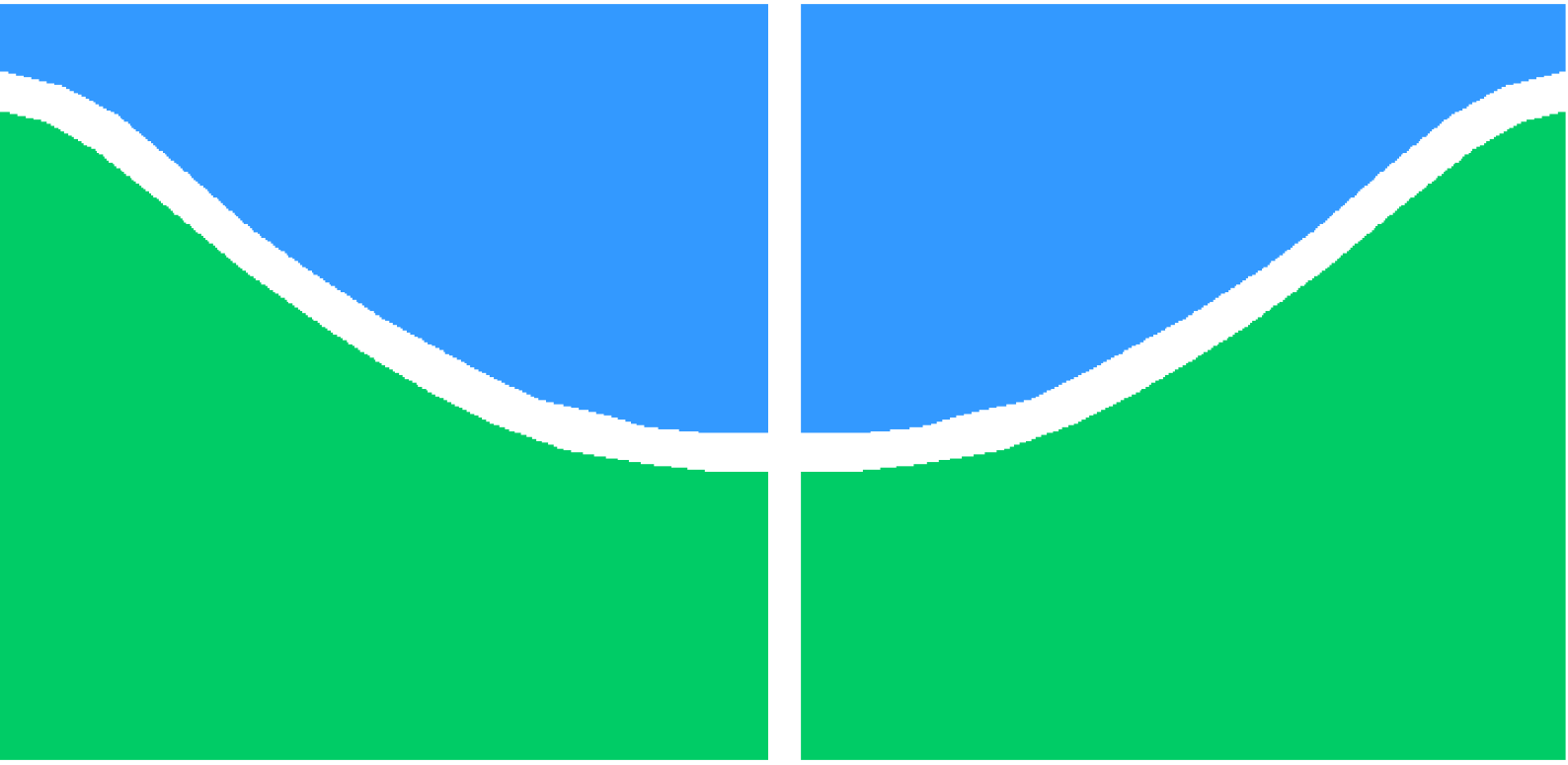

Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Electronic Engineering

BertBR: A Pretrained Language Model for Law Texts

Author: Victor Hugo Ciurlino
Supervisor: Dr. Nilton Correia da Silva

Brasília, DF
2021

Victor Hugo Ciurlino

BertBR: A Pretrained Language Model for Law Texts

Monograph submitted to the undergraduate course in (Electronic Engineering) of Universidade de Brasília, as a partial requirement to obtain the Title of Bachelor in (Electronic Engineering).

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Supervisor: Dr. Nilton Correia da Silva

Co-supervisor: Professor Pierre Gatien Florent André Guillou

Brasília, DF

2021

Victor Hugo Ciurlino
BertBR: A Pretrained Language Model for Law Texts/ Victor Hugo Ciurlino. – Brasília, DF, 2021-
48 p. : il. (algumas color.) ; 30 cm.

Supervisor: Dr. Nilton Correia da Silva

Final paper – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2021.

1. Palavra-chave01. 2. Palavra-chave02. I. Dr. Nilton Correia da Silva. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. BertBR: A Pretrained Language Model for Law Texts

CDU 02:141:005.6

Victor Hugo Ciurlino

BertBR: A Pretrained Language Model for Law Texts

Monograph submitted to the undergraduate course in (Electronic Engineering) of Universidade de Brasília, as a partial requirement to obtain the Title of Bachelor in (Electronic Engineering).

approved work. Brasília, DF, 24th May 2021:

Dr. Nilton Correia da Silva
Advisor

Professor Pierre Gatien Florent André Guillou
Co-advisor

Dr Fabricio Ataides Braz
Guest

Brasília, DF
2021

Acknowledgements

Dedico esse trabalho aos meus pais e irmão por sempre me apoiarem e fornecerem todo o suporte necessário durante toda a minha graduação. Agradeço a minha namorada por ter participado de boa parte dessa caminhada comigo e ter me dado a confiança e o apoio necessário. Agradeço também aos meus amigos que partilharam de todos os momentos na universidade.

Abstract

A aplicação de modelos de machine learning no âmbito jurídico está se tornando algo indispensável na automação e na otimização de processos, tornando possível o desvio de recursos de um trabalho mecânico e podendo concentrar esses recursos na parte mais intelectual do processo. Modelos criados a partir da lingua portuguesa demonstram um bom desempenho quando treinados para sub-tarefas da área de processamento de linguagem natural, tornando possível a extração e a classificação automatizada de documentos jurídicos, otimizando o tempo de processos e melhorando o atendimento de órgãos em que o volume de entrada para avaliação tende a ser maior que a sua vazão para as próximas esferas ou o próprio deferimento do processo. Estes modelos por sí são eficazes, porém parte da interpretação da linguagem jurídica é perdida, visto que a estrutura de sentenças e de documentos completos escritos com esse "dialeto"podem se diferenciar da estrutura normalmente usada e aquela em que os modelos são treinados. Afim de criar um modelo especializado para esse tipo de texto, foi utilizado um modelo BERT(*Bidirectional Encoder Representations from Transformers*) treinado na lingua portuguesa e realizado um processo de pós-treinamento utilizando textos jurídicos, afim de criar e disponibilizar um modelo voltado para esse domínio. O modelo treinado alcançou um F1-Score de 94.39% na subtarefa de reconhecimento de entidades nomeadas.

Palavras-chaves: BERT, NER, Futher pre-train.

Abstract

The application of machine learning models in the legal domain is becoming indispensable in the automation and optimization of processes, making it possible to redirect resources from mechanical work and being able to concentrate these resources in the most intellectual part of the process. Models created from the Portuguese language demonstrate a good performance when trained for sub-tasks in the area of natural language processing, making it possible to automate extract and classification tasks of legal documents, optimizing the time of proceedings and improving the attendance of bodies in which the volume of input for evaluation tends to be greater than its flow to the next spheres or the deferral of the process itself. These models by themselves are effective, but part of the interpretation of the legal language is lost, since the sentence structure and complete documents written with this "dialect" can differ from the structure normally used and the one in which the models are trained. In order to create a specialized model for this type of text, a BERT model (*Bidirectional Encoder Representations from Transformers*) was used, trained in Portuguese and a further pre-training process using legal texts, in order to create and make available a model geared to that domain. The trained model achieved an F1-Score of 94.39 % in the subtask of named entities recognition.

Key-words: BERT, NER, Further pre-train.

List of Figures

List of Tables

List of abbreviations and acronyms

Summary

Introduction

The development of NLP models for legal texts can be a challenge, since the language used for the elaboration of law suits texts carries in their vocabulary words that are not used very often, therefore the trained models currently available has its performance reduced. In order to overcome this challenge, a model for the Portuguese language will be developed for this work,aiming for a model specialized for this specific domain.

