UNIVERSIDADE ESTADUAL DE CAMPINAS

Faculdade de Engenharia Elétrica e de Computação

Fábio Capuano de Souza

# BERTimbau: pretrained BERT models for Brazilian Portuguese

# BERTimbau: modelos BERT pré-treinados para Português Brasileiro

Campinas

2020

Fábio Capuano de Souza

# BERTimbau: pretrained BERT models for Brazilian Portuguese

# BERTimbau: modelos BERT pré-treinados para Português Brasileiro

Dissertation presented to the School of Electrical and Computer Engineering of the University of Campinas in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering, in the field of Computer Engineering.

Dissertação de mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica, na Área de Engenharia de Computação.

Supervisor: Prof. Dr. Roberto de Alencar Lotufo

Co-supervisor Dr. Rodrigo Frassetto Nogueira

Este exemplar corresponde à versão final da tese defendida pelo aluno Fábio Capuano de Souza, e orientada pelo Prof. Dr. Roberto de Alencar Lotufo e coorientada pelo Dr. Rodrigo Frassetto Nogueira.

Campinas

2020

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Rose Meire da Silva - CRB 8/5974

So89b

Souza, Fábio Capuano de, 1993-
BERTimbau: pretrained BERT models for brazilian portuguese / Fábio Capuano de Souza. – Campinas, SP : [s.n.], 2020.

Orientador: Roberto de Alencar Lotufo.
Coorientador: Rodrigo Frassetto Nogueira.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Processamento de linguagem natural (Computação). 2. Redes neurais profundas. 3. Aprendizado de máquina. I. Lotufo, Roberto de Alencar, 1955-. II. Nogueira, Rodrigo Frassetto, 1986-. III. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. IV. Título.

Informações para Biblioteca Digital

**Título em outro idioma:** BERTimbau: modelos BERT pré-treinados para português brasileiro
**Palavras-chave em inglês:**
Natural language processing
Deep neural networks
Machine learning
**Área de concentração:** Engenharia de Computação
**Titulação:** Mestre em Engenharia Elétrica
**Banca examinadora:**
Roberto de Alencar Lotufo [Orientador]
Anderson da Silva Soares
Cícero Nogueira dos Santos
**Data de defesa:** 18-12-2020
**Programa de Pós-Graduação:** Engenharia Elétrica

**Identificação e informações acadêmicas do(a) aluno(a)**
- ORCID do autor: https://orcid.org/0000-0003-0174-4506
- Currículo Lattes do autor: http://lattes.cnpq.br/2511138737338839

## COMISSÃO JULGADORA - DISSERTAÇÃO DE MESTRADO

**Candidato:** Fábio Capuano de Souza **RA:** 116735
**Data da defesa:** 18/12/2020

**Dissertation Title:** "BERTimbau: pretrained BERT models for Brazilian Portuguese".

**Titulo da Dissertação:** "BERTimbau: modelos BERT pré-treinados para Português Brasileiro".

# Acknowledgements

Firtly, I would like to thank my family: my parents João and Maria Silvia, and my brother Danilo, who provided all the opportunities and support my whole life without hesitation. I am priviledged to have you as family.

Secondly, I would like to thank my supervisor Prof. Lotufo and my co-supervisor Dr. Nogueira, for the continuous support throughout this research and writing of this dissertation and papers. Your guidance and vast knowledge were crucial to the development and achievements of this work. Dr. Nogueira, your excitement about researching is inspiring. In particular, I would like to emphasize my appreciation to Prof. Lotufo, for being a role model of respectfulness, calmness and mentorship. I could not have imagined having a better advisor.

Lastly, I would like to thank my friends. Guilherme and Giulia, for being by my side since childhood and sharing countless good times. Rodrigo, David, Louise, Renata and Giovanna, for always encouraging me and supporting me to pursue this degree. The DDG crew, for making my days better by always bringing a smile to my face and whose friendships grow stronger every year.

*“For millions of years, mankind lived just like the animals.*
*Then something happened which unleashed the power of our imagination.*
*We learned to talk and we learned to listen.”*
*(Stephen Hawking)*

# Abstract

Recent advances in language representation using neural networks and deep learning have made it viable to transfer the learned internal states of large pretrained language models (LMs) to downstream natural language processing (NLP) tasks. This transfer learning approach improves the overall performance on many tasks and is highly beneficial when labeled data is scarce, making pretrained LMs valuable resources specially for languages with few annotated training examples. In this work, we train BERT (Bidirectional Encoder Representations from Transformers) models for Brazilian Portuguese, which we nickname BERTimbau. We evaluate our models on three downstream NLP tasks: sentence textual similarity, recognizing textual entailment, and named entity recognition. Our models improve the state-of-the-art in all of these tasks, outperforming Multilingual BERT and confirming the effectiveness of large pretrained LMs for Portuguese. We release our models to the community hoping to provide strong baselines for future NLP research.

**Keywords**: Natural Language Processing; Deep neural networks; Machine learning; Language Model; Named Entity Recognition; Sentence Textual Similarity; Recognizing Textual Entailment.

# Resumo

Os avanços recentes em representação de linguagem usando redes neurais e aprendizado profundo permitiram que os estados internos aprendidos por grandes modelos de linguagem (ML) pré-treinados fossem usados no tratamento de outras tarefas finais de processamento de linguagem natural (PLN). Essa abordagem de transferência de aprendizado melhora a performance em diversas tarefas e é bastante benéfica quando há escassez de dados rotulados, fazendo com que MLs pré-treinados sejam recursos de grande utilidade, especialmente para línguas cujos conjuntos de dados de treinamento possuam poucos exemplos anotados. Nesse trabalho, nós treinamos modelos BERT (*Bidirectional Encoder Representations from Transformers*) para Português brasileiro, os quais apelidamos de BERTimbau. Nós avaliamos os modelos em três tarefas finais de PLN: similaridade semântica, inferência textual e reconhecimento de entidades nomeadas. Nossos modelos desempenham melhor do que o estado da arte em todas essas tarefas, superando o BERT multilíngue e confirmando a efetividade de grandes MLs para Português. Nós disponibilizamos nossos modelos para a comunidade de modo a promover boas bases de comparação para pesquisas futuras em PLN.

**Palavras-chaves**: Processamento de Linguagem Natural; Redes neurais profundas; Aprendizado de máquina; Modelo de Linguagem; Reconhecimento de Entidades Nomeadas; Similaridade semântica; Inferência textual.

# List of Figures

# List of Tables

# List of Acronyms

| | |
|---|---|
| ASSIN2 | *Segunda Avaliação de Similaridade Semântica e Inferência Textual* |
| BERT | Bidirectional Encoder Representation from Transformers |
| BPE | Byte Pair Encoding |
| CRF | Conditional Random Fields |
| LM | Language Model |
| MLM | Masked Language Modeling |
| MSE | Mean Squared Error |
| NER | Named Entity Recognition |
| NLI | Natural Language Inference |
| NLP | Natural Language Processing |
| NSP | Next Sentence Prediction |
| OOV | Out-of-vocabulary |
| RTE | Recognizing Textual Entailment |
| STS | Sentence Textual Similarity |
| TPU | Tensor Processing Unit |

# Contents

# 1 Introduction

Early deep neural network systems for natural language processing were proposed as an alternative to rule-based systems or classical machine learning approaches that combine heavy feature engineering with standard classification algorithms, such as Support Vector Machines (CORTES; VAPNIK, 1995). While the latter approach requires a rich set of hand-designed features tailored by domain specialists for each end task, the deep neural network approach proposes to pre-process the inputs as little as possible and train a model in an end-to-end fashion, learning to extract and compose relevant features automatically from data (COLLOBERT; WESTON, 2008; COLLOBERT *et al.*, 2011). This characteristic provides higher domain and language independence, making it directly applicable to diverse tasks and contributing to increase the popularity of neural NLP models.

A crucial component of most neural NLP models are word vector representations. Words are usually treated as atomic units associated to indices in a vocabulary. Each word is mapped to a dense vector representation known as word embeddings —a row of a matrix used as a look-up table. These representations are then fed to a neural model, such as a recurrent neural network. These vectors constitute a relevant part of the model's parameters and are optimized and learned during training. Since the first proposals, it has been shown that sharing word embeddings between tasks is beneficial (COLLOBERT; WESTON, 2008). Intuitively, multi-task learning stimulates learning representations that are useful to many tasks.

An important landmark for neural NLP models was initializing the word embeddings with pretrained representations from unsupervised tasks, such as word2vec (MIKOLOV *et al.*, 2013) and GloVe (PENNINGTON *et al.*, 2014). These embeddings are word-level representations trained on large corpora that capture semantic and syntactic features of words. Initializing the model with rich word vectors can push the performance of many NLP tasks compared to random initialization (KIM, 2014).

However, word embeddings are context agnostic. More recently, the internal states of language models were leveraged to extract richer word representations in context, such as ELMo (PETERS *et al.*, 2018) and Flair Embeddings (AKBIK *et al.*, 2018). These contextual embeddings are drop-in replacements for classical embeddings and improved the state-of-the-art on several language understanding tasks.

Despite these advances, a major limitation persisted. These word representations are used as input features to task-specific models that still have to be trained from scratch on each task of interest. This implies having to learn all model parameters using limited labeled training examples, which hinders the application of large models on scenarios of data scarcity and can lead to overfitting to training data.

Language modeling pretraining (DAI; LE, 2015) using only unlabeled data was shown to provide useful model initialization for all parameters, but only recently this transfer learning approach became largely adopted. Transfer learning consists of first training a model on a data-rich source task and then fine-tuning it on tasks of interest. The strategy of fine-tuning a large pretrained language model (LM) achieved state-of-the-art performances on a variety of NLP tasks (DEVLIN *et al.*, 2018; RADFORD *et al.*, 2018; RAFFEL *et al.*, 2019; YANG *et al.*, 2019). Aside from bringing performance improvements, transfer learning reduces the amount of labeled data needed for supervised learning on downstream tasks (HOWARD; RUDER, 2018; PETERS *et al.*, 2018).

Pretraining these large language models, however, is a resource-intensive process that requires huge amounts of unlabeled data and specialized hardware, with reports of models being trained using thousands of GPUs or TPUs and hundreds of GBs of raw textual data (LIU *et al.*, 2019; RAFFEL *et al.*, 2019). This resource barrier has limited the availability of these models, early on, to English, Chinese and multilingual models.

BERT (DEVLIN *et al.*, 2018), which uses the Transformer architecture (VASWANI *et al.*, 2017), among with its derived models, such as RoBERTa (LIU *et al.*, 2019) and Albert (LAN *et al.*, 2019), is one of the most adopted models. Despite having a multilingual BERT[1] model (mBERT) trained on 104 languages, much effort has been devoted on pretraining monolingual BERT and BERT-derived models on other languages, such as French (MARTIN *et al.*, 2019), Dutch (DELOBELLE *et al.*, 2020; VRIES *et al.*, 2019), Spanish (CAñETE *et al.*, 2020), Italian (POLIGNANO *et al.*, 2019), and others (BALY *et al.*, 2020; KURATOV; ARKHIPOV, 2019; NGUYEN; NGUYEN, 2020). Even though it is unfeasable to train monolingual models for every language, these works are motivated by the superior performance and resource efficiency of monolingual models compared to mBERT.

Large pretrained LMs can be valuable assets especially for languages that have few annotated resources but abundant unlabeled data, such as Portuguese. With that in mind, we train BERT models for Brazilian Portuguese — which we nickname BERTimbau— using data from brWaC (FILHO *et al.*, 2018), a large and diverse corpus of web pages. We evaluate our models on three NLP tasks: sentence textual similarity, recognizing textual entailment, and named entity recognition. BERTimbau improves the state-of-the-art on these tasks over multilingual BERT and previous monolingual approaches, confirming the effectiveness of large pretrained LMs for Portuguese. We make BERTimbau models available to the community on open-source libraries as to provide strong baselines for future research on NLP.

[1] https://github.com/google-research/bert/blob/master/multilingual.md

## 1.1 Objectives

In this work, we investigate the transfer learning capabilities of large neural language models for NLP tasks in Portuguese. First, we train monolingual BERT models of two size variants for Brazilian Portuguese. Then, we evaluate our models on downstream tasks and compare the results to multilingual models and previous monolingual approaches.

## 1.2 Contributions

The main contributions of this work are the assessment of the effectiveness of deep-learning-based transfer learning approaches for NLP tasks in Portuguese and open-sourcing our developed resources to the community. Our contributions can be outlined as follows:

- Resource-intensive pretraining of BERT models for Brazilian Portuguese using brWaC, a large corpus of unlabeled data.
- Evaluation on three downstream NLP tasks.
- State-of-the-art performances on ASSIN2 and First HAREM/MiniHAREM datasets.
- Experiments with fine-tuning and feature-based approaches for BERT.
- Comparison of the trained Portuguese models to the available multilingual model.
- Assessment of the vocabulary impacts on the models' performances.
- Open-sourcing resources to the community.

## 1.3 Organization of the thesis

This thesis is organized as follows: in Chapter 2, we review relevant concepts used throughout this work, such as the BERT architecture and its pretraining procedures. in Chapter 3, we present and discuss the related work. In Chapter 4, we describe BERTimbau's pretraining methods, such as the pretraining data, and the vocabulary generation, and the evaluation procedures, such as evaluation tasks, datasets, architectures and metrics. Then, in Chapter 5, we describe our experiments and present and analyze our results. Lastly, we make our conclusions in Chapter 6.

# 2 Concepts overview

## 2.1 Language modeling

The task of language modeling consists of estimating the joint probability of a sequence of tokens, $P(x_1, \ldots, x_N)$. In neural language modeling (BENGIO *et al.*, 2003), it is often factorized as

$$P(\mathbf{x}) = \prod_{t=1}^{N} P(x_t \mid x_1, \ldots, x_{t-1}) \tag{2.1}$$

With the factorization, the problem reduces to estimating each conditional factor, that is, predicting the next token given a history of previous tokens. This formulation sees words from left to right and is called forward language model. Similarly, a backward LM sees the tokens in reversed order, which results in predicting the previous token given the future context, $P(x_t \mid \mathbf{x}_{>t})$.

## 2.2 Transfer learning

The most established way to reduce the data requirements and improve the performance of neural networks is to resort to transfer learning techniques. Transfer learning consists of training a base model on a base dataset and task and then transferring the learned features to another model to be trained on a target task of interest (YOSINSKI *et al.*, 2014). In other words, the second model has its weights initialized with the weight values of the trained base model. The base dataset is generally much larger than the target dataset and the tasks are often related, as the main goal is to learn features from the base task that are useful for both tasks. In computer vision, an effective recipe for transfer learning is to pretrain a model on ImageNet (DENG *et al.*, 2009) dataset, which contains 1 million labeled images for classification in 1000 classes, and then fine-tune it on a task of interest. This way, the model can leverage the learned representations from the base task and only a fraction of the model parameters have to be learned from scratch to the final task and dataset.

For NLP, the choice of a base task is an active research area. Language modeling pretraining (DAI; LE, 2015) has been successfully proposed as a base task. It has been shown that the features learned by this general task is highly transferable to a wide range of downstream tasks, resembling a multitask objective that allows zero-shot learning on many tasks (RADFORD *et al.*, 2019). Its self-supervised nature is also beneficial, since training examples can be automatically generated from raw textual data, which is usually readily available is most languages and domains and cheaper to gather, as opposed to other supervised tasks that require manually labeled examples.

## 2.3 Subword tokenization

Subword tokenization has a vocabulary of subword units that can comprise characters, sequences of characters of variable length and even entire words. The tokenization process consists of segmenting the text into subword units (also referred to as subtokens or subword tokens from hereon). The vocabulary is often generated in a data-driven iterative process, such as the adapted Byte Pair Encoding (BPE) algorithm (SENNRICH *et al.*, 2016a). In BPE, the vocabulary is initialized with all characters in a corpus, then the corpus is tokenized and the most frequent pair of adjacent symbols is merged and included in the vocabulary. This is repeated until a desired vocabulary size is reached. This method, along with character-level tokenization, is more robust to out-of-vocabulary (OOV) words, since the worst case scenario when tokenizing an arbitrary word is to segment it into characters.

