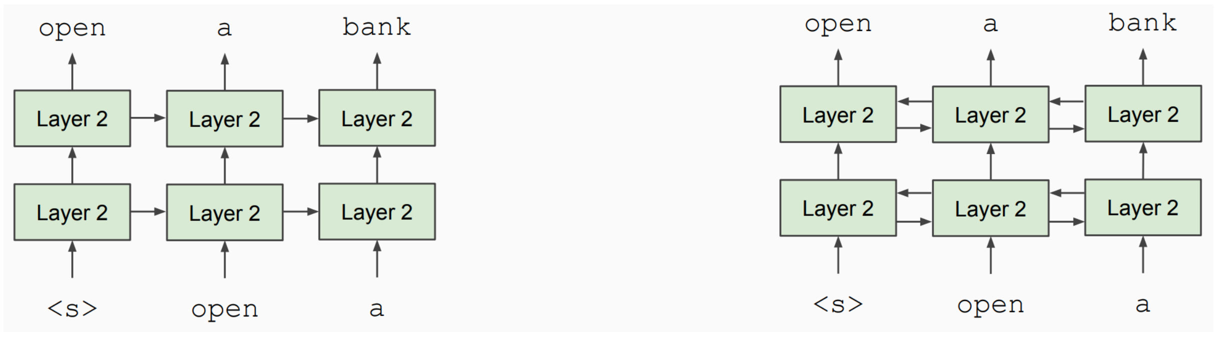


**Figure 2.7:** Unidirectional and Bidirectional training, on the left and right, respectively. [2]

BERT manages to treat this issue by using a masking technique. In a normal pre-training cycle, BERT (which relies on unsupervised training) goes through all of the unlabeled text from which it is learning from and, for 15% of the tokens in the text, it applies a mask (by replacing those tokens with a [MASK] token). Only these masked tokens are to be predicted.

By masking a small percentage of tokens, the inevitability of predicting a word that was there to see all along during training disappears. Since the prediction is only done on masked tokens, and not on the entire text input, it creates a mismatch between the pre-training and the fine-tuning, where there is no masking due to it being supervised training. To solve this issue, the masking strategy is not only focused on replacing tokens with a [MASK] token — this is done on 80% of the masked tokens (80% of 15% = 12% of the entire text). For another 10% of the tokens (1.5% of the entire text) they get replaced with another random token in the text, and in the remaining 10% (1.5%) it just leaves the original token unchanged. The purpose of this last percentage is to create a bias towards the actual observed word so that the system can actually learn the correct prediction.

The masking is applied to 15% of the tokens since this was the number that achieved a better performance. The problem with a small percentage of masked tokens is that the learning curve gets much bigger. On the other hand, a greater percentage of masked tokens results in a lack of context, making it harder to predict a word by its surroundings.

Another task for which BERT also pre-trains is the prediction of next sentences. This is especially important to determine the relation between two sentences. It becomes relevant in tasks such as Question Answering and Natural Language Inferencing.

In order to represent the words that the system receives as input, BERT creates word embeddings based on three components. The first component is the token embedding. This is the value that is assigned to the token, in a fixed vocabulary that BERT uses. The second, Segment Embedding, is simply an embedding of the segment to which the word belongs. These are not to be confused with the default sentence embeddings that BERT creates in the embedding for the [CLS] token that appears at the start of every sentence and collects information about the word tokens in the sentence by means of average pooling. Finally, we have the position embedding, which is precisely what its name states, an embedding of the word position in the input.

And so, the input embeddings are the sum of the token embeddings, the segment embeddings and the position embeddings. Figure 2.8 describes an example of the embedding of a sentence.

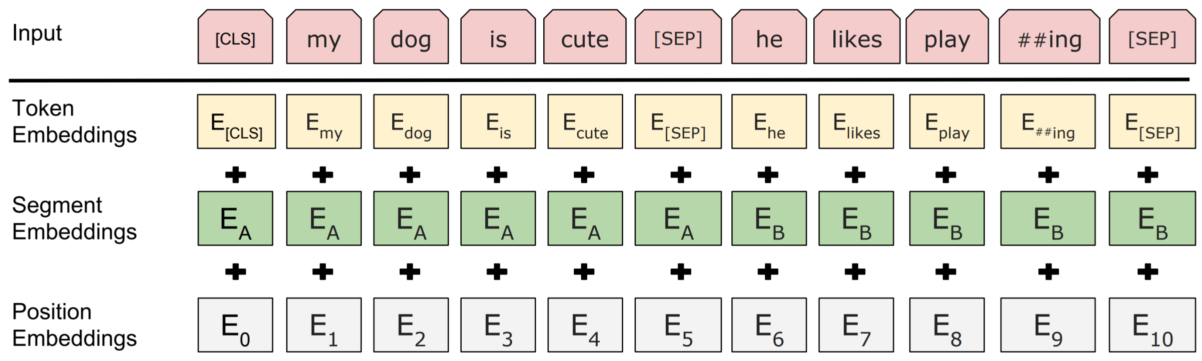

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ $E_{my}$ $E_{dog}$ $E_{is}$ $E_{cute}$ $E_{[SEP]}$ $E_{he}$ $E_{likes}$ $E_{play}$ $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
$E_A$ $E_A$ $E_A$ $E_A$ $E_A$ $E_A$ $E_B$ $E_B$ $E_B$ $E_B$ $E_B$
Position Embeddings
$E_0$ $E_1$ $E_2$ $E_3$ $E_4$ $E_5$ $E_6$ $E_7$ $E_8$ $E_9$ $E_{10}$

**Figure 2.8:** The three components of word embeddings - Token, Segment and Position. [2]

These embeddings are then fed to the neural network by the encoder, and its outcome is then received by the decoder.

### 2.2.3.C BERTimbau

BERTimbau [10] is a BERT model pretrained specifically for the Portuguese language, using a Brazilian Portuguese corpus, Brazilian Web as Corpus (BrWaC) [3], which is composed of 3.53 million documents extracted from pages in websites written in Brazilian Portuguese.

The corpus contains a highly diverse content, ensured by the amount of annotated categories from various sites (see Figure 2.9). The quality of the corpus comes from the pre-processing methodology used such as identification of URLs, removal of duplicated content and annotation of syntactic information.

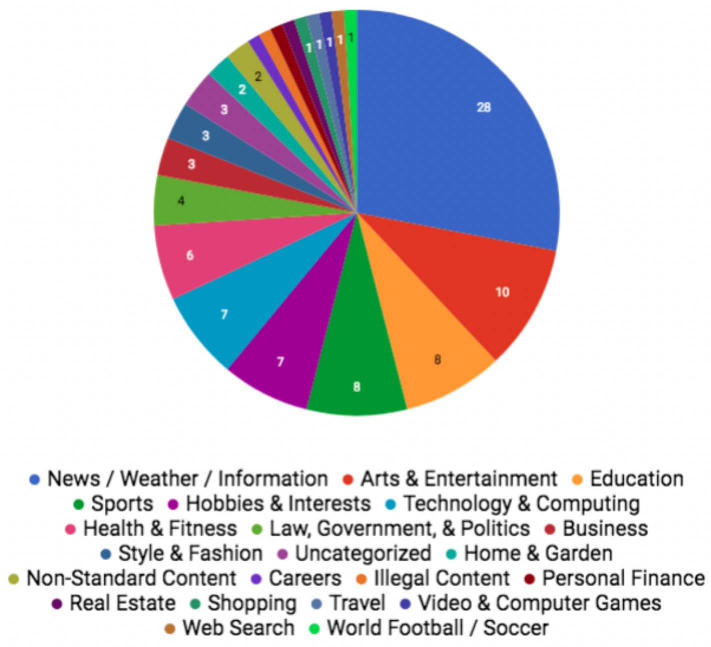


**Figure 2.9:** Annotated categories of the 100 most frequent websites in BrWaC. [3]

## 2.3 Inverse Cloze Task

Originally proposed by [11], Inverse Cloze Task (ICT) is a technique based on the standard Cloze task [12]. In the standard technique, the goal is to predict a masked piece of text out of the available context in a sentence. In the inverse approach, the goal is precisely to predict the context out of a sentence. This is done by training a model on a set of Question-Answer (QA) pairs that are generated from a large corpus, in which the question in each pair is a sentence in the corpus and the answer is the sentence that comes before or after, when available. So, as an example, we have the following excerpt:

(...) Zebras have four gaits: walk, trot, canter and gallop. They are generally slower than horses, but their great stamina helps them outrun predators. When chased, a zebra will zigzag from side to side. (...)

In Table 2.1 we can see the the pairs that are generated from this excerpt following an ICT training technique.

This training technique is especially helpful for training language models that will be used in the task of information retrieval since it can create semantic associations between sentences that would otherwise be ignored as semantic pairs by the generic pretraining done to the model.

**Table 2.1:** QA Pairs Example.

| Question | Answer |
|---|---|
| Zebras have four gaits: walk, trot, canter and gallop. | They are generally slower than horses, but their great stamina helps them outrun predators. |
| They are generally slower than horses, but their great stamina helps them outrun predators. | Zebras have four gaits: walk, trot, canter and gallop. |
| They are generally slower than horses, but their great stamina helps them outrun predators. | When chased, a zebra will zigzag from side to side |
| When chased, a zebra will zigzag from side to side | They are generally slower than horses, but their great stamina helps them outrun predators. |

# 3 Related Work on Information Retrieval

**Contents**

In this chapter, we expose some of the related work on the topic of information retrieval, as well as state-of-the-art approaches.

## 3.1 Information Retrieval

Information Retrieval is the procedure in which a system retrieves information from a collection of resources when given a requirement (usually an expression or a query). This task can be applied to numerous domains and is an important aspect of our day-to-day lives.

### 3.1.1 Traditional Approaches to Legal Information Retrieval

Legal Information Retrieval (LIR) is a specific type of Information Retrieval and, therefore, requires different approaches to the way the text is searched. Usually, legal documents are written in a very formal language but search queries written by regular citizens tend to be in a more informal language. This type of imbalance creates a mismatch of vocabulary that damages search if not attended to. The current LIR systems are mostly based on two techniques — Boolean Search and Manual Classification.

In Boolean Search, documents are scanned through in order to find an existing match in the search terms, according to the logical conditions imposed. A user may specify terms such as specific words or judgments by a specific court. These terms are combined with logic symbols such as AND, OR, NOT in order to provide a search on the content of the texts. They are widely implemented by services such as Westlaw[1], LexisNexis[2], and FindLaw[3], which are American legislation search services. This kind of search performs poorly when literal term matching is done for query processing, due to synonymy and ambivalence of words.

Manual classification is a technique that is used to overcome the limitations of Boolean searches. This technique relies on classifying case laws and statutes into computer structures. This structuring is done according to the way a legal professional would organize them and it also attempts to link the texts based on their subject or area. It allows the texts to be classified and it makes it easier to extract knowledge for a search. The reason why this technique may be considered unsustainable is because there is an increasing amount of legal texts and not enough legal professionals or time to classify them.

When we look back from LIR into the broad spectrum of Information Retrieval, other options appear. These options do not have a focus on the legal subject but rather on generic text documents. In spite of the particularities of legal documents, such as the connections between different documents, for instance, we can also look at legal articles as text documents — therefore expanding the field of research. When we do this, we are met with several other studied alternatives.

[1] https://legal.thomsonreuters.com/en/products/westlaw
[2] https://www.lexisnexis.com/en-us/gateway.page
[3] https://www.findlaw.com/

The Okapi Best Matching 25 (BM25) [13], or rather just BM25 is one of those alternatives and it is widely used as an information retrieval ranking algorithm. This algorithm is still used today by search engines to determine the relevance of entries to the searched query, along with TF-IDF (Term frequency - inverse document frequency), on which it relies, as we will see.

