EMBEDDING GENERATION FOR TEXT CLASSIFICATION OF USER REVIEWS IN BRAZILIAN PORTUGUESE: FROM BAG-OF-WORDS TO TRANSFORMERS

Frederico Dias Souza

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: João Baptista de Oliveira e Souza Filho

Rio de Janeiro
Dezembro de 2022

EMBEDDING GENERATION FOR TEXT CLASSIFICATION OF USER REVIEWS IN BRAZILIAN PORTUGUESE: FROM BAG-OF-WORDS TO TRANSFORMERS

Frederico Dias Souza

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Orientador: João Baptista de Oliveira e Souza Filho

Aprovada por: Prof. João Baptista de Oliveira e Souza Filho, D.Sc.
Prof. José Gabriel Rodriguez Carneiro Gomes, Ph.D.
Prof. Thiago Alexandre Salgueiro Pardo, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
DEZEMBRO DE 2022

Dias Souza, Frederico

Embedding generation for Text Classification of User Reviews in Brazilian Portuguese: From Bag-of-Words to Transformers/Frederico Dias Souza. – Rio de Janeiro: UFRJ/COPPE, 2022.

XV, 87 p.: il.; 29, 7cm.

Orientador: João Baptista de Oliveira e Souza Filho

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2022.

Referências Bibliográficas: p. 75 – 87.

1. Machine Learning. 2. Deep Learning. 3. Natural Language Processing. 4. Sentiment Analysis. 5. Text Classification. I. de Oliveira e Souza Filho, João Baptista. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

"Não há saber mais ou saber
menos: há saberes diferentes."
Paulo Freire

Agradecimentos

Agradeço aos meus pais, irmã e namorada pelo irrestrito apoio, tornando a trajetória consideravelmente mais fácil.

Agradeço ao meu professor orientador João Baptista pela agradável parceria e absoluta dedicação ao longo dessa trajetória. Graças a ele, obtive enorme desenvolvimento técnico e profissional e consegui publicar em importantes conferências da área.

Agradeço aos demais professores, funcionários e colegas da COPPE por viabilizarem esse sonho.

Agradeço a todos que tornam possível a universidade pública, gratuita e de qualidade.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

EMBEDDING GENERATION FOR TEXT CLASSIFICATION OF USER REVIEWS IN BRAZILIAN PORTUGUESE: FROM BAG-OF-WORDS TO TRANSFORMERS

Frederico Dias Souza

December/2022

Advisor: João Baptista de Oliveira e Souza Filho

Department: Electrical Engineering

Text Classification is one of the most classical and studied Natural Language Processing (NLP) tasks. To classify documents accurately, a common approach is to provide a robust numerical representation, a process known as embedding. Embedding is a key NLP field that faced a significant advance in the last decade, especially after the popularization of Deep Learning models for solving NLP tasks, including Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Transformer-based Language Models (TLMs). Despite achievements, the literature regarding generating embeddings for Brazilian Portuguese texts still needs further investigation compared to the English language. Therefore, this work provides an experimental study of embedding techniques targeting a binary sentiment classification of user reviews in Brazilian Portuguese. This analysis includes classical (Bag-of-Words) to state-of-the-art (Transformer-based) NLP models. We evaluate the models over five open-source datasets containing pre-defined partitions to encourage reproducibility. The Fine-tuned TLMs attain the best results for all cases, followed by the Feature-based TLM, LSTM, and CNN, with alternate ranks depending on the database under analysis.

Contents

List of Figures

List of Tables

List of Abbreviations

BERT	Bidirectional Encoder Representations from Transformers, p. 22
BoW	Bag-of-Words, p. 8
CNN	Convolutional Neural Network, p. 2
DL	Deep Learning, p. 2
GCN	Graph Convolutional Network, p. 17
GNN	Graph Neural Network, p. 17
GRU	Gated Recurrent Unit, p. 13
LR	Logistic Regression, p. 8
LSA	Latent Semantic Analysis, p. 9
LSTM	Long Short-Term Memory, p. 13
MLM	Masked Language Model, p. 23
ML	Machine Learning, p. 2
NER	Named Entity Recognition, p. 23
NLP	Natural Language Processing, p. 1
NMT	Neural Machine Translation, p. 18
NSP	Next Sentence Prediction, p. 23
QRNN	Quasi-Recurrent Neural Networks, p. 17
RNN	Recurrent Neural Network, p. 2
SIF	Smooth Inverse Frequency, p. 11
T5	Text-to-Text Transfer Transformer, p. 24

TF-IDF Term Frequency-Inverse Document Frequency, p. 8

TLM Transformer-based Language Model, p. 2

Chapter 1

Introduction

The opinion has a significant influence over the human behavior. Nowadays, every company, brand, and even political group struggles to know individual and collective opinions, aiming to leverage their business [1]. For instance, streaming services are often interested in discovering user's opinions about a movie to feed their recommendation systems. In turn, e-commerce platforms always desire to be aware of customers impressions and satisfaction about products and services to improve user experience.

With the growing volume of data automatically generated due to users' interaction with a myriad of digital platforms, such opinions can be automatically mined by analysing their reviews, not requiring the conception of specific surveys or interviews. For this reason, sentiment analysis over user reviews is of particular relevance to companies' decision-making, leading to the current urgent need of algorithms able to perform this task in a large scale.

In fact, making computers to "understand" the content of a text is highly challenging. To accomplish this task, the first step commonly involves transforming texts into numbers [2]. The process of providing a vector representation to a word or document is known as embedding. It figures out as one of the key parts of automatically inferring sentiments from texts, representing the core study of this work.

1.1 Natural Language Processing and Machine Learning

Natural Language Processing (NLP) refers to a set of techniques involving applying statistical methods, with or without linguistics insights, taking unstructured natural language data as input, and making the human language accessible to computers [3, 4]. This understanding of the text by machines consists of transforming texts into

useable computational representations and using discrete or continuous structures, such as vectors or tensors, graphs, and trees [2].

According to Eisenstein [3], NLP is a broad research field with many related areas. One of them is Computational Linguistics, which focuses on studying the language itself and computational methods for a clearer understanding of its functioning. On the other hand, NLP is focused on developing and analyzing algorithms targeting the numerical representation of the human language. Of course, understanding linguistic concepts may contribute to the success of this tasks but what really matters is how well the computational model solves some specific problem. Another area closely related to NLP is Machine Learning (ML), representing the basis of most modern NLP applications, which allows building systems from past data and examples. Most of the current NLP research can be considered applied ML. In this work, the focus is not on the study of the Portuguese language itself; it is only used as a proxy for evaluating different ML techniques.

The best ML methods for dealing with language data include supervised algorithms [4]. Supervised learning refers to a data modelling case which the ground truth (or target) is available for all dataset instances. For example, in document classification, the target is a categorical label, and it represents how this document must be classified regarding a predefined set of labels [2]. This example illustrates an NLP task referred to as Text Classification (TC), which involves assigning labels to textual units, and it represents this dissertation's main object of study.

TC represents one of the most fundamental NLP tasks and may include sentiment analysis, topic analysis, question answering, and natural language inference as primary objectives. Particularly, sentiment analysis, also called opinion mining, targets to infer people's opinions expressed in textual data, predicting if they are positive or negative [5–7].

NLP solutions exploiting ML models have been existing for a long time. However, in the last decade, a brand of algorithms popularly known as Deep Learning (DL) achieved impressive results, a process started with the Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) [2]. More recently, the Transformers architecture revolutionized the NLP area by proposing Transformer-based Language Models (TLMs), responsible for redefining the state-of-the-art in many tasks. TLMs have the distinguishing characteristic of processing large portions of texts in parallel on a much deeper semantic level than previous NLP models, one of the primary reasons for their performance. Moreover, such models also have the benefit of addressing a range of NLP problems through transfer learning.

Regarding Brazilian Portuguese, several datasets, pre-trained resources, and models have been developed in the recent years for sentiment analysis in Brazilian Portuguese texts, as can be seen in Pereira's work [8] and *Opinando* project

[9], proposed by the University of São Paulo (USP). However, to our best knowledge, despite such valuable resources, studies comparing embeddings generated by the current state-of-the-art models (Transformed-based) with more consolidated approaches, such as Bag-of-Words and classical Deep Learning models, still require further exploration compared to the English language.

1.2 Research Hypotheses

Regarding the predictive performance of the analyzed methods, we expect the following ordering for all datasets: Bag-of-Words $<$ Classical Deep Learning $<$ Feature-based Transformer-based Language Model $<$ Fine-tuned Transformer-based Language Model. Within Bag-of-Words models, we expect TF-IDF models to perform better than Bag-of-Word Vectors. As for Classical Deep Learning, Convolutional Neural Networks should perform less than or equal to Recurrent Neural Networks since the texts are generally short and should benefit less from the second's greater learning capacity.

As for the Feature-based Transformer-based Language Model, the different embedding extraction techniques analyzed should significantly influence the predictive power and vary according to the pre-trained model adopted. Finally, the Fine-tuned Transformer-based Language Model might overcome all previous approaches. However, it may reach results close to the feature-based model when we adopt a more elaborate feature extraction process.

Finally, as for datasets, those with longer sentences should benefit more from more complex models. In addition, the domain must also have some influence. That is, the models may struggle in datasets that involve reviews with greater subjectivity, such as movie analysis.