In this paper it will be studied Transformers architecture to better understand how BERT and transfer learning method works. The technique known as further training will also be studied, enabling the use of a BERT model already trained for the Portuguese language to create a model for the specific scenario mentioned in the first paragraph. In the end, the results of this work will be presented and discussed.

To better understand the objective of this paper, it is necessary to present some previous concepts, in order to apply knowledge in the development of analyzes in the future. First, it is presented the method BERT, explaining step-by-step and its architecture and some key concepts, such as self-attention, multi-Head attention and positional encoding. Then it presents the pre-training method and how it works in BERT, explaining Masked Language Modeling and Next Sentence Prediction method. After explained BERT, it is presented pos training methods as further pre-train and fine-tuning, explaining how it affects the outcome, explaining the subtask Name Entity Recognition. Then, the DATASET used for training is presented, describing how it was designed and finally, the results and the analysis of the results after the implementation of the study are presented.

Part I

Background

1 BERT

As showed in (DEVLIN et al., 2019), BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks. BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering, Named Entity Recognition and language inference, without substantial task specific architecture modifications.

There are two steps in this framework explained in (COLLOBERT; WESTON, 2008): pre-training and fine-tuning. During pre-training, the model is trained on unlabeled data over different pre-training tasks. For fine-tuning, the BERT model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks. Each downstream task has separate fine-tuned models, even though they are initialized with the same pre-trained parameters. A distinctive feature of BERT is its unified architecture across different tasks. There is minimal difference between the pre-trained architecture and the final downstream architecture.

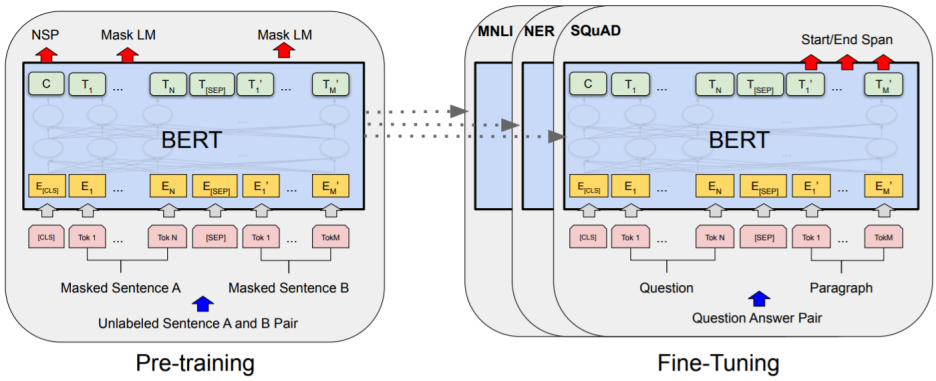

Figure 1 – Overall pre-training and fine-tuning procedures for BERT Source: (DEVLIN et al., 2019)

1.1 MODEL ARCHITECTURE

(DEVLIN et al., 2019) Explain BERT's model architecture as a multi-layer bidirectional Transformer encoder based on the original implementation described in (VASWANI et al., 2017).

The dominating trend in these models is to build complex, deep text representation models, for example, with convolutional networks (PARIKH et al., 2016) or long short-term memory networks (HOCHREITER; SCHMIDHUBER, 1997) with the goal of deeper sentence comprehension. While these approaches have yielded impressive results, they are often computationally very expensive, and result in models having millions of parameters (excluding embeddings).

Most competitive neural sequence transduction models have an encoder-decoder structure as as described in (VASWANI et al., 2017). Here, the encoder maps an input sequence of symbol representations (x1, ..., xn) to a sequence of continuous representations z = (z1, ..., zn). Given z, the decoder then generates an output sequence (y1, ..., ym) of symbols one element at a time. At each step the model is auto-regressive, consuming the previously generated symbols as additional input when generating the next. The Transformer follows this overall architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder, respectively.

Extended Neural GPU (KAISER; SUTSKEVER, 2015), ByteNet (KALCHBRENNER et al., 2016) and ConvS2S (GEHRING et al., 2017), all of which use convolutional neural networks as basic building block, computing hidden representations in parallel for all input and output positions. In these models, the number of operations required to relate signals from two arbitrary input or output positions grows in the distance between positions, linearly for ConvS2S and logarithmically for ByteNet. This makes it more difficult to learn dependencies between distant positions (HOCHREITER et al., 2001). In the Transformer this is reduced to a constant number of operations, at the cost of reduced effective resolution due to averaging attention-weighted positions, an effect we counteract with Multi-Head Attention.

(VASWANI et al., 2017) Also shows some advantages to choosing self-attention layers over recurrent and convolutional layers. One is the total computational complexity per layer. Another is the amount of computation that can be parallelized, as measured by the minimum number of sequential operations required. The third is the path length between long-range dependencies in the network. Learning long-range dependencies is a key challenge in many sequence transduction tasks. One key factor affecting the ability to learn such dependencies is the length of the paths forward and backward signals have to traverse in the network. The shorter these paths between any combination of positions in the input and output sequences, the easier it is to learn long-range dependencies.

(VASWANI et al., 2017) describes self-attention, sometimes called intra-attention, as an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence. Self-attention has been used successfully in a variety of tasks including reading comprehension, abstractive summarization, textual entailment and learning task-independent sentence representations.

1.1.1 Transformer architecture

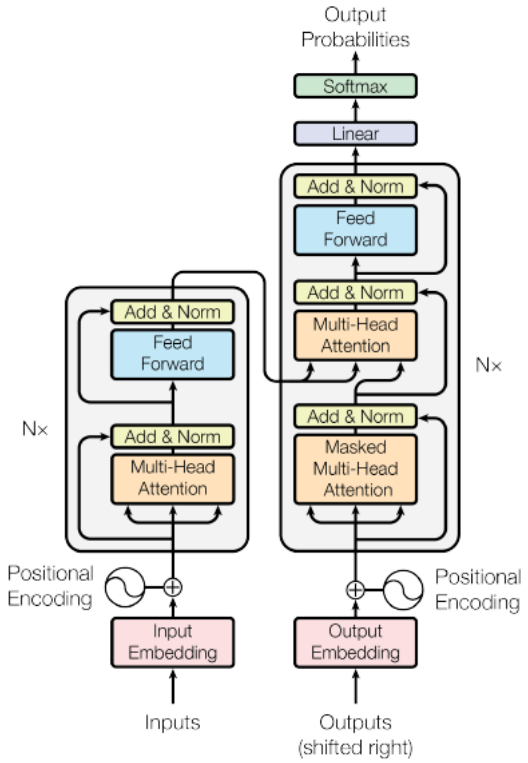

Figure 2 – Transformer architecture Source: (VASWANI et al., 2017)

1.1.1.1 Encoder

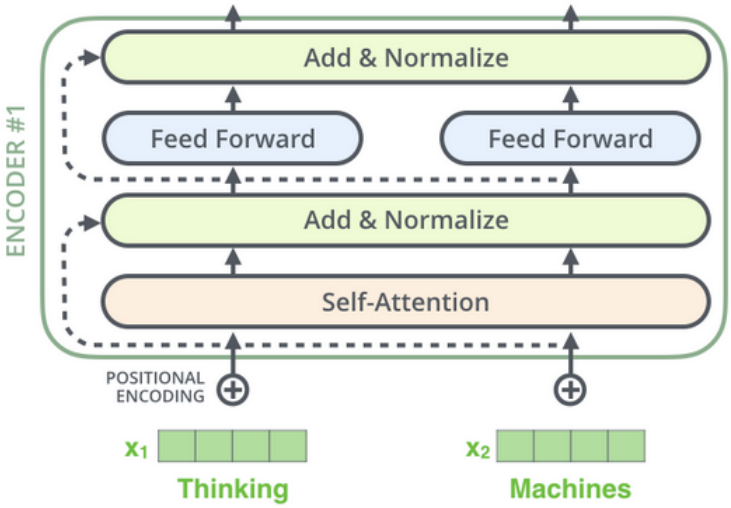

Figure 3 – Encoder structure Source: (JALAMMAR, 2019)

(JALAMMAR, 2019) describe the encoder structure as an stack of N = 6 identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple position-wise fully connected feed-forward network. The encoders are all identical in structure (yet they do not share weights). The encoder's inputs

first flow through a self-attention layer – a layer that helps the encoder look at other words in the input sentence as it encodes a specific word. The outputs of the self-attention layer are fed to a feed-forward neural network. The exact same feed-forward network is independently applied to each position. The decoder has both those layers, but between them is an attention layer that helps the decoder focus on relevant parts of the input sentence. It is added a residual connection around each of the two sub-layers, followed by layer normalization. That is, the output of each sub-layer is $LayerNorm(x + Sublayer(x))$, where $Sublayer(x)$ is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension $d_{model} = 512$.