BM25 is a bag-of-words retrieval algorithm, which is defined by the representation of text as a set (or *bag*) of words while disregarding their syntax or context. The most popularized version of BM25, introduced in TREC 1994 (a conference on text retrieval) is the following:

Given a query $Q$, containing keywords $q_1, q_2, ..., q_n$, the BM25 score of a document $D$ is defined as:

$$score(D,Q) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{f(q_i,D) \cdot (k_1 + 1)}{f(q_i,D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{avgdl})} \tag{3.1}$$

where $f(q_i, D)$ is the term frequency of $q_i$ in the document $D$, $k_1$ and $b$ are optimization parameters, $|D|$ is the length of the document $D$ in words and $avgdl$ is the average document length in the set of documents. $IDF(q_i)$ is the inverse document frequency of the term $q_i$. It is used as a weight function and it is defined as:

$$IDF(q_i) = log\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} \tag{3.2}$$

where $N$ is the cardinality of the set of documents and $n(q_i)$ is the number of documents that carry the term $q_i$.

In a paper [14] released for the COLIEE workshop in 2019, this algorithm was used to retrieve legal information based on query search. In the third task (first task of statute law) of the competition, they used the BM25 function to derive the score of each document based on a searched query. It delivered promising results but was still not able to be set apart from other information retrieval techniques — their results were not largely separated.

Representation models are one of those alternative information retrieval techniques and their goal is to process information in the documents and queries in order to represent them in a different way. There are three main groups of representation models. The set-theoretic models are the ones that aim to represent documents and queries as sets of words. In this group we find the standard boolean model, which was mentioned previously in the context of legal information retrieval. It is one of the most simple and inexpensive representation models available.

Then we have the algebraic group of models, that represent documents and queries as embeddings with the objective of defining the similarity between documents and queries based on their vectors.

Finally, we have the probabilistic models where the similarities between documents and queries are given by probability of text usage and relevance. In this category we can fit language models, which are a very specific case that has been expanded given their importance in NLP tasks. We will approach systems of this kind in the next subsection.

## 3.2 Natural Language Processing Approaches to Semantic Textual Similarity

NLP is the subfield of Artificial Intelligence (AI) that is focused on the computation of human language. This includes their representation, analysis, processing and generation, in a digital form. This subfield has revealed to be quite relevant in the creation of new ways of interpreting and representing legal text in a way that normal citizens with no introduction to fundamental law concepts can understand and query certain laws by using natural language understanding. For this complex task, there are a few factors that weigh in the virtuous operation of a NL-based law query system.

The order of the words in a sentence is a very important factor in its semantic value. As we saw in the last section, the first few sentence encoders that were released - some that are essentially repurposed word encoders - have fallen into the same predicament. The lack of word order encoding.

Two sentences with the exact same words, but different ordering, can have equally different meanings. And the context defined by that order can influence the meaning of a specific word. For instance, when an individual says, *I went for a run*, we assume that the word *run* has a meaning related to the sport. When another individual says, *You have to run this on your computer*, the meaning is completely different from the other sentence. Not only is it being used as a verb, the meaning of it is no longer related to the sport, but rather to a computational task. And, even in a case where the word *run* was used as a verb connected to the sport, it could still be differentiated from the computational meaning due to the associated words, for most of the cases.

### 3.2.1 Universal Sentence Encoder

Google's Universal Sentence Encoder (USE) [15] was one of the language models that were created to mitigate the issue stated above, able to consider the order between words. There are two USE options available to use, each based on a different model.

**Deep Averaging Network (DAN):** The first, DAN, averages together the input embeddings for words and bi-grams and then passes them through a feed-forward Deep Neural Network (DNN) to produce the sentence embeddings. By considering bi-grams, it allows the representation of pairs of words in a sentence embedding.

**Transformer:** The second one, Transformer, constructs sentence embeddings using the encoding sub-graph of the Transformer architecture. It uses attention to compute context aware representations of words in a sentence. These representations are then converted to a fixed length sentence encoding vector.

**DAN vs Transformer:** This last model is much more complex than the first one and more resource demanding. It does have a slightly higher accuracy but the DAN model achieves a compute time that is linear to the length of the input sequence. This encoder is also available in a version pre-trained for Portuguese (Multilingual) [16] in the Transformer variant and in a reduced accuracy version that uses a Convolutional Neural Network (CNN).

### 3.2.2 InferSent

Another very popular sentence encoder model is InferSent [17]. This model has proven to be better at certain key aspects of sentence representations, when compared to USE. More specifically in terms of distinguishing between a sentence and its negation. USE, however, performs better when estimating the STS between embeddings and also seems to excel at detecting relevant word order differences.

### 3.2.3 Language Issue

There are plenty of state-of-the-art models that have been benchmarked in competitions but the majority of them, including some of the best performers, have only been pre-trained in English or Chinese datasets, as is the case with ERNIE 2.0 [18] and XLNet [19], with this last one only being pretrained for the English language. Since pre-training a language model of this calibre has computational and time limitations, it is simply unfeasible to pre-train these language models for Portuguese data.

Some other alternative models that are pre-trained for the Portuguese language include Multilingual BERT (M-BERT), which is essentially BERT (introduced in Chapter 2) pre-trained for multiple languages. BERT has an STS score slightly lower than the ones mentioned before, but still quite relevant. However, this difference in the STS score is due to the fact that BERT excels at generating word embeddings but not so much at their sentence counterparts.

The reason why this difference exists is because BERT is not trained for semantic sentence similarity, as opposed to Google's USE or InferSent models, so its sentence embeddings are not semantically meaningful to use in a scenario where we evaluate and compare the semantic value of two sentences.

However, there are ways to bypass this constraint. The most popular one, as we have seen in the previous section, is the embeddings averaging strategy. It consists of averaging the embeddings of the words in a sentence. Another alternative would be to use the token embedding of the special token [CLS] as the sentence embedding. This token appears at the start of a sentence and collects information about the whole sentence due to the bidirectional exchanges in the neural network.

### 3.2.4 Sentence-BERT

A team of researchers from Ubiquitous Knowledge Processing Lab (UKP) developed a system, Sentence-BERT [4], heavily based on the BERT model that has averaged promising results in all of the SemEval[4] editions from 2012 to 2016, especially when compared to BERT's averaged embeddings or CLS embeddings — and even when compared to USE and Infersent. (See Figure 3.1).

| Model | STS12 | STS13 | STS14 | STS15 | STS16 | STSb | SICK-R | Avg. |
|---|---|---|---|---|---|---|---|---|
| Avg. GloVe embeddings | 55.14 | 70.66 | 59.73 | 68.25 | 63.66 | 58.02 | 53.76 | 61.32 |
| Avg. BERT embeddings | 38.78 | 57.98 | 57.98 | 63.15 | 61.06 | 46.35 | 58.40 | 54.81 |
| BERT CLS-vector | 20.16 | 30.01 | 20.09 | 36.88 | 38.08 | 16.50 | 42.63 | 29.19 |
| InferSent - Glove | 52.86 | 66.75 | 62.15 | 72.77 | 66.87 | 68.03 | 65.65 | 65.01 |
| Universal Sentence Encoder | 64.49 | 67.80 | 64.61 | 76.83 | 73.18 | 74.92 | **76.69** | 71.22 |
| SBERT-NLI-base | 70.97 | 76.53 | 73.19 | 79.09 | 74.30 | 77.03 | 72.91 | 74.89 |
| SBERT-NLI-large | 72.27 | **78.46** | **74.90** | 80.99 | 76.25 | **79.23** | 73.75 | 76.55 |
| SRoBERTa-NLI-base | 71.54 | 72.49 | 70.80 | 78.74 | 73.69 | 77.77 | 74.46 | 74.21 |
| SRoBERTa-NLI-large | **74.53** | 77.00 | 73.18 | **81.85** | **76.82** | 79.10 | 74.29 | **76.68** |

**Figure 3.1:** Spearman rank correlation *c* between the cosine similarity of sentence representations and the gold labels (label that classifies two sentences on the basis of their similarity) for various STS tasks. Performance is reported by convention as *c* x 100. STS12-STS16: SemEval 2012-2016, STSb: STSbenchmark, SICK-R: SICK relatedness dataset. [4]

Sentence-BERT, or SBERT, is essentially a modified pre-trained BERT model that uses siamese and triplet network structures in order to extract sentence embeddings that are semantically relevant.

The system trains by encoding two sentences in the siamese way, i.e., running two identical networks adjusted with the same parameters on two different inputs — in this case two BERT networks receive each a sentence to encode. These encodings are then passed through a pooling process that the team behind SBERT found to have a better performance with a mean agreggation strategy, as opposed to a max or [CLS] vector strategy. Like we have seen previously, it consists of averaging the token embeddings of a sentence. That way, all of the token embeddings that BERT outputs are joined into one vector, consequently joining a layer of the size of the sequence of tokens into one output. Finally, both sentence encodings are then compared by calculating their cosine similarity. The whole process is depicted in the diagram in Figure 3.2.

The training is done using a combination of two datasets — the Stanford Natural Language Inference (SNLI) [20] and the Multi-Genre NLI [21]. The SNLI has a body of 570,000 sentence pairs annotated with the labels contradiction, entailment and neutral. MultiNLI contains 430,000 sentence pairs and covers a range of genres of spoken and written text.

When developing SBERT, the UKP team compared the performance of different metrics in the evaluation of each sentence pair similarity. They experimented with the most used one, cosine-similarity, but also tried to use negative Manhattan and Euclidean distances. After the experiments, they concluded

[4]an international workshop on semantic evaluation (http://alt.qcri.org/semeval2020/)

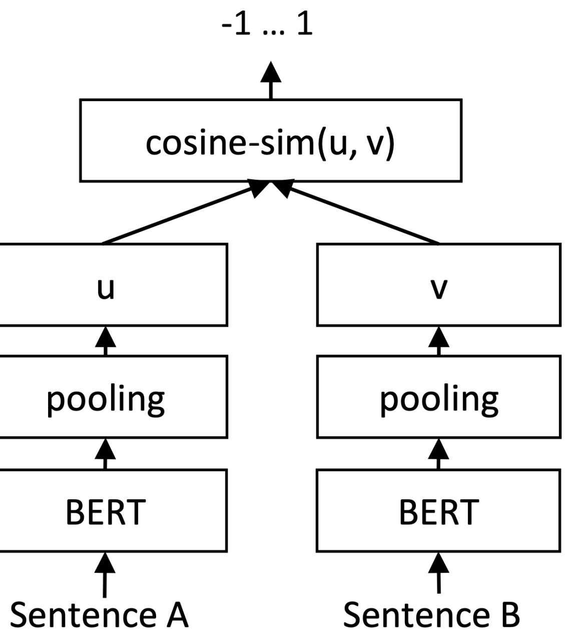


**Figure 3.2:** SBERT architecture for STS tasks. [4]

that the metric used, between the three, wasn't relevant given that the results were roughly the same. Therefore, they continued to use cosine-similarity as the metric of STS.

They also considered using a regression function that maps sentence embeddings to similarity scores but refrained from doing so given the resource exhaustion that would occur.