1.3 Objectives and Contributions

Due to the growing importance of inferring users' opinions in large amounts of data and motivated by the necessity of experimental studies including the more recent NLP models with Brazilian Portuguese, this work aims at conducting a comprehensive experimental study of embedding alternatives targeting the sentiment classification task for Brazilian Portuguese texts. We contemplated from traditional solutions to the state-of-the-art models in five open-source databases to ensure the generality and reproducibility of the findings.

Although some works [10, 11] emphasize the importance of including a class representing neutrality in the sentiment analysis task, we decided here to focus only on the positive and negative reviews, disregarding those neutral, thus assuming a

binary sentiment analysis task. These classes represent a more direct user feedback about e-commerce transactions that often requires a more effective response from customer satisfaction services, a central aspect in future applications aimed by us, in opposition to the still relevant but fuzzy neutral category.

In the following, we provide a brief overview of the central objectives and contributions of this thesis:

1. It collects five public annotated datasets of user reviews in Brazilian Portuguese targeting sentiment classification. We propose pre-defined partitions for each target (polarity and 1-5 rating), hoping to encourage other interested researchers to evaluate the performance of alternative models in the same partitions, making easier a further analysis and direct comparisons with the results reported here. These partitions are stored in a public repository [1].

2. It provides a comprehensive study of feature generation techniques for text classification in Brazilian Portuguese, covering from strategies based on corpus statistics to transfer learning approaches, including word embedding strategies and Transformer-based Language Models. This study also addresses intrinsic embeddings generated by an end-to-end optimization of models like CNNs, RNNs, and Fine-tuned TLMs.

3. It evaluates the predictive performance of Transformer-based Language Models available for Brazilian Portuguese. Despite the recent advances in the NLP area, open-source models in this language have emerged only recently. Therefore, more systematic studies are lacking, especially regarding the text classification task. Thus, this study includes three multilingual and seven Portuguese TLMs.

4. It aims to contribute to practitioners when choosing a feature extraction model for text classification, providing some insights into experimental trade-offs between the predictive performance and the computational resources required by some state-of-the-art models.

5. This master's dissertation generated two works published in international conferences and one journal article, as described in Appendix A.

1.4 Thesis Outline

In the second chapter, "Bibliographic Review", we briefly describe previous studies on some NLP topics and works of interest for this dissertation. After contextualizing

[1] `https://www.kaggle.com/datasets/fredericods/ptbr-sentiment-analysis-datasets`

the text classification methods, we visit some crucial references with different ML models to address the sentiment classification task. Many of the models evaluated in this work are based on the references cited in this chapter.

In the third chapter, "Datasets", we describe the process of collecting and pre-processing the annotated corpus considered in this work. Besides representing a fundamental step of this study, its simple compilation already represents a small advance for the NLP community.

In the fourth chapter, entitled "Text Classification Pipeline", we explain all steps from the text classification pipeline exploited in this work, detail all experiments, and clarify how the models were trained and evaluated.

In the fifth chapter, named "Results and Discussion", we conduct a systematical evaluation of the predictive performance of different models and discuss the results in terms of the potentials and drawbacks of each method.

Finally, in the "Conclusion" chapter, we summarize the main findings of this work and point out future research directions.

Chapter 2

Bibliographic Review

This chapter provides a brief overview over the references that guided this work. The first section focuses on proposals for addressing text classification and sentiment analysis tasks. In the second section, we report the main works on automatic text classification with machine learning. Finally, the third section concentrates on the Transformer architecture and its main developments, which have revolutionized the NLP field.

2.1 Text Classification and Sentiment Analysis

Text Classification is the most essential NLP task. It assigns labels to textual units such as sentences, paragraphs, or documents. We can stratify text classification into different subtasks (sentiment analysis, topic analysis, question answering, natural language inference) better described below [5] [6] [7].

- **Sentiment Analysis:** this research field aims to develop methods capable of automatically analyzing people's opinions expressed in texts [1]. A common branch of this area is to infer the people's opinions expressed in textual data by defining an equivalent binary or multi-class classification problem. Binary sentiment analysis classifies texts into positive and negative classes, while multi-class sentiment analysis into classes of grading sentiment intensity;

- **Topic Analysis:** it aims to identify the central theme or topics in a text, such as whether the product review is about "customer support" or "ease of use";

- **Question Answering**. There are two types of QA tasks: extractive and generative. The first is essentially a text classification task: the system must identify the most adequate answer for a given question from a set of possible

answers. Generative QA corresponds to a text generation task, as it requires the generation of the whole responses;

- **Natural Language Inference:** it aims at predicting if the meaning of one text can be inferred from another. An NLI system assigns to a pair of text units labels, such as implication, contradiction, and neutral.

We can perform text categorization through manual annotation or automatic labeling. The second way is becoming increasingly crucial as the volume of textual data grows. According to Medhat et al. [12], there are two types of automatic sentiment classification techniques: machine learning and lexicon-based approaches (Figure 2.1).

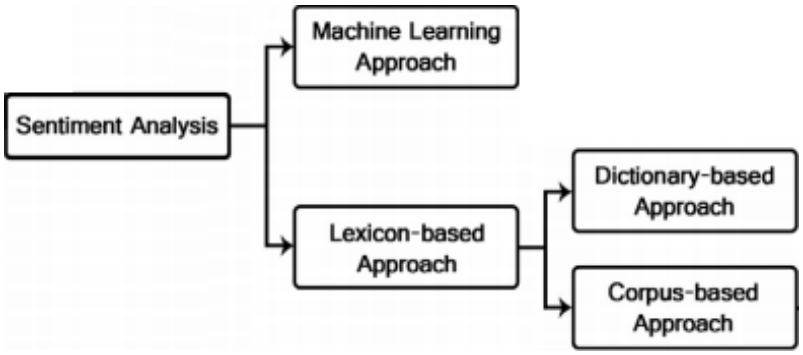

Figure 2.1: Sentiment Analysis techniques. Adapted from Medhat et al. [12]

The Lexicon-based approach relies on a sentiment lexicon, a collection of pre-compiled sentiment terms, and can be further split into dictionary and corpus-based approaches. The dictionary-based approach manually collects a set of opinion words with known orientations. Its primary disadvantage is that the opinion words do not vary with the context since they are predefined in the dictionary. The Corpus-based approach solves this problem by finding opinion words with context-specific orientations [12].

Machine learning-based text categorization automatically learns how to produce classifications solely based on past observations rather than relying on manually crafted rules. ML systems infer intrinsic relationships between texts and labels using pre-labeled cases as training data [5].

This thesis focuses on ML-based embedding techniques, aiming to obtain meaningful features for the binary sentiment classification task. The following sections briefly describe the prior machine learning and deep learning works related to NLP, from the most basic and traditional to the current state-of-the-art, which have inspired the choices for the techniques adopted in this dissertation.

2.2 Automatic Text Classification with ML

2.2.1 Bag-of-Words

Bag-of-Words (BoW) is the most basic feature-based approach for automatic text classification, producing a simple vector representation for each word or the entire document that ignores the surrounding context, word order, and semantic relations [5]. Usually, it exploits statistics extracted from the corpus, such as the word frequency or the term frequency-inverse document frequency (TF-IDF), or may consider aggregated pre-trained word vectors as features, as discussed further in the following.

Commonly, the BoW embedding is followed by a classical ML classifier, such as *Logistic Regression* (LR), *Support Vector Machines*, *Gradient Boosting Decision trees*, or *Random Forests*, when applied to sentiment classification tasks. Since most of these models are fast and straightforward to implement and train, they may constitute a handy baseline. Regardless of their simplicity, such methods can achieve high performance for simple texts, comparable to or even better than more complex alternatives. A drawback of BoW models is not efficiently accomplishing new tasks, in contrast with modern transfer learning approaches (Transformer-based Language Models). Also, they do not benefit from the large amounts of training data available nowadays due to their limited model capacity. [5].

Since the shift in the ML paradigm motivated by the AlexNet [13], the state-of-the-art models in NLP and Computer Vision mostly include DL architectures [5]. Despite often involving a more challenging and slower training phase, such architectures can easily learn complex patterns and scale to large datasets. Compared to the classical models, the DL counterparts do not require a hand-crafted feature extraction since features are automatically learned during model training [6].

2.2.2 TF-IDF

The TF-IDF is a document embedding approach based on the frequency of occurrence of each word into a collection of N documents (corpus) [14]. The dimensionality of a document vector produced by the TF-IDF is equal to the vocabulary size, wherein the i-component of the vector related to the jth document vector is given by

$$w_{ij} = tf_{ij}\ idf_i, \tag{2.1}$$

where tf_{ij} is the frequency of occurrence of the ith word into the jth document, while idf_i, as defined by Eq. (2.2), denotes the inverse of the ratio between the number of documents containing the ith word, denoted as df_i, and the total number of documents N.

$$idf_i = log(\frac{N}{df_i}). \tag{2.2}$$

This embedding often results in very sparse vectors, favouring frequent words of the same document and penalizing those present in many documents [15]. The vocabulary may also include sequences of words with an arbitrary length n, referred to as n-grams, in addition to single document words. To constrain the vocabulary size, one may consider just taking the most frequent n-grams or exploiting Chi-squared and ANOVA F-value tests [16] to select the best subset of words from the corpus to integrate the dictionary. Singular Value Decomposition (SVD) may also be explored for deriving low-dimensional representations over TF-IDF embeddings. This combined approach is often known as Latent Semantic Analysis (LSA) [17]. As Zhang et al. [18] shows, the models using TF-IDF can achieve better predictive results than more complex approaches, such as CNNs and RNNs in some applications.

