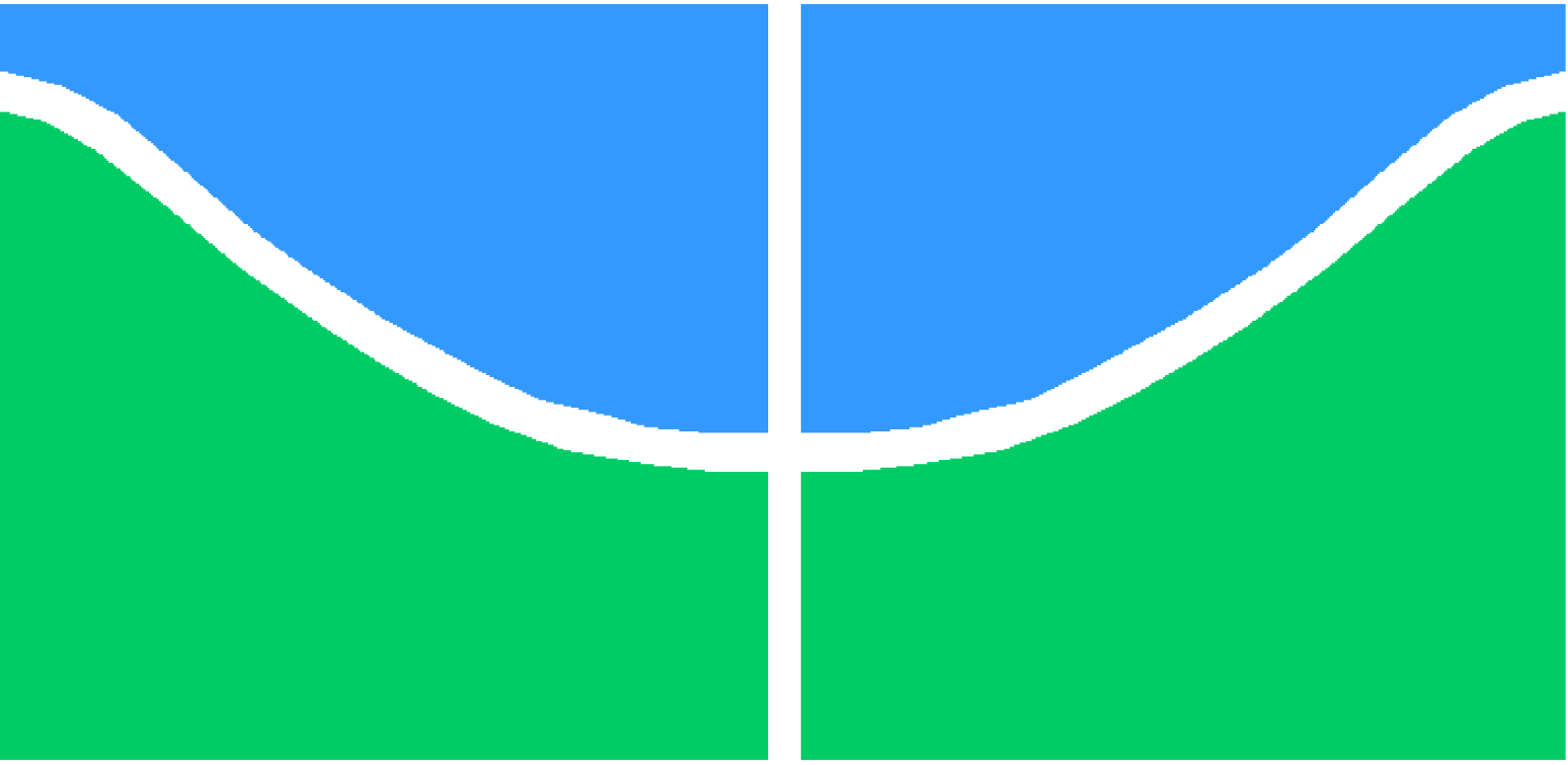

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Electronic Engineering

Development and evaluation of model methodology for Named-entity recognition using BERT

Author: Victor Hugo Ciurlino

advisor: Dr. Nilton Correia da Silva

Brasília, DF

2020

Victor Hugo Ciurlino

Development and evaluation of model methodology for Named-entity recognition using BERT

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

Orientador: Dr. Nilton Correia da Silva

Coorientador: Professor Pierre Gatien Florent André Guillou

Brasília, DF

2020

Victor Hugo Ciurlino
Development and evaluation of model methodology for Named-entity recognition using BERT/ Victor Hugo Ciurlino. – Brasília, DF, 2020-
40 p. : il.