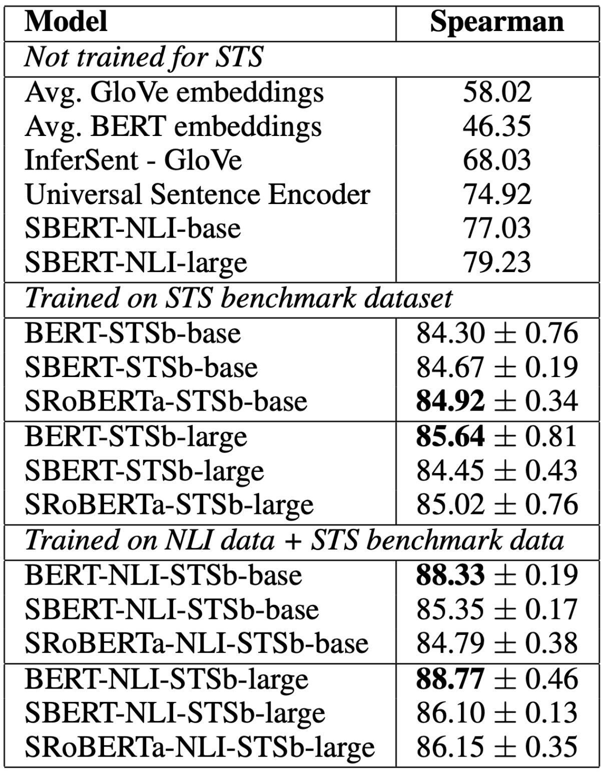
When comparing the performance of SBERT, in STS tasks, two different strategies of training were used — Unsupervised and Supervised Learning. For the unsupervised approach, SBERT only retained the knowledge that it had gained with the pre-training from BERT (based off of Wikipedia) and NLI data. To evaluate this system, three datasets were used — STS tasks 2012-2016[5], the STS benchmark [22] and the SICK-Relatedness [23] datasets. These three datasets include labels for sentence pairs that define, on a scale of 0 to 5, how semantically related they are. SBERT was able to outperform both InferSent and USE on most of the datasets, with the exception of the SICK-R dataset, in which USE gained an edge due to its training on a variety of diverse datasets that seemed to better fit the data in SICK-R. The results can be seen in Figure 3.1.

For the supervised learning, SBERT was fine-tuned on the training set of the STS benchmark dataset using cosine similarity as the metric for sentence embedding similarity alongside a mean squared error loss function to assess the quality of each prediction. This dataset has proven to be very popular in the evaluation of supervised datasets given the quality of the sentence pairs and its dimensions — it is composed of 8628 sentence pairs that are divided into three categories (*captions*, *news* and *forums*).

Apart from the fine-tuning done only on STSb, another experiment was done by training on the NLI

[5]http://alt.qcri.org/semeval2020/

dataset first and then the STSb. This latter option resulted in a considerable improvement. In the paper it was also found that using RoBERTa, a BERT based language model, (instead of BERT) did not make much difference in the final results. These findings are displayed in Figure 3.3.



| **Model** | **Spearman** |
|---|---|
| *Not trained for STS* | |
| Avg. GloVe embeddings | 58.02 |
| Avg. BERT embeddings | 46.35 |
| InferSent - GloVe | 68.03 |
| Universal Sentence Encoder | 74.92 |
| SBERT-NLI-base | 77.03 |
| SBERT-NLI-large | 79.23 |
| *Trained on STS benchmark dataset* | |
| BERT-STSb-base | 84.30 $\pm$ 0.76 |
| SBERT-STSb-base | 84.67 $\pm$ 0.19 |
| SRoBERTa-STSb-base | **84.92** $\pm$ 0.34 |
| BERT-STSb-large | **85.64** $\pm$ 0.81 |
| SBERT-STSb-large | 84.45 $\pm$ 0.43 |
| SRoBERTa-STSb-large | 85.02 $\pm$ 0.76 |
| *Trained on NLI data + STS benchmark data* | |
| BERT-NLI-STSb-base | **88.33** $\pm$ 0.19 |
| SBERT-NLI-STSb-base | 85.35 $\pm$ 0.17 |
| SRoBERTa-NLI-STSb-base | 84.79 $\pm$ 0.38 |
| BERT-NLI-STSb-large | **88.77** $\pm$ 0.46 |
| SBERT-NLI-STSb-large | 86.10 $\pm$ 0.13 |
| SRoBERTa-NLI-STSb-large | 86.15 $\pm$ 0.35 |

**Figure 3.3:** Evaluation on the STS benchmark test set. SBERT was fine-tuned on the STSb dataset, SBERT-NLI was pretrained on the NLI datasets, then fine-tuned on the STSb dataset. [4]

## 3.3 Quin

Quin [5] is a fact-checking system that was developed during the outbreak of COVID-19 with the purpose of providing the public with an automated fact-checking system capable of examining the veracity of claims surrounding the topic of COVID-19. It was later repurposed as a general fact-checking system, capable of verifying open-domain claims.

Quin works in three stages (see Figure 3.4) — in the first, the query goes through a BM25 sparse retriever, from which the top scoring 500 results are extracted. In the second stage, parallel to the first, the query is encoded using QR-BERT, a BERT model specifically designed to work in the context of question answering, trained with a large dataset constructed using **NLI!** (**NLI!**). After the query is encoded, it goes through a Facebook AI Similarity Search (Faiss) index to search for the passages that best resemble the query in semantic value, using a cosine similarity function to compare between embeddings, and the best 500 results are extracted from this stage.

The union set of the results from stages 1 and 2 are then used in the third and final stage, where these go through a relevance classifier, that is essentially a BERT model fine-tuned on a large dataset of query-passage pairs that applies a linear transformation to the embedding of the [CLS] token in order to retrieve a score from each query-result pair in the union set. These results are then reordered according to the score attributed by the relevance classifier, and this is the final order used to display the results in the search engine.

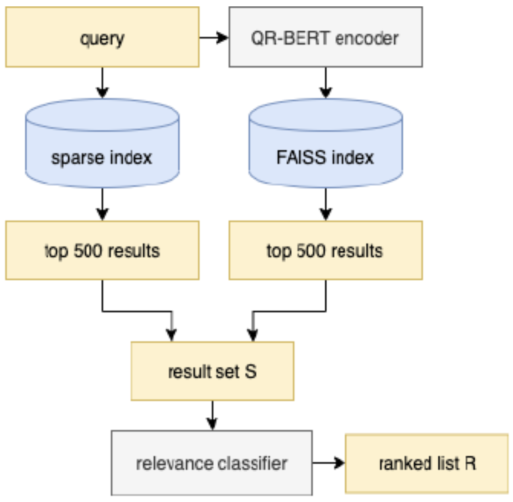


**Figure 3.4:** Diagram of the architecture of Quin. [5]

This system is very relevant to our problem, given that the motivation is somewhat analogous to ours, albeit in a different context, and it combines traditional information retrieval with a NLP-based approach that is capable of assisting a traditional keyword search with a semantic component, hence the reason we decided to base our system off of this approach.

# 4 A Legal Corpus for the Portuguese Consumer Law

**Contents**

In order to apply NL models based on DNN models such as the transformer-based ones, one important requirement is the existence of a dataset that can be used to train and validate the model.

So, one of the first challenges that was addressed in this thesis was the creation of a Legal Corpus for the Portuguese Consumer Law. In this chapter, we will describe the several steps that were needed to create this corpus.

First, we will describe how the initial set of documents were analyzed and how we decided to structure the corpus in terms of what is the minimal amount of information. Then, we will also address the creation of a separate corpus, made up of annotations that were created automatically and manually, and that were crucial in the training and evaluation of the system that we developed. The process in which these annotations were created will also be detailed.

## 4.1 Defining a Common Structure for Legal Acts and Legal Segments

The set of documents selected for this law corpus was selected with the help of DGC's team who provided us with a document that has all the law documents generally necessary to answer questions on the topics regarding consumer law.

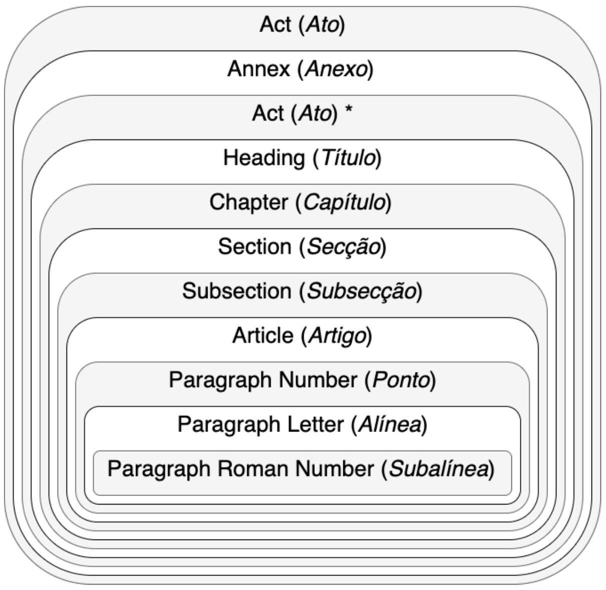
Those documents are extracted directly from the DRE database that contains all of the law documents existing in the Official Portuguese Gazette (*Diário da República*, in Portuguese).

After analysing and performing a short study on the corpus of law documents, we were able to conclude that, in average, a law document contains dozens of articles, and most of them surpass the character limit in the language model (BERT) that we are using in Legal Semantic Search Engine (LeSSE) (our search system, which will be addressed in further detail in Chapter 5). This language model can only take as input segments of 512 tokens[1], the corpus needed to be divided into smaller segments to accommodate for the segment size restriction. Therefore, each act (law document) was divided into several segments, which are generally paragraphs and each one of them has information about its location in the document.

When writing a new act, lawmakers use a Legistics Guide[2] which is used to write and structure an act according to a convention. In this guide, the advisable structure for an act is the one in Figure 4.1. An example of the aforementioned structure is present in Figure 4.2.

Inside an article exists a substructure composed of paragraph numbers, paragraph letters and paragraph roman numbers, as well as normal paragraphs which are called segments. These are organized

[1]a BERT token is generally smaller than a word — a word can often be divided into several tokens
[2]The most recent guide available to the general public can be found at
https://www.parlamento.pt/ArquivoDocumentacao/Documents/AR_Regras_Legistica.pdf



* This act is different from the parent one and is usually a republication of another act, therefore, it will never have any annexes.

**Figure 4.1:** Act Structure.

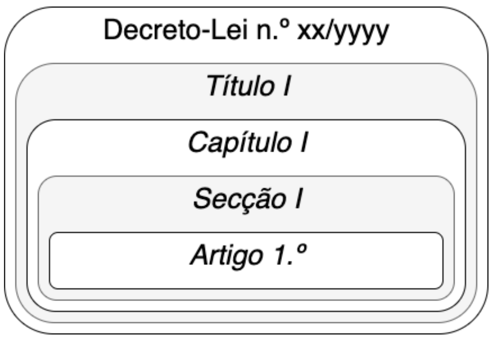


**Figure 4.2:** Act Structure Example.

according to their hierarchical order (i.e. segments are inside paragraph roman numbers, which go inside paragraph letters and these go inside paragraph numbers[3])

Since this structural information is necessary to locate an article, or a specific paragraph, it is also included in each segment's information. Therefore, each segment has the information relative to its act name, act id (which is a non repetitive id used in the DRE database, from where the acts are extracted), publishing date, ELI URL (a DRE.pt URL for the act, which is used to point to the full version of an act), summary of the act (it gives a brief description of the topics covered by the act), annex, heading, chapter, section, subsection, article, paragraph number, paragraph letter, paragraph roman number.

First, the extraction script checks if an act is out of force by using the *Vigência* attribute associated to it. When it is, the act is ignored and is not added to the corpus. When it is not, the act id is used to extract the act, in full, from the DRE database. After that, the script checks whether the act arrives fragmented (already separated into articles, by law experts) or not. If it is fragmented, then the separation into divisions is already done and, therefore, the only task that needs to be done is the segmentation of each article, i.e. breaking each article into multiple segments, along with its divisions. In case the act is not fragmented, it must first be separated into divisions, and then the text of each division gets segmented.