2.2.3 Word Vectors

Instead of compiling textual statistics from the corpus to obtain meaningful word vectors, it is possible to train a language model that learns from the corpora the probability of occurence of a word given the words belonging to its context. In this case, the model must represent the words as dense vectors that maximize the correct prediction of their context words. Roughly, this approach turns the language generation process into a classification task and allows the prediction error calculated at each iteration to be used for updating model parameters using a gradient-based rule that penalizes proportionally to the compiled error. This is the basic concept of backpropagation, widely used to train neural models [19]. These word embeddings are the most classic pre-trained resources used in different NLP tasks by various architectures.

This idea was initially developed by Collobert et al. [20] and improved by Mikolov et al. [21], author of the Word2vec model, that proposed two different algorithms: CBOW and Skip-Gram, depicted in Figure 2.2. CBOW aims to predict a center word from the surrounding context words based on their embeddings. Skip-gram does the opposite, predicting the probability of context words based on a center word.

In addition, this work also proposed two strategies to reduce the computational

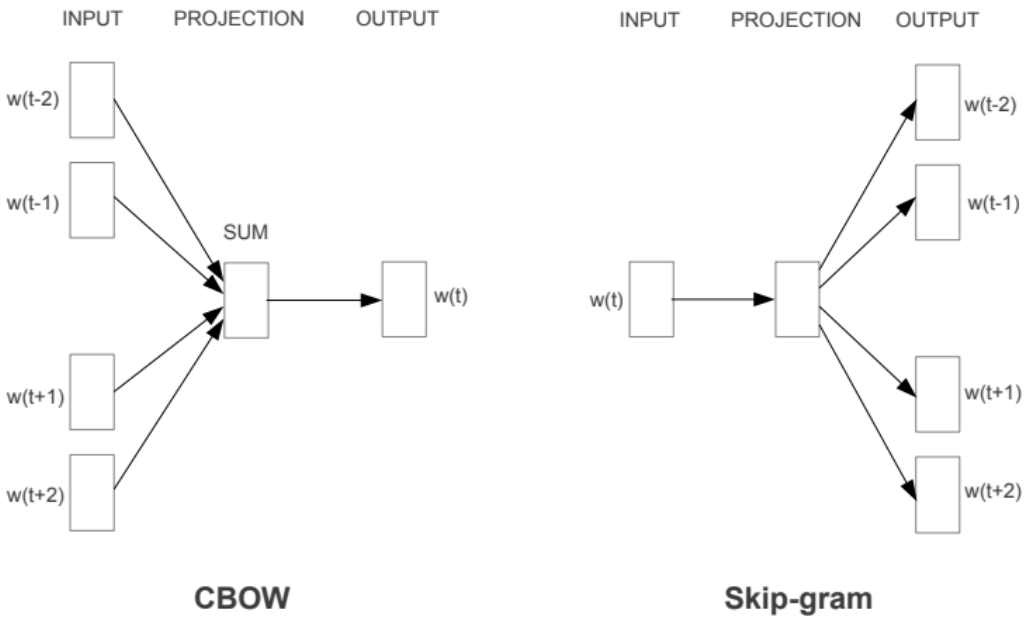

Figure 2.2: CBOW and Skip-Gram architectures [21].

burden from training these models: negative sampling and hierarchical softmax. In the naive implementation of both Word2Vec variants, at each model iteration, it is necessary to compute the softmax probabilities for all words integrating the vocabulary, which can be huge. Negative sampling replaces this costly procedure by randomly selecting words as counter-examples and thus using them to optimize the objective function. When training the network, the label is a one-hot vector, where the vector position corresponding to the target word is 1, and all the other countless words are 0. The negative sampling randomly selects just a tiny number of negative words to update the weights. Conversely, hierarchical softmax uses a binary tree in the vocabulary evaluation process, reducing the time complexity to $O(log(|V|)$ [19].

GloVe [22] is another widely used word vector algorithm. It combines the advantages of the two prominent families of models dedicated to learning embeddings in the literature: the global matrix factorization approaches, such as LSA (Latent Semantic Analysis), and the local context window methods, like Word2vec. The first efficiently accounts for general text statistical information, but it does poorly on the word analogy task. The second one does better on the analogy task, but it makes inefficient use of the statistics of the corpus. The authors proposed a specific weighted least squares model that trains over global word-word co-occurrence counts, which considers the context windows, and thus efficiently uses corpus statistics [23].

Finally, FastText [24] is a word embeddings model that aims to balance the predictive performance with the number of model parameters. The model is based on the skip-gram algorithm, where each word is represented as a bag of character n-grams. A vector representation is associated with each character n-gram and words

are represented by the sum of these atomic representations.

2.2.4 Document Embeddings composed by Word Vectors

The most basic approach to generate document embeddings exploring pre-trained word vectors is averaging the embeddings of all the words present in the document, thus giving an equal importance to all of them [25]. Nonetheless, some words do not carry high semantic value, being meaningless for text classification. In this sense, Singh et al. [26] analyzed different graded schemes to account for the relative relevance of the words in a given corpus. They reported promising results for a weighted average of word vectors with IDF factors. Arora et al. [27] go further in this idea with SIF (smooth inverse frequency) embeddings. They computed a weighted average of the word vectors in the document using a weighting scheme similar to TF-IDF. Subsequently, they removed the projections of the resulting vectors on their first singular vector, which can be viewed as a form of denoising.

There are other ways to produce document embeddings through word vectors without requiring further neural training. Gupta et al. [28] developed the Graded Weighted Bag of Words Vector, which first represents the document in a lower-dimensional space related to semantic clusters inferred by a k-means clustering procedure. Then it concatenates those representations with inverse cluster frequency weighting factors. Mekala et al. [29] proposed the Sparse Composite Document Vectors, a feature vector formation technique that uses soft-clustering to build a distributional representation that captures multiple semantic contexts. This work exploits Gaussian Mixture Models for soft-clustering the words in topics, considering the probabilities of a given word of being part of each cluster for composing the vectorial document representation.. Since these document representations may show many values close to zero, the components below a certain threshold are set to zero. Gupta et al. [30] proposed something analogous to the latter model with P-SIF, a variant of SIF [27], which adopts the concept of partition word average to represent the different topics in the sentences accurately. One of the critical differences to SCDV is that, instead of adopting GMM to represent the cluster/partitions, it uses the concept of sparse coding.

Instead of just aggregating word vectors, it is also possible to learn document embeddings similarly to how the word vectors are learned. Classical approaches are the Distributed Memory Model Paragraph Vectors and the Distributed BoWs paragraph vectors, proposed by Le and Mitolov [31]. The first one is based on predicting the next context word using vectors of words and paragraphs. The second one, a more straightforward and lighter version since it ignores the context words, learns how to predict words randomly sampled from the paragraph in the output. A

text window and a random word are sampled at each iteration to form a classification task. This idea is similar to the Skip-Gram approach from the Word2Vec algorithm. Following a similar idea, the Neural Tensor Skip-Gram model [32] can learn multiple embeddings per word and aggregate them to form the document embedding through a tensor layer.

2.2.5 Recurrent Neural Networks

Typically, RNNs can better exploit more complex data patterns than feedforward neural networks or bag-of-words approaches, accessing more effectively their mutual dependencies, thus better capturing the sentence context [5]. An RNN aims to learn a sequence representation by keeping a hidden state vector that stores the current state of the sequence and is updated considering both the current input vector and the previously hidden state vector [2].

Figure 2.3 shows the basic 1-layer Elman RNN architecture [33], also called Vanilla RNN, where each vertical rectangular box represents the hidden state vector at a time step t. For an easy understanding, it is common to unroll the "RNN" neuron feedforward and backward signal propagation, interpreting this structure as a multilayer architecture. In this case, each layer may have several neurons (dimensions), which conduct linear matrix operations on their inputs followed by a non-linear operation σ. There are also two inputs to the hidden layer at each time step: the output of the previous layer $\boldsymbol{h_{t-1}}$, and the input at that time-step x_t, that, in the case of NLP applications, may correspond to word vectors. As shown in Eq. (2.3), these two inputs are multiplied by a weight matrix $\boldsymbol{W^{(hh)}}$ and a weight matrix $\boldsymbol{W^{(hx)}}$, respectively, to produce the new hidden layer value $\boldsymbol{h_t}$ [34].

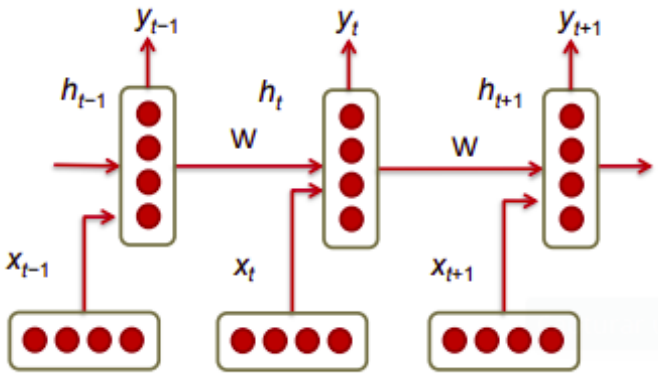

Figure 2.3: A Recurrent Neural Network. Three time-steps are shown [34].

$$\boldsymbol{h_t} = \sigma(\boldsymbol{W^{(hh)}}\boldsymbol{h_{t-1}} + \boldsymbol{W^{(hx)}}\boldsymbol{x_t}) \tag{2.3}$$

Although the Vanilla RNNs can theoretically detect long-term dependencies, two issues often preclude them from doing so: the inability to retain long-term information and the gradient stability. Concerning the first one, as the Vanilla RNNs update the hidden state vector, there is no direct control over which values are retained or discarded in each hidden state update. The unstable gradients are related to the episodes wherein the gradients spiral out of control to zero or infinity values, named vanishing and exploding gradients, respectively. As long as we add time steps to the RNN, the network eventually becomes untrainable due to the multiplicative nature of the backpropagation, which successively multiplies an incresing number of smaller or larger gradients with the addition of more layers [35] [2].

The Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) layers came up to overcome this problem. Both models involve the concept of gating. To intuitively understand the gating process, suppose we add two quantities: a and b, but we want to control how much b gets into this sum. In this case, it is possible to rewrite the sum $a + b$ as $a + \lambda b$, where λ is a value between 0 and 1. In the latter, the variable λ acts as a gate that controls the amount of b getting into the sum. Naturally, this is more complicated in LSTM and GRU models since the gating functions are parameterized and learnable [2]. Here, we are going to focus on the LSTM model.

The most popular variant of the RNNs is the Long Short-Term Memory, firstly proposed by Hochreiter and Schmidhuber [36], aiming to mitigate the gradient vanishing and exploding problems related to the RNNs [5]. The Figure 2.4 illustrates this architecture in more details.

The **new memory cell**, described by Eq. (2.7), uses the word vector $\boldsymbol{x_t}$ and the past hidden state $\boldsymbol{h_{t-1}}$ to generate a new memory $\widetilde{\boldsymbol{c}}_{\boldsymbol{t}}$ thats accounts for aspects of the input vector. The **input gate** output, given by Eq. (2.4), considers the current input vector and the past hidden state to determine how much of the new memory cell computation must be considered when finally updating the memory cell content. In turn, the **forget gate** output, computed according to Eq. (2.5), is similar to the input gate. However, it evaluates how much of the current memory cell content must be kept when updating the memory cell. The **final memory cell** update is given by Eq. (2.8), mixing information from the past and new memory content, modulated by the forget and input gates, respectively. The **output/exposure gate** output, computed by Eq. (2.6), addresses how the final memory cell content output must compose the new hidden state. Finally, the LSTM cell output is given by Eq. (2.9). Note that in the former equations $\boldsymbol{W}$ and $\boldsymbol{U}$ represent the weight matrices related to each cell and gate [34].

$$\boldsymbol{i_t} = \sigma(\boldsymbol{W^{(i)}x_t} + \boldsymbol{U^{(i)}h_{t-1}}) \tag{2.4}$$

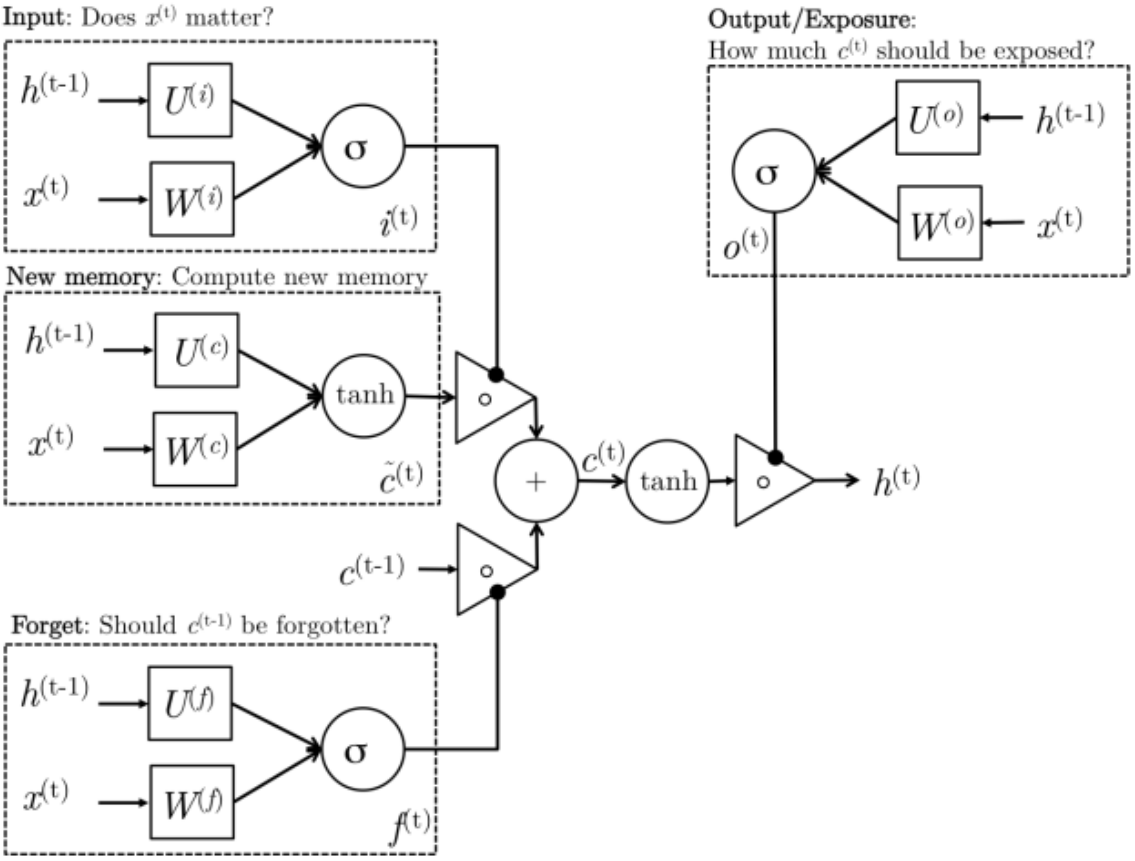

Figure 2.4: LSTM cells and gates in more details [34].

$$\boldsymbol{f_t} = \sigma(\boldsymbol{W^{(f)}x_t} + \boldsymbol{U^{(f)}h_{t-1}}) \tag{2.5}$$

$$\boldsymbol{o_t} = \sigma(\boldsymbol{W^{(o)}x_t} + \boldsymbol{U^{(o)}h_{t-1}}) \tag{2.6}$$

$$\boldsymbol{\widetilde{c}_t} = tanh(\boldsymbol{W^{(c)}x_t} + \boldsymbol{U^{(c)}h_{t-1}}) \tag{2.7}$$

$$\boldsymbol{c_t} = \boldsymbol{f_t} \cdot \boldsymbol{c_{t-1}} + \boldsymbol{i_t} \cdot \boldsymbol{\widetilde{c}_t} \tag{2.8}$$

$$\boldsymbol{h_t} = \boldsymbol{o_t} \cdot \boldsymbol{tanh(c_t)} \tag{2.9}$$

Later, many variants and applications of this model were studied in the context of text classification. Liu et al. [37] innovated by training an LSTM-based architecture in a multi-task learning framework instead of just training it on a single-task supervised objective. This model was trained jointly on four different text classification tasks. Nowak et al. [38] and Wang et al. [39] applied LSTM to short text sentiment classification using pre-trained word embeddings.

2.2.6 Convolutional Neural Networks

While RNNs are popular in NLP applications motivated by their requisites of sequential processing, CNNs work better when detecting local position-invariant patterns. These patterns could represent key phrases that express a particular sentiment like "I like" or a topic like "endangered species". This fact makes CNNs suitable for TC [5] [40].

A typical CNN architecture integrates convolutional layers, pooling operations, and fully connected layers [2]. Regarding the convolution operation, Figure 2.5 depicts an example of a 1D convolution applied to a sentence. In the example, each word is represented by a 4-dimensional word embedding submitted to a 3-dimensional filter/kernel. The filter is stored as a matrix that must be slide over the input matrix. Typically, convolutions have up to three dimensions. For textual applications, the most common CNN application involves one-dimensional convolution since this operation runs only on the direction of text reading. In contrast, it is more common in Computer Vision applications to operate in two dimensions, i.e., in the vertical and horizontal image directions.

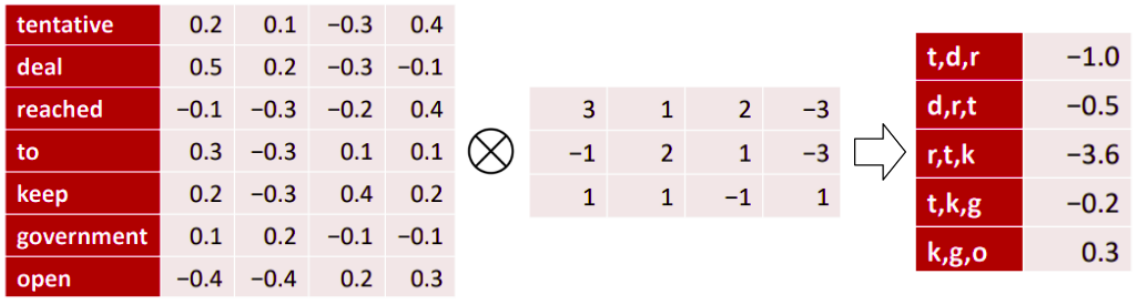

Figure 2.5: 1D convolution for texts. Adapted from [41].