Orientador: Dr. Nilton Correia da Silva

Final paper – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2020.

I. Dr. Nilton Correia da Silva. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Development and evaluation of model methodology for Named-entity recognition using BERT

CDU 02:141:005.6

Errata

Victor Hugo Ciurlino

Development and evaluation of model methodology for Named-entity recognition using BERT

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, 14th December 2020:

Dr. Nilton Correia da Silva
Advisor

Professor Pierre Gatien Florent André Guillou
Co-advisor

Dr Fabricio Ataides Braz
Guest

Brasília, DF
2020

Agradecimentos

Resumo

No âmbito jurídico, a leitura e extração de dados de peças dentro de um processo podem estender o período desde a entrada do processo até a sua conclusão. Tal tarefa atualmente é efetuada por pessoas, onde a metodologia consiste em abrir cada documento e, dentro das peças do mesmo, extrair informações chaves dos envolvidos naquele processo. Esse processo pode ser efetuado por máquinas, reduzindo o trabalho manual e otimizando o tempo para a extração dos dados. Foi realizado o treinamento de um modelo BERT(*Bidirectional Encoder Representations from Transformers*) pré treinado para a língua portuguesa, utilizando uma base de dados voltada para textos jurídicos disponibilizada pelo laboratório de ciências da computação da Universidade de Brasília. O modelo alcançou um *F1 score* de 0.918 e uma acurácia de 0.981. BERT se mostrou um modelo com grande potencial, alcançando uma pontuação acima de 90% com apenas 3 gerações de treinamento.

Palavras-chaves: BERT, NER, Name Entity Recognition,

Abstract

This is the english abstract.

Key-words: latex. abntex. text editoration.

List of Figures

List of Tables

List of abbreviations and acronyms

Summary

Introduction

Training a model for the NLP scenario from scratch is a costly process, both in terms of resources and time. For a long time, this was the only way to develop a model. However, in 2017 it was presented in an article "attention is all you need" developed by the google team, an architecture that would revolutionize the way that it create NLP models. Transformer is a deep learning model that use transfer learning from a pre-trained model for specific sub tasks. In other words, it is possible to use a model that has already been trained and, with just one extra step, train the model for a specific subtask.

In this paper, it will be studied and fine tuned BERT models to identify Named Entities, a subtask of information extraction from unstructured texts, from legal texts. It will be used pre-trained models in 3 different languages (Portuguese, English and Spanish) and a dataset provided by AILAB. After the fine-tuning, an evaluation of each model will be made individually and combining the English and the Spanish for comparison purposes.

The development of this approach allows to create models that reach the state of the art with less resource, opening possibilities for future models, since the methodology can be applied to all models with allows transfer learning.

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, the concept and history of NER is presented, then we present the architecture of Transformers, explaining how each step from it works. And finally, a schedule with the planning of the development of the tasks already carried out and the tasks to be carried out for the next semester.

Part I

Background

1 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*. This dataset is composed by 66 legal 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 (NerReIIberLEF2019) " 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, 2020) outperforms the previous state-of-the-art model (BiLSTMCRF+FlairBBP). it was trained three distinct models: Mul-

tilingual 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.

2 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.

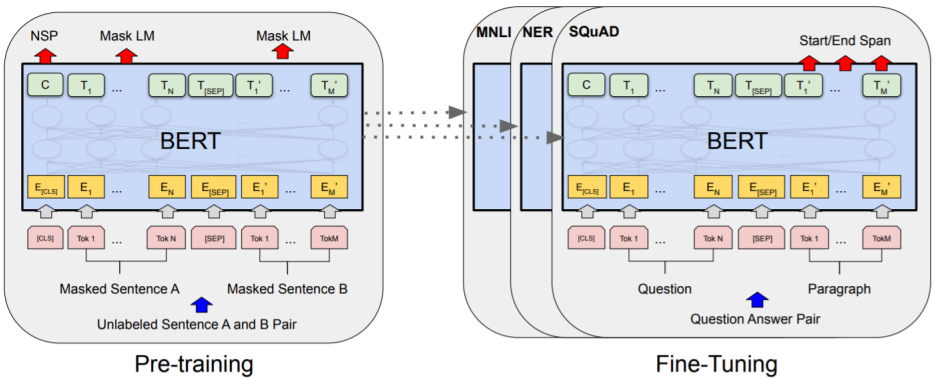

Figure 1 – Overall pre-training and fine-tuning procedures for BERT Source: (DEVLIN et al., 2019)

2.1 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, albeit 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.