1.1.1.2 Decoder

The decoder is also composed of a stack of N = 6 identical layers as shown at (JALAMMAR, 2019). In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder. The self-attention sub-layer in the decoder stack is modified to prevent positions from attending to subsequent positions so it will only consider tokens that was already decoded. This masking, combined with fact that the output embeddings are offset by one position, ensures that the predictions for position i can depend only on the known outputs at positions less than i.

1.1.1.3 Attention

(VASWANI et al., 2017) describes attention function as a mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

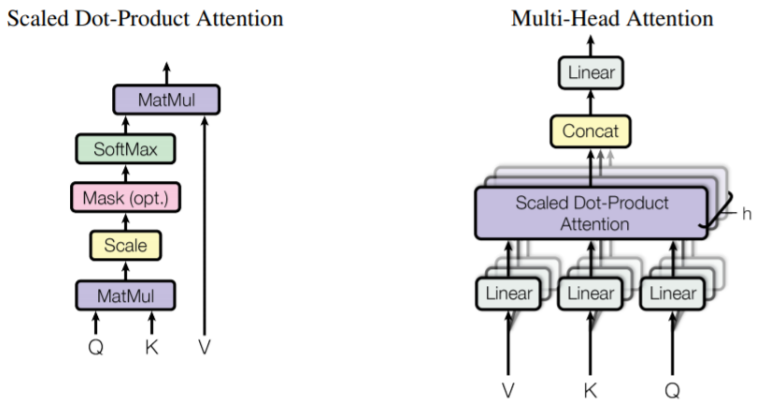

Figure 4 – (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel. Source: (VASWANI et al., 2017)

The input consists of queries and keys of dimension d_k, and values of dimension d_v as shown at (VASWANI et al., 2017). The dot products of the query with all keys is computed, divide each by $\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values, that pack all together into a matrix Q. The keys and values are also packed together into matrices K and V . The matrix of outputs compute as:

$$Attention(Q, K, V) = softmax(\frac{Q.K^T}{\sqrt{d_k}}).V \tag{1.1}$$

A self-attention layer connects all positions with a constant number of sequentially executed operations, whereas a recurrent layer requires $O(n)$ sequential operations. In terms of computational complexity, self-attention layers are faster than recurrent layers when the sequence length n is smaller than the representation dimensionality d, which is most often the case with sentence representations used by state-of-the-art models in machine translations, such as word-piece and byte-pair representations. To improve computational performance for tasks involving very long sequences, self-attention could be restricted to considering only a neighborhood of size r in the input sequence centered around the respective output position. This would increase the maximum path length to $O(n/r)$ as shown at (VASWANI et al., 2017).

1.1.1.4 Multi-Head Attention

Instead of performing a single attention function with $d_{model} - dimensional$ keys, values and queries, it beneficial to linearly project the queries, keys and values h times with different, learned linear projections to d_k and d_v dimensions, respectively. On each of these projected versions of queries, keys and values it is performed multiple attention functions in parallel, yielding $d_v - dimensional$ output values. Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O \tag{1.2}$$

Where

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{1.3}$$

The Multi-head attention is used in three different ways:

- In encoder-decoder attention layers. The queries come from the previous decoder layer, and the memory keys and values come from the output of the encoder. This allows every position in the decoder to attend over all positions in the input sequence. This mimics the typical encoder-decoder attention mechanisms in sequence-to-sequence models.

- The encoder contains self-attention layers. In a self-attention layer all of the keys, values and queries come from the same place, in this case, the output of the previous layer in the encoder. Each position in the encoder can attend to all positions in the previous layer of the encoder.

- Similarly, self-attention layers in the decoder allow each position in the decoder to attend to all positions in the decoder up to and including that position. We need to prevent leftward information flow in the decoder to preserve the auto-regressive property. We implement this inside of scaled dot-product attention by masking out (setting to $-\infty$) all values in the input of the softmax which correspond to illegal connections.

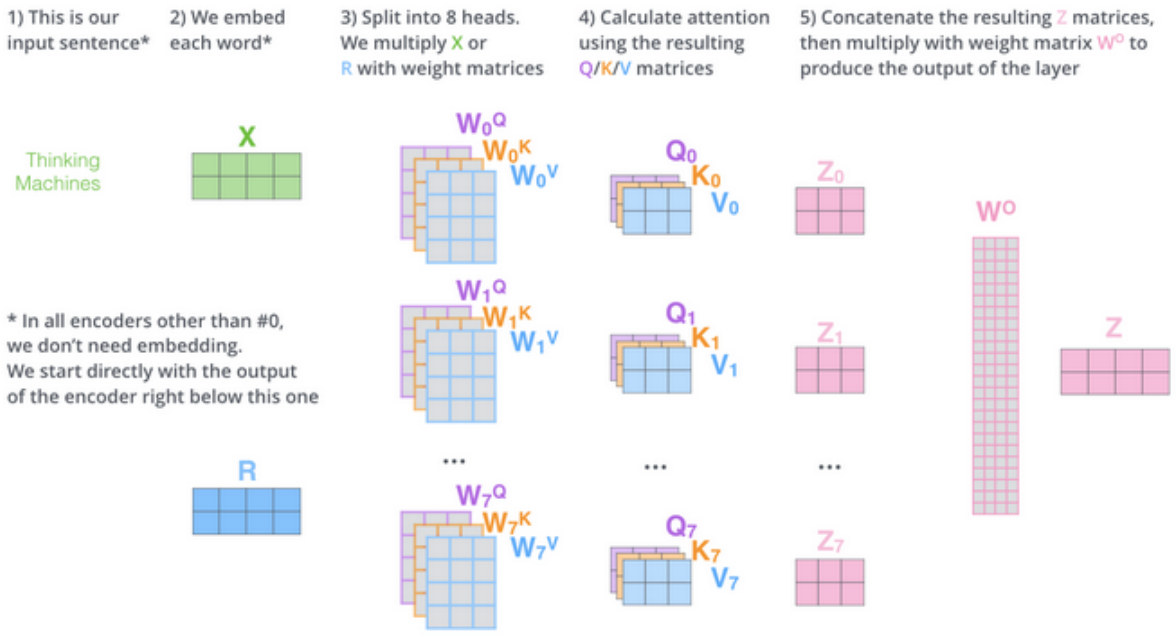

Figure 5 – Multi-head attention diagram Source: (JALAMMAR, 2019)

1.1.1.5 Feed-forward networks

In addition to attention sub-layers, each of the layers in our encoder and decoder contains a fully connected feed-forward network, which is applied to each position separately and identically as described at (VASWANI et al., 2017). This consists of two linear transformations with a ReLU activation in between.

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{1.4}$$

While the linear transformations are the same across different positions, they use different parameters from layer to layer. Another way of describing this is as two convo-

lutions with kernel size 1. The dimensionality of input and output is $d_{model} = 512$, and the inner-layer has dimensionality $d_{ff} = 2048$.

1.1.1.6 Positional enconding

In order for the model to make use of the order of the sequence,(VASWANI et al., 2017) explain the need to inject some information about the relative or absolute position of the tokens in the sequence. To this end, it is added "positional encodings" to the input embeddings at the bottoms of the encoder and decoder stacks. The positional encodings have the same dimension d_{model} as the embeddings, so that the two can be summed. There are many choices of positional encodings, learned and fixed.