The process of segmenting the text of an act is very similar to a parsing task. The act is iterated line by line, and each line is compared to a set of titles to determine whether the line begins a division in the act (one of the mentioned above). In Table 4.1 we can find all of the possible regular expressions that trigger the start of a division, along with examples of those expressions. The script also checks if a line is part of the signature of the act (the ending of an act) and, in case it is, it discards those lines so that they are not used in training and in the search. After all of the parsing is done on the act, the segments are added to the corpus, stored in a file in text format.

**Table 4.1:** Act Division Triggers

| Division | Regular Expression | Example |
|---|---|---|
| Annex | ˆ(ANEXO—Anexo)( N.º \d+— [MDCLXVI]+)? | *ANEXO II* |
| Heading | ˆ(TÍTULO—Título) [MDCLXVI]+ | *Título VII* |
| Chapter | ˆ(CAPÍTULO—Capítulo) [MDCLXVI]+ | *CAPÍTULO I* |
| Section | ˆ(SECÇÃO—Secção) [MDCLXVI]+(-[A-Z])? | *Secção IV-A* |
| Subsection | ˆ(SUBSECÇÃO—Subsecção) [MDCLXVI]+ | *Subsecção XII* |
| Article | Artigo | *Artigo 2.º* |

The main difficulty in the development of the script was constructing all of the regular expressions that caught every trigger expression for the start of each division. This was a crucial task, given that without it, the structure of an act would be completely ruined.

As of writing this document, the Consumer Law corpus is composed by 237 different acts, and the segmentation task resulted in a total of around 70,200 segments (roughly 296 segments per act, in

[3]Note: Some of these divisions could be omitted, such as paragraph letters and roman numbers, for instance.

average).

An additional corpus was added, in the context of the Retirement Law (*Estatuto da Aposentação*) and the acts in that corpus were segmented in the same way. This corpus was added to see how well the system was able to work with a different corpus, in a different context. Being a smaller corpus, with only 2 acts, the segmentation resulted in 583 segments (around 291 segments per act, in average).

## 4.2 Manual Annotations

In order to evaluate the performance of the system and compare the results of the differently fine-tuned systems, it was important to have a dataset of golden labels for QA pairs, manually annotated by experts in Portuguese law.

The first step to construct this dataset was to put together a set of queries for annotation. We did this by extracting search queries from the DRE search logs, which we had access given by INCM. The search logs included the queries inserted into the DRE search engine by real users, as well as the name of the act that the user accessed at the end of the search. Given the information in the search logs, we proceeded to filter the search queries based on the act that the user accessed, with the help of the initial consumer law corpus that we had, i.e., the name of each act in a search log was compared to the names of the acts in the initial corpus. If the name of the act in the search log was included in the corpus, then it meant that the query would probably be relevant to the consumer law domain. These queries were then pre-processed in order to filter out any mistakes, or irrelevant terms.

The next step was annotating the collected queries and, for this task, we had the help of Helder Santos, a jurist at INCM, Ana Catarina Fonseca and Ana Filipa Claro, from DGC, whom we sent the set of questions and, in return, we received the answers to those questions in the form of law annotations (references to specific passages in the corpus). These annotations contain information about the location of the passage in the corpus — this information refers to the name of the act and the divisions inside it (annex, heading, chapter, etc). In the end of the annotation process, we received 321 annotated queries pointing to a total of 175.544 segments of the corpus, which means that each query would point to 546 segments, in average. The queries-to-segments ratio is very low, and this is explained by the fact that, for the most part, a query points to several segments (e.g. a query could be annotated with an article, and that article could point to 10 segments of the corpus), and that explains why a single query could point to over 500 segments.

An example of an annotation would be:

**Query:** *portabilidade operador*

**Annotation:** *Decreto-Lei n.º 56/2010, Artigo 4.º*

Since *Artigo 4.º* of *Decreto-Lei n.º 56/2010* only contains one line, this annotation would generate only one pair in the dataset:

**Query:** *portabilidade operador*

**Segment:** *O período de fidelização não pode ter duração superior a 24 meses.*

## 4.3 Automatic Annotations

Because the manually constructed annotation dataset was insufficient to cover all of the search subjects, two additional datasets were created to ensure that the model was fine-tuned on data that covered all of the topics in the corpus.

### 4.3.1 ICT Dataset

The ICT dataset was created using a technique with the same name mentioned in Chapter 2, but slightly altered to account for the type of training that needed to be done. In the original article, the ICT dataset was used to pre-train the model but, in this particular case, the dataset was only used to fine-tune the model. Since the original purpose was to train the model from the start, the authors needed diversified data to generalize better in the predictions, but in a fine-tuning task, the purpose is to adapt the model to a particular task or dataset. In this case we are fine-tuning the model on a dataset where the answer (in the QA pair) should always be a full segment, in order to assign every question to a line in the corpus — as every answer to the automatically generated questions should be in the corpus. Whereas in the original ICT technique the dataset was built by pairing each sentence in a line with the rest of the sentences, for this dataset it was essential that the answer should not be the remaining sentences in the line, but the whole line itself.

Another key difference between the original technique and this one, is that, because the lines of the corpus are mostly sentences, they need to be separated into portions by dividing them using commas (,) and semi-colons (;).

The line from the corpus:

*É na sequência da mencionada disposição legal que se torna agora necessário estabelecer os requisitos específicos da instalação, classificação e funcionamento daqueles empreendimentos turísticos para que, mediante o seu cumprimento, possam ser classificados numa das categorias previstas.*

would give way to 4 QA pairs, where the questions would be:

*É na sequência da mencionada disposição legal que se torna agora necessário estabelecer os requisitos específicos da instalação*

*classificação e funcionamento daqueles empreendimentos turísticos para que*

*mediante o seu cumprimento*

*possam ser classificados numa das categorias previstas.*

And the answer to form a pair with each of these questions would be the original line. Using this technique, we were able to generate 211.794 question-answer pairs.

### 4.3.2 Semantic Pairs Dataset

The other dataset was built by another masters student, as the main work of her thesis [24], and it required creating a dataset of QA pairs, automatically generated from the corpus.

The system is composed of three models — each one of those models corresponds to a Nested Named Entity Recognition (NER) task and is responsible for predicting a different set of labels (norms, semantic roles and named-entities).

The model architecture is the same for the three models and is based on the Named Entity Recognition as Dependency Parsing [25] — it consists of a BERTimbau (Chapter 2) model used for generating embeddings, followed by a Bidirectional Long Short-Term Memory (BiLSTM) to generate a contextualized representation of each word. These are then fed to two separate FNN, one to generate the representation of all the possible spans of text at the start of the entity, and the other to generate the same but for the spans of text at the end of the entity. This improves the prediction accuracy since the context at the start and end of the entity name are different. Lastly, the two representations go through a classifier model to generate the scores for all possible spans that could constitute a named entity. These are then ranked by their category score and after filtering according to some constraints, the final named entities are used to generate and identify QA pairs.

The system goes through two stages. In the first stage, it predicts the norms in the lines of the corpus, and then for those predicted norms, it predicts the remaining labels (semantic roles and mentioned entities) involved. In the second stage, QA pairs are generated from those detected labels.

For the following segment:

*1 - As pousadas devem cumprir os requisitos fixados no anexo i da presente portaria: a) Para a atribuição da categoria de 4 estrelas, quando instaladas em edifícios classificados como de interesse nacional ou de interesse público;*

The model would detect this segment as an obligation (semantic role) and generate the following questions from it:

*Que deve As pousadas fazer quando instaladas em edifícios classificados como de interesse nacional ou de interesse público ?*

*Que deve As pousadas fazer ?*

*Quando deve As pousadas cumprir os requisitos fixados no anexo i da presente portaria ?*

*Quem deve cumprir os requisitos fixados no anexo i da presente portaria quando instaladas em edifícios classificados como de interesse nacional ou de interesse público ?*

*Quem deve cumprir os requisitos fixados no anexo i da presente portaria ?*

Each of these question would, in turn, be paired with the original segment as the answer. Following this approach, we were able to generate 57.839 question-answer pairs.

# 5 Legal Semantic Search Engine

**Contents**

The goal of this thesis is to introduce a system that merges a common document retrieval technique with semantic search abilities on the Portuguese consumer law. In this chapter, we will start by looking at LeSSE, a search engine we specifically created to answer questions on the topic of Portuguese consumer law that uses state-of-the-art search techniques, and is based on Quin [5].

This system was developed for this thesis in the context of the project *Descodificar a Legislação*, a research collaboration between INESC-ID and INCM. The goal of this project was to make the Consumer Law more accessible and understandable by the Portuguese citizens. This was done by combining popular document retrieval techniques with the recent advancement of Machine Learning and NLP to provide semantic search capabilities to the search system.

This chapter starts with an overview description of the system, that will broadly explain the pipeline step by step. Then, we will address each system component in a more detailed way, as well as the training process for the language model used in some of the components. Ultimately, we have a description of how the results are selected, organized and presented to the user.

## 5.1 System Overview

Our system takes a hybrid approach, in which it combines a traditional (syntactic) information retrieval algorithm with a semantic search. The syntactic search allows the users to search for literal terms, such as names or titles included in the legislation, and the semantic search assists in case the answer contains juridical jargon that the user did not use in the query. The semantic search is able to identify synonymous words and expressions that the user may be interested in. A visual description of the system architecture can be found in Figure 5.1.

First, the system starts by pre-processing all of the law documents (acts) before any search is to be initiated. This will save time and resource exhaustion since all of the acts will be used in each search, therefore avoiding repetitive computations, given that they are always stored in the database. The search query will also undergo pre-processing, but since the queries will be inevitably different each time a search is performed, it is done during search time.

The pre-processing is the same for the acts and the query with the exception of one step — the acts go through text segmentation first, in which the text of each act is separated into segments, which are pieces of text separated by line breaks, and each one of them contains information about its location in the document (e.g. chapter 3, section 2, article 24) so that they can be later referenced in the results.