2.1.1 Transformer architecture

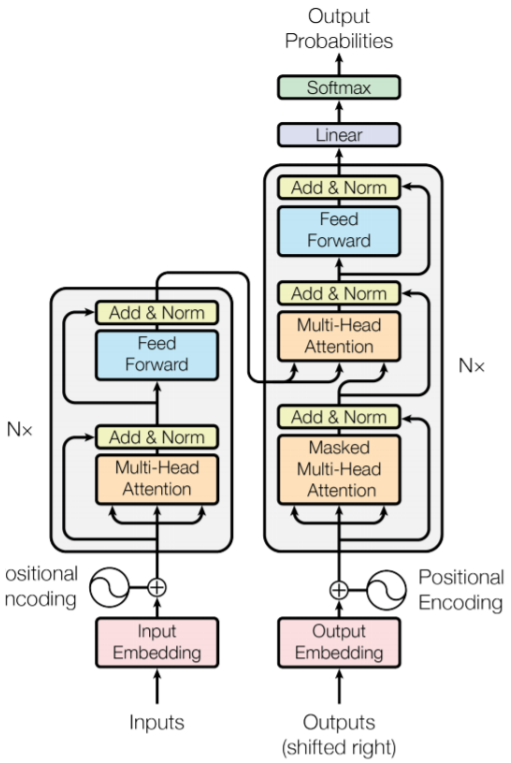

Figure 2 – Transformer architecture Source: (VASWANI et al., 2017)

2.1.1.1 Encoder

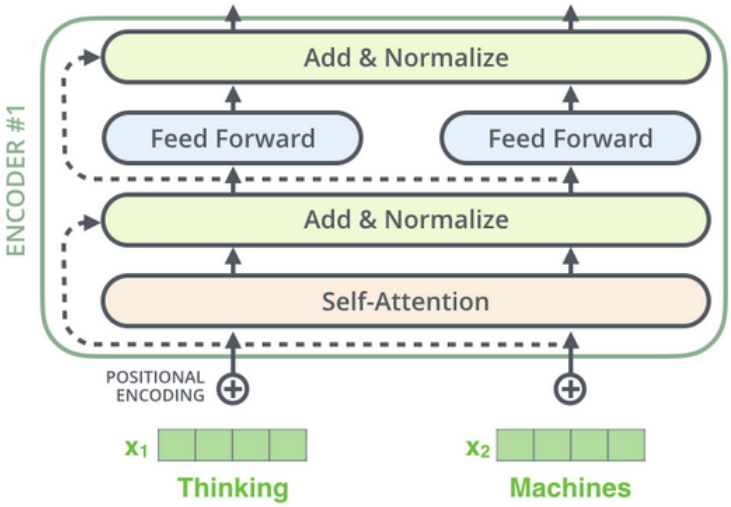

Figure 3 – Encoder structure Source: (JALAMMAR, 2019)

(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 $d_{model} = 512$.

2.1.1.2 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.

2.1.1.3 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.

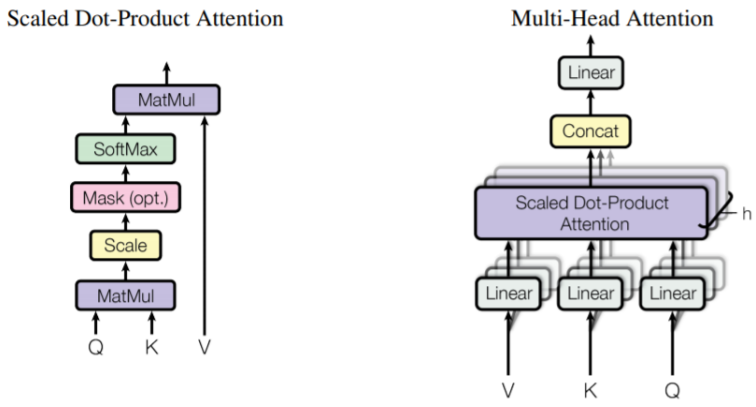

Figure 4 – (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel. Source: (VASWANI et al., 2017)

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:

$$Attention(Q, K, V) = softmax(\frac{Q.K^T}{\sqrt{d_k}}).V \tag{2.1}$$

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).

2.1.1.4 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.