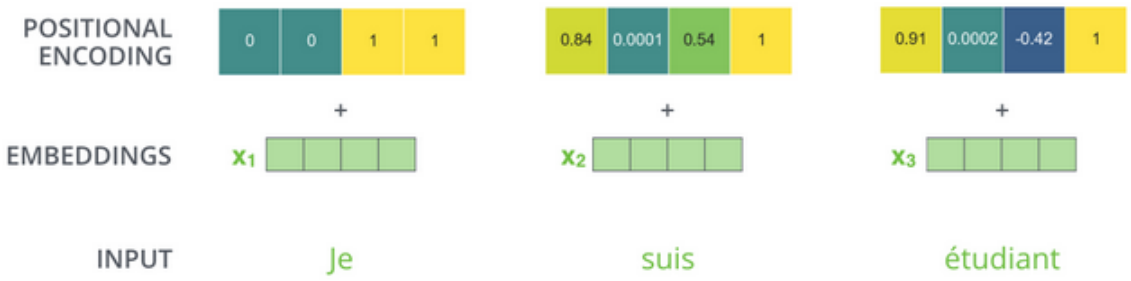

Figure 6 – Positional enconding Source: (JALAMMAR, 2019)

The transformer's original positional encoding scheme has two key properties. First, every position has a unique positional encoding, allowing the model to attend to any given absolute position. Second, any relationship between two positions can be modeled by an transform between their positional encodings. The positional encodings take the form:

$$PE_{(pos,2i)} = sin(\frac{pos}{1000^{(\frac{2i}{d_{model}})}}) \tag{1.5}$$

$$PE_{(pos,2i+1)} = cos(\frac{pos}{1000^{(\frac{2i}{d_{model}})}}) \tag{1.6}$$

Where pos is the position and i is the dimension. That is, each dimension of the positional encoding corresponds to a sinusoid. The wavelengths form a geometric progression from 2π to $10000 \cdot 2\pi$

(SHIV; QUIRK, 2019) explain that positional encodings address the power limitations of bag-of-words representations by upgrading the bag of words to a bag of annotated words. The transformer's core attention mechanism is order-agnostic, treating keys as a bag. The calculations performed on any given element of a sequence are entirely independent of the order of the rest of that sequence in that layer; this leaves most of the work

of exploiting positional information to the positional encodings showed by (VASWANI et al., 2017), though decoder-side self-attention masking and auto-regression also play a role.

1.2 PRE-TRAINING METHOD

(DEVLIN et al., 2019) explains the difference between Pre training BERT and others models who used traditional left-to right or right to left language models to pre-train. BERT use two unsupervised tasks: Masked Language Modeling and Next Sentence Prediction(NSP).

1.2.1 Masked language modeling

In order to train a deep bidirectional representation, some percentage of the input tokens is masked at random, and then predict those masked tokens.This procedure is refered as a "masked LM" (MLM), it is often referred to as a *Cloze task* in the literature (Taylor, 1953). In this case, the final hidden vectors corresponding to the mask tokens are fed into an output softmax over the vocabulary, as in a standard LM.

(DEVLIN et al., 2019) show how this process allows the model to obtain a bidirectional pre-trained model, a downside is that are created a mismatch between pre-training and fine-tuning, since the [MASK] token does not appear during fine-tuning. To mitigate this, words with the actual [MASK] token are not always replace "masked". The training data generator chooses 15% of the token positions at random for prediction. If the i-th token is chosen, i-th token with the [MASK] token is replaced 80% of the time, a random token 10% of the time and the unchanged i-th token 10% of the time. Then, T_i will be used to predict the original token with cross entropy loss.

1.2.2 Next sentence prediction

Many important downstream tasks such as Question Answering (QA) and Natural Language Inference (NLI) are based on understanding the relationship between two sentences, which is not directly captured by language modeling. In order to train a model that understands sentence relationships, the model is pre-trained for a binarized next sentence prediction task that can be trivially generated from any monolingual corpus. Specifically, when choosing the sentences A and B for each pre-training example, 50% of the time B is the actual next sentence that follows A (labeled as IsNext), and 50% of the time it is a random sentence from the corpus (labeled as NotNext) as shown at (DEVLIN et al., 2019).

The input embeddings are the sum of the tokens embeddings, the segmentation embeddings and the position embeddings. In prior work, only sentence embeddings are transferred to down-stream tasks, where BERT transfers all parameters to initialize end-task model parameters.

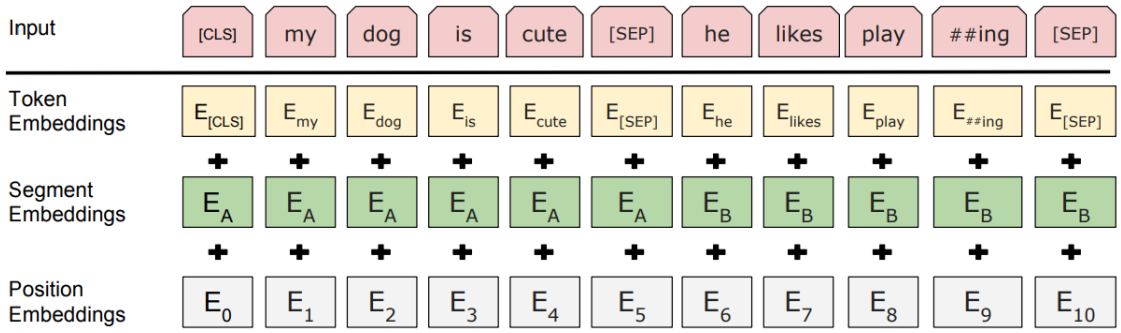

Figure 7 – BERT input representation. Source:(DEVLIN et al., 2019)

1.3 Further pre-train

Training a model for a specific domain can benefit a fine-tuning subtask with a specific vocabulary. As shown in articles, such as SciBERT, FinBERT, BioBERT and ClinicalBERT, it is possible to use a pre-trained model in one language and feed it with a non-labeled domain-specific database, using unsupervised deep learning method to specialize the model. This model can be fine-tuned for sub-tasks and as shown in the articles cited above, achieving an improvement over the BERT model without advanced training for the specific domain.

This process can be done in multiple ways, it is possible to train from scratch, using the domain-specific dataset to train all layers from the model, another method its to train some layers from a pre-trained model and a pre-trained model can be be re-trained using the domain-specific dataset, training all layers. SCIBERT and ClinicalBERT was trained using the original code from BERT (BELTAGY; LO; COHAN, 2019) (HUANG; ALTOSAAR; RANGANATH, 2019) and BioBERT used a pre-trained model in English language to take vantage from weights and the vocabulary, it used a WordPiece Tokenization, which mitigates the out-of-vocabulary issue. (LEE et al., 2019)

In all cases cited above, train a BERT using a domain-specific dataset showed a better result, outperforming BERT for general purpose on down-tasks for the domain.

1.4 Fine-tuning models

As explained at (DEVLIN et al., 2019), Fine-tuning using BERT its a straightforward job. Since BERT uses transfer learning, it uses a pre-trained model as a start point.

The self attention mechanism in the Transformer allows BERT to model many downstream tasks by choosing the right inputs and outputs. There are three ways to fine-tune BERT models: training the entire architecture, Train some layers while freezing others and Train some layers while freezing others and input a trained neural network at the end of the last layer. For each task, its only necessary to plug in the task specific inputs and outputs into BERT and fine-tune all the parameters end-to-end. At the output, the token representations are fed into an output layer for token level tasks and the [CLS] representation is fed into an output layer for classification. Compared to pre-training, fine-tuning is relatively inexpensive.

For NER fine-tuning, the dataset need to be labeled and treated to feed the model. Considering a dataset structured as a table with 3 columns (number of sentence, word and label), it is necessary to build a array of sentences, using words and labels. After the array is built, it is transformed into tokens. From this new array, its applied both mechanisms cited before, Masked Language Modeling and Next Sentence Prediction, each one of then input a different special token to the array so the model can interpret. Finally it is possible to feed the input into model and fine-tuning for the specific task.

1.5 Name Entity Recognition

Named Entity Recognition (NER) is a task in information extraction, consisting in identifying and classifying information elements called named entity. Common categories are person, organization, location, time and numerical expressions. NER systems are often used as the first step in question answering, information retrieval, co-reference resolution, topic modeling, etc.

This term was first cited at the 6^{th} Message Understanding Conference (MUC) in 1996. One of the first research papers in the field was presented by Lisa F. Rau (1991) at the Seventh IEEE Conference on Artificial Intelligence Applications. Rau's paper describes a system to "extract and recognize [company] names". Early NER systems were based on handcrafted rules, lexicons, orthographic features and ontologies. These systems were followed by NER systems based on feature-engineering and machine learning.

Despite being conceptually simple, NER is not an easy task. The category of a named entity is highly dependent on textual semantics and its surrounding context. Moreover, there are many definitions of named entity and evaluation criteria, introducing evaluation complications.

(NADEAU; SEKINE, 2007) Shows that a good proportion of work in this task is devoted to the study of English, but a larger proportion addresses language independence and multilingualism problems. ConLL-2003 studied German in earlier works, Spanish and Dutch are represented by ConLL-2002, Japanese has been studied in MUC-6 conference

and Portuguese is studied in HAREM conference. There is datasets that focus on specific tasks in a process called fine-tuning. For this paper, a dataset for NER will be used in the legal scope, this dataset were build by *AILAB* [1]. This dataset is composed by 66 law suits documents from several Brazilian Courts, containing 318.073 tokens in total as shown at (ARAUJO et al., 2020).