After that, both the segments and the query go through Semantic and Syntactic Pre-Processing — two distinct text processing steps that are needed to prepare the segments and the query for the Semantic Extraction and Syntactic Similarity, respectively. It is worth mentioning that the pre-processing is done at different stages for the segments and the query. The segments are pre-processed when the

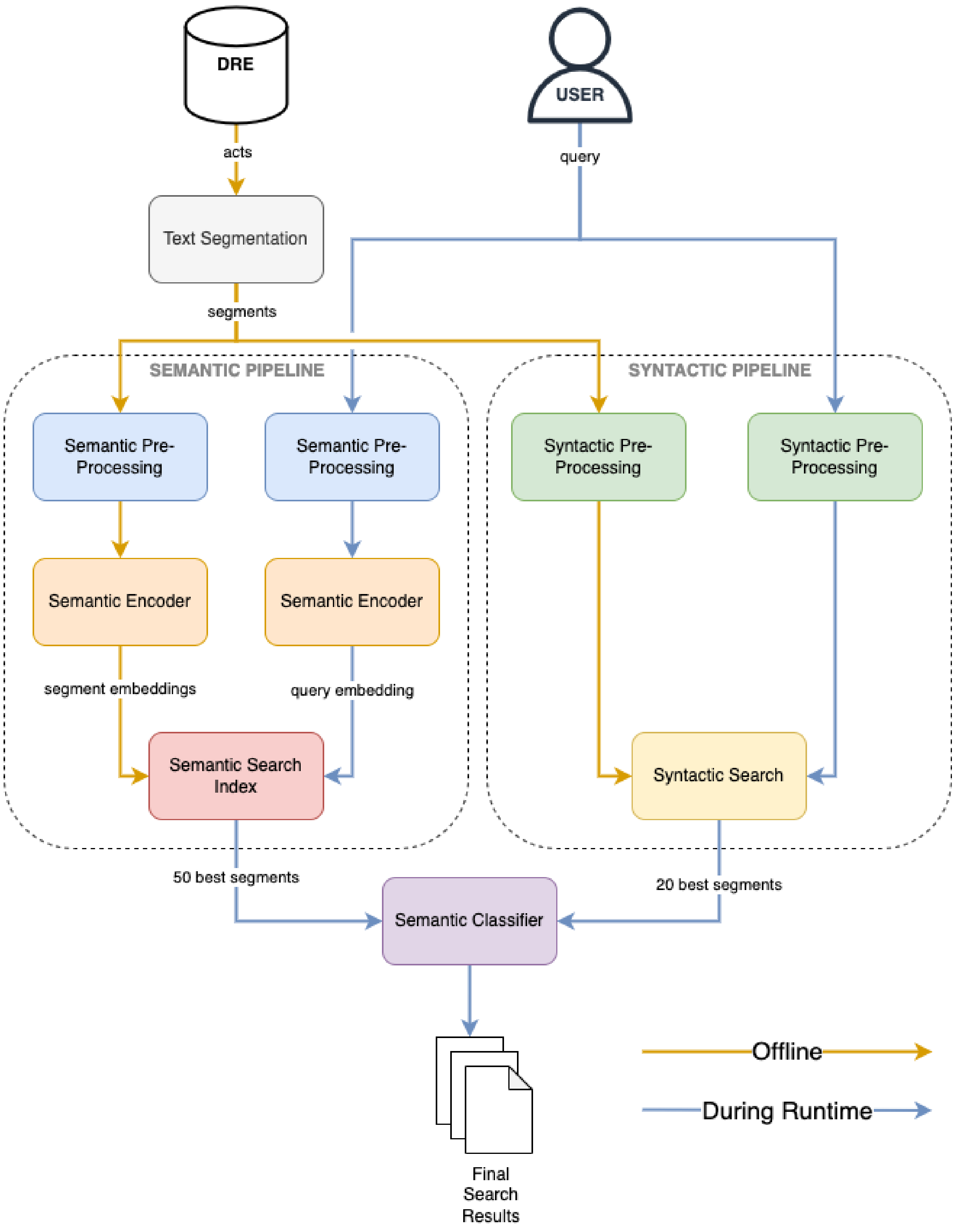


**Figure 5.1:** LeSSE

system is started and the query is pre-processed during runtime, right after it is inserted into the search engine.

After the semantic pre-processing is complete, we move on to the semantic extraction, which is the step in which the segments and the query are encoded into embeddings.

With the segment embeddings that were generated, a search index is created and stored locally in the server that hosts the search service. This allows the similarity search to be more efficient, and given the persistent nature of the corpus, the index won't need to suffer changes as long as the corpus does not change.

Additionally, before the system is up and running, the segments are used to create a structure that holds each word present in the segments (every word in the corpus), that is used by the syntactic similarity algorithm (BM25) to determine a syntactic similarity score for each pair of segment-query.

Once all of this is complete, the system is then ready to receive a query. In search time, the query goes through the syntactic pre-processing and is then used to calculate a similarity score for each segment. The 20 segments with the highest scores are selected. Consecutively, the query also goes through the semantic pre-processing, before being converted into an embedding, that is afterwards used to perform a search in the index that was created with the segments. The search index returns the scores of semantic similarity between each segment and the query, based on the similarity of their corresponding embeddings, and, at that point, the 50 segments with the best scores are selected.

In the final stage, the 50 best segments from the semantic search index and the 20 best segments from the syntactic similarity search are then united into a single set, but since the scores from the syntactic and semantic searches are on different scales and are not comparable, the results are then passed on to a trained semantic similarity model that assigns each result pair with a score that signifies its similarity to the initial query. Ultimately, the results are ordered according to these scores and returned to the user.

## 5.2 Semantic Pipeline

### 5.2.1 Semantic Pre-Processing

As opposed to the pre-processing done for the syntactic search, the one that is used for the semantic search does not involve any of the steps mentioned prior, but instead rests on the Bert Tokenizer from the Hugging Face[1] library. This process has been detailed in Chapter 2.

[1] https://huggingface.co

### 5.2.2 Semantic Encoder

After the semantic pre-processing, we are left with segments (and a query) that are prepared to be received by the Semantic Encoder, which will then generate segment embeddings off of those segments.

Semantic Encoder is, in its core, BERTimbau [10], a BERT model trained on the BrWaC [3], a large corpus that was constructed using the Brazilian Portuguese Web as a source. For the purpose of simplifying the fine-tuning and evaluation, we have used BERTimbau Base (BERT-Base) with hidden size 768.

BERT-Large, with hidden size 1024, is known to present better results than BERT-Base, but since fine-tuning it requires more computational power, BERT-Base came as the best choice in terms of time, performance and computational limitations.

In this step, the ultimate goal is to create a segment embedding, but since BERT only generates token embeddings we will be using the average embedding strategy to generate a segment embedding out of all the tokens in it. Therefore, once BERTimbau finishes embedding the tokens in the segment, all of the token embeddings are then averaged into a single segment embedding array.

Once this has been done, the embedding array is then normalized. This is a requirement for the Faiss [26] index that we are using and it also facilitates the selection of a fixed threshold for the maximum cosine distance between arrays, since these distances will then be comprised between 0 and 1 after the arrays are normalized.

For the purpose of adjusting the Semantic Encoder to the vocabulary used in the law documents and the queries (European Portuguese, formal language in the law documents and informal — sometimes formal — in the queries), we had to fine-tune the model in order to teach it to recognize popular semantic pairs between queries and law segments. To do this we used the Trainer[2] class from the Hugging Face library, providing a training and evaluation loop for PyTorch, optimized for Hugging Face Transformers classes.

To train the Semantic Encoder we used the Manual Annotations dataset (Section 4.2) on a task of sequence classification. The model was trained for 1 epoch, at an initial learning rate of 7.40546e-05, weight decay 0.244911, with a training batch size of 32 per GPU (a total of 64, considering the training was done on two GPUs).

Since we are using the Base version of BERTimbau in the Semantic Encoder, with hidden size 768, the generated segment embeddings are feature vectors with 768 dimensions (features).

After the creation of all of the segment embeddings, they are then stored in a Faiss search index for future use (every time a search query is received).

The reason why we opted for BERTimbau as opposed to M-BERT is because of the way these models were trained. M-BERT was trained for multiple languages, including Portuguese, but in a much

[2]https://huggingface.co/transformers/main_classes/trainer.html#transformers.Trainer

more modest way compared to other languages since the corpus is proportional to the available source material in each language. And since M-BERT needed to be trained in various languages, the trade-off was between number of languages included and the size of the training corpus for each language. Given that M-BERT was specifically designed to be pre-trained in multiple languages, its performance cannot be compared to the one from a model that was specifically trained in one language solely.

### 5.2.3 Semantic Search Index

After every segment in the corpus is encoded, they are added to a Faiss Search Index. The Index being used is the IndexFlatIP — providing an exact search for inner product. Since there is no index that provides an exact search based on the cosine similarity of the arrays, we chose the inner product considering that the cosine similarity is simply the inner product between normalized vectors — and that is why all of the embeddings are normalized before entering the index, in the final stage of the Semantic Encoder.

The Faiss library possesses two functions that easily allow us to convert the index in memory to a binary file when it is created (write_index) and read that binary file and bring the index back to memory upon the initialization of the search system (read_index). This is convenient given that the creation of the index is a task that takes quite a few minutes (20-30) to complete, and would be a waste of time to repeat it every time the system initializes.

After the query is encoded into an embedding, it is then used to search for the 50 arrays that are closest to it (in cosine similarity). We then use a threshold of 0.5 to eliminate any array that had a score below significant — these arrays represent the segments that are not relevant to the query that was searched. The remaining arrays — the ones with a score above 0.5 — are the ones that represent segments that are relevant to the query, and these segments will go to the latter stage in the system — the Semantic Classifier.

## 5.3 Syntactic Pipeline

### 5.3.1 Syntactic Pre-Processing

Before forwarding the query to the syntactic similarity search, the query needs to be processed in order for it to be recognized even when it is not written in the same way that it exists in the dictionary, due to various concerns.

The syntactic similarity search that is used in the system is the BM25 algorithm, which uses a bag-of-words strategy. Essentially, it means we need to divide a phrase into words. The words belonging to the query are therefore compared with the words in the dictionary (corpus/segments) and, for two words to be considered the same, they must share every Unicode[3] character in the same order.

In the syntactic similarity search, the words of each segment and query are used to calculate a similarity score. For two words to be considered the same they have to be be exactly the same — that is, they must share every unicode character in the same order. When two words are compared by the algorithm, one starting with upper case, and the other with lower case, they are deemed unequal.

Because of this constraint, the syntactic search must follow a syntactic pre-processing that consists of five main steps:

**Natural Language Toolkit (NLTK) Word Tokenization:** Using the NLTK word tokenization tool for the Portuguese language, each segment and query is separated into tokens, which are words defined in the NLTK Portuguese dictionary. This helps us construct the bag-of-words necessary for the BM25 algorithm, and is especially helpful in recognizing a word that is separated by a hyphen, for instance, which would have been viewed as two single words by a simple space-separator.

**Removal of Punctuation:** Since the previous step does not remove punctuation, every punctuation character is removed so that it does not count as a word. Otherwise, the segments and the query could be compared using, for instance, the number of commas that they contain, which is not our goal.

**Word Lowering:** Words that contain uppercase letters are lower-cased so they can be identified as being the same word. This ensures that when the user submits a search query that mistakenly includes an uppercase letter in the wrong place, the words in the query can be identified as belonging to the corpus.

**Stop-Word Removal:** In order to perform a relevant keyword search on the corpus, the segments of the corpus and the query need to go through a process of stop-word removal since, by definition, they offer no additional meaning to the sentences. To do this, we use the NLTK Portuguese stop-word dictionary to remove all of the irrelevant words. In this corpus of stop-words, there exist words such as: *a*, *ao*, *aos*, *aquela*, *aquelas*, among many others. The process of removal is quite simple and it involves keeping only the words in the segments/query that are not present in the corpus of stop-words. This a major step in the text processing since words such as *de* appear a lot in the Portuguese language, and the similarity between two phrases (a segment and the query) should not be judged by their amount of *de*s, which provide no contextual information about the phrase.

**Unidecode:** This last step ensures that every word in the segments and the query do not have any special characters such as *é*, *à*, *ç*, among others. It also ensures that the algorithm is able to recognize every word that the user typed with a missing accent. Every word goes through the function unidecode from the Unidecode[4] library.

[3]https://home.unicode.org
[4]https://github.com/avian2/unidecode

### 5.3.2 Syntactic Search

Following the Syntactic Pre-Processing, the processed segments and the query will go to the Syntactic Search, where the query will be compared to the segments, on a syntactic level. For this purpose, we perform a BM25 text search using the BM25Okapi[5] class from the rank_bm25 python library but, before any search is done, at the initialization stage of the search system, the BM25Okapi object is initialized using the corpus, i.e., all of the segments in the corpus that have been pre-processed. Since it only takes a few seconds, there is no need to store the binary object in memory and it is done during loading time, every time the system loads.