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O \tag{2.2}$$

Where

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{2.3}$$

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.

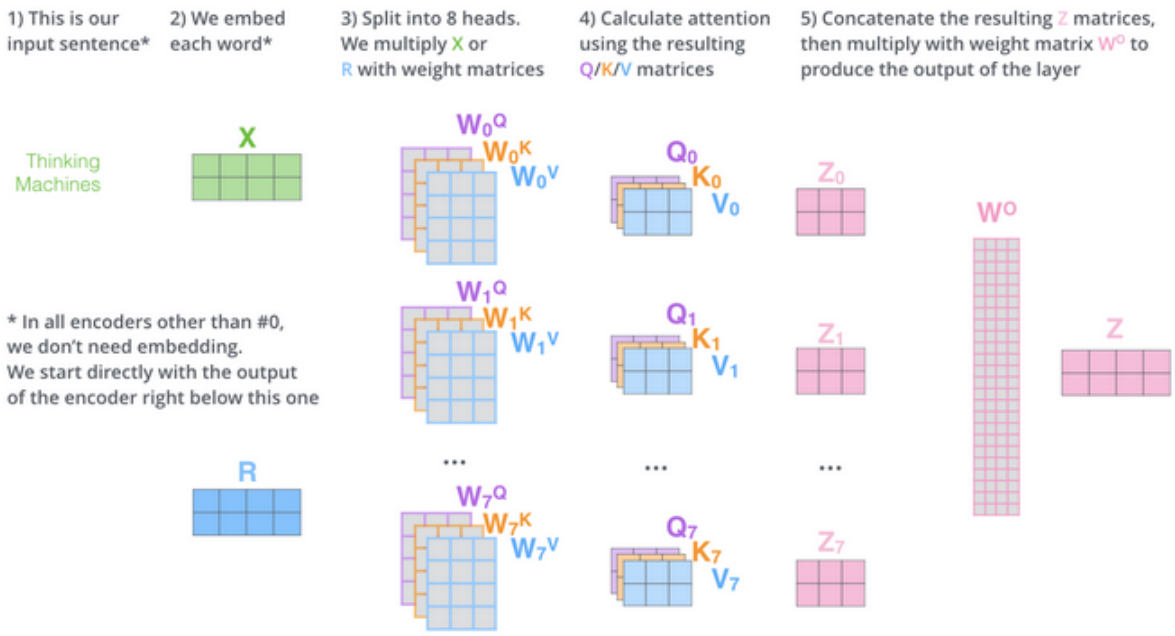

Figure 5 – Multi-head attention diagram Source: (JALAMMAR, 2019)

2.1.1.5 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.

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{2.4}$$

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 $d_{ff} = 2048$.

2.1.1.6 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.

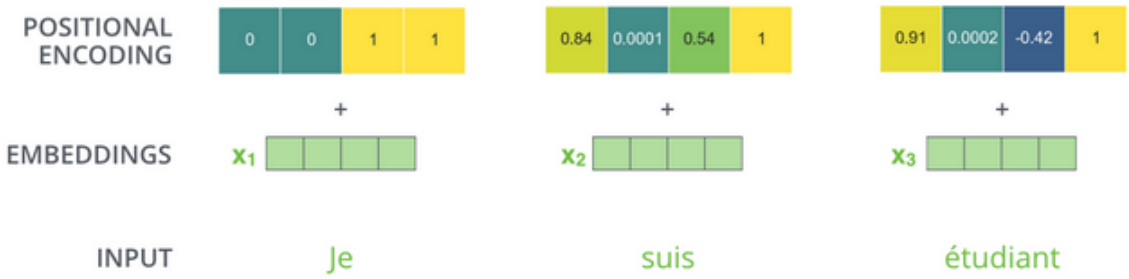

Figure 6 – Positional enconding Source: (JALAMMAR, 2019)

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:

$$PE_{(pos,2i)} = sin(\frac{pos}{1000^{(\frac{2i}{d_{model}})}}) \tag{2.5}$$

$$PE_{(pos,2i+1)} = cos(\frac{pos}{1000^{(\frac{2i}{d_{model}})}}) \tag{2.6}$$

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.

2.2 PRE-TRAINING METHOD

(DEVLIN et al., 2019)explains the difference between Pre training BERT 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).

2.2.1 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.

2.2.2 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.