(SANTOS et al., 2019) presents a system for Portuguese NER proposed by the Shared Task "Portuguese Named Entity Recognition and Relation Extraction Tasks (NerReIberLEF2019) " on IberLEF 2019. It used a BiLSTM-CRF model that receives a compilation of highly representational embeddings: FlairBBP + W2V-SKPG, achieving 74.64% F1-score for general proposes at the Second HAREM using the CoNLL-2002 script to evaluate. (SANTOS; GUIMARAES, 2015) used as approach language-independent NER using a DNN that employs word-and-character-level embeddings to perform sequential classification, achieving a 82.213% F1 score on SPA CoNLL-2002 and 71.23% on HAREM evaluation corpus. (SOUZA; NOGUEIRA; LOTUFO, 2020b) outperforms the previous state-of-the-art model (BiLSTMCRF+FlairBBP). It was trained three distinct models: Multilingual BERT-Base, Portuguese BERT-Base and Portuguese BERT-Large. it achieved 78.67% F1 score using Portuguese BERT-Large on total scenario and 83.24% for selective scenario using CoNLL 2003 evaluation script and MiniHAREM test set.

1.6 Dataset

Iudicium Textum Dataset is a database generated through the scraping process of Supreme Federal Court Agreements, where only digital judgments were considered, ignoring those that were digitized for a better reading of the words and fewer errors associated with the OCR process. Usually, a scanned document can misspell some words, harming the vocabulary used to train the model. This dataset contains 50.928 law suits, in between 2010 and 2018 and 20 gigabytes in PDF.

As shown in (SOUSA; FABRO, 2019), the main document used to compose the database are documents resulting from the trial by higher courts of the Judiciary in Brazil, the choice was made due to the very rigid and well-defined structure.

1.6.1 Extraction

(SOUSA; FABRO, 2019) explains that the decision of the documents to be extracted was based on the study of the quality of the texts, aiming at a better quality of the data and not the quantity. A document whose digitization presents noise impairs the reading of the text, making it harmful to the training of the model to use it, and may create bias with words that do not exist in the Portuguese vocabulary.

[1] UNB artificial intelligence research laboratory

After scrapped, a text cleaning process was carried out, removing signatures, page numbers, footnotes and headings. Some textual components have also been removed, such as locations, dates, final and initial indication of the rapporteur and personal names. However, such textual information can be retrieved in the database itself, since a copy of the full text was kept for each document.

2 Experiment

This experiment was separated into two stages: Further pre-train a bert-large-portuguese-cased model using BERT and (SOUSA; FABRO, 2019) dataset and fine-tuning this model for the subtask Named-entity recognition. For comparison purposes it was fine-tuned a bert-large-portuguese-cased.

For the first stage, a BERT model was imported from Huggingface (SOUZA; NOGUEIRA; LOTUFO, 2020a) and trained using (SOUSA; FABRO, 2019). All layers of the model were trained using the methods described in BERT, masked language modeling and next sentence prediction.

For the first step, it was used the default setup:

- evaluation_strategy = "epoch",
- learning_rate=2e-5,
- weight_decay=0.01

Some challenges in relation to this first stage were mainly due to the requirement of high computational power. Such an experiment could have been developed using colabs, however some configurations so that the GPU was not deallocated and the machine did not go down were necessary, some configurations even as an auto-clicker to avoid inactivity. In addition, the resource in time gave up the use of the platform, although it was free, since the development time for the elaboration of the experiment and the study about it would be key to the conclusion of the CBT in time. Therefore, it was requested to use the GPU available by the AILAB laboratory for training the model.

Before starting training the model, it was necessary to perform a processing on the database, unifying all available texts, which were in several folders separated by process into a single text file. The database had both pdf and txt files, for this first moment only text files were used ignoring PDF files, because they have to go through an optical character recognition process before they can be used in the training.

There were some setbacks in this stage, since the training a unsupervised model itself was expensive in time and after 2 days,a conflict in the version of transformers packages occurred, generating a significant delay in this first stage. At the end it was possible to export the trained model, unfortunately due to the high time required, it was not possible to experiment training another model with other parameters in order to analyze.

Second it was used a specific data set, made available by the laboratory *AILAB* for law suits texts in Portuguese/Brazil. There are 13 different labels, wich are: B-LOCAL, I-PESSOA, I-LOCAL, I-LEGISLACAO, B-LEGISLACAO, O, B-ORGANIZACAO, I-ORGANIZACAO, B-PESSOA, I-JURISPRUDENCIA, B-TEMPO, I-TEMPO and B-JURISPRUDENCIA, where B stands for begin and I for inside the entity. For a example, if the phrase "Meu nome é Victor Ciurlini e eu estudo na Universidade de Brasília", the model will return Victor as B-PESSOA, Ciurlini as I-PESSOA, Universidade as B-LOCAL and de Brasília I-LOCAL. note that if multiple words is used to describe a entity, all of then will receive one label.

for this second step there was also a data processing, since for the BERT fine-tuning requires a specific structure to take full advantage of all the resources of this method, it was necessary to generate a new column assigning for each sentence an ID, indicating for each word which sentence it belong. It was separate the text into sentence in all ending points (commas, periods, etc) and for each sentence a unique ordered ID. The individual words were separated from the sentences and assigned the ID corresponding to the sentence that word belonged to. at the end of the procedure, the database was in the following format:

Word	**Tag**	**Sentence #**
STJ	B-ORGANIZACAO	8
,	O	8
REsp	B-JURISPRUDENCIA	8
n	I-JURISPRUDENCIA	8
.	I-JURISPRUDENCIA	8
1694984/MS	I-JURISPRUDENCIA	8
,	O	8
Rel	O	8
.	O	8
Ministro	O	8
LUIS	B-PESSOA	8
FELIPE	I-PESSOA	8
SALOMÃO	I-PESSOA	8
,	O	8
QUARTA	B-ORGANIZACAO	8
TURMA	I-ORGANIZACAO	8
,	O	8
julgado	O	8
em	O	8
14/11/2017	B-TEMPO	8
,	O	8
DJe	O	8
01/02/2018	B-TEMPO	8

Table 1 – Fine-tuning table for NER

Some pipelines for fine-tuning were made available by Huggingface, however it was developed a algorithm from scratch, both for the deeper understanding of the methodology, as well as for greater control of the steps, in order to visualize the entrance and output for each step until complete fine-tuning of the model. This choice allowed to generate metrics for future analysis.

For the Optmizer definition, a pipeline was developed in order to test a greater combination of parameters, noting results so that the way the changes affect the F1-score of the model was analyzed. 5 values were tested for each learning rate and epsilon parameter and then iterating between them, a total of 25 tests. the choice of the epoch number was made based on an experiment, where it was noted that in 4 generations, the loss in validation start to increase, with 5 continuing to grow and with 2 it did not converge to the best model, so 3 epoch were frozen for all experiments.

After the experiment, graphs were generated in order to better analyze the results of each experiment, analyzing the general f1-score and each class for the subtask. These visualizations enabled a deeper analysis of the model, understanding where the model failed and where it can improve its performance.

Unlike those models cited above, it was used 500.000 epochs, avoiding overfitting the model. The further trained took 96 hours, using a GPU GP102-TITAN Xp, with 33 MHz clock and 64 bits width (made available by the laboratory *AILAB*)

For the second step, it was used a algorithm were developed in *google colabs* platform. The algorithm is descriptive, showing step-by-step of the NER fine-tuning procedure. It will be fine-tuned two models, a bert-large-portuguese-cased and the further pre-trained model with domain-specific dataset.

aiming for the best possible model, a logic was developed in the training stage that performs the variation of 2 parameters within the AdamW Optmizer, the loss rate and epsilon. 5 values were tested for each parameter, whose values are close to the value used as the default and a combination was made between them to contemplate all possible pairs, totaling 25 tests.

For evaluation and comparison between those two models, it was used the metric F1-SCORE, since that it has a balanced relationship between precision and recall.

$$F_1 = \frac{tp}{tp + 1/2(fp - fn)} \tag{2.1}$$

where

- tp: True Positive
- fp: False Positive

- fn: False negative

3 RESULTS AND DISCUSSION

All fine-tuning and training data metrics were exported to generate graphical visualizations in order to allow a better analysis of the results obtained.