WordPiece (SCHUSTER; NAKAJIMA, 2012b) and SentencePiece (KUDO; RICHARDSON, 2018) are commonly used subword-level tokenization methods. In WordPiece, text is first divided into words and the resulting words are then segmented into subword units. For each segmented word that is composed of multiple subword units, all units following the first one are prefixed with "`##`" to indicate word continuation. SentencePiece takes a different approach: white space characters are replaced by a meta symbol "▁" (`U+2581`) and the sequence is then segmented into subword units. The meta symbol marks where words start and allows for lossless detokenization. Possible tokenizations of the phrase "Oscar Niemeyer was born in 1907" in WordPiece and SentencePiece, supposing the words "Niemeyer" and "1907" are not in the vocabulary, are, respectively:

"Oscar" "Nie" "##meyer" "was" "born" "in" "19" "##07"

"▁Oscar" "▁Nie" "meyer" "▁was" "▁born" "▁in" "▁19" "07"

## 2.4 BERT

BERT (Bidirectional Encoder Representation from Transformers) (DEVLIN *et al.*, 2018) is a language model based on the Transformer Encoder (VASWANI *et al.*, 2017) architecture. BERT's main contribution is establishing a framework to pretrain deep bidirectional word representations that are jointly conditioned on both left and right contexts, in contrast to preceding works on language modeling that employ unidirectional LMs. For example, OpenAI GPT (RADFORD *et al.*, 2018) pretrains a Transformer Decoder using left-to-right language modeling, and ELMo (PETERS *et al.*, 2018) concatenates representations from independent left-to-right and right-to-left language models. BERT demonstrates that bidirectionality is important for both sentence-level and token-level tasks by improving the state-of-the-art on several benchmarks. This unidirectionality limitation is overcome by using a modified language modeling objective called Masked Language Modeling, that resembles a denoising objective and which we detail in Section 2.4.2.

Before delving into specifics of the model architecture, BERT can be seen as a black box model that maps a sequence of tokens $\mathbf{x}$ into a sequence of encoded token representations:

$$(x_1, \ldots, x_n) \mapsto (\mathbf{c}, \mathbf{T}_1, \ldots, \mathbf{T}_n) \tag{2.2}$$

where $x_i$ is a token of a vocabulary $\mathcal{V}$ of size $V$, $\mathbf{T}_i \in \mathbb{R}^H$ is the encoded representation of the i-th token $x_i$ in the sequence $\mathbf{x}$, $\mathbf{c} \in \mathbb{R}^H$ is an aggregate representation of the entire sequence, and $H$ is the model's hidden size. To apply BERT to a task of interest, the representations $\mathbf{c}$ or $\mathbf{T}_i$ are used as inputs to a task specific model, which can be as simple as a linear transformation.

BERT usage in downstream tasks is composed of two stages: pretraining and fine-tuning. In the pretraining stage, the model is trained from scratch on self-supervised tasks to learn useful representations $\mathbf{c}$ and $\mathbf{T}_i$. This stage is computationally intensive and has to be performed only once. In the fine-tuning stage, a task specific model is attached to the pretrained BERT and the whole model is trained on the task of interest. In the following subsections, we describe BERT's architecture, the pretraining procedure and the fine-tuning on downstream tasks.

### 2.4.1 Model architecture

BERT consists of a Transformer Encoder, which is a part of the Transformer architecture (VASWANI *et al.*, 2017) that comprises an Encoder and a Decoder. The Transformer was proposed as an alternative to convolutional models and well established recurrent models, such as LSTM and GRU. By relying only on self-attention mechanisms instead of recurrence, Transformer models can see all input tokens at once and, hence, can model dependencies in long sequences in constant time while enabling much higher parallelization. These features enable the training of deeper models and longer sequences. Figure 1 shows a diagram of BERT's architecture. Each component will be described in the following sections.

#### 2.4.1.1 Input representation

BERT can receive as input a single sentence or a sentence pair. Borrowing the notation of the original work, throughout the rest of this work, a "sentence" can be any arbitrary contiguous text span, rather than a linguistic sentence. A "sequence", in its turn, refers to the input token sequence, which can be composed of one or two sentences.

An input sequence is generated by packing the input sentences using two special tokens, `[CLS]` and `[SEP]`. A single sentence is represented as

$$\texttt{[CLS]}\ x_1\ \cdots\ x_n\ \texttt{[SEP]}\ ,$$

and a sentence pair, as

$$\texttt{[CLS]}\ x_1\ \cdots\ x_n\ \texttt{[SEP]}\ y_1\ \cdots\ y_m\ \texttt{[SEP]}\ .$$

The `[SEP]` token is simply a separator to mark the end of a sentence. The motivation of the `[CLS]` token will be discussed in Section 2.4.1.5.

#### 2.4.1.2 Input embeddings

A sequence of tokens has to be converted to vector representations before inputting into the model. In addition to simple token embeddings, that maps each token of the vocabulary to a corresponding embedding vector, BERT embeds each token using 2 extra embeddings: position and segment embeddings.

The inclusion of position embeddings is directly associated to the non-sequential nature of the Transformer architecture. While recurrent models inherently take sequence order into account by consuming one word at a time, the Transformer operates on sets of vectors and has no notion of sequence order. Therefore, some information about the relative or absolute position of the tokens in the sequence has to be included at the input. This position embedding encodes absolute position information by associating each position $i \in \{1, \cdots, S\}$ to a learned vector embedding, and replaces the Transformer's fixed positional encoding. The maximum sequence length $S$ is a hyperparameter and is set to 512 in the original work.

Segment embeddings, in their turn, are related to the input representation and are used to distinguish two sentences A and B inside a sequence. Each token of sentence A is embedded using segment A embedding, and each token of sentence B is embedded using segment B embedding. Table 1 shows an example of the embedding process for a sentence pair. Formally, the embedding vector for a token $x_i$ is given by

$$\mathbf{E}(x_i) = \text{LayerNorm}(\mathbf{E}_V^{x_i} + \mathbf{E}_{pos}^{i} + \mathbf{E}_{seg}^{A|B}) \in \mathbb{R}^H\ , \tag{2.3}$$

where LayerNorm is layer normalization (BA *et al.*, 2016), $\mathbf{E}_V \in \mathbb{R}^{V\times H}$ is the matrix of token embeddings for the vocabulary $\mathcal{V}$, $\mathbf{E}_{pos} \in \mathbb{R}^{S\times H}$ is the matrix of position embeddings,

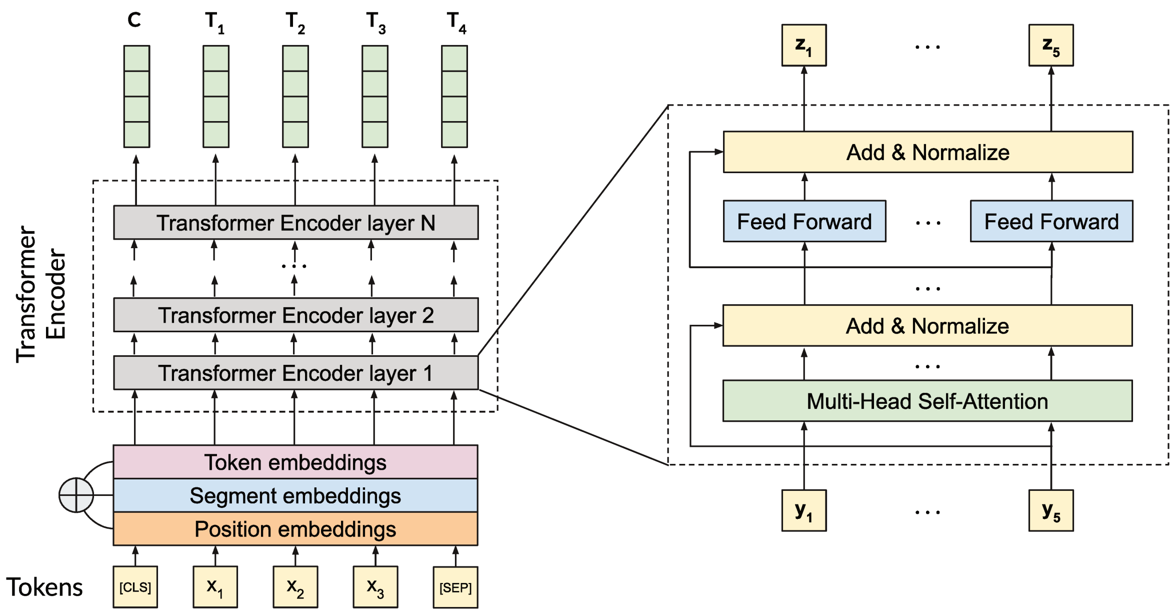

Figure 1 – Diagram of BERT model for an input sequence of 5 tokens (left), with a zoom representing the operations inside of a Transformer Encoder layer (right).

$\mathbf{E}_{seg} \in \mathbb{R}^{2\times H}$ is the matrix of segment embeddings, and the superscript represents row indexing to select the corresponding token id, position and segment for token $x_i$.

Table 1 – Example of token, position and segment embeddings encoding for the sentence pair "*Ana foi ao mercado.*" and "*Ela comprou pão.*". Matrices $\mathbf{E}_V$, $\mathbf{E}_{pos}$ and $\mathbf{E}_{seg}$ refer to Eq. (2.3).

| Token | [CLS] | Ana | foi | ao | mercado | . | [SEP] | Ela | comprou | pão | . | [SEP] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Token id | 100 | 3412 | 262 | 320 | 2918 | 119 | 102 | 1660 | 10107 | 14525 | 119 | 102 |
| Position | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Segment | A | A | A | A | A | A | A | B | B | B | B | B |
| Token embedding | $\mathbf{E}_V^{100}$ | $\mathbf{E}_V^{3412}$ | $\mathbf{E}_V^{262}$ | $\mathbf{E}_V^{320}$ | $\mathbf{E}_V^{2918}$ | $\mathbf{E}_V^{119}$ | $\mathbf{E}_V^{102}$ | $\mathbf{E}_V^{1660}$ | $\mathbf{E}_V^{10107}$ | $\mathbf{E}_V^{14525}$ | $\mathbf{E}_V^{119}$ | $\mathbf{E}_V^{102}$ |
| Position embedding | $\mathbf{E}_{pos}^{1}$ | $\mathbf{E}_{pos}^{2}$ | $\mathbf{E}_{pos}^{3}$ | $\mathbf{E}_{pos}^{4}$ | $\mathbf{E}_{pos}^{5}$ | $\mathbf{E}_{pos}^{6}$ | $\mathbf{E}_{pos}^{7}$ | $\mathbf{E}_{pos}^{8}$ | $\mathbf{E}_{pos}^{9}$ | $\mathbf{E}_{pos}^{10}$ | $\mathbf{E}_{pos}^{11}$ | $\mathbf{E}_{pos}^{12}$ |
| Segment embedding | $\mathbf{E}_{seg}^{A}$ | $\mathbf{E}_{seg}^{A}$ | $\mathbf{E}_{seg}^{A}$ | $\mathbf{E}_{seg}^{A}$ | $\mathbf{E}_{seg}^{A}$ | $\mathbf{E}_{seg}^{A}$ | $\mathbf{E}_{seg}^{A}$ | $\mathbf{E}_{seg}^{B}$ | $\mathbf{E}_{seg}^{B}$ | $\mathbf{E}_{seg}^{B}$ | $\mathbf{E}_{seg}^{B}$ | $\mathbf{E}_{seg}^{B}$ |

#### 2.4.1.3 Transformer Encoder

A Transformer Encoder is a stack of $L$ identical layers that alternates self-attention and feed-forward operations on an input sequence. Each encoder layer maps an input sequence $\mathbf{X} = (\mathbf{x_1}, \ldots, \mathbf{x_n})$ to a sequence of encoded representations $\mathbf{Z} = (\mathbf{z_1}, \ldots, \mathbf{z_n})$. An encoder layer is composed of two sub-layers: a Multi-Head Self-Attention and a Feed Forward layers. Each sub-layer has a residual connection, dropout (SRIVASTAVA *et al.*, 2014) and is followed by layer normalization. That is, the output of a sub-layer is given by

$$\mathrm{SubLayer}(\ell, \mathbf{U}) = \mathrm{LayerNorm}(\mathrm{Dropout}(\ell(\mathbf{U})) + \mathbf{U})\,, \tag{2.4}$$

where $\ell(\,\cdot\,)$ is either a Multi-Head Self-Attention or a Feed Forward layer and $\mathbf{U} \in \mathbb{R}^{n\times H}$ is either the output sequence of the previous encoder layer or of the previous sub-layer. To facilitate the residual connections, both sub-layers' inputs and outputs have dimension $H$.

The Multi-Head Self-Attention layer — that will be described in the next subsection— receives as input all tokens from the output of the previous encoder layer. In the attention layer, any position of the sequence can attend to all positions of the previous encoder layer. In other words, the attention mechanism allows each position of the sequence to incorporate contextual information from across the sequence.

The Feed Forward layer, in its turn, is composed of two consecutive fully-connected layers and is parameterized as

$$\mathrm{FNN}(\mathbf{u}) = \mathrm{Act}(\mathbf{u}\,\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2\,, \tag{2.5}$$

where $\mathbf{u} \in \mathbb{R}^{H}$, $\mathbf{W}_1 \in \mathbb{R}^{H\times d_{ff}}$, $\mathbf{b}_1 \in \mathbb{R}^{d_{ff}}$, $\mathbf{W}_2 \in \mathbb{R}^{d_{ff}\times H}$, $\mathbf{b}_2 \in \mathbb{R}^{H}$, $d_{ff}$ is called intermediate dimension and $\mathit{Act}(\,\cdot\,)$ is an activation function — ReLU in the original Transformer and GeLU (HENDRYCKS; GIMPEL, 2016) in BERT.

Note that the input $\mathbf{u}$ is a vector and the layer is applied to each position of the sequence separately and identically, as opposed to the attention that operates on the entire sequence simultaneously. The output $\mathbf{Z}^{(i)}$ of the i-th Encoder layer is given by

$$\mathbf{Z}^{(0)} = \mathbf{E}(\mathbf{x}) \tag{2.6}$$

$$\mathbf{Y}^{(i)} = \mathrm{SubLayer}(\mathrm{MultiHead}(\mathbf{Z}^{(i-1)},\, \mathbf{Z}^{(i-1)},\, \mathbf{Z}^{(i-1)}),\, \mathbf{Z}^{(i-1)}) \tag{2.7}$$

$$\mathbf{Z}^{(i)} = \mathrm{SubLayer}([\mathrm{FFN}(\mathbf{y}_1^{(i)}),\, \ldots,\, \mathrm{FFN}(\mathbf{y}_n^{(i)})],\, \mathbf{Y}^{(i)}) \tag{2.8}$$

$$\mathbf{Z} = \mathbf{Z}^{(L)} \tag{2.9}$$

where $\mathbf{E}(\mathbf{x}) \in \mathbb{R}^{n\times H}$ is the embedded input token sequence, the superscript $(i)$ indicates the i-th layer, subscript $i$ indicates i-th position in the sequence and $[\,\cdot\,]$ represents concatenation operation.

#### 2.4.1.4 Multi-Head Self-Attention

This subsection describes Multi-Head Self-Attention in a bottom-up approach by first defining an attention mechanism. The attention mechanism was first proposed in neural machine translation (BAHDANAU *et al.*, 2014) to enable models to relate signals from arbitrary positions in long sequences using a constant number of operations. Let $\mathbf{q}$ be a query vector and $(\mathbf{k}_i, \mathbf{v}_i)$ be key-value vector pairs. The output of an attention mechanism is a linear combination of the value vectors,

$$\mathbf{a} = \sum_i \alpha_i \mathbf{v}_i\,, \tag{2.10}$$

where $\alpha_i$ is the weight associated to value $\mathbf{v}_i$. The weights are computed by

$$\alpha_i = \frac{\exp(g(\mathbf{q}, \mathbf{k}_i))}{\sum_{i'} \exp(g(\mathbf{q}, \mathbf{k}_i'))}\,, \tag{2.11}$$

where $g$ is a scoring or compatibility function that is calculated with the query and all keys. Transformer uses Dot-Product attention (LUONG *et al.*, 2015), whose compatibility function is the dot-product of query and key vectors, and introduces an extra scaling factor:

$$g(\mathbf{q}, \mathbf{k}_i) = \frac{\mathbf{q}^\intercal \mathbf{k}_i}{\sqrt{d_k}}\,, \tag{2.12}$$

where $d_k$ is the dimension of the query and key vectors. The scaling factor was included to reduce the magnitude of the dot products for large values of $d_k$, which could push the softmax function (see Eq. 2.11) into regions where it has small gradients (VASWANI *et al.*, 2017).

In practice, dot-product attention can be computed on a set of queries simultaneously by packing the queries, keys and values in matrices. Let $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ be the query, key and value matrices. The matrix of outputs is computed by:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\intercal}{\sqrt{d_k}}\right)\mathbf{V}\,, \tag{2.13}$$

where $\mathbf{Q} \in \mathbb{R}^{n \times d_k}$ , $\mathbf{K} \in \mathbb{R}^{m \times d_k}$, $\mathbf{V} \in \mathbb{R}^{m \times d_v}$, $n$ is the number of queries, $m$ is the number of keys-value pairs and softmax is performed on each row. The attention output has dimension $\mathbb{R}^{n \times d_v}$.