## 5.4 Semantic Classifier

At this stage, the segments that were chosen from the Semantic and the Syntactic Searches are collected into a set of segments that will be reordered by the Semantic Classifier, which is none less than a BERTimbau model trained in the same way that the one in Semantic Encoder is fine-tuned. However, their purposes are separate — the Semantic Encoder uses a base BertModel class to generate the embedding for each sentence and then they are compared in the search index based on their cosine similarity whereas the Semantic Classifier uses a BertForSequenceClassification class and the semantic similarity score is calculated by applying a softmax function to the logits that are returned by the model, upon receiving both sentences (segment and query) as input.

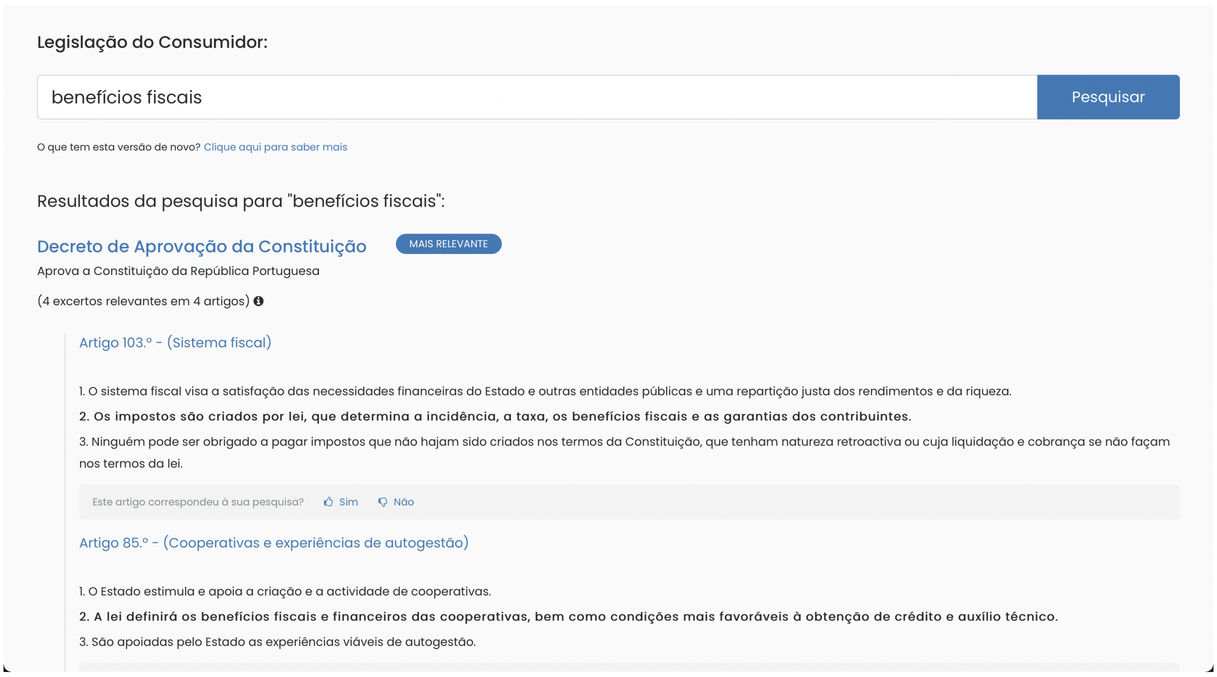
The Encoder + Search Index combination is used at a stage where comparison speed needs to be high, since we need to compare the query to every single segment in the corpus. At a later stage, in the Semantic Classifier where, at most, we have 70 segments, we are allowed to use a BertForSequenceClassification model to compare the query to the final segments and classify them in the order of relevance.

## 5.5 Results Selection and Presentation

The final results are shown in the following manner: The acts are ordered according to the sum of the scores of its segments, in a descending order. Thus, the act with the highest sum of its segments' scores is at the top of the results list. In each act, the ordering of the articles follows the same strategy — the articles in each act are ordered according to the sum of its segments' scores.

In Figure 5.2 we have an example of the results interface where we searched for the query terms "benefícios fiscais" and it returned an act in first place named Decreto de Aprovação da Constituição, and inside it two articles, Artigo 103º and Artigo 85º.

[5]https://pypi.org/project/rank-bm25/


Legislação do Consumidor:

benefícios fiscais

Pesquisar

O que tem esta versão de novo? Clique aqui para saber mais

Resultados da pesquisa para "benefícios fiscais":

Decreto de Aprovação da Constituição MAIS RELEVANTE

Aprova a Constituição da República Portuguesa

(4 excertos relevantes em 4 artigos)

Artigo 103.º - (Sistema fiscal)

1. O sistema fiscal visa a satisfação das necessidades financeiras do Estado e outras entidades públicas e uma repartição justa dos rendimentos e da riqueza.

**2. Os impostos são criados por lei, que determina a incidência, a taxa, os benefícios fiscais e as garantias dos contribuintes.**

3. Ninguém pode ser obrigado a pagar impostos que não hajam sido criados nos termos da Constituição, que tenham natureza retroactiva ou cuja liquidação e cobrança se não façam nos termos da lei.

Este artigo correspondeu à sua pesquisa? Sim Não

Artigo 85.º - (Cooperativas e experiências de autogestão)

1. O Estado estimula e apoia a criação e a actividade de cooperativas.

**2. A lei definirá os benefícios fiscais e financeiros das cooperativas, bem como condições mais favoráveis à obtenção de crédito e auxílio técnico.**

3. São apoiadas pelo Estado as experiências viáveis de autogestão.

**Figure 5.2:** Search Results Example

## 5.6 Model Training

As previously stated, the language model that was used for both the Semantic Classifier and Semantic Encoder was a BERTimbau base model fine-tuned on the annotations dataset. For this fine-tuning task, we were able to use a machine with 2 NVIDIA GeForce RTX 3090 Graphics Processing Unit (GPU)s, each with 24GB of memory and 10496 cores.

Prior to training, we ran a hyper-parameter optimization in order to optimize the learning capabilities of the model and its performance on unseen data. We did this by using a function in the Trainer class from the HuggingFace python library, which we also used to train the model. This class provides an Application Programming Interface (API) for feature-complete training in PyTorch, which simplifies much of the process and abstracts the researcher from the training loop, focusing on the optimization process. The API also supports distributed training on multiple GPUs/Tensor Processing Unit (TPU)s, which was beneficial to us since we used 2 GPUs instead of just one, and we would like them to be used in parallel to guarantee that the training process finishes faster.

### 5.6.1 Hyperparameter Optimization Techniques

In practice, hyperparameter optimization is nothing but a search for the set of parameters that optimize the training process and maximize (or minimize) the validation metrics, which in our case are the validation accuracy and loss. Usually, hyperparameter optimization can be done using various techniques, but the three most commonly used ones are Hand Tuning, Random (or Grid) search and Bayesian search.

These three techniques, however, possess two downsides. The first downside is the time it takes to find the hyperparameters that optimize the performance of the model. With Hand Tuning, the researcher must try a set of hyper parameters by training the model with that set and then evaluate the performance of the model. The researcher must keep this cycle until they are satisfied with the performance of the model. This task can take a lot of time, and can sometimes even take weeks or months to reach the perfect set of hyper parameters.

Bayesian Optimization is a technique that automates the hand tuning process by using the Bayes theorem to make changes to the hyper parameters at the end of each training run, which makes it a sequential training optimization, and therefore it can also become a very slow process.

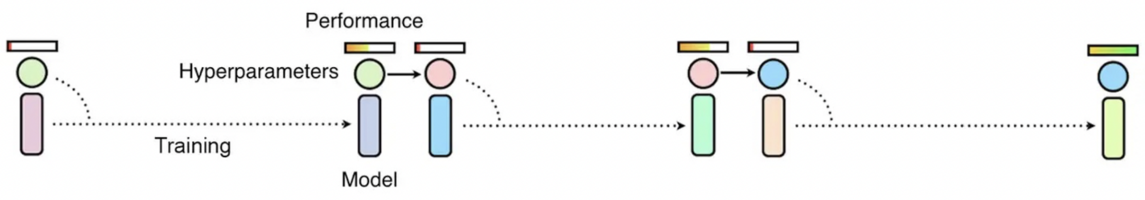


**Figure 5.3:** Bayesian Optimization

In Grid Search, a set of neural networks are trained independently in parallel and, at the end, the hyperparameters from the model with the highest performance are selected. The number of different combinations of hyperparameters can go around the dozens or even hundreds, which means that dozens or hundreds of models will be trained and only a small fraction of those will have a high performance. This, in turn, means that most of the models that were trained will not be used and, therefore, a high amount of computer resources were wasted training them — making this the second downside to using one of the most commonly used optimization techniques.

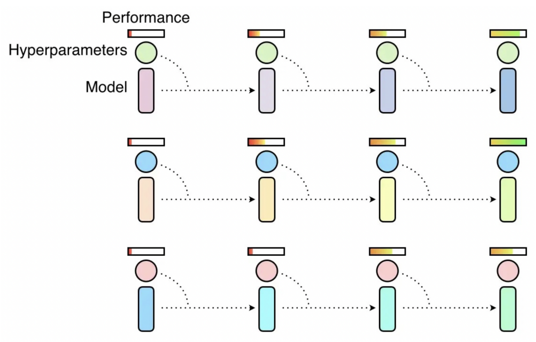


**Figure 5.4:** Grid Search

Population Based Training (PBT) [27] is an optimization technique that combines Grid search and Hand Tuning. PBT starts in the same way of Grid Search — by training a set of models with random hyperparameter values — but instead of training the set of models independently, it allows the models to share information with each other and reconfigure their training hyperparameters according to the most promising models.

Given these advantages, we decided to optimize the hyperparameters by using a PBT algorithm that was available as a scheduler option with the same name, in the Ray Tune library, that is integrated into the Trainer class function hyperparameter_search.

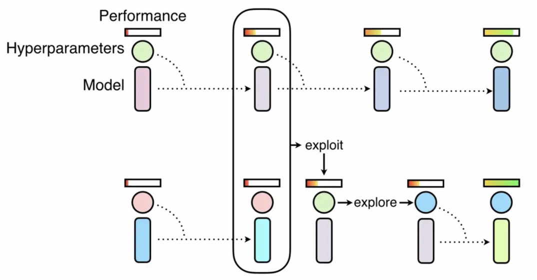


**Figure 5.5:** PBT

### 5.6.2 Model Optimization and Training

In order to find out which type of data would train the model in the best way possible for this task, we studied the use of 4 datasets for the Consumer Law Corpus — Manual Annotations, ICT, Semantic Pairs and Manual Annotations + Semantic Pairs — and for the Retirement Corpus we used 2 datasets, Manual Annotations and Semantic Pairs.

To choose which training dataset would return the best results we followed the same evaluation procedure for all of them. We first divided the datasets into 3: The training set, used to train the model; the validation set, that was used to evaluate the performance of the model during training; and the test set, which was used to evaluate the performance of the model on unseen data (different from the validation set), after the training is complete.

The Manual Annotations dataset was divided into 2 subsets — training and validation/test. Since we did not have many golden labels, we decided to split the dataset cautiously, prioritizing the validation/test set given that these sets had to have manually annotated queries, in order to check the real performance of the system. The validation and test sets are the same one given the same reason (not enough manual annotations). Instead of using percentages to split the dataset, we used the number of queries in the dataset — 100 for the validation/test sets, and the remaining (221) for the training set. We chose this query-split approach as opposed to a traditional percentage-split approach, given that, if we had chosen to divide to set according to traditional percentages, such as 80-10-10 (training-validation-test), we would have only had 32 queries to test the performance of the system, which is just not enough, given that there are plenty of topics in the consumer law domain and, with 32 queries, the probability of covering all of those topics is very low.