#	Loss Rate	epsilon	F1 Score Total	F1 Score for each entity					
				jurisprudencia	legislacao	local	organizacao	pessoa	tempo
1	0.00001	1.000e-06	0.941592	0.926293	0.915361	0.935574	0.913793	0.969450	0.976092
2	0.00001	1.000e-07	0.944528	0.912752	0.935433	0.903409	0.917981	0.968528	0.994172
3	0.00001	1.000e-08	0.941594	0.903297	0.922118	0.915068	0.915682	**0.970134**	0.994172
4	0.00001	1.000e-09	0.951529	0.930693	**0.946203**	0.935211	0.927626	0.961224	0.993344
5	0.00001	1.000e-10	0.949785	0.932018	0.927673	0.933718	0.928627	0.962360	0.992519
6	0.00002	1.000e-06	0.947053	0.920177	0.944532	0.913295	0.925955	0.960082	0.991694
7	0.00002	1.000e-07	0.950587	0.931718	0.929134	0.927954	0.930124	0.966361	0.993344
8	0.00002	1.000e-08	0.950887	0.933775	0.939873	0.916427	0.926905	0.966223	0.993367
9	0.00002	1.000e-09	0.945713	0.914607	0.925984	0.916427	0.925117	0.962360	**0.995844**
10	0.00002	1.000e-10	0.947782	0.929360	0.919431	0.913793	0.927287	0.962963	**0.995844**
11	0.00003	1.000e-06	0.951369	0.942350	0.918083	0.923077	0.927514	0.968335	**0.995844**
12	0.00003	1.000e-07	**0.952595**	0.942350	0.923567	0.922190	0.928739	0.969325	**0.995844**
13	0.00003	1.000e-08	0.951882	0.942605	0.939683	0.922636	0.924647	0.962206	0.994172
14	0.00003	1.000e-09	0.943262	0.937294	0.916535	0.911111	0.912941	0.954918	0.994172
15	0.00003	1.000e-10	0.946719	**0.943522**	0.906832	0.906077	0.926366	0.958848	**0.995844**
16	0.00004	1.000e-06	0.947094	0.933921	0.925750	0.938202	0.913887	0.960082	**0.995844**
17	0.00004	1.000e-07	0.947211	0.934950	0.924765	0.916201	0.918189	0.962051	**0.995844**
18	0.00004	1.000e-08	0.951261	0.935094	0.915361	**0.947977**	0.929577	0.965306	0.995008
19	0.00004	1.000e-09	0.950562	0.935236	0.910236	0.934844	**0,932912**	0.967078	0.993333
20	0.00004	1.000e-10	0.945116	0.935841	0.920188	0.924370	0.907137	0.969072	0.993333
21	0.00005	1.000e-06	0.945244	0.936123	0.914557	0.916667	0.922231	0.956967	0.993333
22	0.00005	1.000e-07	0.941702	0.937294	0.903125	0.923944	0.902326	0.962887	**0.995844**
23	0.00005	1.000e-08	0.947113	0.943396	0.909091	0.926554	0.917187	0.964029	**0.995844**
24	0.00005	1.000e-09	0.946937	0.937017	0.919937	0.923513	0.912335	0.967413	**0.995844**
25	0.00005	1.000e-10	0.947901	0.935982	0.937799	0.918768	0.918448	0.959184	0.992506

Table 2 – Results table fine-tuning BertBR

As shown above, the best model found was using $lossrate = 0.00003$ and $epsilon = 1.10^{-7}$ as parameter for AdamW optimizer, which achieved a general F1-score of 0.9525. for each entity, the best f1-score its found as described above.

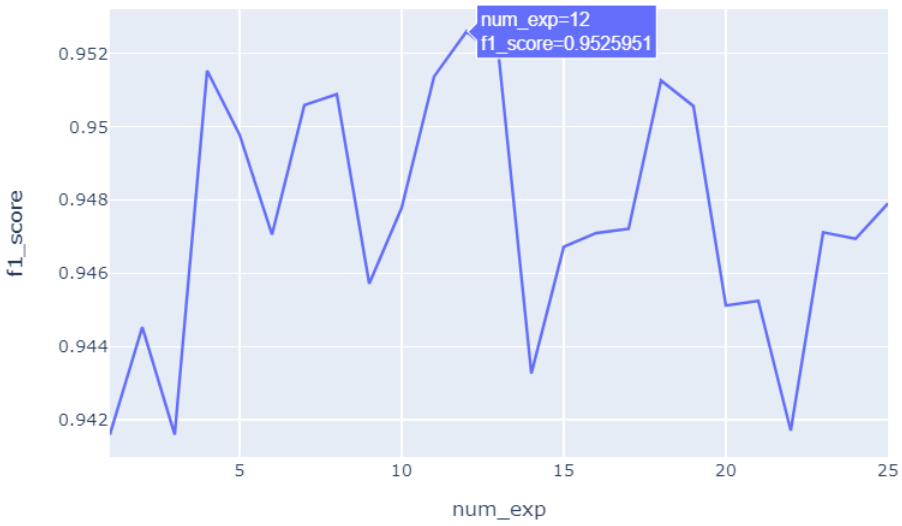

Figure 8 – F1-score evolution per experiment

For visualization purposes, table 2 was presented in a graph, and it can be perceived how the alteration of the parameters affect the results in the fine tuning step in the final f1-score of each experiment. This F1-score is a metric generated from other f1-scores generated by the precision and calculated recall of the individual classification of each class of named entity.

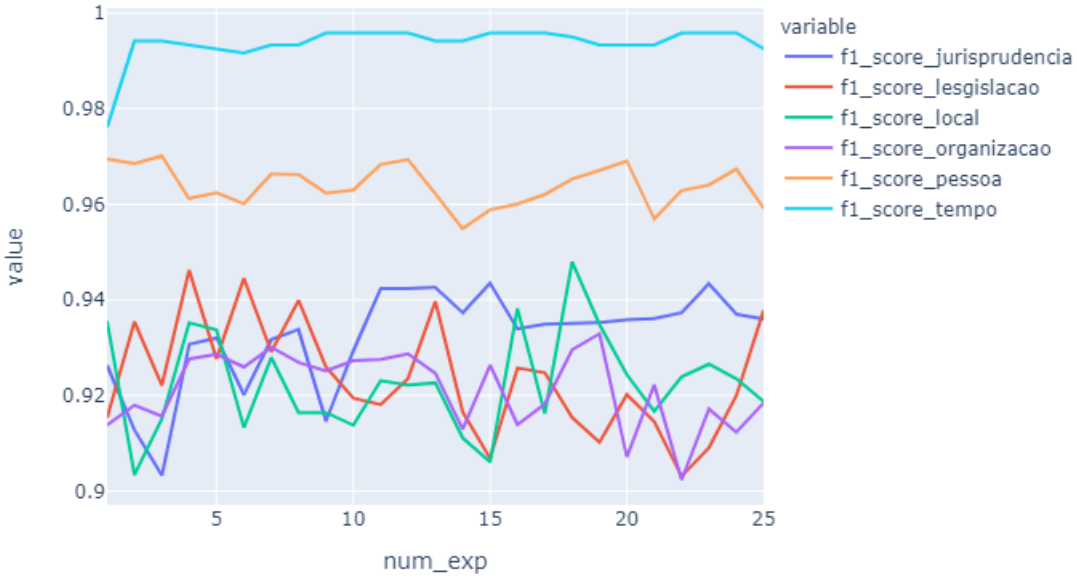

Figure 9 – F1-score evolution per experiment per class

Analyzing the graph 3 , it is noted that the time class is the most constant during training, keeping its f1-score above 98%. This is due to the fact that its specific structure with limited formats for identification, having less chance of being confused with other possible classes. It is also noted that, even they have an acceptable f1-score, the identification of the jurisprudence, legislation, organization and local classes presents a greater challenge for the model.