Multi-head attention (VASWANI *et al.*, 2017) consists of performing several attention functions in parallel for $A$ individual *heads*, and then combining its outputs. To keep computational cost constrained, one can apply linear projections on the queries, keys and values to reduced dimensions $d'_q$, $d'_k$ and $d'_v$, respectively. The Multi-Head Attention can be defined as:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = [head_1; \ldots; head_A]\mathbf{W}^O, \tag{2.14}$$

$$head_j = \text{Attention}(\mathbf{Q}\mathbf{W}_j^Q, \mathbf{K}\mathbf{W}_j^K, \mathbf{V}\mathbf{W}_j^V), \tag{2.15}$$

where $[\,\cdot\,]$ is concatenation operation, $\mathbf{W}_j^Q \in \mathbb{R}^{d_i \times d'_q}$, $\mathbf{W}_j^K \in \mathbb{R}^{d_i \times d'_k}$, $\mathbf{W}_j^V \in \mathbb{R}^{d_i \times d'_v}$ are the input projection matrices for head $j$, $\mathbf{W}^O \in \mathbb{R}^{Ad'_v \times d_o}$ is the output projection matrix, and $d_i$ and $d_o$ are the input and output dimensions, respectively. To keep $d_i = d_o = H$, the inner dimensions are set to $d'_q = d'_k = d'_v = H/A$. Finally, self-attention implies that queries, keys and values vectors are all projections from the same input vectors using distinct linear projection matrices, as denoted by Eq. (2.7). Figure 2 shows three-dimensional tensor representations of an input sequence as it is processed inside BERT.

#### 2.4.1.5 Output representation

As described in 2.4.1.1, the `[CLS]` token is inserted as the first token of every input sequence. Its purpose is to produce an aggregate encoded representation $\mathbf{c}$ of the entire sequence to be used in sequence-level tasks. Given the output of the Transformer Encoder block $\mathbf{Z} = (\mathbf{z}_1, \cdots, \mathbf{z}_n)$, the encoded representation of `[CLS]` is pooled using a linear layer with hyperbolic tangent activation

$$\mathbf{c} = \tanh(\mathbf{z}_1 \mathbf{W}_c + \mathbf{b}_c)\,, \tag{2.16}$$

where $\mathbf{c}, \mathbf{z}_1, \mathbf{b}_c \in \mathbb{R}^H$, and $\mathbf{W}_c \in \mathbb{R}^{H \times H}$.

For token-level tasks, the encoded representation $\mathbf{T}_i$ of each token is taken directly from its corresponding position in $\mathbf{Z}$, as illustrated in Figure 1.

### 2.4.2 Pretraining stage

In the pretraining stage, BERT is trained on two self-supervised tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). Each pretraining example is generated by concatenating two sentences A, with tokens $(a_1, \ldots, a_m)$ and B, with tokens $(b_1, \ldots, b_o)$, as

$$x = (\texttt{[CLS]}\ a_1\ \ldots\ a_m\ \texttt{[SEP]}\ b_1\ \ldots\ b_o\ \texttt{[SEP]})\,. \tag{2.17}$$

Given a sentence A from the corpus, in 50% of the time the sentence B is the sentence that follows sentence A, forming a contiguous piece of text, and 50% of the time B is a random sentence sampled from a distinct document of the corpus. This choice defines the ground truth label for the NSP task

$$y_{\text{NSP}} = \mathbb{1}(\text{B is the continuation of A}) \tag{2.18}$$

where $\mathbb{1}(\cdot)$ is the indicator function.

For MLM, each example is then corrupted by first selecting a random set of positions (integers from 1 to $n = |x|$) $\mathbf{m} = [m_1, \ldots, m_k]$, $k = \lceil 0.15n \rceil$. The tokens in each of these positions are then replaced by 1 of 3 options: a special `[MASK]` token with 80% probability, a random token from the vocabulary with 10% probability or, otherwise, keeping the original token. In this selection step we use whole word masking: if a token from a word composed of multiple subword units is chosen to be corrupted, all other subword units are also corrupted. The original tokens from the positions $m_i$ are saved and serve as labels for the MLM task

$$y_{\text{MLM}} = [x^*_{m_1}, \ldots, x^*_{m_k}]\,. \tag{2.19}$$

The final pretraining example can be represented as a tuple $(x^{\text{corrupted}}, \mathbf{m}, y_{\text{MLM}}, y_{\text{NSP}})$.

The corrupted sequences $x^{\text{corrupt}}$ are used as inputs to BERT and the output encoded representations are used as inputs for the pretraining tasks' heads that are attached during this pretraining stage.

#### 2.4.2.1 MLM head and loss function

The MLM task consists of, for every corrupted position $i \in \mathbf{m}$, predicting the original token $x_i^*$ back by a classification over the entire vocabulary $\mathcal{V}$ of size $V$. For each corrupted position $i$, the MLM head computes the classification probability distribution $p_{M_i}$ over the vocabulary tokens by

$$\mathbf{h}_{Mi} = \text{LayerNorm}(\text{GeLU}(\mathbf{T}_i\,\mathbf{W}_M + \mathbf{b}_m)) \tag{2.20}$$

$$p_{Mi} = \text{Softmax}(\mathbf{h}_{Mi}\mathbf{E}_V^T + \mathbf{b}_v)\,, \tag{2.21}$$

where $\mathbf{W}_M \in \mathbb{R}^{H\times H}$, $\mathbf{b}_m \in \mathbb{R}^H$, $\mathbf{b}_v \in \mathbb{R}^V$, and $\mathbf{E}_V \in \mathbb{R}^{V\times H}$ is the matrix of input token embeddings, that is, is a dense layer with tied weights with the input token embeddings.

The MLM loss for an example is the mean cross entropy loss over all corrupted positions

$$\mathcal{L}_{MLM}(x^{\text{corrupt}}, \theta) = \frac{1}{k}\sum_{i\in\mathbf{m}} -log\; p_{Mi}(x_i^* \mid x^{\text{corrupt}})\,, \tag{2.22}$$

where $p_{Mi}(x_i^* \mid x^{\text{corrupt}})$ is the model's estimated probability for the original token $x_i^*$ at position $i$ and $k$ is the number of masked tokens.

#### 2.4.2.2 NSP head and loss function

The NSP task is a binary classification to predict if the sentence B is the actual continuation of sentence A or a random sentence. The NSP head consists of a single linear layer that projects the vector $\mathbf{c}$ to the probabilities of the two classes, using a softmax activation:

$$p_N = \text{Softmax}(\mathbf{c}\mathbf{W}_N + \mathbf{b}_n)\,, \tag{2.23}$$

where $\mathbf{W}_N \in \mathbb{R}^{H\times 2}$ and $\mathbf{b}_n \in \mathbb{R}^2$. Again, the NSP loss for an example is the cross entropy loss

$$\mathcal{L}_{NSP}(x^{\text{corrupt}}, \theta) = -\left(y_{NSP}\; log\; p_N(\hat{y}_{NSP} = 1) + (1 - y_{NSP})\; log(1 - p_N(\hat{y}_{NSP} = 0))\right)\,. \tag{2.24}$$

The total pretraining loss is the sum of MLM and NSP losses.

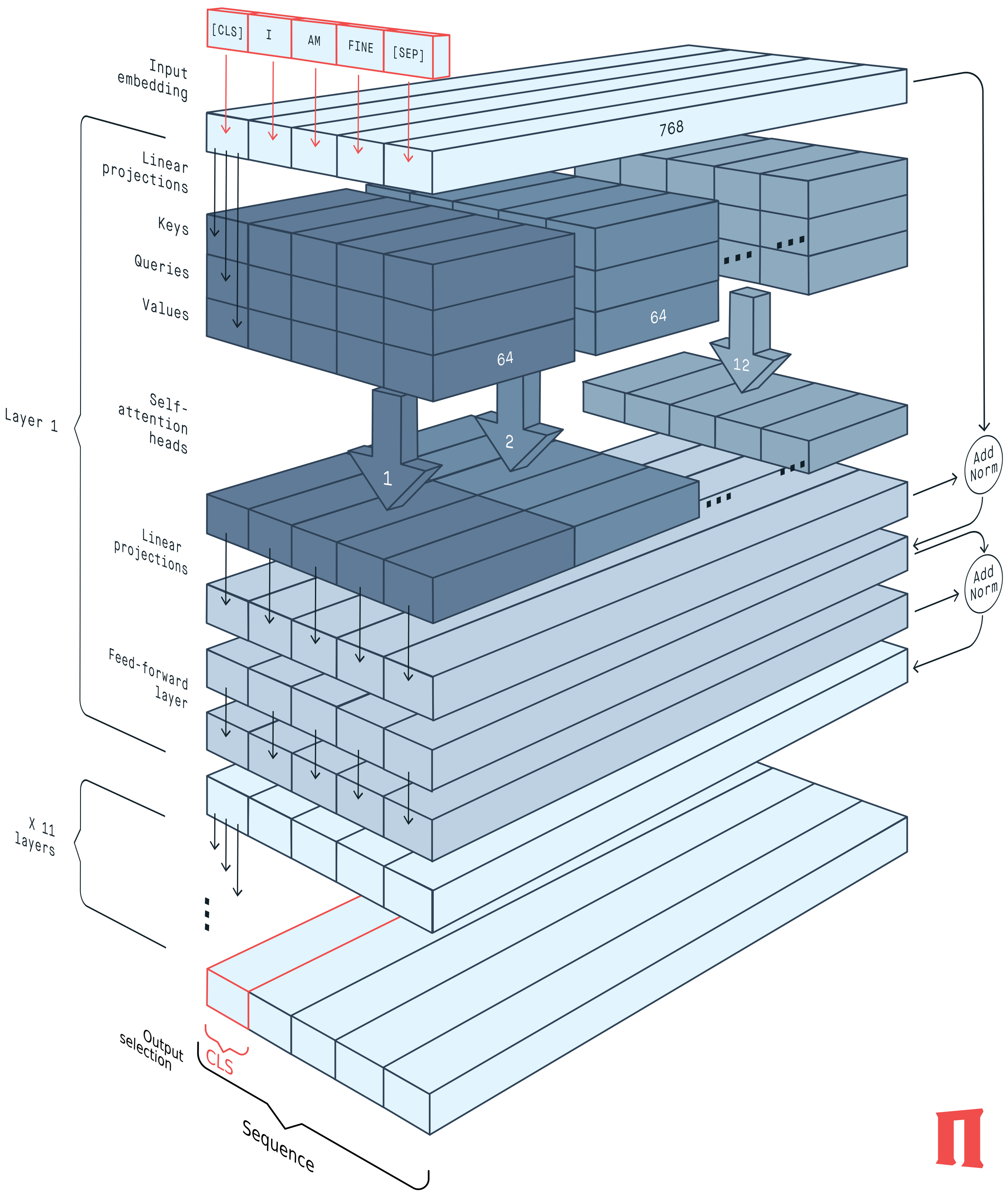

Figure 2 – Three dimensional tensor representation of the operations performed by a BERT model for a single input sequence with the words “I am fine”. Each attention head is represented by a distinct shade of blue. The tensor dimensions, the number of attention heads and encoder layers refer to the BERT Base variant (12 layers, hidden dimension 768, attention dimension 64, 12 attention heads). Figure produced by Peltarion company and published on its social networks.

# 3 Related Work

## 3.1 Word vector representations

Word vector representations are a crucial component of many neural NLP models. Classic word embeddings (MIKOLOV *et al.*, 2013; PENNINGTON *et al.*, 2014) are static non-contextualized word-level representations that capture semantic and syntactic features using large corpora. More recently, contextual embeddings, such as ELMo (PETERS *et al.*, 2018) and Flair Embeddings (AKBIK *et al.*, 2018), leverage the internal states of language models to extract richer word representations in context. These embeddings are used as features to task-specific models and consist of a shallow transfer learning approach, since downstream models have to be trained from scratch on each task of interest.

## 3.2 Deeper transfer learning

Deeper transfer learning techniques for NLP emerged by successfully fine-tuning large pretrained LMs with general purpose architectures, such as the Transformer (VASWANI *et al.*, 2017), replacing task-specific models. Language modeling pretraining is shown to resemble a multitask objective that allows zero-shot learning on many tasks (RADFORD *et al.*, 2019). This pretraining stage benefits from diverse texts and can be further improved by additional pretraining with unlabeled data of downstream tasks' domains (GURURANGAN *et al.*, 2020).

## 3.3 Representations and language models for Portuguese

Several static word embeddings for Portuguese have been trained and evaluated over the past years (SANTOS; ZADROZNY, 2014; HARTMANN *et al.*, 2017). Recent works also explored and compared contextual embedding techniques. ELMo and Flair Embeddings trained on a large Portuguese corpora achieve good results on the named entity recognition task (CASTRO *et al.*, 2019b; CASTRO *et al.*, 2019a; SANTOS *et al.*, 2019a; SANTOS *et al.*, 2019b). A comparison of ELMo and multilingual BERT using a contextual embeddings setup shows superior performance of Portuguese ELMo on semantic textual similarity task when no fine-tuning is used (RODRIGUES *et al.*, 2020c). Portuguese language models for fine-tuning purposes, which is the topic of this work, is still an area not much explored. Concurrent to this work, T5 models for Portuguese, PTT5, were trained and evaluated on semantic textual similarity and recognizing textual entailment tasks (CARMO *et al.*, 2020).

# 4 BERTimbau: pretraining and evaluation

In this chapter, we describe the procedures to pretrain and evaluate BERT models for Brazilian Portuguese.

## 4.1 Pretraining

Our approach closely replicates BERT's architecture and pretraining procedures with few changes. In this section, we describe the procedures of vocabulary generation and adaptation, and gathering and preprocessing of unlabeled data, which are the steps required to pretrain the models.

We train BERTimbau models on two sizes: Base (12 layers, 768 hidden dimension, 12 attention heads, and 110M parameters) and Large (24 layers, 1024 hidden dimension, 16 attention heads and 330M parameters). The maximum sentence length is set to $S = 512$ tokens. We train cased models only since we focus on general purpose models and capitalization is relevant for tasks like named entity recognition (CASTRO *et al.*, 2018; DEVLIN *et al.*, 2018).

### 4.1.1 Vocabulary generation

We generate a cased Portuguese vocabulary of 30,000 subword units using the SentencePiece library (KUDO; RICHARDSON, 2018) with the BPE algorithm (SENNRICH *et al.*, 2016b) and 2,000,000 random sentences from Portuguese Wikipedia articles. The resulting vocabulary is then converted to WordPiece (SCHUSTER; NAKAJIMA, 2012a) format for compatibility with original BERT code.

#### 4.1.1.1 SentencePiece to WordPiece conversion

To convert the generated SentencePiece vocabulary to WordPiece format, we follow BERT's tokenization rules. Firstly, all BERT special tokens are inserted ([CLS], [MASK], [SEP], and [UNK]) and all punctuation characters of mBERT's vocabulary are added to the Portuguese vocabulary. Then, since BERT splits the text at whitespace and punctuation prior to applying WordPiece tokenization in the resulting chunks, each SentencePiece token that contains punctuation characters is split at these characters, the punctuations are removed and the resulting subword units are added to the vocabulary.[1]

[1] Splitting at punctuation implies no subword token can contain both punctuation and non-punctuation characters.

Finally, subword units that do not start with "▁" are prefixed with "##" and the "▁" symbol is removed from the remaining tokens.

### 4.1.2 Pretraining data

For pretraining data, we use the brWaC (FILHO *et al.*, 2018) corpus (Brazilian Web as Corpus), a crawl of Brazilian webpages which contains 2.68 billion tokens from 3.53 million documents and is the largest open Portuguese corpus to date. On top of its size, brWaC is composed of whole documents and its methodology ensures high domain diversity and content quality, which are desirable features for BERT pretraining.

We use only the document body (ignoring the titles) and we apply a single post-processing step on the data to remove *mojibakes*[2] and remnant HTML tags using the `ftfy` library (SPEER, 2019). The final processed corpus has 17.5GB of raw text. We split the corpus into chunks of 50MB and generate pretraining examples independently for each file as described in 2.4.2, with a duplication factor of 10. That is, we run example generation 10 times for each 50MB file, producing distinct sentence pairs for NSP task and token masks for MLM task. For maximum sequence length of 128 tokens, a total of $4.29 \times 10^8$ examples are generated, and, for maximum length 512, a total of $1.58 \times 10^8$ examples.

## 4.2 Evaluation

Once pretrained, we evaluate our models on 3 downstream NLP tasks: Sentence Textual Similarity (STS), Recognizing Textual Entailment (RTE), and Named Entity Recognition (NER). To evaluate BERTimbau on downstream tasks, we remove the MLM and NSP classification heads used during pretraining stage and attach a relevant head required for each task. We then fine-tune our models on each task or pair of tasks. Similar to pretraining, sentence-level tasks are performed on the encoded representation of the `[CLS]` special token, $\mathbf{c}$, and token-level tasks use the encoded representation of each relevant token, $\mathbf{T}_i$.

In the following sections we briefly define each evaluation task, the datasets, the architecture modifications and the training and evaluation procedures.