The other datasets (ICT and Semantic Pairs) were also divided into 2 subsets — training and validation — but in this case following percentages, 80% and 20%, respectively. The reason why we didn't divide these datasets into 3 was because we only needed one test set to test all the models, and we chose the annotations dataset because that was the most reliable since it included manual annotations created by experts, therefore the veracity of those annotations is guaranteed, as opposed to the ones automatically generated, which might not be correct.

Before the division, every dataset is shuffled in order to guarantee that every training batch is representative of the dataset, as a whole. If we did not shuffle the data, we would risk creating batches that have similar data, which would set off the gradient estimate and, therefore, lead the training of the model in the wrong direction.

Then, we ran a hyperparameter optimization for each dataset separately. This step was necessary due to the scale and diversity of data present in each dataset. A selection of parameters and their model performance are displayed in Table 5.1. The final training hyperparameters were chosen according to the accuracy, while monitoring the loss value which is calculated with a Cross Entropy loss function. This function is defined as:

$$loss = -w_y \cdot \log \frac{\exp(x_y)}{\sum_{c=1}^{C} \exp(x_c)} \cdot y \tag{5.1}$$

where $x$ is the input, $y$ is the target, $w$ is the weight and $C$ is the number of classes, which in our case is 2 (0 — not relevant, and 1 — relevant).

**Table 5.1:** Example of Training Hyperparameters for Manual Annotations of the Consumer Law Corpus

| Weight Decay | Learning Rate | Training Batch Size Per GPU | Epochs | Validation Accuracy | Validation Loss |
|---|---|---|---|---|---|
| 0.261884 | 2.42913e-05 | 32 | 4 | 0.837746 | 0.934985 |
| 0.244911 | 7.40546e-05 | 32 | 1 | 0.843621 | 0.634938 |
| 0.204092 | 9.25682e-05 | 16 | 1 | 0.84328 | 0.638671 |
| 0.261884 | 1.61942e-05 | 32 | 4 | 0.83131 | 1.09068 |
| 0.0596527 | 9.41399e-05 | 16 | 1 | 0.818426 | 0.500694 |

# 6 System Evaluation

**Contents**

In this chapter, we analyze the results upon evaluating LeSSE on different datasets and we discuss the influence held by certain factors on the performance of each test configuration.

## 6.1 Performance of LeSSE in the Portuguese Consumer Law

The primary purpose of the evaluation was to compare the performance of LeSSE in the Portuguese Consumer Law corpus to the baseline, BM25. After choosing the best model from each training session with the different datasets, we compared their performance on the same test set. In the following table, we present the accuracy results for the different combinations of search algorithms and training datasets used. The accuracy is divided into 4 categories (TOP 1, 3, 5 and 12) — each TOP $x$ category represents the percentage of test queries that the system got right in the first $x$ results. And so, for instance, this means that the Baseline system was able to fetch at least one of the correct results, in the first 3 search results, in 70.0% of the test queries (TOP 3/Baseline). The accuracy is measured by comparing the act in the result with the act in the golden label.

The third column shows the results obtained with the LeSSE system without fine-tuning (training) the model. Despite the fact that these results are not the best, this option manages to achieve a better result than the baseline. The fourth column shows the performance of LeSSE when the models were trained using the Manual Annotations dataset, and these are the best results achieved, especially when comparing with the baseline.

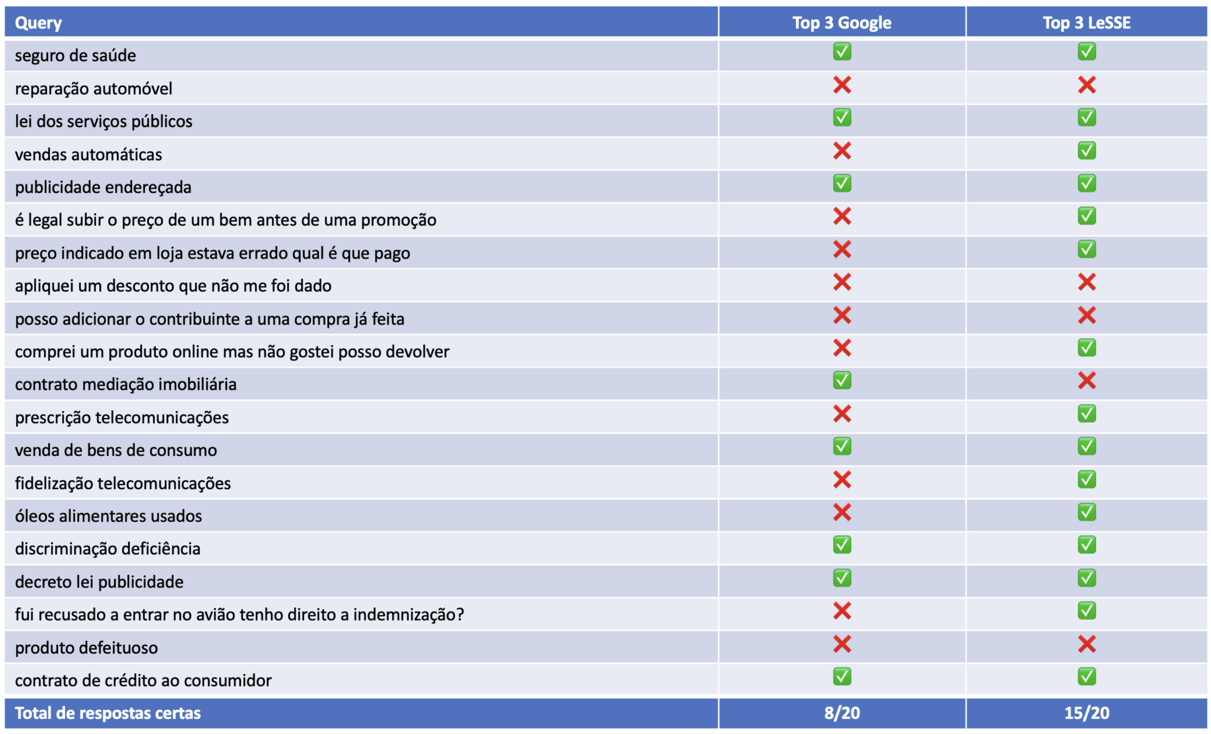
**Table 6.1:** LeSSE Accuracy in the Portuguese Consumer Law

| Accuracy (%) | | | |
|---|---|---|---|
| Results Measure | Baseline (BM25) | LeSSE | |
| | | No Training | Trained with Manual Annotations |
| TOP 1 | 42.0 | 44.0 | 55.0 |
| TOP 3 | 70.0 | 74.0 | 89.0 |
| TOP 5 | 71.0 | 88.0 | 96.0 |
| TOP 12 | 75.0 | 95.0 | 99.0 |

The reason why all the accuracy percentages in Table 6.1 are whole numbers is because the test set from the consumer law corpus had 100 query-answer pairs. So, for instance, this would mean that the baseline search algorithm (BM25) got the first result right in 42 out of the 100 test queries (TOP 1, Baseline (BM25)).

After comparing the different iterations of the system for the Consumer Law context, we chose the best configuration based on the TOP 3 measure, which was deemed more relevant in the search task.

After comparing the system to the baseline and its iterations, we ran a small test to evaluate the performance of Google on the search of some queries and then we ran those same queries on our system, with the full pipeline, and fine-tuned on the manual annotations dataset. The results from this experiment, as well as the queries that were tested in both search systems, are described in Figure 6.1.



| Query | Top 3 Google | Top 3 LeSSE |
|---|---|---|
| seguro de saúde | ✓ | ✓ |
| reparação automóvel | ✗ | ✗ |
| lei dos serviços públicos | ✓ | ✓ |
| vendas automáticas | ✗ | ✓ |
| publicidade endereçada | ✓ | ✓ |
| é legal subir o preço de um bem antes de uma promoção | ✗ | ✓ |
| preço indicado em loja estava errado qual é que pago | ✗ | ✓ |
| apliquei um desconto que não me foi dado | ✗ | ✗ |
| posso adicionar o contribuinte a uma compra já feita | ✗ | ✗ |
| comprei um produto online mas não gostei posso devolver | ✗ | ✓ |
| contrato mediação imobiliária | ✓ | ✗ |
| prescrição telecomunicações | ✗ | ✓ |
| venda de bens de consumo | ✓ | ✓ |
| fidelização telecomunicações | ✗ | ✓ |
| óleos alimentares usados | ✗ | ✓ |
| discriminação deficiência | ✓ | ✓ |
| decreto lei publicidade | ✓ | ✓ |
| fui recusado a entrar no avião tenho direito a indemnização? | ✗ | ✓ |
| produto defeituoso | ✗ | ✗ |
| contrato de crédito ao consumidor | ✓ | ✓ |
| **Total de respostas certas** | **8/20** | **15/20** |

**Figure 6.1:** Google vs. LeSSE

This was an important experiment since it showed that LeSSE is clearly more apt to answer queries on the topic of the Portuguese Consumer Law than Google, the most used search engine in Portugal, which is also used to search DRE documents. Google was only able to find 8 results out of 20 of the TOP 3 search results, while LeSSE was able to find 15 out of those same 20, which is a significant outcome. In order to restrict the search domain of Google, we added the term ”dre” before every query.

## 6.2 Performance of LeSSE in the Absence of Manual Annotations

The second evaluation sought to answer whether the LeSSE would be ready to answer questions correctly when there would be no manual annotations or jurists to annotate them. This was a relevant question to pose, since there may come a scenario where there are no manual annotations available for a specific domain in the Portuguese Law, as it is the case now for any domain other than the ones we have worked with — Consumer Law and Retirement Law. To prevent this issue, we studied the option of using automatically generated annotations, by following two generation techniques — ICT and Semantic Pairs generation. Additionally, we also tried a mixed configuration of Manual Annotations and Automatic Annotations (Semantic Pairs) to see whether it improved upon the Manual Annotations configuration. And so, upon training LeSSE with these automatically generated datasets, we compared their performance against the Baseline.

**Table 6.2:** Testing Accuracy with Automatic Annotation Datasets

| Accuracy (%) | | | | | | |
|---|---|---|---|---|---|---|
| Results Measure | Baseline (BM25) | LeSSE | | | | |
| | | No Training | Trained with Manual Annotations | Trained with Automatic Annotations | | |
| | | | | ICT Dataset | Semantic Pairs | Manual Annotations + Semantic Pairs |
| TOP 1 | 42.0 | 44.0 | 55.0 | 45.0 | 51.0 | 50.0 |
| TOP 3 | 70.0 | 74.0 | 89.0 | 76.0 | 77.0 | 88.0 |
| TOP 5 | 71.0 | 88.0 | 96.0 | 84.0 | 90.0 | 96.0 |
| TOP 12 | 75.0 | 95.0 | 99.0 | 95.0 | 95.0 | 98.0 |

In Table 6.2, the fifth column presents the accuracy results of LeSSE when trained with the ICT dataset. Despite not being the best in the bunch, it managed to surpass LeSSE when no fine-tuning was done (when comparing TOP3), and it also showed a significant improvement over the baseline. This means that it could be a good option when there are no manual annotations to fine-tune on, since it is relatively fast and inexpensive to generate.

The sixth column contemplates the results from training with Semantic Pairs, automatically generated from the corpus. When comparing with the manual annotations dataset in the TOP3 category, training with this dataset did not improve the performance of the model, but it did perform better than LeSSE when trained with the ICT dataset. It also showed to be far superior the baseline, and even scored higher than LeSSE with no training, which was indicative that it could be useful in a scenario where there are no annotations. The generation of this dataset was the product of a master thesis from another student, and it required the help from two linguists that annotated the segments with syntactic and semantic functions of the words and expressions, so its generation was slower and more expensive than the ICT one.