This way, the 1D convolutional operation resembles extracting textual features based on moving n-gram windows. For example, a 2-dimensional kernel extracts features from bi-grams, while a 3-dimensional kernel processes tri-grams, and so on. For this reason, when one considers filters of different dimensions in the CNN architecture, it allows extracting information of different context windows from the text, automatically learning weights to properly address each one of these cases.

There are several relevant and classical studies applying CNNs to text classification. Kim [42] reports a series of experiments with single-layer CNNs using pre-trained word vectors. He analyzed the use of word vectors with fine-tuning and without fine-tuning, and both approaches concatenated, which brought significant improvements. Zhang and Wallace [43] published a follow-up work conducting a sensitivity analysis of one-layer CNNs, analyzing the effects of architectural issues on model performance. The authors considered the influence of the input word

vectors (Word2vec and GloVe), region size, number of feature maps for each filter region size, activation function, pooling strategy, and regularization. The base CNN architecture adopted in this study is illustrated in Figure 2.6. The model adopted three filter sizes (2, 3, 4), with two filters each. These filters are convolved with the sentence matrix and produce feature maps that pass through 1D-max pooling operations, generating feature vectors based on all six maps. The six resulting features are then concatenated and fed to the softmax layer that finally classifies the sentence.

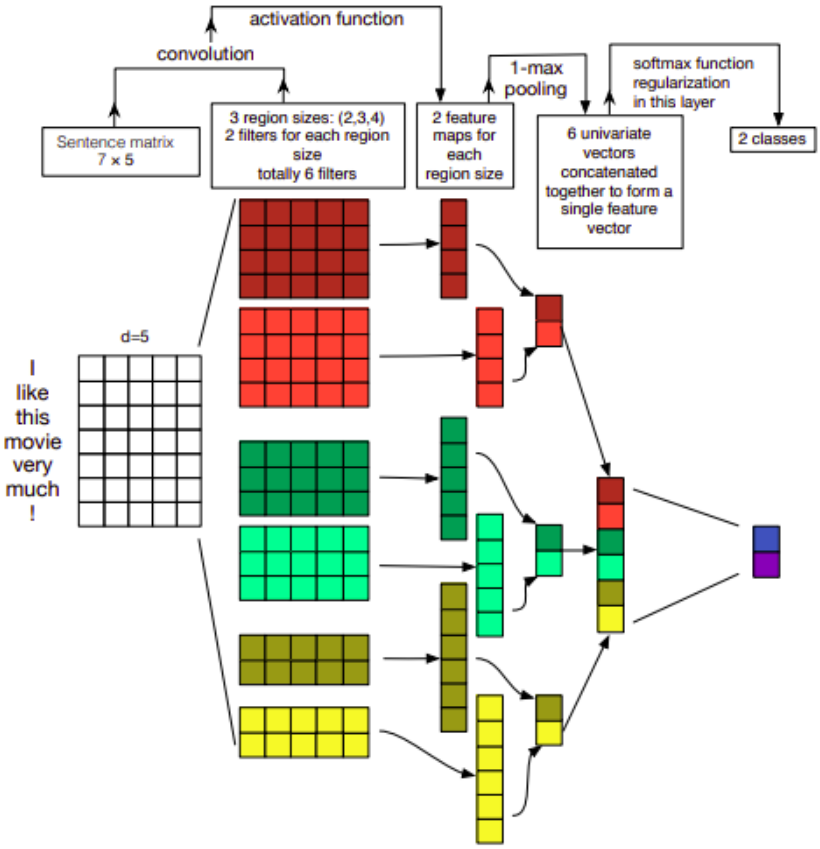

Figure 2.6: The base CNN architecture adopted in [43].

While the more classic approaches treat the corpus at the word level, Zhang et al. [18] proposed a character-level convolutional neural network for text classification, achieving competitive results at the time without the use of any pre-trained resources. Conneau et al. [44] dramatically increased the scale of CNNs models for text classification by proposing the Very Deep Convolutional Networks for Text Classification (VDCNN), inspired in the large CNN-based architectures explored in Computer Vision. The VDCNN operates on the character level, and the authors evaluated different configurations, especially regarding the number of convolutional layers (raised up to 49), showing significant performance improvements up to 29 layers.

2.2.7 Recurrent and Convolutional Neural Networks

As convolutional and recurrent networks can learn different patterns and have different computational complexities, combining them can be a good alternative to benefit from both strengths and mitigate the negative points of each approach.

Zhou et al. [45] introduced the C-LSTM, which uses a CNN to extract a sequence of higher-level representations that is fed into an LSTM model to obtain the sentence embeddings. C-LSTM can capture both the local features of phrases and the global (and temporal) sentence semantics. Lee and Dernoncourt [46] also combined these two models but considered a short text classification task involved in a dialog prediction task. The model consists of two steps: the first generates a vector representation for each short text using either the RNN or CNN architecture. The second one is responsible for classifying the current short text based on the vectorial representation of the current and of a few preceding short texts.

Bradbury et al. [47] proposed the quasi-recurrent neural networks (QRNNs), an approach to neural sequence modeling that alternates convolutional layers, which are applied in parallel across the timesteps, and a minimalist recurrent pooling function applied in parallel across the channels. The model addresses the drawbacks of both CNNs and RNNs. Like CNNs, QRNNs allow a parallel computation across the timestep and minibatch dimensions. Similarly to RNNs, QRNNs' outputs depend on the overall order of the words in the sequence. The authors reported a better predictive and computational performance than traditional LSTM models.

2.2.8 Graph Neural Networks

Another interesting approach that has been gaining more attention in NLP applications is the Graph Neural Networks (GNN). Graphs are commonly used to represent interactions between problem instances, and the natural language contains internal graph structures, such as syntactic and semantic parse trees, responsible for establishing syntactic and semantic relations among the words that integrate a sentence [48].

Firstly introduced by Gori et al. [49] and refined by Scarselli et al. [50], GNNs may be considered as extensions of the more traditional Deep Learning models, like CNNs, RNNs, and autoencoders to handle graph data. The graph convolution is typically performed by taking the weighted average of the neighborhood information of each node (the message passing paradigm [48]). The Graph Convolutional Networks (GCNs) are the most popular GNNs due to their effectiveness in different applications [48] [5].

Text classification is a typical application of GCNs in NLP. Peng et al. [51] proposed a graph-CNN-based deep learning model that first converts a text into a

graph of words and then uses graph operations to convolve the graph. The representation of texts in graphs of words has the advantage of capturing non-consecutive and long-distance semantics. In a same fashion, Yao et al. [52] proposed a GCN for text classification that first builds a single text graph for a corpus based on word co-occurrence and document relations, then trains a Text Graph Convolutional Network.

2.2.9 Attention

Neural Machine Translation (NMT) models used to explore the sequence-to-sequence architecture following an encoder-decoder scheme, according to which the decoder has its hidden state initialized with a fixed-length context vector (represented by the hidden state vector of the encoder) [53]. This practice represents a bottleneck in this architecture, since it limits the amount of information available to the decoder and, consequently, the network performance.

In this sense, Bahdanau et al. [54] proposed to allow the model automatically search for parts of the source sentence that are most relevant in the prediction of some target word. Hence, this resource reduces the need for the encoder structure to squash all the source sentence's information into a fixed-length vector.

This feature is called Attention. It is generally defined as a technique to express a given query vector in terms of a set of value vectors (typically by a weighted sum) [53] [55].

In detail, Figure 2.7 shows the Sequence-to-Sequence with Attention architecture proposed by Bahdanau et al. [54]. In the decoding phase, first, the attention scores are calculated, following the Eq. (2.10), where $\boldsymbol{s_t}$ is the decoder hidden state at the step t, and $\boldsymbol{h_i}$ are the encoder hidden states ($1 \leq i \leq N$). After that, these scores are normalized to result in the attention distribution, described by Eq. (2.11), and used to calculate the context vector or attention output, according to Eq. (2.12), considering a weighted sum of the encoder hidden states. Finally, the attention output is concatenated with the decoder hidden state [53, 56].

$$\boldsymbol{e^t} = [\boldsymbol{s_t^T h_1}, \cdots, \boldsymbol{s_t^T h_i}, \cdots, \boldsymbol{s_t^T h_N}] \tag{2.10}$$

$$\boldsymbol{\alpha^t} = softmax(\boldsymbol{e^t}) \tag{2.11}$$

$$\boldsymbol{a_t} = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h_i} \ . \tag{2.12}$$