### 4.2.1 Sentence Textual Similarity and Recognizing Textual Entailment

Sentence Textual Similarity is a regression task that measures the degree of semantic equivalence between two sentences in a numeric scale. Recognizing Textual En-

[2] Mojibake is a kind of text corruption that occurs when strings are decoded using the incorrect character encoding. For example, the word "codificação" becomes "codificaÃ§Ã£o" when encoded in UTF-8 and decoded using ISO-8859-1.

tailment, also known as Natural Language Inference (NLI), is a classification task of predicting if a given premise sentence entails a hypothesis sentence.

#### 4.2.1.1 Dataset and metrics

We use the dataset of the ASSIN2 shared task (REAL *et al.*, 2020), which contains 10,000 sentence pairs with STS and RTE annotations. The dataset is composed of 6500 train, 500 validation and 3000 test examples.

STS scores are continuous values in a scale of 1 to 5, where a pair of sentences with completely different meanings have a score of 1 and virtually equivalent sentences have score of 5. STS performance is evaluated using Pearson's Correlation as primary metric and Mean Squared Error (MSE) as secondary metric.

RTE labels are simply `entailment` and `non-entailment`. RTE performance is evaluated using macro F1-score as primary metric and accuracy as secondary metric. Examples from ASSIN2 dataset can be seen in Table 2, which contains sentence pairs and their corresponding gold labels for both tasks.

#### 4.2.1.2 Tasks' heads and loss functions

Given an example with a premise sentence and a hypothesis sentence, we concatenate the two sentences as of 2.4.1.1 and feed the sequence into BERTimbau. We attach two independent linear layers on top of BERTimbau in a multitask scheme, both receiving the vector $\mathbf{c}$ as input:

$$\hat{y}_{STS} = \mathbf{c}\mathbf{W}_S + \mathbf{b}_S \tag{4.1}$$

$$\hat{p}_{RTE} = \text{Softmax}(\mathbf{c}\mathbf{W}_R + \mathbf{b}_R)\,, \tag{4.2}$$

where $\hat{y}_{STS}$ is the model's prediction for STS relatedness score, $\hat{p}_{RTE}$ is the model's predicted probabilities for the two RTE classes, $\mathbf{W}_S \in \mathbb{R}^{H\times 1}$, $\mathbf{b}_S \in \mathbb{R}$, $\mathbf{W}_R \in \mathbb{R}^{H\times 2}$ and $\mathbf{b}_R \in \mathbb{R}^2$.

We train using MSE loss for STS and cross-entropy loss for RTE. The final loss is the sum of both losses with equal weight.

### 4.2.2 Named Entity Recognition

The task of named entity recognition consists of identifying text spans that mentions named entities (NEs) and classifying them into predefined categories. There are a variety of definitions for the "named entity" expression, such as proper names and *rigid designators* (NADEAU; SEKINE, 2007). These definitions, however, are often loosened for practical reasons. Common general NE categories are person, organization, location, temporal expressions and numerical expressions, such as money, quantities of other units

Table 2 – Five samples of ASSIN2 dataset. Each sample is composed of a sentence pair and its gold STS relatedness score (a contiguous value from 1 to 5) and RTE label (Entailment or None).

| **Gold STS/RTE** | **Sentence pair** |
|---|---|
| 5.0 / Entailment | A: *Os meninos estão de pé na frente do carro, que está queimando.*<br>B: *Os meninos estão de pé na frente do carro em chamas.* |
| | A: The boys are standing in front of the car, which is burning.<br>B: The boys are standing in front of the burning car. |
| 4.0 / Entailment | A: *O campo verde para corrida de cavalos está completamente cheio de jóqueis.*<br>B: *Os jóqueis estão correndo a cavalos no campo, que é completamente verde.* |
| | A: The green field for horse races is completely full of Jockeys.<br>B: The Jockeys are racing horses on the field, which is completely green. |
| 3.0 / Entailment | A: *A gruta com interior rosa está sendo escalada por quatro crianças do Oriente Médio, três meninas e um menino.*<br>B: *Um grupo de crianças está brincando em uma estrutura colorida.* |
| | A: Four middle eastern children, three girls and one boy, are climbing on the grotto with a pink interior.<br>B: A group of kids is playing in a colorful structure. |
| 2.0 / None | A: *Não tem nenhuma pessoa descascando uma batata.*<br>B: *Uma pessoa está fritando alguma comida.* |
| | A: There is no one peeling a potato.<br>B: A person is frying some food. |
| 1.0 / None | A: *Um cachorro está correndo no chão.*<br>B: *A menina está batucando suas unhas.* |
| | A: A dog is running on the ground.<br>B: The girl is tapping her fingernails. |

and percentages. Domain-specific entity categories can also be defined, such as "protein", "chemical" and "cell type" that are found in works in the biomedical field (NADEAU; SEKINE, 2007).

Formally, a NER system has to perform the following task: given a tokenized text composed of a sequence of $n$ tokens $(x_1, \ldots, x_n)$, the system has to output triples $(t_s, t_e, k)$ where $t_s, t_e \in \{1, \ldots, n\}$ are the start and end token indices of an entity, respectively, and $k$ is a named entity class. For instance:

Figure 3 – Examples of named entity recognition tag sequences for the sentence "James L. was born in Washington, DC" using the IOB2 and BILOU schemes with person (PER) and location (LOC) classes.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Sentence** | | | | | | | | | |
| James | L | . | was | born | in | Washington | , | DC | . |
| **Tagging in IOB2 scheme** | | | | | | | | | |
| B-PER | I-PER | I-PER | O | O | O | B-LOC | O | B-LOC | O |
| **Tagging in BILOU scheme** | | | | | | | | | |
| B-PER | I-PER | L-PER | O | O | O | U-LOC | O | U-LOC | O |

| James | L | . | was | born | in | Washington | , | DC | . |
|---|---|---|---|---|---|---|---|---|---|
| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ | $x_{10}$ |

$$\downarrow$$

(1, 3, PERSON) $\longrightarrow$ "James L."
(7, 7, LOCATION) $\longrightarrow$ "Washington"
(9, 9, LOCATION) $\longrightarrow$ "DC"

This example emphasizes that NER outputs should locate each named entity in the input text, and not simply output categorized substrings, such as a list of person names and locations.

NER is commonly modeled as a sequence tagging task that performs unified entity identification and classification. Given a sequence of tokens $(x_1, \ldots, x_n)$, the model has to output a sequence of tags $(y_1, \ldots, y_n)$, where each token is assigned a tag of a predefined tag vocabulary according to a tagging scheme and the entity classes.

Common tagging schemes are IOB2 (TJONG *et al.*, 1999) and IOBES/BILOU. Each scheme defines a tag vocabulary and tag transition constraints. The IOB2 tagging scheme defines the tags `{B-x, I-x, O}` . The `B-x` tag indicates the beggining of an entity of class "x". `I-x` marks succeeding tokens inside the same entity of class "x" and must follow a `B-x` tag. The `O` tag is used for outside tokens that do not belong to entities. The final tag vocabulary is composed by the the `O` tag and `{B-, I-}` tags for each named entity class, allowing entity identification and classification to be performed jointly.

The IOBES/BILOU scheme extends IOB2 by including Ending/Last and Single/Unit tags. The `E-/L-` tag is used to mark an entity's final token and `S-/U-` marks single-token entities. Tagging output examples for IOB2 and BILOU schemes are shown in Figure 3.

To perform NER using BERT, we follow the original work and cast it as a sequence tagging task using the IOB2 tagging scheme. Below we describe the datasets.

Table 3 – Dataset statistics for the HAREM I corpora. The Tokens column refers to whitespace and punctuation tokenization.

| **Dataset** | **Documents** | **Tokens** | **Entities in scenario** | |
|---|---|---|---|---|
| | | | **Selective** | **Total** |
| First HAREM | 129 | 95585 | 4151 | 5017 |
| MiniHAREM | 128 | 64853 | 3018 | 3642 |

#### 4.2.2.1 NER datasets

We use the Golden Collections of the First HAREM (SANTOS *et al.*, 2006) and Mini HAREM evaluation contests, which we refer hereafter as First HAREM and Mini-HAREM. Both datasets contain multidomain documents annotated with 10 NE classes: Person, Organization, Location, Value, Date, Title, Thing, Event, Abstraction, and Other. Examples from First HAREM dataset are shown in Table 4.

We use First HAREM as train set and Mini HAREM test set. We employ the datasets on two distinct scenarios: a Total scenario that considers all 10 classes, and a Selective scenario that includes only 5 classes (Person, Organization, Location, Value, and Date). This setup of train/test sets and distinct scenarios follows previous works (SANTOS *et al.*, 2019a; CASTRO *et al.*, 2018; SANTOS; GUIMARAES, 2015) and aims to facilitate the comparison of results. Table 3 presents some dataset statistics. We set aside 7% of First HAREM documents as a holdout validation set.

**Preprocessing** — The HAREM datasets are annotated taking into consideration vagueness and indeterminacy in text, such as ambiguity in sentences. This way, some text segments contain <ALT> tags that enclose multiple alternative named entity identification solutions. Additionally, multiple categories may be assigned to a single named entity.

To model NER as a sequence tagging problem, we must select a single truth for each undetermined segment and/or entity. To resolve each <ALT> tag in the datasets, our approach is to select the alternative that contains the highest number of named entities. In case of ties, the first one is selected. To resolve each named entity that is assigned multiple classes, we simply select the first valid class for the scenario.

An example annotation of HAREM that contains multiple solutions, in XML format, is:

```
<ALT>
<EM CATEG="PER|ORG">Governo de Cavaco Silva</EM>|
<EM CATEG="ORG">Governo</EM> de <EM CATEG="PER">Cavaco Silva</EM>
</ALT>
```

where <EM> is a tag for named entity and "`|`" identifies alternative solutions. This annotation can be equally interpreted as containing the following NEs:

1. 1 NE: Person "Governo de Cavaco Silva"
2. 1 NE: Organization "Governo de Cavaco Silva"
3. 2 NEs: Organization "Governo" and Person "Cavaco Silva"

The defined heuristics select the third solution in the example above as the ground-truth.

#### 4.2.2.2 NER evaluation metrics

NER performance is evaluated using CoNLL 2003 (SANG; MEULDER, 2003) evaluation script,[3] that computes entity-level precision, recall, and micro F1-score on exact matches. In other words, precision is the percentage of named entities predicted by the model that are correct, recall is the percentage of corpus entities that were correctly predicted and F1-score is the harmonic mean of precision and recall.

Formally, considering a set of ground-truth named entity triples $\mathcal{T}$ and a set of predicted triples $\hat{\mathcal{T}}$, the metrics are computed as:

$$TP = |\mathcal{T} \bigcap \hat{\mathcal{T}}| \tag{4.3}$$

$$FP = |\hat{\mathcal{T}} - \mathcal{T}| \tag{4.4}$$

$$FN = |\mathcal{T} - \hat{\mathcal{T}}| \tag{4.5}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.6}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.7}$$

$$\text{F1-score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}\,, \tag{4.8}$$

where TP, FP and FN stand for True Positives, False Positives and False Negatives, respectively.

#### 4.2.2.3 NER architectures and loss functions

Given a sequence of tokens $\mathbf{x} = (x_1, \ldots, x_n)$ and a corresponding sequence of ground-truth tags $\mathbf{y} = (y_1, \ldots, y_n)$, the direct approach to perform NER using BERT is to feed the BERT-encoded token representations $\mathbf{T} = (\mathbf{T}_1, \ldots, \mathbf{T}_n)$ to a classification model that projects each token's encoded representation to the tag space, i.e. $\mathbb{R}^H \mapsto \mathbb{R}^K$, where $K$ is the number of tags and depends on the the number of classes and on the tagging scheme. In the simplest architecture, the classification model is a single linear

[3] <https://www.clips.uantwerpen.be/conll2002/ner/bin/conlleval.txt>

layer and tag predictions are made independently for each position,

$$\mathbf{P} = \mathbf{T}\mathbf{W}_{cls} \tag{4.9}$$

$$\hat{\mathbf{y}} = \underset{j}{\operatorname{argmax}}\, \mathbf{P} \tag{4.10}$$

where $\mathbf{W}_{cls} \in \mathbb{R}^{H \times K}$, $\mathbf{P} \in \mathbb{R}^{n \times K}$ is a matrix of tag scores for each token, argmax is applied on the tags dimension and $\hat{\mathbf{y}} \in \{1, \ldots, K\}^n$ is the sequence of predicted tags. In this setup, the model is trained by minimizing the cross-entropy loss.

Since Linear-Chain Conditional Random Fields (CRF) (LAFFERTY *et al.*, 2001) is widely adopted to enforce sequential classification in sequence labeling tasks (SANTOS; GUIMARAES, 2015; LAMPLE *et al.*, 2016; AKBIK *et al.*, 2018), we also experiment with employing a CRF layer. In this setup, the output scores of the classification model, $\mathbf{P}$, are fed to a CRF layer, whose parameters are a matrix of tag transitions $\mathbf{A} \in \mathbb{R}^{K+2,K+2}$. The matrix $\mathbf{A}$ is such that $A_{i,j}$ represents the score of transitioning from tag $i$ to tag $j$. $\mathbf{A}$ includes 2 additional states: start and end of sequence.

For an input sequence $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ with a corresponding matrix of tag scores $\mathbf{P}$ and a sequence of tag predictions $\mathbf{y} = (y_1, \ldots, y_n)$, the score of the sequence is defined as

$$s(\mathbf{X}, \mathbf{y}) = \sum_{i=0}^{n} A_{y_i, y_{i+1}} + \sum_{i=1}^{n} P_{i, y_i}\,, \tag{4.11}$$

where $y_0$ and $y_{n+1}$ are start and end tags, respectively, and $P_{i,y_i}$ is the score of tag $y_i$ for the *i-th* token. During training, the model is optimized by maximizing the log-probability of the correct tag sequence, which follows from applying softmax over all possible tag sequences' scores:

$$\log(p(\mathbf{y} \mid \mathbf{X})) = s(\mathbf{X}, \mathbf{y}) - \log \left( \sum_{\tilde{\mathbf{y}} \in \mathbf{Y}_{\mathbf{X}}} e^{s(\mathbf{X}, \tilde{\mathbf{y}})} \right) \tag{4.12}$$

where $\mathbf{Y}_{\mathbf{X}}$ are all possible tag sequences. The summation in Eq. (4.12) is computed using dynamic programming. During evaluation, the most likely sequence $\hat{\mathbf{y}}$ is obtained by Viterbi decoding. We refer readers to (LAMPLE *et al.*, 2016) for further explanation of CRF.

It is important to note that subword tokenization requires tag predictions and losses to be computed only for the first subtoken of each word, ignoring word continuation tokens. This applies to both architectures described in this section, especially for Eqs. (4.10) to (4.12).

### 4.2.3 Document context and max context evaluation for token-level tasks

In token-level tasks such as NER, we use document context for input examples instead of sentence context to take advantage of longer contexts when encoding token

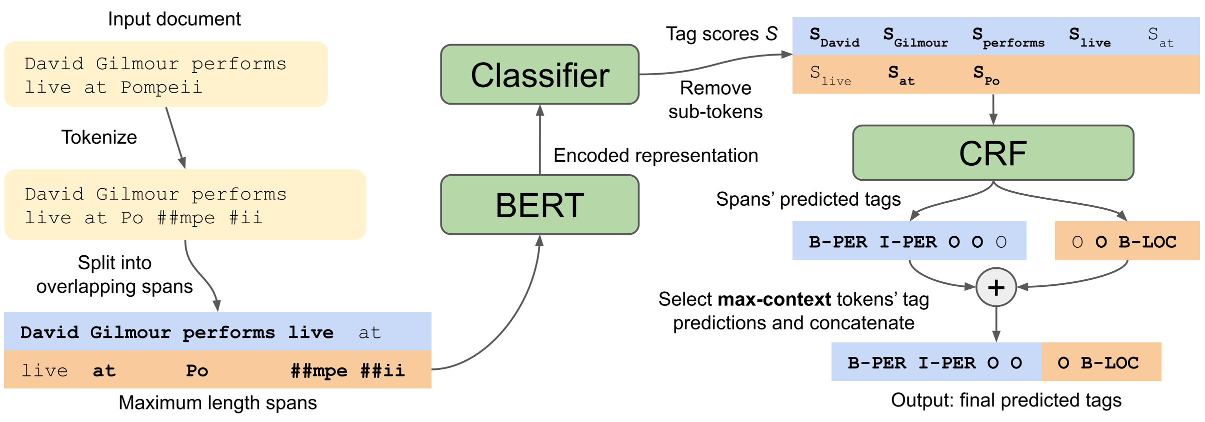
Input document
David Gilmour performs live at Pompeii
Tokenize
David Gilmour performs live at Po ##mpe #ii
Split into overlapping spans
David Gilmour performs live at
live at Po ##mpe ##ii
Maximum length spans
BERT
Encoded representation
Classifier
Tag scores S
Remove sub-tokens
S David S Gilmour S performs S live S at
S live S at S Po
CRF
Spans' predicted tags
B-PER I-PER O O O
O O B-LOC
Select max-context tokens' tag predictions and concatenate
+
B-PER I-PER O O
O B-LOC
Output: final predicted tags

Figure 4 – Illustration of the proposed method for the NER task described in 4.2.3. Given an input document, the text is tokenized using WordPiece (SCHUSTER; NAKAJIMA, 2012a) and the tokenized document is split into overlapping spans of the maximum length using a fixed stride (= 3, in the example). Maximum context tokens of each span are marked in bold. The spans are fed into BERT and then into the classification model, producing a sequence of tag scores for each span. The scores of subtoken entries (starting with ##) are removed from the spans and the remaining tags scores are passed to the CRF layer — if it is employed, otherwise the highest tag scores are used independently. The maximum context tokens are selected and concatenated to form the final predicted tags.

representations from BERT. Following the approach of original BERT work (DEVLIN *et al.*, 2018) on the SQuAD dataset, examples longer than $S$ tokens are broken into spans of length up to $S$ using a stride of $D$ tokens. Each span is used as a separate example during training. During evaluation, however, a single token $x_i$ can be present in $N = \frac{S}{D}$ multiple spans $s_j$, and so may have up to $N$ distinct predictions $y_{i,j}$. Each token's final prediction is taken from the span where the token is closer to the central position, that is, the span where it has the most contextual information. Figure 4 illustrates this procedure.