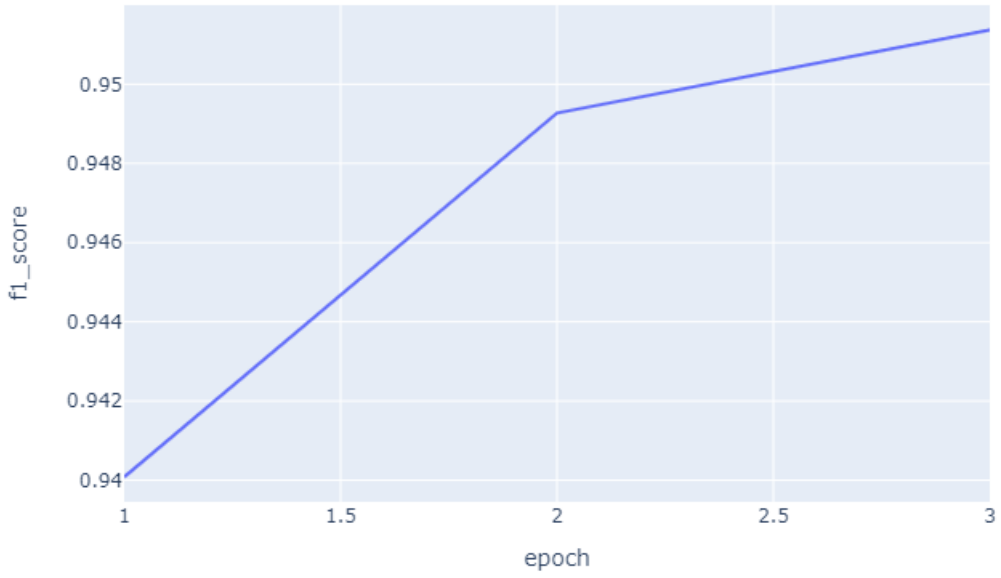

Figure 10 – F1-score evolution per epoch of the best model

Analyzing the results of experiment 12, which achieved the best f1-score and is therefore considered the best model, we noticed a significant improvement from the first generation to the second and a less attenuated improvement from the second generation to the third.

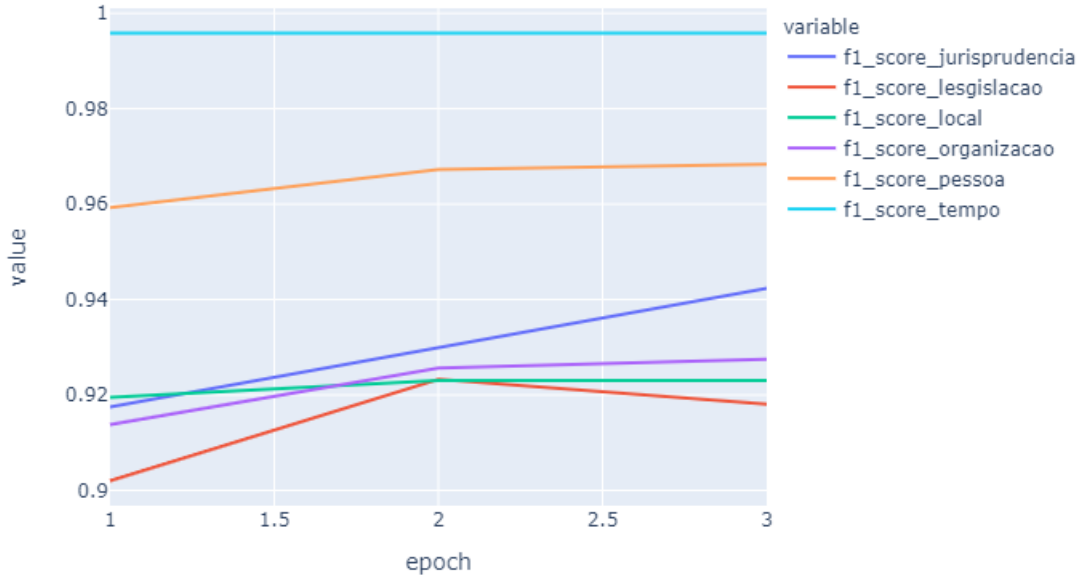

Figure 11 – F1-score evolution per epoch per class of the best model

Regarding the individual analysis of the classes, it is noted that the result shown for all models is reflected in the analysis of this specific model. It is observed that for the time class, the model performs with an f1-score of 99%. Such value would generate a suspicion if this value were referred to accuracy, which does not consider the miss-classification in this class, however for f1-score, this value represents the high success rate in the classification along with the low miss-classification.

Labels	**Precision**	**Recall**	**F1-score**
JURISPRUDENCIA	0.96	0.91	0.94
LEGISLACAO	0.99	0.88	0.91
LOCAL	0.98	0.92	0.92
ORGANIZACAO	0.98	0.91	0.92
PESSOA	0.95	0.96	0.96
TEMPO	0.96	0.99	0.99

Table 3 – BertBR metrics

Labels	precision	recall	f1-score
JURISPRUDENCIA	0.95	0.91	0.93
LEGISLACAO	0.93	0.93	0.93
LOCAL	0.89	0.88	0.89
ORGANIZACAO	0.92	0.89	0.90
PESSOA	0.99	0.98	0.98
TEMPO	0.99	0.99	0.99

Table 4 – large-bert-model metrics

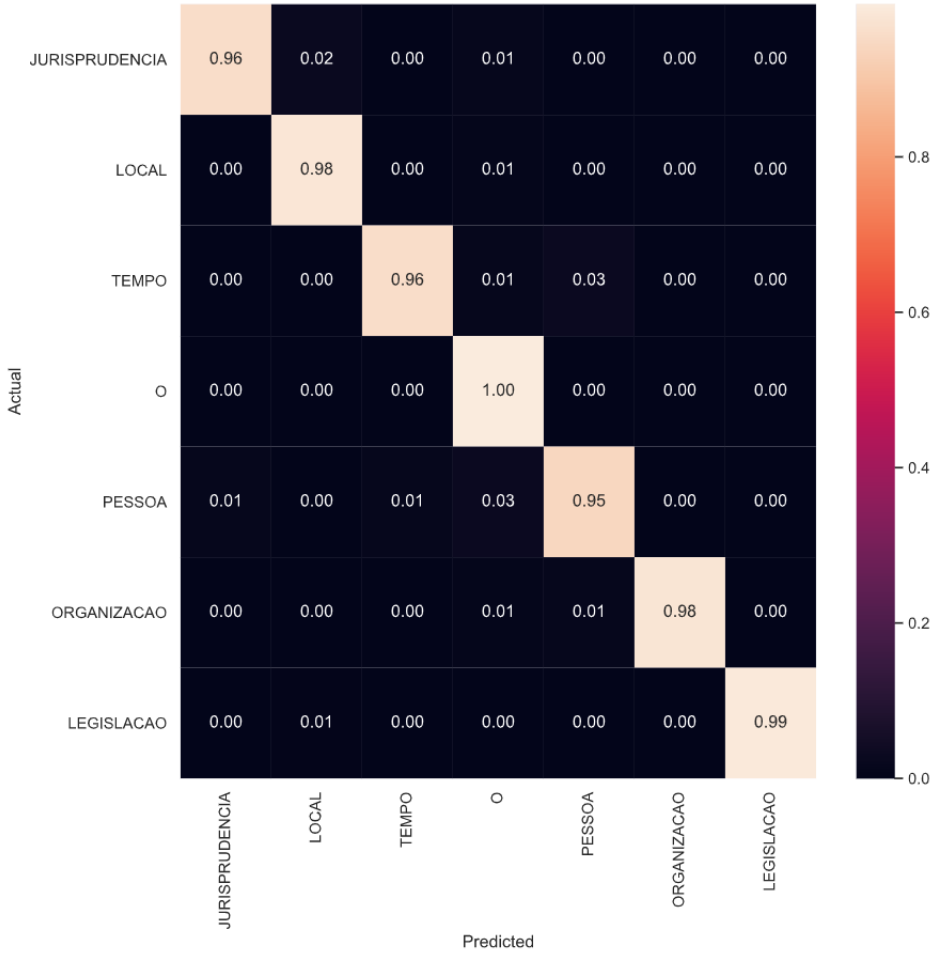

Figure 12 – Confusion matrix BertBR

Analyzing the confusion matrix, it is possible to observe the classes in which the model miss-classificated. In BertBR model, it is noted that in 2% of cases, it classified a word as "LOCAL", but when it was "JURISPRUDENCIA". It can also be noted that the highest miss-classification is 3% and belongs to the "person-time" and "person-object" label conflicts. When observing the column of objects, a pattern occurs where it classifies as an object words that when he ranks without a good accuracy, which is in accordance with the training, since most of the training classification labels are object.