Despite initially developed in the Computer Vision area and originally applied

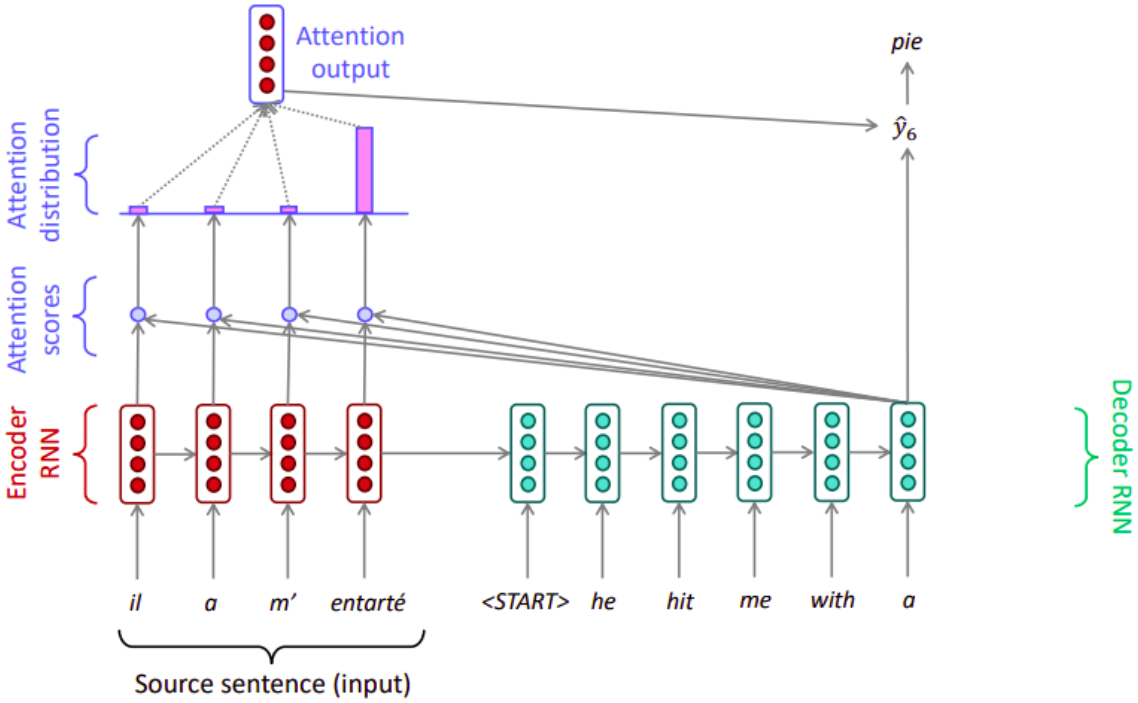

Figure 2.7: Sequence-to-sequence with attention (we showed only the last step of the decoding phase). Extracted from [56].

for NMT, Attention is a Deep Learning technique that can be applied to other NLP tasks and even tabular data [55, 57]. Thus, several deep learning models that explore Attention for text classification have been proposed recently [5].

Wang et al. [58] proposed an attention-based LSTM architecture, shown in Figure 2.8, focusing on the aspect-level sentiment classification task, a fine-grained task in sentiment analysis that focuses on identifyng the polarity of different text segments. The attention mechanism enables the model to concentrates on different parts of the sentence when different polarities are concerned.

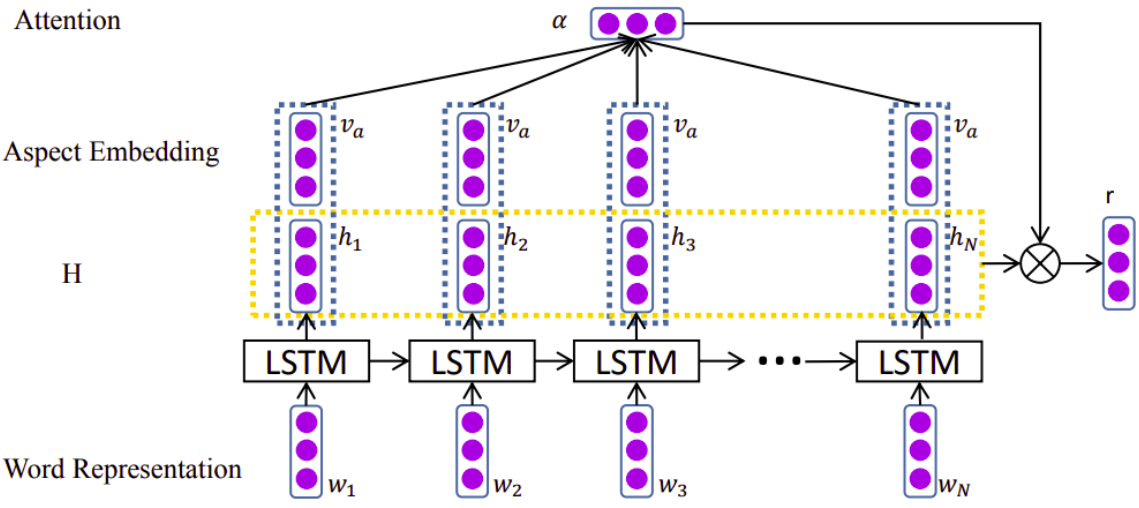

Figure 2.8: Attention-based LSTM architecture [58]

Liu and Guo [59] also adopted an LSTM-based architecture with Attention called Attention-based Bidirectional Long Short-Term Memory with Convolution layer (AC-BiLSTM), a mixture of RNN, CNN, and Attention. The convolutional layer extracts higher-level phrase representations from the word vectors, and the BiLSTM is used to access the preceding and succeeding context representations. The Attention mechanism provides a different focus to the information outputted from the hidden layers of the BiLSTM model.

Another useful attention-based model for document classification was the hierarchical attention network presented by Yang et al. [60]. This model contains a hierarchical structure that mimics the structure observed in the documents and has two levels of attention mechanisms applied at word and sentence levels. The Attention layer enables the model to respond differently, according to the relevance of each text segment in the vectorial document representation.

2.3 Transformers

2.3.1 Vanilla Transformers

Recurrent neural networks used to be the basic building block of successful NLP solutions when modelling textual dependencies and semantic contexts. However, the sequential processing inherent to the RNNs is computationally inefficient, as it is only possible to compute a specific hidden state after the previous one has been determined, which makes any algorithm parallelization difficult [61]. As mentioned in the former section, there have been some efforts trying to overcome this issue, such as QRRNs, but with a moderate success.

Vaswani et al. [62] revolutionized the NLP field by proposing the Transformers Deep Learning architecture, which only relies on the attention mechanism and feed-forward networks, thus not requiring any recurrence or convolution. Until the paper's publication, this mechanism has been extensively used in sequence modeling, but mostly in conjunction with RNNs [61].

The Transformers replace the recurrence with a multi-head self-attention mechanism, allowing an extensive model parallelization and, consequently, the model training to explore a massive amount of data [62]. As a result, the complexity of NLP models could vastly scale, and big datasets can be addressed. The possibilities opened by the massive parallel processing available nowadays allowed the Transformers to be applied to the development of large pre-trained models, thus enabling a broad adoption of the transfer learning strategy in many NLP applications, as observed years before in the Computer Vision area [63].

Figure 2.9 presents the general structure of the Vanilla Transformer proposed

by Vaswani et al. [62], following an encoder-decoder structure. The encoder is formed by stacked blocks, each composed of self-attention modules and feed-forward layers. The decoder has a similar structure to encoder but with an encoder-decoder-attention layer between these two. For simplification, the image reduced the number of blocks to two, despite the smaller configuration proposed in the original paper explore six for the encoder and decoder each.

The positional encoding refers to vectors added to each input token embedding to signalize the position of each one in the sequence. Instead of simply adopting integer index values, which may create problems for variable length sequences, the Transformers use a clever positional encoding scheme based on trigonometric functions [64].

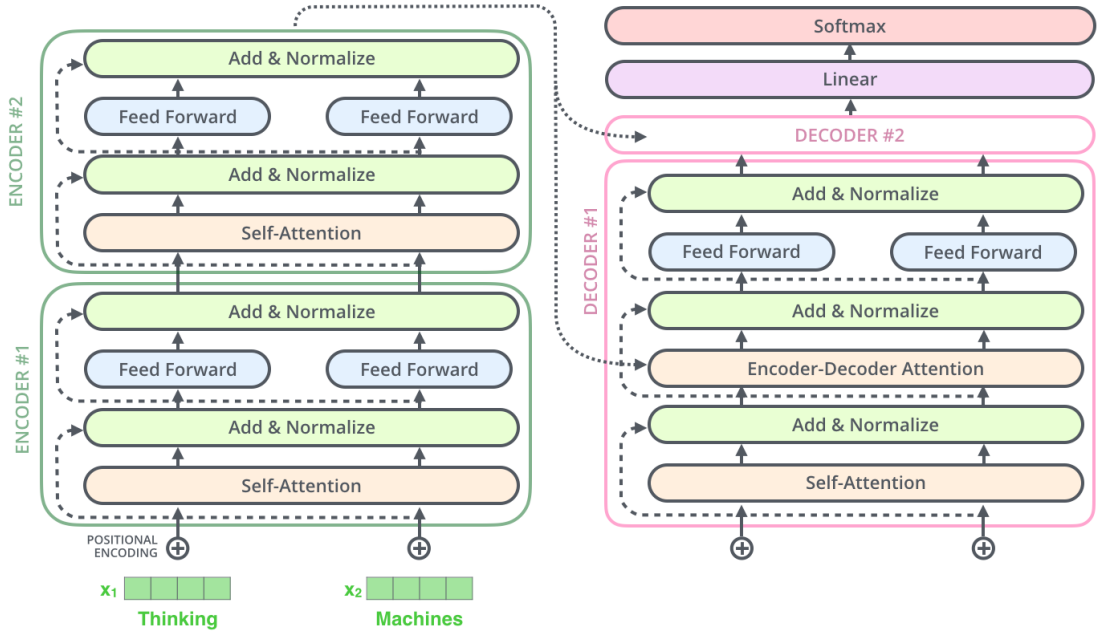

Figure 2.9: Transformer architecture overview. Extracted from [65].