Table 4 – FirstHAREM dataset samples. Gold named entities are enclosed by brackets with subscripted labels.

| |
|---|
| A onça, ou jaguar, é um mamífero ([Panthera]$_{\text{THING}}$ onca), da ordem dos carnívoros, família dos felídeos, encontrado em todo o continente americano, dos [EUA]$_{\text{LOC}}$ à [Argentina]$_{\text{LOC}}$ e em todo o [Brasil]$_{\text{LOC}}$. English translation: The jaguar is a mammal ([Panthera]$_{\text{THING}}$ onca), of the order of carnivores, family of felids, found throughout the American continent, from the [USA]$_{\text{LOC}}$ to [Argentina]$_{\text{LOC}}$ and throughout [Brazil]$_{\text{LOC}}$. |
| [Almeida Henriques]$_{\text{PER}}$ ([A.H.]$_{\text{PER}}$): O [CEC]$_{\text{ORG}}$ foi criado numa lógica de unir as associações da [Região Centro]$_{\text{LOC}}$, quer sejam industriais, quer sejam comerciais, quer sejam agrícolas. English translation: [Almeida Henriques]$_{\text{PER}}$ ([A.H.]$_{\text{PER}}$): The [CEC]$_{\text{ORG}}$ was created in a logic of uniting the associations of the Center Region, whether industrial, commercial or agricultural. |
| Entre os mais importantes destacam-se o de [Shanta Durga]$_{\text{TITLE}}$ e o de [Shri Munguesh]$_{\text{TITLE}}$, construidos há [400 anos]$_{\text{VALUE}}$. English translation: Among the most important are [Shanta Durga]$_{\text{TITLE}}$ and [Shri Munguesh]$_{\text{TITLE}}$, built [400 years]$_{\text{VALUE}}$ ago. |
| Para aqueles que vão participar do processo seletivo, o professor de [Direito Previdenciário]$_{\text{ABS}}$ [Fábio Zambite]$_{\text{PER}}$ dá uma dica importante: os candidatos devem estudar com bastante atenção o [Decreto 3.048/99]$_{\text{TITLE}}$, que aprova o [Regulamento da Previdência Social]$_{\text{TITLE}}$. English translation: For those who are going to participate in the selection process, Professor of [Social Security Law]$_{\text{ABS}}$ [Fábio Zambite]$_{\text{PER}}$ gives an important tip: candidates must carefully study [Decree 3.048/99]$_{\text{TITLE}}$, which approves the [Social Security Regulation]$_{\text{TITLE}}$. |
| [A Mulher no Inicio do Novo Século]$_{\text{EVENT}}$ Dia [15 de Maio]$_{\text{TIME}}$, pelas [9.30H]$_{\text{TIME}}$, no [Cine-Teatro Caridade]$_{\text{LOC}}$, em [Moura]$_{\text{LOC}}$ irá realizar-se um Fórum intitulado [A Mulher no Inicio do Novo Século]$_{\text{ABS}}$, tendo como organização a [Câmara Municipal de Moura]$_{\text{ORG}}$ e a colaboração da [Associação de Mulheres do Concelho de Moura]$_{\text{ORG}}$. English translation: [Women at the Beginning of the New Century]$_{\text{EVENT}}$ On the [15th of May]$_{\text{TIME}}$, at [9.30 am]$_{\text{TIME}}$, at the [Cine-Teatro Caridade]$_{\text{LOC}}$, in [Moura]$_{\text{LOC}}$, a Forum entitled [Women at the Beginning of the New Century]$_{\text{ABS}}$ will take place, organized by the [Moura City Council]$_{\text{ORG}}$ with the collaboration of the [Moura's Women Association Board]$_{\text{ORG}}$. |
| [Touro]$_{\text{OTHER}}$ é o signo seguinte. O sol o visita entre [21 de abril]$_{\text{TIME}}$ e [21 de maio]$_{\text{TIME}}$, domicílio de [Vênus]$_{\text{THING}}$. English translation: [Taurus]$_{\text{OTHER}}$ is the next sign. The sun visits him between [April 21]$_{\text{TIME}}$ and [May 21]$_{\text{TIME}}$, home of [Venus]$_{\text{THING}}$. |

# 5 Experiments

In this section, we present the experimental setup and results for BERT pretrainings and evaluation tasks. We conduct additional experiments to explore the usage of BERTimbau as a fixed extractor of contextual embeddings, the impact of the long pretraining stage and the impacts of the vocabulary and tokenization on the evaluation tasks' metrics.

## 5.1 Pretrainings

Following Devlin et al. (DEVLIN *et al.*, 2018), models are pretrained for 1,000,000 steps. We use a peak learning rate of 1e-4, with learning rate warmup over the first 10,000 steps followed by a linear decay of the learning rate over the remaining steps.

For BERTimbau Base models, the weights are initialized with the checkpoint of Multilingual BERT Base, with the exception of the word embeddings and MLM head bias, $\mathbf{E}_V$ and $\mathbf{b}_m$, that are of a different vocabulary and are randomly initialized. We use a batch size of 128 and sequences of 512 tokens the entire training. This training takes 4 days on a TPU v3-8 instance and performs about 8 epochs over the pretraining data.

For BERTimbau Large, the weights are initialized with the checkpoint of English BERT Large, again discarding $\mathbf{E}_V$ and $\mathbf{b}_m$. Since it is a bigger model with longer training time, we follow the instructions of the original work and use sequences of 128 tokens in batches of size 256 for the first 900,000 steps and then sequences of 512 tokens and batch size 128 for the last 100,000 steps. This training takes 7 days on a TPU v3-8 instance and performs about 6 epochs over the training data.

Training loss curves for both pretrainings are shown in Figure 5. It can be seen that there is a sharp decrease in loss over the initial 100k steps, which can be interpreted as the models learning the word embeddings that are initialized randomly. The losses slowly decrease afterwards until the end of the training. A steep decrease in BERTimbau Large loss can be noticed in step 900,000, which marks the beginning of the pretraining using sequences of 512 tokens. It is worth noting that while the smoothed curve appears to have room for further training, this is actually an effect of the large smoothing factor — the non-smoothed curve shows the loss rapidly decreases and varies around a new plateau. BERTimbau Large reaches a MLM accuracy of 70% and NSP accuracy of 98.5%, while BERTimbau Base reaches 66.8% and 98.2%, respectively.

Note that in the calculation of the number of epochs, we are taking into consideration a duplication factor of 10 when generating the input examples. This means

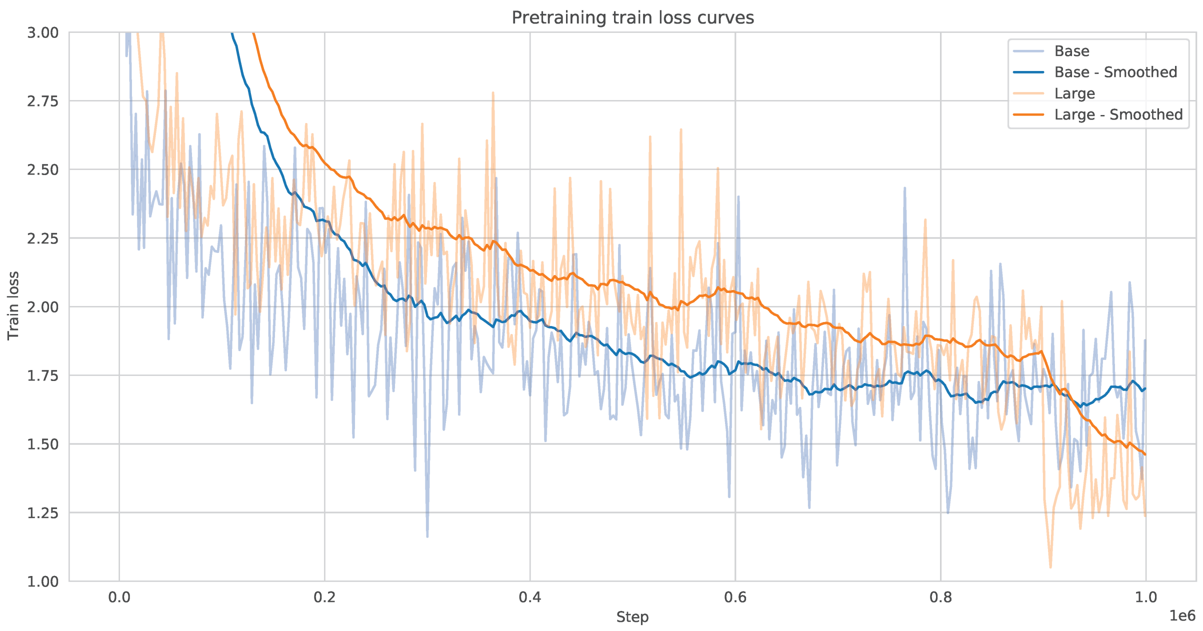

Figure 5 – Training loss curves for BERTimbau Base and BERTimbau Large pretrainings. Smoothed curves are exponential moving averages with smoothing factor $\alpha = 0.95$. A sharp decrease in BERTimbau Large training loss can be noticed after step 900,000, when training begins using sequences of 512 tokens.

that under 10 epochs, the same sentence is seen with different masking and sentence pair in each epoch, which is effectively equal to dynamic example generation proposed by RoBERTa (LIU *et al.*, 2019).

## 5.2 Fine-tunings on evaluation tasks

For all evaluation experiments, we use a learning rate schedule of warmup over the first 10% steps followed by linear decay of the learning rate over the remaining steps. Similar to pretraining, we use BERT's AdamW optimizer implementation with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and L2 weight decay of 0.01. We perform early stopping and select the best model on the validation set of each task. While the pretraining experiments require TPU devices or dozens of GPUs in parallel to meet the memory requirements and a reasonable training time, the following fine-tuning experiments can be run on consumer-grade GPUs with 8GB memory taking advantage of gradient accumulation. The only exceptions are the NER experiments using BERTimbau-Large, which require at least 12GB.

## 5.3 STS and RTE tasks

For this experiment, we train BERTimbau Base with learning rate of 4e-5 and batch size 32 for 10 epochs, and BERTimbau Large with learning rate of 1e-5, batch size

8 for 5 epochs. We also train mBERT to compare it to BERTimbau models. mBERT is trained using learning rate of 1e-5 and batch size 8 for 10 epochs.

### 5.3.1 Results

Table 5 – Test scores for STS and RTE tasks on ASSIN2 dataset. We compare our models to the best published results. Best scores in bold. Reported values are the average of multiple runs with different random seeds. Star (⋆) denotes primary metrics. †: ensemble technique. ‡: extra training data.

| Row | Model | STS | | RTE | |
|---|---|---|---|---|---|
| | | Pearson (⋆) | MSE | F1 (⋆) | Accuracy |
| 1 | mBERT + RoBERTa-Large-en (Averaging) (RODRIGUES *et al.*, 2020b) † | 0.83 | 0.91 | 84 | 84.8 |
| 2 | mBERT + RoBERTa-Large-en (Stacking) (RODRIGUES *et al.*, 2020b) † | 0.785 | 0.59 | 88.3 | 88.3 |
| 3 | mBERT (STS) and mBERT-PT (RTE) (RODRIGUES *et al.*, 2020a) ‡ | 0.826 | 0.52 | 87.6 | 87.6 |
| 4 | USE+Features (STS) and mBERT+Features (RTE) (FONSECA; ALVARENGA, 2020) | 0.800 | **0.39** | 86.6 | 86.6 |
| 5 | mBERT+Features (FONSECA; ALVARENGA, 2020) | 0.817 | 0.47 | 86.6 | 86.6 |
| 6 | mBERT (ours) | 0.809 | 0.58 | 86.8 | 86.8 |
| 7 | BERTimbau Base | 0.836 | 0.58 | 89.2 | 89.2 |
| 8 | BERTtimbau Large | **0.852** | 0.50 | **90.0** | **90.0** |

Our results for both tasks are shown in Table 5. We compare our results to the best-performing submissions to official ASSIN2 competition. All compared works employ either mBERT or a Transformer-based architecture in their approaches. In the following paragraphs, we refer to each work using their corresponding row numbers in Table 5.

BERTimbau models achieve the best results on the primary metrics of both STS and RTE tasks, with the large model performing significantly better than the base variant. The previous highest scores (rows 1 and 2) for both STS Pearson's correlation and RTE F1 score are from ensemble techniques that combine mBERT fine-tuned on original ASSIN2 data and an English RoBERTa-Large fine-tuned on ASSIN2 data automatically translated to English. The averaging ensemble uses 2 models and the stacking ensemble uses 10 distinct fine-tuned models — 5-fold stacking which results in 5 mBERT and 5 RoBERTa trained models. While this approach shows an interesting application of English models to Portuguese tasks, our BERTimbau models achieve higher performance using a single model and, hence, demand lower compute resources in both fine-tuning and inference stages.

Regarding our implementation using mBERT (row 6), it presents a lower performance compared to BERTimbau models, which highlights the benefits of Portuguese pretraining of BERTimbau. For STS task, we note that mBERT achieves the same MSE as BERTimbau Base, even though Pearson correlation is lower. Comparing it to other works' approaches, better performances are achieved using extra supervised training data and further pretraining of mBERT on Portuguese data (row 3), and also by combining it with hand-designed features (rows 4 and 5).

## 5.4 NER

In this section, we refer to the 2 architectures defined in 4.2.2.3 as BERT and BERT-CRF. Long examples are broken into spans using a stride of $D = 128$ as explained in Section 4.2.3.

The model parameters are divided in two groups with different learning rates: 5e-5 for BERT model and 1e-3 for the classifier. We train BERT models for up to 50 epochs using a batch size of 16. BERT-CRF models are trained for up to 15 epochs.

In addition to BERTimbau Base and Large, we also train mBERT to compare monolingual versus multilingual model performances. mBERT is fine-tuned with the same hyperparameters.

It is common in NER for the vast majority of tokens not to belong to named entities (and have tag label "O"). To deal with this class imbalance, we initialize the classifier's bias term of the "O" tag with a value of 6 in order to promote a better stability in early training (LIN *et al.*, 2017). We also use a weight of 0.01 for "O" tag losses.

When evaluating, we produce valid predictions by removing all invalid tag transitions for the IOB2 scheme, such as "I-" tags coming directly after "O" tags or after an "I-" tag of a different class. This post-processing step trades off recall for a possibly higher precision.