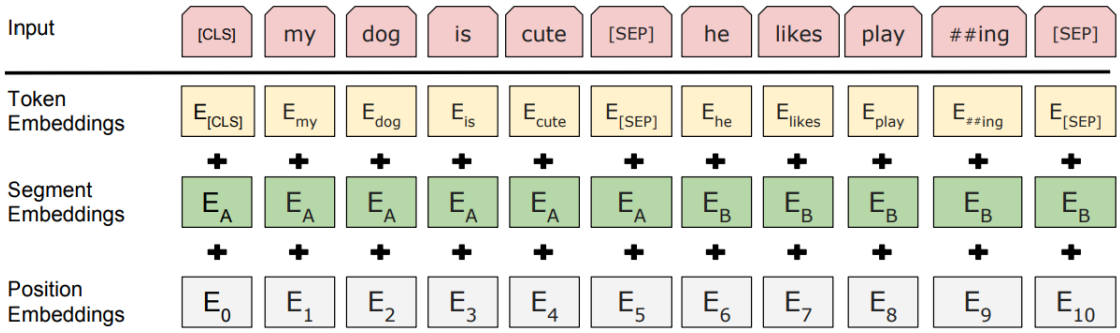

Figure 7 – BERT input representation. Source:(DEVLIN et al., 2019)

2.3 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 down-stream 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, training a few neural network. 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.

3 EVOLUTION

For this paper, it will be used a specific data set, made available by the laboratory *AILAB* for legal texts in Portuguese/Brazil. there is 13 different labels, with 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. The data set must be translated to English and Spanish before fine-tuning on the respective models.

Three models will be fine-tuned, using the entire architecture train approach, each one in a different language(Portuguese, English and Spanish), using this data set, All models will be trained using the same data set and optimizer for comparison purposes. AdamW will be used as optimizer, it has an excellent training speed and achieves good results with very little effort. The dataset will be splitted into Training, validation and test (80%, 10%, 10%, respectively), Allowing the models to be trained and validated for further analysis from the obtained results. For classification tasks, F1-SCORE it is commonly used, since that it has a balanced relationship between precision and recall.

$$F_1 = \frac{tp}{tp + 1/2(fp - fn)} \tag{3.1}$$

where

- tp: True Positive
- fp: False Positive
- fn: False negative

After the fine-tuning process, all metrics will be compared to determine the feasibility of the methodology. All steps will be done using *Google Colabs*, a online jupyter notebook that allows using a free GPU. For further application, a API will be developed for tests using all trained models.

3.1 SCHEDULE

Tasks	2020										2021			
	3	4	5	6	7	8	9	10	11	12	1	2	3	4
1.Theorical study														
2. Prototype development														
2.1 Scope definition														
2.2 Dataset definition														
2.3 Dataset preparation														
2.3.1 Data Cleanse														
2.3.2 New Column sentence counter														
2.4 Model definition														
2.5 Model configuration														
2.6 Model Finetuning														
2.7 Evaluation results														
2.8 Export and test Model														
3. TCC1 Writting														
3.1 Prepare LateX model														
3.2 Bibliography research														
3.3 Development Writting														
3.4 Schedule definition														
3.5 Introduction														
4.0 TCC1 Presentation														
5. New models research														
6. Prepare dataset for Other languages														
7. Models configuration														
8. Models Finetuning														
9. Models evaluation														
10. API development														
11. Article Writting														
12. TCC2 Writting														
13. TCC2 Presentation														

Table 1 – Schedule

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 25.

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 27.

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 13, 27, 34, and 35.

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 28.

HOCHREITER, S. et al. *Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies*. 2001. Cited on page 28.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural Computation*, v. 9, n. 8, p. 1735–1780, 1997. Cited on page 28.

JALAMMAR. *The Illustrated Transformer*. 2019. Disponível em: <http://jalammar.github.io/illustrated-transformer/>. Cited 5 times on page 13, 29, 30, 32, and 33.

KAISER Łukasz; SUTSKEVER, I. *Neural GPUs Learn Algorithms*. 2015. Cited on page 28.

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 28.

NADEAU, D.; SEKINE, S. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, v. 30, 01 2007. Cited on page 25.

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 28.

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 25.

SANTOS, J. et al. Multidomain contextual embeddings for named entity recognition. Pontifical Catholic University of Rio Grande do Sul, 2019. Cited on page 25.

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 33.

SOUZA, F.; NOGUEIRA, R.; LOTUFO, R. Portuguese named entity recognition using bert-crf. University of Campinas, 2020. Cited on page 25.

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 13, 27, 28, 29, 30, 31, 32, 33, and 34.