The self-attention mechanism is described by the set of Eqs. (2.13-2.16). The matrices $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ correspond to the query, key, and value matrices, respectively. They are obtained by multiplying the matrix of word representations $\boldsymbol{X}$ by their respective weight matrices $\boldsymbol{W}$. The softmax part explored to compute $\boldsymbol{Z}$ describes how much each line of $\boldsymbol{V}$ must be accounted for when expressing the corresponding line of $\boldsymbol{X}$. The parameter d_k is the dimension of the key vectors (equal to 64 in the case of the Vanilla Transformer), and it was introduced to improve gradient stability during models' training. Finally, the self-attention output is determined by the sum of the value vectors weighted by their respective softmax scores.

$$\boldsymbol{Q} = \boldsymbol{X}\ \boldsymbol{W}^{\boldsymbol{Q}} \tag{2.13}$$

$$\boldsymbol{K} = \boldsymbol{X}\ \boldsymbol{W^K} \tag{2.14}$$

$$\boldsymbol{V} = \boldsymbol{X}\ \boldsymbol{W^V} \tag{2.15}$$

$$\boldsymbol{Z} = softmax(\frac{\boldsymbol{Q}\ \boldsymbol{K^T}}{\sqrt{d_k}})\ \boldsymbol{V} \tag{2.16}$$

The Transformer architecture introduced the concept of multi-head self-attention, which exploits eight concatenated self-attention heads. After this concatenation, the output is multiplied by a weight matrix $\boldsymbol{W^o}$ to reduce the dimensionality and to define the final matrix $\boldsymbol{Z}$ that captures information from all the attention heads. This multi-head mechanism enlarges the model's ability to focus on different embedding positions and gives the attention layer multiple representation subspaces [65].

Transfer learning became a prominent approach in NLP a few years before the proposition of the Transformers architecture. Initial transfer learning solutions considered smaller models with pre-trained word vectors and, later on, more complex models, such as ELMo [66] and ULMFiT [66]. These two models rely on LSTMs to predict the term around a particular central word/token. However, it was only with the Transformers that NLP models could considerably scale up. It is noteworthy that, despite having been initially conceived for text modeling, nowadays, the Transformers also have achieved the state-of-the-art in many Computer Vision tasks [67].

2.3.2 Transformer-based Large Language Models

Large pre-trained Transformer-based language models represent the state-of-the-art in many NLP tasks. Several variations have gained notoriety in recent years, such as GPT [68], GPT2 [69], GPT3 [70], BERT [71], BART [72], RoBERTa [73], T5 [74], among others, with a considerable increase in the number of parameters over the years, as shown in Figure 2.10. Until now, Megatron-Turing NLG [75] is the world's largest language model, with 530 billion parameters.

Devlin et al. [71] proposed the BERT (Bidirectional Encoder Representations from Transformers), one of the most popular Transformer-based architectures and object of analysis in this work. BERT features an encoder structure similar to Vanilla Transformers without the decoder, with the number of layers and hidden size equal to 12 layers and 768, in the case of BERT Base, or 24 and 1024, for BERT Large. It has innovated by using bidirectional self-attention, which allowed this model to learn the context of tokens situated on the left and on the right of

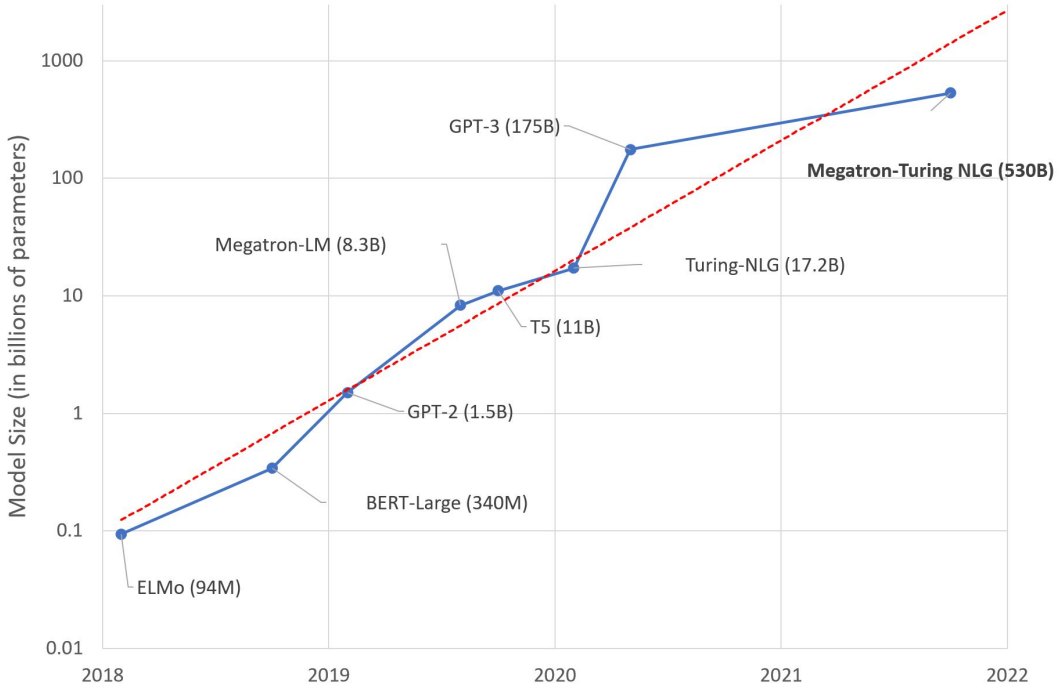

Figure 2.10: Trend of sizes of state-of-the-art NLP models over time [75]

each token, unlike GPT, where the unidirectional self-attention only allows a token to make use of the context from tokens situated before it.

BERT pre-trained parameters may be obtained by exploring two unsupervised approaches: masked language model (MLM) and next sentence prediction (NSP). The first is a strategy to circunvent the unidirectionality of the next token prediction task, allowing the bidirectional context processing characteristic of BERT.

In MLM, tokens can be masked (i.e., replaced by the special token [MASK]) or replaced with a random token, and the model must predict which token is correct for that position. In turn, the NSP aims to predict if a particular sentence succeeds the previous one, thus allowing the model to learn the semantic relationships between the sentences integrating a document. For this objective, the authors introduced the special token [CLS] at the beginning of each sentence. This token carries information from the entire sentence, a reason behind its relevance for the text classification task, as we will see throughout this work.

Transfer-learning with BERT may consider two design options: pre-trained and fine-tuning. The pre-trained approach assumes BERT as a fixed-model for producing "unsupervised" features; therefore, only the model stacked over it is trained for a target application. Conversely, the fine-tuning strategy focus on updating BERT weights using labelled data for a specific task. Surprisingly, the authors presented competitive results for the Named Entity Recognition (NER) task compared to the state-of-the-art by just exploiting the pre-trained approach. Figure 2.11 presents an

example of feature extraction using BERT targeting a text classification task.

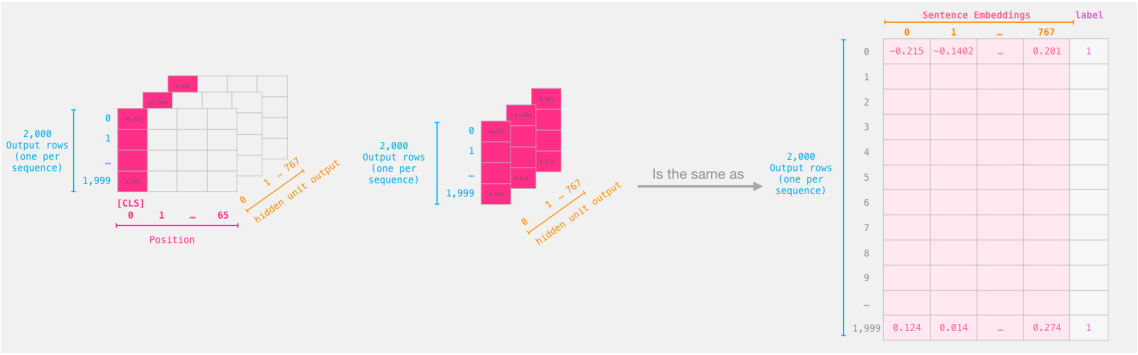

Figure 2.11: BERT feature extraction scheme. In the example, BERT Base (embedding size equal to 768) was chosen, a corpus with 2000 documents and sentences padded to 66 tokens. Each document is represented by the embedding corresponding to the CLS token, that is, the first token of the document. Only the last layer of the BERT model is considered. Adapted from [65].

Some recently developed language models follow a multilingual approach to address languages other than English, which has been the focus of a considerable research effort nowadays. The BERT authors have also open-sourced a variant with a multilingual purpose (m-BERT), trained in more than 100 languages, including Portuguese. Another notable multilingual model is the XLM-RoBERTa [76], a multilingual version of the RoBERTa, trained on contents of 100 languages. One of the main differences between BERT and RoBERTa is the masking phase. BERT is based on static masking, i.e., the masking process is performed during data preprocessing, resulting in a single static mask. RoBERTa adopts dynamic masking, generating the masking pattern every time a sequence is fed into the model. This difference helps to scale the training and input data size [73].

In addition, the Text-to-Text Transfer Transformer (T5) [74] is a framework that renders different language tasks in a text-to-text format. As shown in Figure 2.12, T5 brings together different NLP tasks, which involve texts as input, such as translation, question answering, classification, regression, and summarization, and must result in the generation of some target text. This makes the same model usable for different applications. T5 has three learning objectives: language modeling (predicting the next word), masked language modelling (masking tokens from the input text randomly and predicting the original text, as in BERT), and deshuffling (randomly reordering the input tokens and predicting which order is the correct). The authors provided five versions of different scales: small, base, large (similar to BERT), as well as models with 3 and 11 billion parameters.