Table 6 – Results of NER task (Precision, Recall and micro F1-score) on the test set (MiniHAREM). Best results in bold. Reported values are the average of multiple runs with different random seeds. Star (⋆) denotes primary metrics.

| **Row** | **Architecture** | **Total scenario** | | | **Selective scenario** | | |
|---|---|---|---|---|---|---|---|
| | | **Prec.** | **Rec.** | **F1 (⋆)** | **Prec.** | **Rec.** | **F1 (⋆)** |
| 1 | CharWNN (SANTOS; GUIMARAES, 2015) | 67.2 | 63.7 | 65.4 | 74.0 | 68.7 | 71.2 |
| 2 | LSTM-CRF (CASTRO *et al.*, 2018) | 72.8 | 68.0 | 70.3 | 78.3 | 74.4 | 76.3 |
| 3 | BiLSTM-CRF+FlairBBP (SANTOS *et al.*, 2019a) | 74.9 | 74.4 | 74.6 | 83.4 | 81.2 | 82.3 |
| 4 | mBERT | 71.6 | 72.7 | 72.2 | 77.0 | 78.8 | 77.9 |
| 5 | mBERT + CRF | 74.1 | 72.2 | 73.1 | 80.1 | 78.3 | 79.2 |
| 6 | BERTimbau Base | 76.8 | 77.1 | 77.2 | 81.9 | **82.7** | 82.2 |
| 7 | BERTimbau Base + CRF | 78.5 | 76.8 | 77.6 | 84.6 | 81.6 | 83.1 |
| 8 | BERTimbau Large | 77.9 | **78.0** | 77.9 | 81.3 | 82.2 | 81.7 |
| 9 | BERTimbau Large + CRF | **79.6** | 77.4 | **78.5** | **84.9** | 82.5 | **83.7** |

### 5.4.1 Results

The main results of our NER experiments are presented in Table 6. We compare the performances of our models on the two scenarios (total and selective) defined in Section 4.2.2.1 to results of previous works. The models of rows 1 to 3 show the progress of neural network approaches for this dataset over the recent years. The previous best result (row 3),

Table 7 – NER performances (Precision, Recall and F1-score) on the test set (MiniHAREM) using BERTimbau as contextual embeddings in a feature-based approach. Star (⋆) denotes primary metrics.

| **Architecture** | **Total scenario** | | | **Selective scenario** | | |
|---|---|---|---|---|---|---|
| | **Prec.** | **Rec.** | **F1 (⋆)** | **Prec.** | **Rec.** | **F1 ⋆)** |
| mBERT + BiLSTM-CRF | 74.7 | 69.7 | 72.1 | 80.6 | 75.0 | 77.7 |
| BERTimbau Base + BiLSTM-CRF | 78.3 | 73.2 | 75.6 | 84.5 | 78.7 | 81.6 |
| BERTimbau Large + BiLSTM-CRF | 77.4 | 72.4 | 74.8 | 83.0 | 77.8 | 80.3 |

achieved by BiLSTM-CRF+FlairBBP model, uses Portuguese Flair Embeddings, which are contextual embeddings extracted from character-level language models (AKBIK *et al.*, 2018).

Our best model, BERTimbau Large + CRF (row 9), outperforms the best published results improving the F1-score by 3.9 points on the total scenario and by 1.4 point on the selective scenario. Interestingly, Flair embeddings outperform BERT models on English NER (AKBIK *et al.*, 2018; DEVLIN *et al.*, 2018).

There is a large performance gap between BERTimbau and mBERT, which reinforces the advantages of monolingual models pretrained on multidomain data compared to mBERT, that is trained only on Wikipedia articles. This result is on par with other monolingual BERT works.

The CRF layer consistently brings performance improvements in F1 in all settings. However, F1 increases are pushed by a large boost in precision that is often associated with lower recall. It is worth noting that, without CRF, BERTimbau Large shows a close but inferior performance to the Base variant on the selective scenario. This result suggests that a more controlled fine-tuning scheme might be required in some cases, such as partial layer unfreezing or discriminative fine-tuning (PETERS *et al.*, 2019) — usage of lower learning rates for lower layers —, given that it is a higher capacity model trained on few data.

## 5.5 BERTimbau as contextual embeddings

In this experiment, we evaluate BERTimbau as a fixed extractor of contextual embeddings that we use as input features to train a downstream model on the NER task. This setup can be interesting in lower resource scenarios in which several tasks are to be performed on the same input text: the extraction of contextual embeddings —which is the most expensive stage, —can be computed once and then shared across several smaller task-specific models.

In this feature-based approach, we train a BiLSTM-CRF model with 1 layer and

100 hidden units followed by a linear classifier layer for up to 50 epochs. Instead of using only the hidden representation of BERT's last encoder layer, we sum the last 4 layers, as proposed by the original work (DEVLIN *et al.*, 2018):

$$\mathbf{T}_i^{fixed} = \mathbf{Z}_i^{(L)} + \mathbf{Z}_i^{(L-1)} + \mathbf{Z}_i^{(L-2)} + \mathbf{Z}_i^{(L-3)}\,, \tag{5.1}$$

where $\mathbf{Z}_i^{(j)}$ is the output of the j-th Encoder layer at the i-th position. The resulting architecture resembles the BiLSTM-CRF model (LAMPLE *et al.*, 2016) but using BERT embeddings instead of fixed word embeddings.

### 5.5.1 Results

We present the results on Table 7. Models of the feature-based approach perform significantly worse compared to the ones of the fine-tuning approach. The performance gap is found to be much higher than the reported values for NER on English language (DEVLIN *et al.*, 2018; PETERS *et al.*, 2019) and reaches up to 2 points on BERTimbau Base and 3.5 points on BERTimbau-Large, although it can probably be reduced by further hyperparameter tuning.

In this setup, BERTimbau Base+BiLSTM-CRF achieves similar performances to BiLSTM-CRF+FlairBBP (row 3 of Table 6), which also uses contextual embeddings and a similar architecture. BERTimbau shows a slightly lower F1-score in the Selective scenario but higher F1-score in the Total scenario.

It is worth mentioning that BERTimbau models in this feature-based approach achieve better performances than a fine-tuned mBERT on this same task. While BERTimbau Large is the highest performer when fine-tuned, we observe that it experiences performance degradation when used in this feature-based approach, performing worse than the smaller Base variant but still better than mBERT.

## 5.6 Impact of long pretraining

To assess the impact of long pretraining stage on the performance of downstream tasks, we repeat part of the NER fine-tuning experiment (Section 4.2.2) using intermediate checkpoints of BERTimbau Base pretraining procedure. We train BERT models (without CRF) using the checkpoints of steps 235k, 505k and 700k, which correspond to 23.5%, 50.5% and 70% of the complete pretraining of 1M steps, respectively. All models are trained with the same hyperparameters and experimental setup of Section 4.2.2.

The results are displayed in Figure 6. Performances on the downstream task increase non-linearly with pretraining steps, with diminishing returns as pretraining pro-

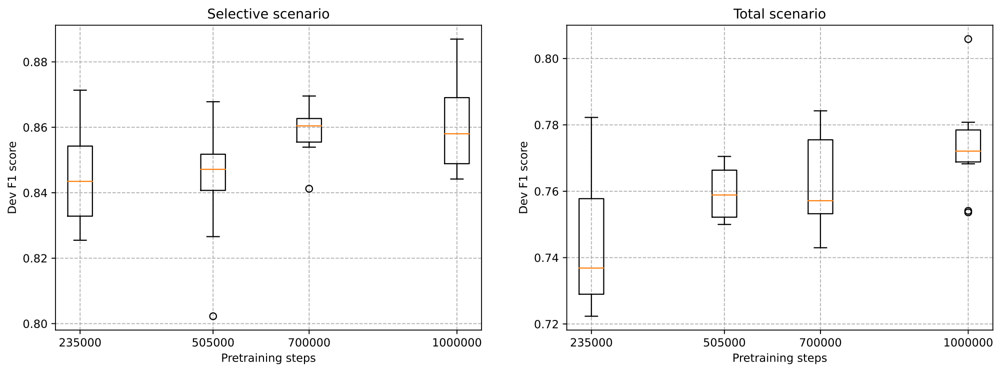

Figure 6 – Performance of BERTimbau Base on NER task using intermediate checkpoints of the pretraining stage. Reported scores on the validation set.

gresses. This is an expected result, as test performance of pretraining tasks are shown to follow a power law on the number of pretraining steps (KAPLAN *et al.*, 2020).

## 5.7 Tokenization analysis

One possible advantage of a monolingual BERT over multilingual BERT can be related to the WordPiece tokenizer vocabulary. The vocabulary size is a hyperparameter that limits the number of distinct recognizable tokens, which affects the size of the input token embedding matrix. Most monolingual BERT models have vocabulary sizes in the range of $30,000$ to $50,000$ tokens (DEVLIN *et al.*, 2018; LIU *et al.*, 2019; LAN *et al.*, 2019; CAñETE *et al.*, 2020). In comparison, mBERT has a vocabulary of $120,000$ tokens, which has to encompass tokens of over 100 languages and a variety of alphabets. When considering the usage of mBERT on a single specific language, the effective vocabulary size is usually much smaller than a monolingual equivalent, resulting in longer tokenized sequences. This happens because, in smaller vocabularies generated by BPE, only very frequent words will be present as individual tokens, causing the tokenization of most words to be composed of multiple subword units.

Considering that dot-product attention layers have quadratic complexity that imposes limitations on input sequence size of BERT and Transformer models in general (VASWANI *et al.*, 2017), a more efficient tokenization that produces shorter sequences allows inputing larger textual context in a sequence of maximum length $S$. This limitation is often encountered in sequence-level tasks such as document classification of long documents (SUN *et al.*, 2019).

One can also hypothesize that a tokenization that often breaks words into multiple subword units imposes a harder task on the model, since instead of receiving an embedding vector that readily represents the original word, the model will receive several vectors, one for each subword unit, that will have to be combined inside the model to form a complete

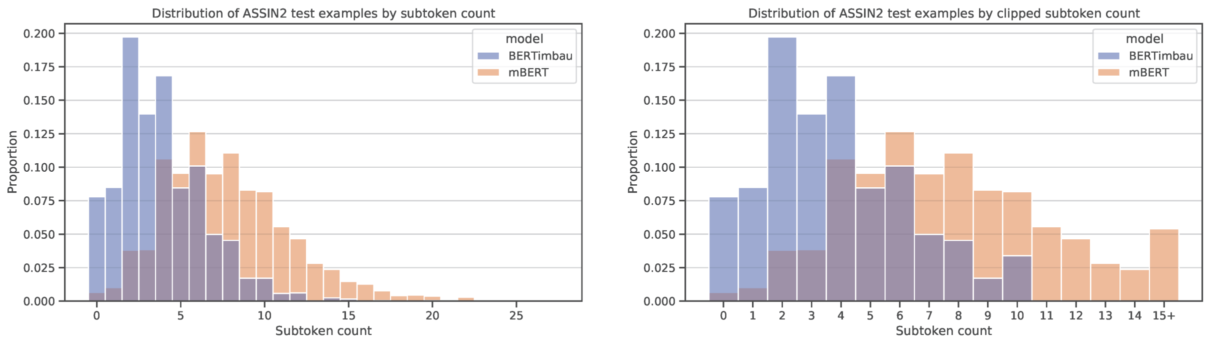

Figure 7 – Distribution of ASSIN2 test set examples binned by subtoken count in the tokenization of the premise and hypothesis texts, for BERTimbau and mBERT (left-side). A subtoken is any word continuation token that starts with “##”. The bin at $x = 0$ contains examples whose premise and hypothesis tokenizations are composed of only whole words. The histogram on the right-side is a clipped version that aggregates the right tail of the distributions in the $x = 10$ and $x = 15$ bins for BERTimbau and mBERT, respectively.

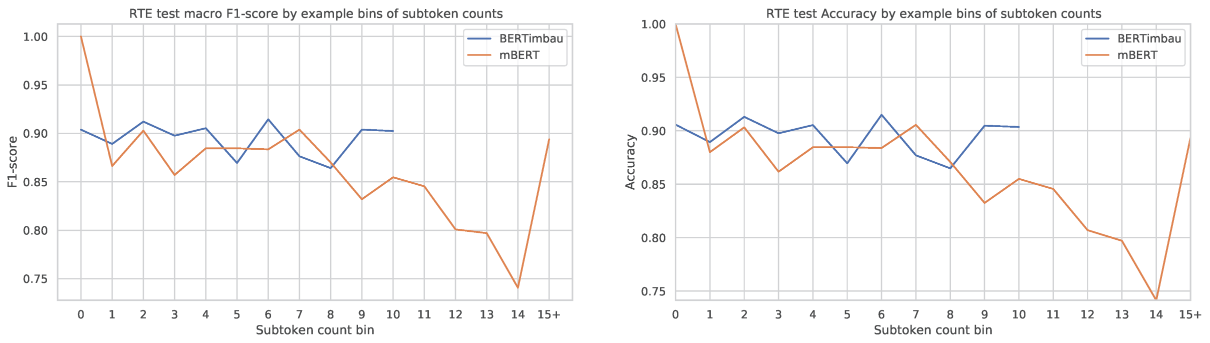

Figure 8 – Metrics of RTE task on ASSIN2 test set examples computed separately for each bin of the distribution of the right side of Figure 7.

representation.

In this experiment, we analyze the tokenizations produced by BERTimbau’s and mBERT’s tokenizers and compare the produced tokenized sequences for the evaluation tasks’ datasets. We compare the sequence lengths for each dataset, how each tokenizer behaves on the most frequent words and assess how subword unit tokenization may affect the performance of each task.

### 5.7.1 Tokenization effects on Sentence Textual Similarity and Recognizing Textual Entailment tasks

To investigate the effects of the tokenization on the RTE and STS tasks, we tokenize the ASSIN2 test set examples using BERTimbau and mBERT tokenizers. The examples are then binned by the subtoken count in the premise and hypothesis texts’ tokenizations, as can be seen in Figure 7. The mBERT tokenizer produces a median of

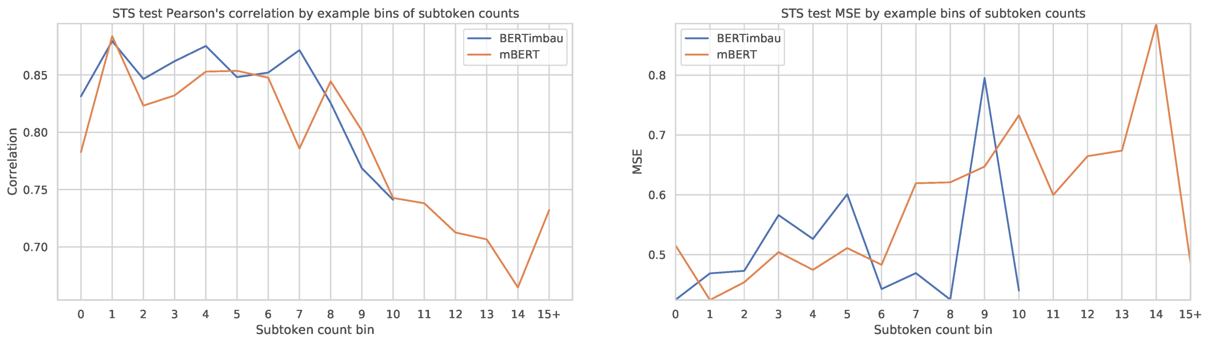

Figure 9 – Metrics of STS task on ASSIN2 test set examples computed separately for each bin of the distribution of the right side of Figure 7. Computation of Pearson's correlation uses global mean values for ground-truth and prediction similarity scores.

7 subtokens per example, while BERTimbau has a distribution skewed to the lower end and a median value of 3.5. Less than 1% of the examples tokenized using the multilingual vocabulary are composed of only whole words, while this proportion is 7.5% for the Portuguese tokenizer. Since both distributions have a right tail of low proportion bins, the right tails are clipped to the bins $x = 10$ for BERTimbau and $x = 15$ for mBERT, as shown in the right side of Figure 7, and these bins are used for the following metrics analysis.

For each example bin of the distribution, we compute the evaluation metrics for RTE and STS tasks to see how performance vary as tokenizations break words into more pieces, as shown in Figures 8 and 9. For the RTE task, there is almost no variation of F1-score and accuracy for BERTimbau as subtoken count increases. For mBERT, it appears to have a performance degradation beginning at the 9 subtokens bin, even though the 15+ bin recovers the performance of the lower bins. It is noticeable that mBERT performs on par with BERTimbau in the lower subtoken bins, and its global metrics are affected by the higher bins with worse performance, that comprise over 30% of the test set. Similar conclusions can be drawn for the STS task metrics, with BERTimbau metrics showing less overall variation while mBERT shows degradation on the higher bins. We argue, however, that it is not possible to draw precise conclusions, since bins at the distribution tails may have as few as 30 to 100 examples and, as such, the metrics of these bins can be dominated by the presence of easy or hard examples or with rarer words.