Mixing the manual annotations with the semantic pairs did not show an improvement over the manual annotations, and we concluded that it had to do with the incongruousness of the data — the manual annotations were quite different from the automatic ones in terms of topics covered, but also in format (the manual ones were more naturally written than the automatic ones), and that created an inconsistent dataset which was not as fit for training as the manual annotations themselves. Another issue with this mixing strategy comes from the fact that the datasets are not balanced in terms of quantity — the automatic annotations far surpassed the manual ones, since those were easier to generate, and that created an imbalance. However, it still managed to score higher than all the other configurations in the TOP3 category (apart from manual annotations), so it could prove to be quite useful when there are not enough manual annotations to cover all of the topics in the domain, or the quantity of annotations is just not enough.

## 6.3 Performance of LeSSE in a Different Law Domain

In addition to the Consumer Law domain, another challenge was defined in the *Descodificar a Legislação* project, which was to make the developed system work with another law corpus (another law domain). This task functioned as a test to see how well LeSSE would work when dealing with another domain or subdomain of the Portuguese Law. For this purpose, we were provided with a few annotations on the domain of Retirement Law (Estatuto da Aposentação) and applied the same methodology that we had done for the Consumer Law corpus. The annotation dataset consisted of 111 queries pointing to 298 segments. This dataset was split similarly to the one used in the consumer law domain, by using a set number of queries for the validation and test sets (the same set for both, due to lack of annotations) and the remaining queries were used for training. In this case, we used 32 queries for the validation and test sets, and the remaining 79 for training.

In Table 6.3, we are presented with the results of the experiment, homologous to the one done for the Consumer Law domain, with the exception of the way in which the accuracy is measured. In the consumer law domain, the accuracy is measured by comparing the act in the result with the act in the golden label, but in this domain we only have 2 acts, which would automatically mean that there is a 50% chance that a random answering search system would get the right result. Since there is no relevance to that kind of metric, we decided to measure the accuracy on the article level — we compared the article in the result with the one on the golden label — and this led to a more fair comparison between domain performances.

**Table 6.3:** LeSSE Accuracy in the Retirement Law

| Accuracy (%) | | | | |
|---|---|---|---|---|
| Results Measure | Baseline (BM25) | LeSSE | | |
| | | No Training | Trained with Manual Annotations | Trained with Automatic Annotations (Semantic Pairs) |
| TOP 1 | 43.8 | 43.8 | 46.9 | 40.6 |
| TOP 3 | 50.0 | 68.8 | 78.1 | 71.9 |
| TOP 5 | 50.0 | 75.0 | 90.6 | 84.4 |
| TOP 12 | 50.0 | 75.0 | 90.6 | 84.4 |

The results obtained from this experiment were much the same — that is, the manual annotations were the best dataset to train on, since it far surpassed the others in any category of accuracy and the automatic annotations (in this case Semantic Pairs were used) proved to be the second best option, with the exception of the TOP1 category, in which it did not score as high as the others, but it was compensated by the other accuracy scores.

# 7 Conclusion

Contents

The proposal of this project was to create a search system that would connect the Portuguese citizens to their Consumer Law by modifying how the search was made. The current Portuguese Consumer Law search engine works by searching keywords in the articles. Our improved version combines a more refined keyword search with a semantic search. For the keyword search we used a well established algorithm called BM25, in its original version, and for the semantic search we used a BERT language model, in a tailored pipeline.

In order to produce the desired results, the language model had to be trained on a corpus that included legislative jargon. This was an important step to ensure that the model would be able to create the right relations between similar words that it had not seen in the pre-training stage (first training with an extensive corpus). It was also important since the pre-training was done using a Brazilian Portuguese corpus, and because we needed it to be able to recognize the European Portuguese vocabulary, it had to be fine-tuned on an European Portuguese corpus.

The training corpus was constructed with the help of jurists from INCM, who annotated questions extracted from the DRE search database with passages from the Portuguese Consumer Law. This allowed us to pair questions with segments (smallest fragment of text in this context) from those passages and those were used, not only to train the model, but also to evaluate its performance during the training stage and to test after it had been trained.

We also tried training the model on other corpora to see how they would influence the performance and how the search engine scaled by training with automatically generated annotations. We observed that LeSSE is quite capable of achieving a high performance when trained with manual annotations (89.0% TOP3 accuracy), especially when compared to the Baseline (70.0%). We also observed that, in the absence of manual annotations the system was able to perform quite well, scoring 76.0% and 77.0%, when trained with an ICT dataset and a Semantic Pairs dataset, respectively.

Ultimately, we also tested the performance of LeSSE in a different law domain. We used a few annotations from the Retirement Law Corpus, which is a smaller corpus than the one originally used (Consumer Law), but, despite having less annotations to train and test on, the experiment proved that the system is fit to work on any domain of the Portuguese law, scoring 78.1% TOP3 accuracy and a decent 71.9% when trained with automatic annotations (in lack of manual ones).

## 7.1 Future Improvements

Despite having achieved good results in the task of searching the Portuguese Consumer Law and even the Retirement Law with natural language, there are still two major improvements that could be made in order to increase the performance of the system.

### 7.1.1 Language Model Size

In our system, we used the BERT model in the base version, which means that the model has 12 encoder layers, as opposed to BERT large which has 24 encoder layers. The large version has a better performance but the training and fine tuning of such a model also requires more computational power and memory. While fine tuning the BERT base version, we found that the resources were being maximized, given the size of the dataset, and so, we did not choose the large version. An additional issue with the large version comes when we generate the search index from the segment embeddings — in the base version, these are arrays of size 768, but in the large version they are 1024 long — and, because of that, the size of the search index would increase, and so would the amount of time it takes to perform a search in the index, adding another limitation to the use of the large option.

### 7.1.2 Number of Manual Annotations

As was mentioned before, we ended up using the manual annotations dataset to train the language model used in our system, and to test the latter. This was important to help the model better understand the jargon used in the legislative context, but the dataset was also essential to test the behaviour of the system with true labels. As we have seen, automatically generated annotations could provide some help in training the model, so they could be combined with manual annotations and even replace them, to some extent, but the dataset used to test the system has to be manually created, in order to ensure that the questions are correctly paired with the segments (answers).

Right now, the manual annotations dataset is composed of 321 annotated queries. Each of these queries has been annotated with one or more segments of the Consumer Law Corpus — in average, each query was annotated with 546 segments. To be able to test the system, we would have liked to use 80% of the queries for training, 10% for evaluating during training and another 10% for testing the system, but since 10% of the 321 queries (32) was not sufficient to cover most of the subjects in the queries, we decided to select the number of testing queries manually, and we set it to 100. Since there were only 221 remaining queries for the training and validation sets, and because the training was much more crucial, we decided to adopt a common approach when dealing with shortage of data — we used the same 100 queries for the validation and testing of the system and the remaining 221 queries were directed to the training set.

We would have liked to use different sets of queries for the validation and the testing of the system, since there is a slight bias the system tends to have towards the validation set, but the manually annotated queries were just not enough to be divided into three separate sets — training, validation and test — without also separating the subjects covered by the annotations into different sets, which was not intended since the ideal split of data is the one that separates the data and its subjects uniformly. In an ideal scenario, we would have liked to have at least 1000 queries annotated, so that both the validation and test sets would be different but this would also require more help from jurists with enough expertise to annotate the queries with the correct segments from the Portuguese Consumer Law.

# Bibliography

[1] 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.

[2] M.-w. C. Kenton, L. Kristina, and J. Devlin, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv:1810.04805 [cs]*, 2017.

[3] J. A. Wagner Filho, R. Wilkens, M. Idiart, and A. Villavicencio, "The BRWAC corpus: A new open resource for Brazilian Portuguese," in *LREC 2018 - 11th International Conference on Language Resources and Evaluation*, 2019.

[4] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using siamese BERT-networks," in *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, 2020.

[5] C. Samarinas, W. Hsu, and M. L. Lee, "Improving evidence retrieval for automated explainable fact-checking," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations*, 2021, pp. 84–91.

[6] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, 2013.

[7] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 2014.

[8] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, 1997.

[9] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems*, 2014.

[10] F. Souza, R. Nogueira, and R. Lotufo, “Bertimbau: Pretrained bert models for brazilian portuguese,” *Intelligent Systems Lecture Notes in Computer Science*, p. 403–417, 2020.

[11] K. Lee, M.-W. Chang, and K. Toutanova, “Latent retrieval for weakly supervised open domain question answering,” *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[12] W. L. Taylor, ““cloze procedure”: A new tool for measuring readability,” *Journalism Quarterly*, vol. 30, no. 4, p. 415–433, 1953.

[13] S. E. Robertson, S. Walker, K. S. Jones, and M. M. Hancock-Beaulieu, “Okapi at TREC-3,” *Proceedings of the Third Text REtrieval Conference*, 1994.

[14] B. Gain, D. Bandyopadhyay, T. Saikh, and A. Ekbal, “Iitp in coliee@ icail 2019: Legal information retrieval using bm25 and bert,” *Competition on Legal Information Extraction/Entailment 2019*, 2019.

[15] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y.-H. Sung, B. Strope, and R. Kurzweil, “Universal Sentence Encoder,” *EMNLP 2018 - Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Proceedings*, 2018.

[16] Y. Yang, D. Cer, A. Ahmad, M. Guo, J. Law, N. Constant, G. H. Abrego, S. Yuan, C. Tar, Y.-H. Sung *et al.*, “Multilingual universal sentence encoder for semantic retrieval,” *arXiv preprint arXiv:1907.04307*, 2019.

[17] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, “Supervised learning of universal sentence representations from natural language inference data,” in *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, 2017.

[18] Y. Sun, S. Wang, Y. Li, S. Feng, H. Tian, H. Wu, and H. Wang, “Ernie 2.0: A continual pre-training framework for language understanding,” *arXiv preprint arXiv:1907.12412*, 2019.

[19] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, “Xlnet: Generalized autoregressive pretraining for language understanding,” in *Advances in neural information processing systems*, 2019, pp. 5754–5764.

[20] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, “A large annotated corpus for learning natural language inference,” in *Conference Proceedings - EMNLP 2015: Conference on Empirical Methods in Natural Language Processing*, 2015.

[21] A. Williams, N. Nangia, and S. R. Bowman, “A broad-coverage challenge corpus for sentence understanding through inference,” in *NAACL HLT 2018 - 2018 Conference of the North American*

*Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 2018.

[22] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia, “SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation,” in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 1–14. [Online]. Available: https://www.aclweb.org/anthology/S17-2001

[23] M. Marelli, S. Menini, M. Baroni, L. Bentivogli, R. Bernardi, and R. Zamparelli, “A SICK cure for the evaluation of compositional distributional semantic models,” in *Proceedings of the 9th International Conference on Language Resources and Evaluation, LREC 2014*, 2014.

[24] M. Duarte, P. A. Santos, J. Dias, and J. Baptista, “Semantic norm recognition and its application to portuguese law,” 2022. [Online]. Available: https://arxiv.org/abs/2203.05425

[25] J. Yu, B. Bohnet, and M. Poesio, “Named entity recognition as dependency parsing,” *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[26] J. Johnson, M. Douze, and H. Jégou, “Billion-scale similarity search with GPUs,” *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

[27] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan *et al.*, “Population based training of neural networks,” *arXiv preprint arXiv:1711.09846*, 2017.