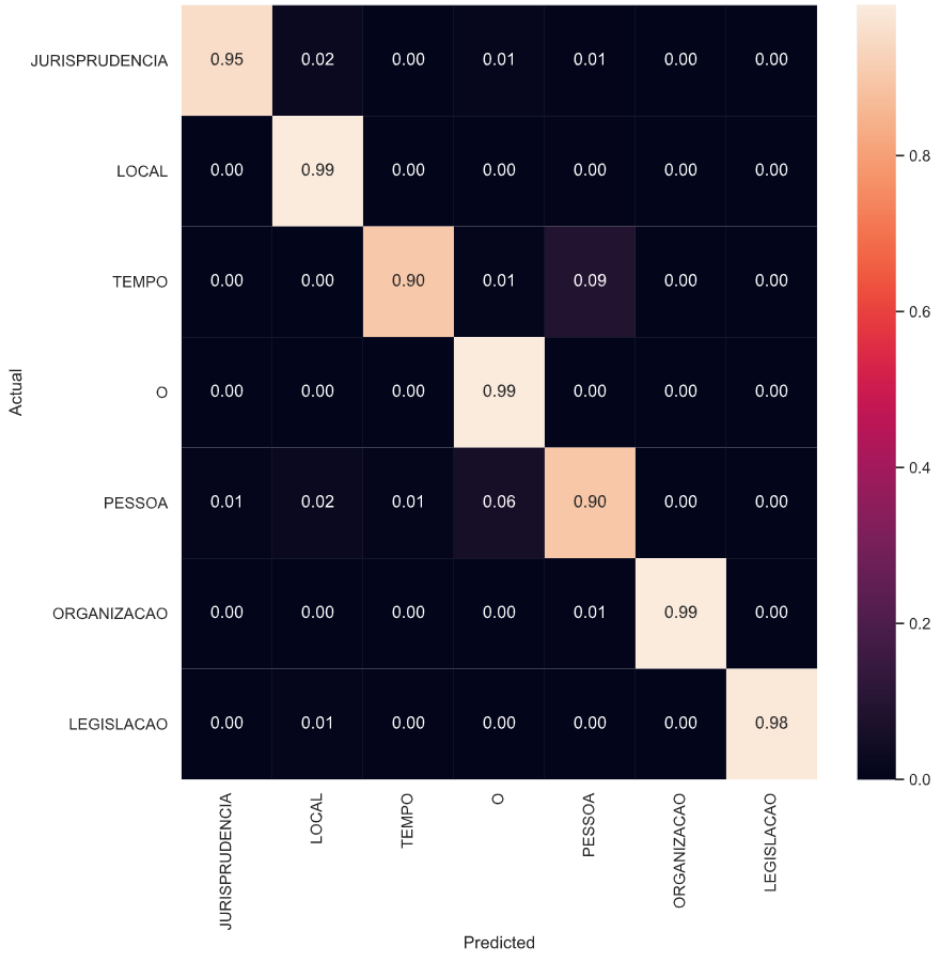

Figure 13 – Confusion matrix large-bert-cased

Analyzing the confusion matrix of the large-bert-cased model, it is possible to notice that miss-classification reaches 9% and is in the same category in which the BertBR model missed. This shows that in the training base there is a close relationship between these two classes, even though visibly the two classes do not seem to be related, even if for the humans it is not possible to perceive the similarity, the models finds some difficulty in effectively differentiating them. this scenario can be suppressed with a larger training base.

Comparing the generated models, it can be seen that bert-large-portuguese-cased overcome BertBR for PESSOA and ORGANIZACAO labels. This is due to the type of word that describes this labels, where bert-large-portuguese-cased can efficiently distinguish it from the others. It is also observed that for the specific law suits terms scenario, the model trained using further training has a greater capacity to distinguish classes correctly.

4 CONCLUSION

In this paper, it was implemented a BERT specialized for law suits domain by further training a bert-large-portuguese-cased using a law suits corpus. This model was fine-tuned for Named entity recognition for comparison purposes and achieved a state-of-the-art score, overcoming the latest model for this specific scenario.

The BertBR showed a improvement performance than PORTUGUESE-BERT, with a 94.45% F1-Score against 95.25%. showing similar results demonstrating results that resemble those shown previously.

This result corroborates with the premises pointed out by the studies that followed the same line, a trained model using a specific domain dataset demonstrates a better performance than one trained for general purpose, making it a valid technique to create models that reach state of the art for sub tasks.

Considering the resources spent, both in time (development and training) and computational resources, it is necessary to assess whether this performance gain justifies the resources cited above. For some applications it is in the developer interest to aim for the highest score possible application, however for simpler applications a model trained from BERT-large-cased is enough and can be developed, trained, exported and deployed in less than 1 day.

The experience of creating this new model has brought a lot of learning through extensive research into the architecture of BERT, as well as methodologies for improving unsupervised learning models. The study of architecture transformers and the learning transfer methodology shows why BERT and other models that use it are at the forefront in research and markets, due to its flexibility and scalability.

For future research, it is possible to analyze more deeply each class of named entity, looking for patterns in the miss-classifications and alternatives overcome the peculiarities found. It is also possible to perform the further training process again by changing the training parameters.

Bibliography

ARAUJO, P. H. L. de et al. Lener-br: a dataset for named entity recognition in brazilian legal text. University of Brasília, 2020. Cited on page 33.

BELTAGY, I.; LO, K.; COHAN, A. *SciBERT: A Pretrained Language Model for Scientific Text*. 2019. Cited on page 31.

COLLOBERT, R.; WESTON, J. A unified architecture for natural language processing: Deep neural networks with multitask learning. In: . New York, NY, USA: Association for Computing Machinery, 2008. (ICML '08), p. 160–167. ISBN 9781605582054. Disponível em: <https://doi.org/10.1145/1390156.1390177>. Cited on page 23.

DEVLIN, J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding. In: *NAACL-HLT*. [S.l.: s.n.], 2019. Cited 4 times on page 11, 23, 30, and 31.

GEHRING, J. et al. Convolutional sequence to sequence learning. *CoRR*, abs/1705.03122, 2017. Disponível em: <http://arxiv.org/abs/1705.03122>. Cited on page 24.

HOCHREITER, S. et al. *Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies*. 2001. Cited on page 24.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural Computation*, v. 9, n. 8, p. 1735–1780, 1997. Cited on page 24.

HUANG, K.; ALTOSAAR, J.; RANGANATH, R. Clinicalbert: Modeling clinical notes and predicting hospital readmission. *CoRR*, abs/1904.05342, 2019. Disponível em: <http://arxiv.org/abs/1904.05342>. Cited on page 31.

JALAMMAR. *The Illustrated Transformer*. 2019. Disponível em: <http://jalammar.github.io/illustrated-transformer/>. Cited 5 times on page 11, 25, 26, 28, and 29.

KAISER Łukasz; SUTSKEVER, I. *Neural GPUs Learn Algorithms*. 2015. Cited on page 24.

KALCHBRENNER, N. et al. Neural machine translation in linear time. *CoRR*, abs/1610.10099, 2016. Disponível em: <http://arxiv.org/abs/1610.10099>. Cited on page 24.

LEE, J. et al. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, Oxford University Press (OUP), Sep 2019. ISSN 1460-2059. Disponível em: <http://dx.doi.org/10.1093/bioinformatics/btz682>. Cited on page 31.

NADEAU, D.; SEKINE, S. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, v. 30, 01 2007. Cited on page 32.

PARIKH, A. P. et al. A decomposable attention model for natural language inference. *CoRR*, abs/1606.01933, 2016. Disponível em: <http://arxiv.org/abs/1606.01933>. Cited on page 24.

SANTOS, C. N. dos; GUIMARAES, V. Boosting named entity recognition with neural character embeddings. Computing Research Repository, arXiv:1505.05008, 2015. Disponível em: <https://arxiv.org/abs/1505.05008.version2.> Cited on page 33.

SANTOS, J. et al. Multidomain contextual embeddings for named entity recognition. Pontifical Catholic University of Rio Grande do Sul, 2019. Cited on page 33.

SHIV, V. L.; QUIRK, C. Novel positional encodings to enable tree-based transformers. In: *NeurIPS 2019*. [s.n.], 2019. Disponível em: <https://www.microsoft.com/en-us/research/publication/novel-positional-encodings-to-enable-tree-based-transformers/>. Cited on page 29.

SOUSA, A. W.; FABRO, M. Iudicium textum dataset uma base de textos jurídicos para nlp. In: . [S.l.: s.n.], 2019. Cited 2 times on page 33 and 35.

SOUZA, F.; NOGUEIRA, R.; LOTUFO, R. BERTimbau: pretrained BERT models for Brazilian Portuguese. In: *9th Brazilian Conference on Intelligent Systems, BRACIS, Rio Grande do Sul, Brazil, October 20-23 (to appear)*. [S.l.: s.n.], 2020. Cited on page 35.

SOUZA, F.; NOGUEIRA, R.; LOTUFO, R. Portuguese named entity recognition using bert-crf. University of Campinas, 2020. Cited on page 33.

VASWANI, A. et al. Attention is all you need. *CoRR*, abs/1706.03762, 2017. Disponível em: <http://arxiv.org/abs/1706.03762>. Cited 9 times on page 11, 23, 24, 25, 26, 27, 28, 29, and 30.