### 5.7.2 Tokenization effects on Named Entity Recognition

Given that Named Entity Recognition is a token-level task and the metrics are computed on entity-level, we take a distinct approach and analyze how the tokenization of the entities' words may affect the model performance. This way, we compare the tokenizations using BERTimbau and mBERT vocabularies on the ground-truth and predicted

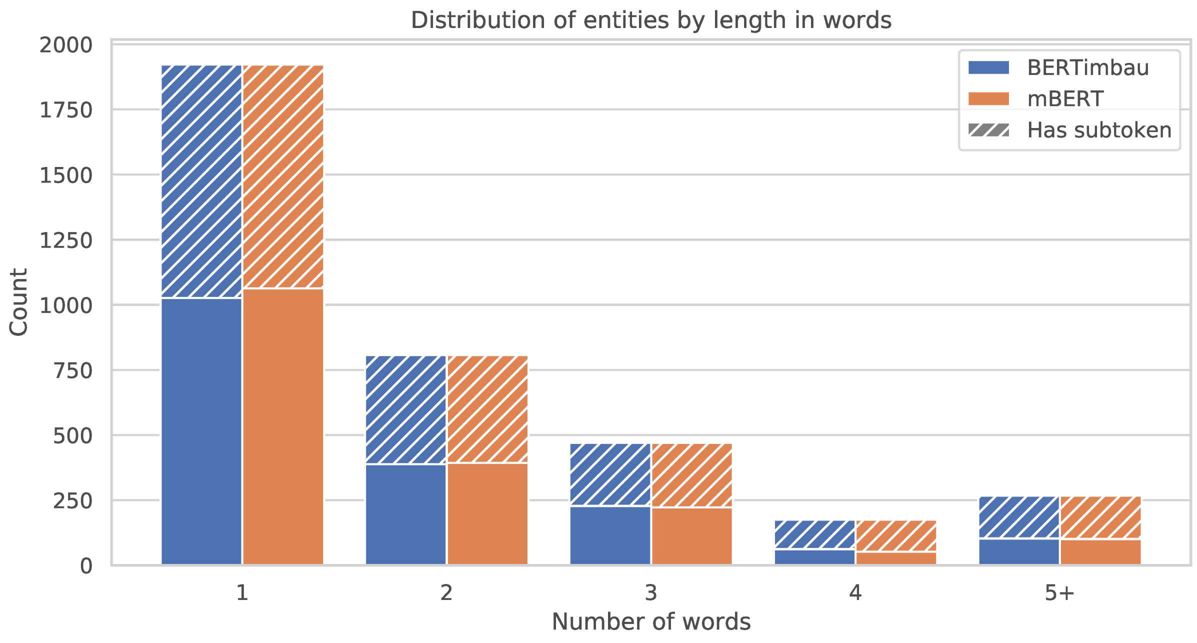

Figure 10 – Distribution of ground-truth entities of Mini HAREM dataset by number of words and presence of subtoken (word continuation token that starts with "##"), for BERTimbau and mBERT.

entities of the test dataset, Mini HAREM. The compared models are the BERT-CRF architecture on the Total scenario.

When casting NER as a sequence tagging problem, it is intuitively expected that longer entities — composed of more words and, hence, encoded as a longer tag sequence — might be harder to predict accurately, since any incorrectly predicted tag yields a wrong entity prediction, hurting both recall and precision. Considering this proposition, we bin the ground-truth and predicted entities by word count and, inside each bin, we distinguish between entities whose tokenization contains only whole words or contains at least one subtoken, as shown in Figure 10 for the ground-truth entities. It is worth emphasizing the definition of word in this context: we consider as words any sequence of characters that are produced by splitting a text into whitespace and punctuation characters, considering each punctuation character as a separate word. The length of an entity in words is independent of the vocabulary. Each word is then tokenized into one or multiple subword units by WordPiece tokenization using the model vocabulary, affecting the presence of subtoken or not.

As can be seen in Figure 10, the proportion of entities that have subtokens inside each bin is very similar between BERTimbau and mBERT vocabularies. This is not unexpected, since it is a general NER dataset and entities often contain proper names which are commonly rarer words. Even though the Portuguese vocabulary contains a larger set of common Portuguese proper names than the multilingual vocabulary, the opposite holds for foreign proper names, for instance, and these effects roughly balance each other out

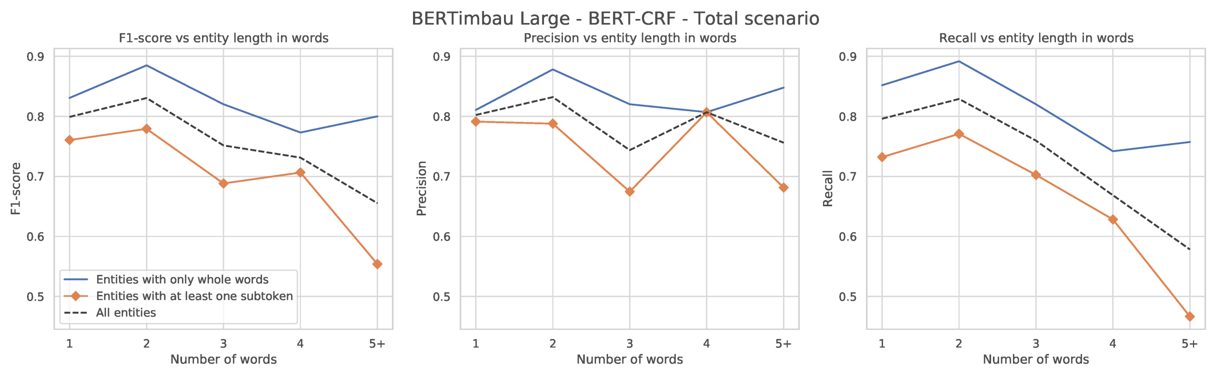

Figure 11 – Metrics of BERTimbau Large (BERT-CRF) on Named Entity Recognition task on the test dataset (Mini HAREM, Total scenario) binned by entity word count.

in this case.

We separately compute NER metrics (F1-score, Precision and Recall) for both analyzed models for each entity bin of Figure 10, as shown for BERTimbau in Figure 11 and for mBERT in Figure 12. By observing the dashed lines of F1-score plots, it can be seen that the proposition of longer entities showing worse performances hold for both models. The most relevant finding is that both models show worse performances to detect entities that have at least one subtoken, across all entity lengths, compared to entities that are composed of only whole words. The degree of performance degradation for increasing entity length is also higher for entities that contain subtokens.

We did not expect the presence of subtokens inside the entities to impact the task performance in such a strong and consistent manner. When defining the NER task, it is often mentioned that the class of a named entity is dependent not only on the entity itself, but also heavily dependent on its surrounding words (YADAV; BETHARD, 2018). For instance, the act of replacing a person's or organization's name in a sentence by another one of the same class should not make it necessarily easier or harder to detect — as long as the meaning and syntax of the sentence are preserved in this process—, since the surrounding context might give enough information to infer the entity class nonetheless. However, this observed performance degradation suggests that the choice of the replacement entity might affect the model's capabilities if it contains subtokens or not.

### 5.7.3 Discussion

Analyzing the impacts of the tokenization on these models can be an area of further research that has not been much explored. SciBERT (BELTAGY *et al.*, 2019) — a BERT trained on scientific articles in English — shows that having an in-domain vocabulary is beneficial, but they argue that larger benefits come rather from fine-tuning

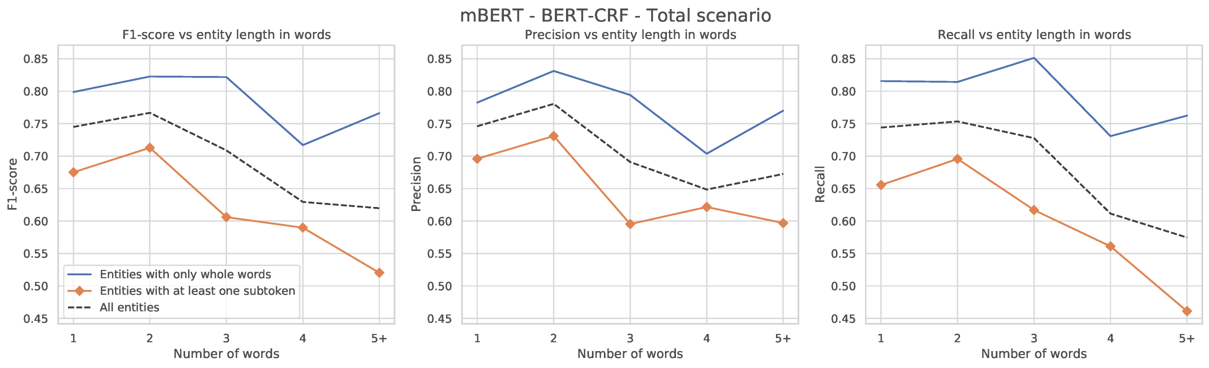

Figure 12 – Metrics of mBERT (BERT-CRF) on Named Entity Recognition task on the test dataset (Mini HAREM, Total scenario) binned by entity word count.

BERT on in-domain data than from a more suited vocabulary. These two factors are present in BERTimbau, since it is trained on more Portuguese data (and more diverse) than mBERT.

However, we believe there are other factors that, along with the tokenization method, might be hurting the performance of these models and can be subjects of research. While subword unit tokenization is a more robust alternative to word-level tokenization, allowing the representation of out-of-vocabulary words and using much smaller vocabularies, there is performance degradation when tokenization deviates from the ideal scenario where words are kept intact. We hypothesize that BERT architecture is not being capable of easily reconstructing a high quality word representation from multiple independent subword units.

# 6 Conclusions

In this work, we advance the study of deep learning models for NLP in Portuguese, especially the usage of pretrained language models in a transfer learning approach. We train BERT models for Brazilian Portuguese and evaluate their performances on three downstream NLP tasks.

In the pretraining stage, we use Wikipedia articles to generate a Portuguese vocabulary and then leverage millions of webpages from the brWaC corpus as unlabeled data to train Portuguese BERT models on self-supervised objectives. On the evaluation stage, we fine-tune the models supervisely on downstream tasks in two distinct experiments.

In the first experiment, we fine-tune our BERTimbau models on the ASSIN2 dataset to jointly solve Sentence Textual Similarity (STS) and Recognizing Textual Entailment (RTE) tasks. BERTimbau achieves state-of-the-art performances in both tasks, surpassing Multilingual BERT (mBERT) and previously published results in the literature, that comprise both Portuguese specific models and multilingual approaches.

In the second experiment, we fine-tune BERTimbau on named entity recognition (NER) task using the FirstHAREM and MiniHAREM datasets. We experiment with two NER architectures: plain BERT and BERT-CRF. Again, our best model achieves state-of-the-art results and shows a large performance improvement over mBERT and the previously best published result, that uses Portuguese Flair embeddings in a contextual embeddings setup, especially in the hardest Total scenario that considers all 10 named entity classes.

In additional experiments, we assess the usage of BERTimbau in a contextual embeddings setup by freezing its weights and training BERTimbau-BiLSTM-CRF models on NER task. Even though there is a notable performance drop, we show that contextual embeddings from BERTimbau Base outperform fine-tuned mBERT models, which can be a lower compute alternative for limited resource scenarios. We also validate the necessity of a long pretraining stage, that had been reported for English and other languages, for our Portuguese models by evaluating the performance of intermediate model checkpoints of the pretraining stage on NER task. Models pretrained for longer times show better performance in the end task, even though the pretraining stage had already started from pretrained checkpoints from mBERT and English BERT.

Lastly, we compare BERTimbau's Portuguese vocabulary to mBERT's multilingual vocabulary by looking at the produced tokenizations on the evaluation tasks. The Portuguese vocabulary produces smaller tokenized sentences, which corresponds to (1) keeping more words intact as a single token and (2) breaking words using a lower average number of subword units per word. We analyze how tokenizing words into multiple subword units might affect the model performance on the end tasks by binning task examples by tokenization statistics and computing evaluation metrics separately for each group. In particular to NER, we notice the models show inferior performance to detect named entities that contain at least one subword unit compared to only whole words. While this phenomenon could be related to the presence of rarer words in these examples, we hypothesize that the BERT architecture is not fully capable of reconstructing word representations from several subword units since it relies heavily on the positional embeddings that might not be sufficient. Further experiments and analyses can be performed to better understand these issues, such as exploring other evaluation tasks, distinct vocabulary sizes and vocabulary generation algorithms, or looking for alternatives to the simple positional embedding.

The field of deep learning applied to NLP is evolving at a rapid pace, with pre-trained language models recently becoming ubiquitous in most state-of-the-art (SOTA) systems. The current trend of publications focus heavily on SOTA by training exponentially larger capacity models that consume huge amounts of data and computational resources. Despite this path achieving unprecedented performances, if it is not closely followed by research aiming to optimize model decisions, such as more efficient architectures and training procedures, it might lead to an overconcentration of resources on a few organizations that have access to the required computational power.

In regards to multilingual models, we notice mBERT is one of the first works in this area. There are more recent models, such as XLM (LAMPLE; CONNEAU, 2019) and XLM-R (CONNEAU *et al.*, 2019), that can be experimented with in future work. These models propose new training procedures that allow greater knowledge sharing across languages without sacrificing much per-language performance, and avoiding vocabulary dilution. Even though Portuguese is heavily present on the internet and, as such, it has enough unlabeled data to train large monolingual language models, the cross-lingual transfer allowed by multilingual models can be extremely beneficial for Portuguese by leveraging labeled datasets of other languages to alleviate the annotated data limitation that is commonly faced by NLP researchers and developers.

Regarding the future of this area, the trend of ever-growing model sizes will probably continue over the next years, since neural networks benefit from higher model capacity specially when larger amounts of data are available. The availability of NLP data is endless when we consider the rate of content generated in the internet, so this is a natural path. Fortunately, there is a strong and active branch of research focused on reducing hardware requirements, such as model distillation, model quantization and optimized architectures.

Since the necessity of labeled datasets is the weaker aspect of deep learning applications in general, every possibility of leveraging transfer learning should be exploited. This way, large multilingual models are a natural path to enable sharing data across languages and alleviate this problem. Monolingual models, instead of trained individually, might be distilled when there is necessity, for instance.

## 6.1 Contributions and publications

The main results of this dissertation were published as an homonymous conference paper in 9th Brazilian Conference on Intelligent Systems (BRACIS), where it was awarded the 1st place in the classification of Best Papers in the conference:

- Souza, F., Nogueira, R., Lotufo, R. (2020, October). *BERTimbau: Pretrained BERT Models for Brazilian Portuguese.* In 9th Brazilian Conference on Intelligent Systems (pp. 403-417). Springer, Cham.

Preliminary results of BERTimbau pretraining and evaluation on NER task were published as a preprint article and has been cited by over 33 works according to Google Scholar:

- Souza, F., Nogueira, R., Lotufo, R. (2019, September). *Portuguese named entity recognition using BERT-CRF.* arXiv preprint arXiv:1909.10649.

An extended abstract was accepted as a talk into the OpenCor[1] 2020 Forum, which aims to facilitate the discovery and access of Latin American and Iberian languages free resources. The forum took place in the PROPOR 2020 conference.

Lastly, our models are available to the community in the Transformers open-source library (WOLF *et al.*, 2019), where they have over 120,000 registered downloads, as shown in Figure 13. Code to reproduce our evaluation experiments is available in our GitHub repository[2].

[1] https://opencor.gitlab.io/
[2] https://github.com/neuralmind-ai/portuguese-bert

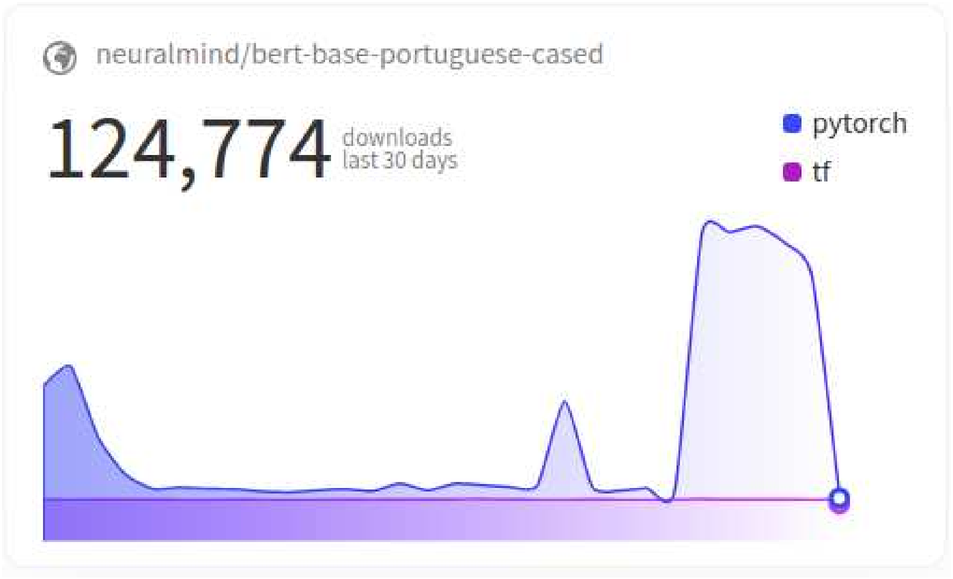

Figure 13 – Registered downloads for BERTimbau Base model in the Transformers library over the period of 20/10/2020 to 20/11/2020, as reported by the library's webpage.

# Bibliography

AKBIK, A.; BLYTHE, D.; VOLLGRAF, R. Contextual string embeddings for sequence labeling. In: *COLING 2018, 27th International Conference on Computational Linguistics.* [S.l.: s.n.], 2018. p. 1638–1649. Cited 4 times on pages 15, 28, 37, and 44.

BA, J. L.; KIROS, J. R.; HINTON, G. E. *Layer Normalization.* 2016. Cited on page 21.

BAHDANAU, D.; CHO, K.; BENGIO, Y. *Neural Machine Translation by Jointly Learning to Align and Translate.* 2014. Cited on page 23.

BALY, F.; HAJJ, H. *et al.* Arabert: Transformer-based model for Arabic language understanding. In: *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection.* [S.l.: s.n.], 2020. p. 9–15. Cited on page 16.

BELTAGY, I.; LO, K.; COHAN, A. Scibert: Pretrained language model for scientific text. In: *EMNLP.* [S.l.: s.n.], 2019. Cited on page 50.

BENGIO, Y.; DUCHARME, R.; VINCENT, P.; JANVIN, C. A neural probabilistic language model. *The journal of machine learning research*, JMLR. org, v. 3, p. 1137–1155, 2003. Cited on page 18.

CARMO, D.; PIAU, M.; CAMPIOTTI, I.; NOGUEIRA, R.; LOTUFO, R. Ptt5: Pretraining and validating the t5 model on brazilian portuguese data. *arXiv preprint arXiv:2008.09144*, 2020. Cited on page 29.

CASTRO, P.; FELIX, N.; SOARES, A. Contextual representations and semi-supervised named entity recognition for portuguese language. In: . [S.l.: s.n.], 2019. Cited on page 28.

CASTRO, P. V. Q. d. *et al.* Aprendizagem profunda para reconhecimento de entidades nomeadas em domínio jurídico. Universidade Federal de Goiás, 2019. Cited on page 28.

CASTRO, P. V. Q. d.; SILVA, N. F. F. d.; SOARES, A. d. S. Portuguese named entity recognition using lstm-crf. In: VILLAVICENCIO, A.; MOREIRA, V.; ABAD, A.; CASELI, H.; GAMALLO, P.; RAMISCH, C.; OLIVEIRA, H. G.; PAETZOLD, G. H. (Ed.). *Computational Processing of the Portuguese Language.* Cham: Springer International Publishing, 2018. p. 83–92. ISBN 978-3-319-99722-3. Cited 3 times on pages 30, 35, and 43.

CAñETE, J.; CHAPERON, G.; FUENTES, R.; PéREZ, J. Spanish pre-trained bert model and evaluation data. In: *to appear in PML4DC at ICLR 2020.* [S.l.: s.n.], 2020. Cited 2 times on pages 16 and 46.

COLLOBERT, R.; WESTON, J. A unified architecture for natural language processing: Deep neural networks with multitask learning. In: *Proceedings of the 25th international conference on Machine learning.* [S.l.: s.n.], 2008. p. 160–167. Cited on page 15.

COLLOBERT, R.; WESTON, J.; BOTTOU, L.; KARLEN, M.; KAVUKCUOGLU, K.; KUKSA, P. Natural language processing (almost) from scratch. *Journal of machine learning research*, v. 12, n. Aug, p. 2493–2537, 2011. Cited on page 15.

CONNEAU, A.; KHANDELWAL, K.; GOYAL, N.; CHAUDHARY, V.; WENZEK, G.; GUZMÁN, F.; GRAVE, E.; OTT, M.; ZETTLEMOYER, L.; STOYANOV, V. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*, 2019. Cited on page 53.

CORTES, C.; VAPNIK, V. Support-vector networks. *Machine learning*, Springer, v. 20, n. 3, p. 273–297, 1995. Cited on page 15.

DAI, A. M.; LE, Q. V. Semi-supervised sequence learning. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2015. p. 3079–3087. Cited 2 times on pages 16 and 18.

DELOBELLE, P.; WINTERS, T.; BERENDT, B. RobBERT: a Dutch RoBERTa-based Language Model. *arXiv preprint arXiv:2001.06286*, 2020. Cited on page 16.

DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K.; FEI-FEI, L. Imagenet: A large-scale hierarchical image database. In: IEEE. *2009 IEEE conference on computer vision and pattern recognition*. [S.l.], 2009. p. 248–255. Cited on page 18.

DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. Cited 8 times on pages 16, 19, 30, 38, 40, 44, 45, and 46.

FILHO, J. W.; WILKENS, R.; IDIART, M.; VILLAVICENCIO, A. The brwac corpus: A new open resource for brazilian portuguese. In: . [S.l.: s.n.], 2018. Cited 2 times on pages 16 and 31.

FONSECA, E.; ALVARENGA, J. P. R. Wide and deep transformers applied to semantic relatedness and textual entailment. In: OLIVEIRA, H. G.; REAL, L.; FONSECA, E. (Ed.). *Proceedings of the ASSIN 2 Shared Task: Evaluating Semantic Textual Similarity and Textual Entailment in Portuguese*. [s.n.], 2020. (CEUR Workshop Proceedings, 2583), p. 68–76. ISSN 1613-0073. Available at: <http://ceur-ws.org/Vol-2583/>. Cited on page 42.

GURURANGAN, S.; MARASOVIĆ, A.; SWAYAMDIPTA, S.; LO, K.; BELTAGY, I.; DOWNEY, D.; SMITH, N. A. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*, 2020. Cited on page 28.

HARTMANN, N.; FONSECA, E.; SHULBY, C.; TREVISO, M.; RODRIGUES, J.; ALUISIO, S. Portuguese word embeddings: Evaluating on word analogies and natural language tasks. *arXiv preprint arXiv:1708.06025*, 2017. Cited on page 28.

HENDRYCKS, D.; GIMPEL, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. Cited on page 23.

HOWARD, J.; RUDER, S. Universal language model fine-tuning for text classification. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. [S.l.: s.n.], 2018. p. 328–339. Cited on page 16.

KAPLAN, J.; MCCANDLISH, S.; HENIGHAN, T.; BROWN, T. B.; CHESS, B.; CHILD, R.; GRAY, S.; RADFORD, A.; WU, J.; AMODEI, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. Cited on page 46.

KIM, Y. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014. Cited on page 15.

KUDO, T.; RICHARDSON, J. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018. Cited 2 times on pages 19 and 30.

KURATOV, Y.; ARKHIPOV, M. Adaptation of deep bidirectional multilingual transformers for russian language. *arXiv preprint arXiv:1905.07213*, 2019. Cited on page 16.

LAFFERTY, J. D.; MCCALLUM, A.; PEREIRA, F. C. N. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In: *Proceedings of the Eighteenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001. (ICML '01), p. 282–289. ISBN 1558607781. Cited on page 37.

LAMPLE, G.; BALLESTEROS, M.; SUBRAMANIAN, S.; KAWAKAMI, K.; DYER, C. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016. Version 3. Available at: <http://arxiv.org/abs/1603.01360>. Cited 2 times on pages 37 and 45.

LAMPLE, G.; CONNEAU, A. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*, 2019. Cited on page 53.

LAN, Z.; CHEN, M.; GOODMAN, S.; GIMPEL, K.; SHARMA, P.; SORICUT, R. Albert: A lite BERT for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019. Cited 2 times on pages 16 and 46.

LIN, T.-Y.; GOYAL, P.; GIRSHICK, R.; HE, K.; DOLLÁR, P. Focal loss for dense object detection. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2017. p. 2980–2988. Cited on page 43.

LIU, Y.; OTT, M.; GOYAL, N.; DU, J.; JOSHI, M.; CHEN, D.; LEVY, O.; LEWIS, M.; ZETTLEMOYER, L.; STOYANOV, V. Roberta: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019. Cited 3 times on pages 16, 41, and 46.

LUONG, T.; PHAM, H.; MANNING, C. D. Effective approaches to attention-based neural machine translation. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, 2015. p. 1412–1421. Cited on page 24.

MARTIN, L.; MULLER, B.; SUÁREZ, P. J. O.; DUPONT, Y.; ROMARY, L.; CLERGERIE, É. V. de la; SEDDAH, D.; SAGOT, B. CamemBERT: a tasty French language model. *arXiv preprint arXiv:1911.03894*, 2019. Cited on page 16.

MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. Cited 2 times on pages 15 and 28.

NADEAU, D.; SEKINE, S. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, John Benjamins, v. 30, n. 1, p. 3–26, 2007. Cited 2 times on pages 32 and 33.

NGUYEN, D. Q.; NGUYEN, A. T. PhoBERT: Pre-trained language models for Vietnamese. *arXiv preprint arXiv:2003.00744*, 2020. Cited on page 16.

OLIVEIRA, H. G.; REAL, L.; FONSECA, E. (Ed.). *Proceedings of the ASSIN 2 Shared Task: Evaluating Semantic Textual Similarity and Textual Entailment in Portuguese, Extended Semantic Web Conference*, (CEUR Workshop Proceedings, 2583). [s.n.], 2020. ISSN 1613-0073. Available at: <http://ceur-ws.org/Vol-2583/>. No citations in the text.

PENNINGTON, J.; SOCHER, R.; MANNING, C. Glove: Global vectors for word representation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).* Doha, Qatar: Association for Computational Linguistics, 2014. p. 1532–1543. Available at: <https://www.aclweb.org/anthology/D14-1162>. Cited 2 times on pages 15 and 28.

PETERS, M.; NEUMANN, M.; IYYER, M.; GARDNER, M.; CLARK, C.; LEE, K.; ZETTLEMOYER, L. Deep contextualized word representations. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers).* [S.l.: s.n.], 2018. p. 2227–2237. Cited 4 times on pages 15, 16, 20, and 28.

PETERS, M. E.; RUDER, S.; SMITH, N. A. To tune or not to tune? adapting pretrained representations to diverse tasks. In: *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019).* [S.l.: s.n.], 2019. p. 7–14. Cited 2 times on pages 44 and 45.

POLIGNANO, M.; BASILE, P.; GEMMIS, M. de; SEMERARO, G.; BASILE, V. AlBERTo: Italian BERT Language Understanding Model for NLP Challenging Tasks Based on Tweets. In: *Proceedings of the Sixth Italian Conference on Computational Linguistics (CLiC-it 2019).* CEUR, 2019. v. 2481. Available at: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85074851349&partnerID=40&md5=7abed946e06f76b3825ae5e294ffac14>. Cited on page 16.

RADFORD, A.; NARASIMHAN, K.; SALIMANS, T.; SUTSKEVER, I. *Improving language understanding with unsupervised learning.* [S.l.], 2018. Cited 2 times on pages 16 and 20.

RADFORD, A.; WU, J.; CHILD, R.; LUAN, D.; AMODEI, D.; SUTSKEVER, I. Language models are unsupervised multitask learners. *OpenAI Blog*, v. 1, n. 8, p. 9, 2019. Cited 2 times on pages 18 and 28.

RAFFEL, C.; SHAZEER, N.; ROBERTS, A.; LEE, K.; NARANG, S.; MATENA, M.; ZHOU, Y.; LI, W.; LIU, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019. Cited on page 16.

REAL, L.; FONSECA, E.; OLIVEIRA, H. G. The assin 2 shared task: A quick overview. In: _____. [S.l.: s.n.], 2020. p. 406–412. ISBN 978-3-030-41504-4. Cited on page 32.

RODRIGUES, R.; COUTO, P.; RODRIGUES, I. Ipr: The semantic textual similarity and recognizing textual entailment systems. In: OLIVEIRA, H. G.; REAL, L.; FONSECA, E. (Ed.). *Proceedings of the ASSIN 2 Shared Task: Evaluating Semantic Textual Similarity and Textual Entailment in Portuguese.* [s.n.], 2020. (CEUR Workshop Proceedings, 2583), p. 39–47. ISSN 1613-0073. Available at: <http://ceur-ws.org/Vol-2583/>. Cited on page 42.

RODRIGUES, R.; SILVA, J. da; CASTRO, P.; FELIX, N.; SOARES, A. Multilingual transformer ensembles for portuguese natural language tasks. In: OLIVEIRA, H. G.; REAL, L.; FONSECA, E. (Ed.). *Proceedings of the ASSIN 2 Shared Task: Evaluating Semantic Textual Similarity and Textual Entailment in Portuguese.* [s.n.], 2020. (CEUR Workshop Proceedings, 2583), p. 27–38. ISSN 1613-0073. Available at: <http://ceur-ws.org/Vol-2583/>. Cited on page 42.

RODRIGUES, R. C.; RODRIGUES, J.; CASTRO, P. V. Q. de; SILVA, N. F. F. da; SOARES, A. Portuguese language models and word embeddings: Evaluating on semantic similarity tasks. In: QUARESMA, P.; VIEIRA, R.; ALUÍSIO, S.; MONIZ, H.; BATISTA, F.; GONÇALVES, T. (Ed.). *Computational Processing of the Portuguese Language.* Cham: Springer International Publishing, 2020. p. 239–248. ISBN 978-3-030-41505-1. Cited on page 28.

SANG, E. F. T. K.; MEULDER, F. D. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003.* [s.n.], 2003. p. 142–147. Available at: <https://www.aclweb.org/anthology/W03-0419>. Cited on page 36.

SANTOS, C. D.; ZADROZNY, B. Learning character-level representations for part-of-speech tagging. In: PMLR. *International Conference on Machine Learning.* [S.l.], 2014. p. 1818–1826. Cited on page 28.

SANTOS, C. N. d.; GUIMARAES, V. Boosting named entity recognition with neural character embeddings. *arXiv preprint arXiv:1505.05008*, 2015. Version 2. Available at: <https://arxiv.org/abs/1505.05008>. Cited 3 times on pages 35, 37, and 43.

SANTOS, D.; SECO, N.; CARDOSO, N.; VILELA, R. HAREM: An advanced ner evaluation contest for Portuguese. 2006. Cited on page 35.

SANTOS, J.; CONSOLI, B.; SANTOS, C. dos; TERRA, J.; COLLONINI, S.; VIEIRA, R. Assessing the impact of contextual embeddings for portuguese named entity recognition. In: *8th Brazilian Conference on Intelligent Systems, BRACIS, Bahia, Brazil, October 15-18.* [S.l.: s.n.], 2019. p. 437–442. Cited 3 times on pages 28, 35, and 43.

SANTOS, J.; TERRA, J.; CONSOLI, B. S.; VIEIRA, R. Multidomain contextual embeddings for named entity recognition. In: *IberLEF@SEPLN.* [S.l.: s.n.], 2019. Cited on page 28.

SCHUSTER, M.; NAKAJIMA, K. Japanese and Korean voice search. In: IEEE. *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* [S.l.], 2012. p. 5149–5152. Cited 3 times on pages , 30, and 38.

SCHUSTER, M.; NAKAJIMA, K. Japanese and korean voice search. In: IEEE. *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.], 2012. p. 5149–5152. Cited on page 19.

SENNRICH, R.; HADDOW, B.; BIRCH, A. Neural machine translation of rare words with subword units. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, 2016. p. 1715–1725. Cited on page 19.

SENNRICH, R.; HADDOW, B.; BIRCH, A. Neural machine translation of rare words with subword units. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, 2016. p. 1715–1725. Available at: <https://www.aclweb.org/anthology/P16-1162>. Cited on page 30.

SPEER, R. *ftfy*. 2019. Zenodo. Version 5.5. Available at: <https://doi.org/10.5281/zenodo.2591652>. Cited on page 31.

SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, JMLR. org, v. 15, n. 1, p. 1929–1958, 2014. Cited on page 22.

SUN, C.; QIU, X.; XU, Y.; HUANG, X. How to fine-tune bert for text classification? In: SPRINGER. *China National Conference on Chinese Computational Linguistics*. [S.l.], 2019. p. 194–206. Cited on page 46.

TJONG, E. F.; SANG, K.; VEENSTRA, J. Representing text chunks. In: *Ninth Conference of the European Chapter of the Association for Computational Linguistics*. Bergen, Norway: Association for Computational Linguistics, 1999. Available at: <https://www.aclweb.org/anthology/E99-1023>. Cited on page 34.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, Ł.; POLOSUKHIN, I. Attention is all you need. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2017. p. 5998–6008. Cited 6 times on pages 16, 19, 20, 24, 28, and 46.

VRIES, W. d.; CRANENBURGH, A. v.; BISAZZA, A.; CASELLI, T.; NOORD, G. v.; NISSIM, M. BERTje: A Dutch BERT Model. *arXiv preprint arXiv:1912.09582*, dez. 2019. Available at: <http://arxiv.org/abs/1912.09582>. Cited on page 16.

WOLF, T.; DEBUT, L.; SANH, V.; CHAUMOND, J.; DELANGUE, C.; MOI, A.; CISTAC, P.; RAULT, T.; LOUF, R.; FUNTOWICZ, M.; DAVISON, J.; SHLEIFER, S.; PLATEN, P. von; MA, C.; JERNITE, Y.; PLU, J.; XU, C.; SCAO, T. L.; GUGGER, S.; DRAME, M.; LHOEST, Q.; RUSH, A. M. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019. Cited on page 54.

YADAV, V.; BETHARD, S. A survey on recent advances in named entity recognition from deep learning models. In: *Proceedings of the 27th International Conference on Computational Linguistics*. [S.l.: s.n.], 2018. p. 2145–2158. Cited on page 50.

YANG, Z.; DAI, Z.; YANG, Y.; CARBONELL, J.; SALAKHUTDINOV, R.; LE, Q. V. XLNet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019. Cited on page 16.

YOSINSKI, J.; CLUNE, J.; BENGIO, Y.; LIPSON, H. How transferable are features in deep neural networks? In: *Advances in neural information processing systems*. [S.l.: s.n.], 2014. p. 3320–3328. Cited on page 18.