Finally, it is essential to emphasize that, despite the notable advances, Bender et al. [77] call the TLMs "stochastic parrots", since they are very good at reproducing patterns in the training data but do not necessarily produce coherent texts and can

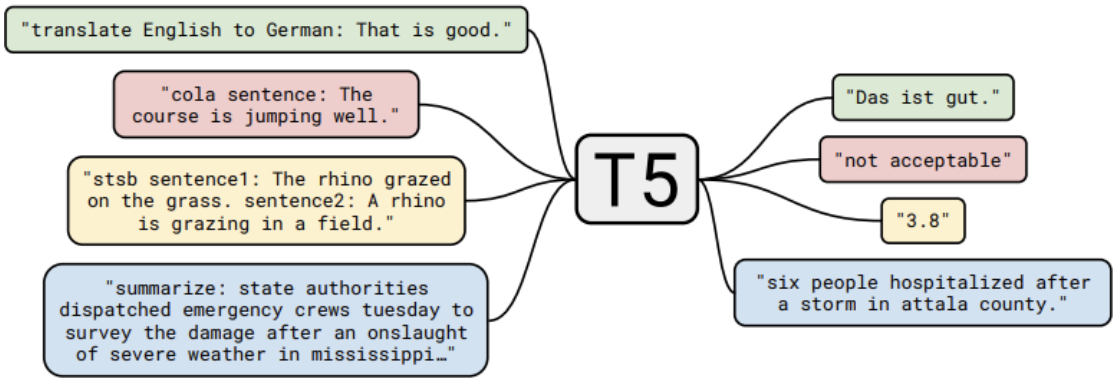

Figure 2.12: Diagram of the T5 framework. Extracted from [74].

propagate unwanted biases.

2.3.3 Brazilian Portuguese TLMs

Souza et al. [78] open-sourced the models BERTimbau Base and Large, trained exclusively on Brazilian Portuguese corpora. As pretraining data, they used the brWaC [79] corpus, the most extensive open Portuguese corpus to date, composed of documents with a high domain diversity and content quality. The authors evaluated the BERTimbau on three NLP tasks (semantic textual similarity, textual entailment recognition, and named entity recognition), reporting advances in the state-of-the-art for all of these tasks.

Despite being a recent model, the BERTimbau has already been applied to other tasks. Lopes et al. [80] fine-tuned the m-BERT and BERTimbau models to an aspect extraction task, whereas Leite at al [81] to toxic sentence classification, outperforming other bag-of-words solutions. Jiang et al. [82] and Neto et al. [83] evaluated fine-tuning the BERTimbau to an irony detection task. Carriço and Quaresma et al. [84] exploited different ways of extracting features from the BERT output layer (CLS token, vector maximum, and vector average), considering a semantic similarity task.

The BertPT and AlbertPT, developed by Feijó and Moreira [85], are other remarkable large language models focused on the Brazilian Portuguese language. The models were trained using corpora from different domains and styles, such as Wikipedia, news articles, movie subtitles, research abstracts, and European Parliament sessions. The models outperformed the baselines on some natural language understanding tasks.

Additionally, Paulo et al. [86] developed the BERTaú, a BERT Base variant trained with data from Itaú (the largest Latin American bank) virtual assistant, and reported better results than the BERTimbau and m-BERT for the NER task. Carmo

et al. [87] open-sourced the PTT5 model, a T5 model trained on the brWaC corpus, the same used to train the BERTimbau, achieving similar results in a semantic similarity task.

Chapter 3

Datasets

Despite the recent advances in NLP and the release of several studies with corpora in Brazilian Portuguese using more advanced techniques, our language still needs datasets focusing on text classification with predefined partitions. Thus, a crucial part of this work involved collecting, preprocessing, and publishing in a public repository different open-source annotated datasets for this task in Brazilian Portuguese[1]. Furthermore, we have explored these datasets to evaluate the different machine learning models presented in this work.

3.1 Dataset Collection

This work considered five user reviews annotated datasets: *Olist* [88], *B2W* [89], *Buscapé* [90], *UTLC-Apps*, and *UTLC-Movies* [91]. Table 3.1 shows two samples of each dataset, one with positive and the other with negative polarity.

3.1.1 Olist

In 2018, Olist, the largest department store in Brazilian marketplaces, launched the "Brazilian E-Commerce Public Dataset by Olist" [88] on Kaggle, a database with approximately 100,000 orders from 2016 to 2018 provisioned by several marketplaces in Brazil. Among the different datasets available, this work adopts the *olist order reviews dataset*, which disposes of the user comments plus a label with a satisfaction rate ranging from 1 to 5.

3.1.2 B2W

In 2019, B2W Digital, one of the most prominent Latin American e-commerce, released the *B2W Reviews01* [89], an open corpus of product reviews with more

[1] `https://www.kaggle.com/datasets/fredericods/ptbr-sentiment-analysis-datasets`

Table 3.1: Some examples of users' review (in Portuguese) for each database, with the corresponding polarity.

Database	User review	Polarity
Olist	*"O produto chegou no prazo combinado. Recomendo a loja"*	1
	"Solicitei devolução! No site marcava um tamanho e veio menor"	0
Buscapé	*"otimo pra quem quer uma foto com qualidade boa sem embaçados"*	1
	"tv com imagem escura, e veio sem os itens da fabrica"	0
B2W	*"Excelente produto, recomendo a família, e amigos. E a entrega foi rápida."*	1
	"Ainda não recebi o produto, portanto não posso avaliá-lo!"	0
UTLC-Apps	*"muito bom gostei de mais estão de parabéns"*	1
	"horrível não consigo entrar no app e ninguém me responde no e-mail"	0
UTLC-Movies	*"Obrigado Miyazaki por esse filme tão sutil, tão profundo e tão lindo..."*	1
	'Um dos piores desfechos de filme que já vi na vida."	0

than 130,000 user reviews. This dataset has two target features: the binary label "recommend to a friend" and a user rate from 1 to 5 stars. This work only considered the user rate.

3.1.3 Buscapé

As described by Hartmann et al. [90], the *Corpus Buscapé* is a large corpus of product reviews in Portuguese, crawled in 2013, integrating more than 80,000 samples from the *Buscapé*, a product and price search website. Unlike the datasets previously described, the Opinando labels' range is from 0 to 5, leading us to remove the comments rated as zero[2].

3.1.4 UTLC-Apps and UTLC-Movies

The *UTLCorpus* [91] is the most extensive set considered here, having more than 2 million reviews. It includes movie reviews collected from the *Filmow*, a famous movie Social network, and mobile apps comments collected from the Google Play Store. Here, the *UTLCorpus* was split into two different datasets: the *UTLC-Movies* and the *UTLC-Apps*. Similar to the Buscapé database, reviews with a rating equal to 0 were excluded.

3.2 Preprocessing and Analysis

For all datasets, the target values were defined by the polarity of the review, computed using the corresponding rating values as follows: we assumed a given review as positive when associated with 4 and 5 stars; negative for 1 and 2 stars, and excluded 3 stars occurrences.

The consolidated dataset includes two additional columns, one for each partition modality, to indicate the fold index associated with each instance. Such partitions were defined to maintain the original label stratification per each fold, aiming to make easier the adoption of the k-fold cross-validation or the training-validation-testing hold-out scheme by the practitioner. In this work, the first eight folds were assumed to define the training set, the ninth fold as the validation set, and the tenth as the testing set.

Besides, we included a column with the user review tokenized to facilitate anyone wishing to skip this preprocessing step. In this process, we lower-cased all the strings and converted their corresponding letters to the English alphabet, removing

[2]We removed zero-rating reviews to be on a similar scale to others. However, it would be interesting to include these reviews for future works since the score 0 of Buscapé can be as negative as the reviews with a score of 1 of the other datasets since they are the worst score.

the accents and converting all the occurrences of "ç" by "c". After that, we obtained tokens with 2 to 30 alphanumeric characters, disregarding special characters and removing the stop words. Samples with no comments or null labels were also excluded.

Table 3.2 summarizes the number of classes, samples, and tokens for each dataset. For the sake of comparison, we also included similar numbers for the classic English benchmarks *AG's News*, *DBPedia*, *Yelp Review* (Full and Polarity), and *Amazon Reviews* (Full and Polarity). One may quickly note that the datasets in English do not include a validation set and are generally more significant than the Brazilian counterparts.

Table 3.2: Number of samples of the Brazilian Portuguese datasets after preprocessing and some English datasets widely used for text classification.

Dataset	Classes	Train samples	Validation samples	Test samples
Olist (Polarity)	2	30 k	4 k	4 k
Olist (Rating)	5	33 k	4 k	4 k
B2W (Polarity)	2	93 k	12 k	12 k
B2W (Rating)	5	106 k	13 k	13 k
Buscapé (Polarity)	2	59 k	7 k	7 k
Buscapé (Rating)	5	68 k	8 k	8 k
UTLC-Apps (Polarity)	2	775 k	97 k	97 k
UTLC-Apps (Rating)	5	832 k	104 k	104 k
UTLC-Movies (Polarity)	2	952 k	119 k	119 k
UTLC-Movies (Rating)	5	1190 k	149 k	149 k
All combined (Polarity)	2	1909 k	239 k	239 k
All combined (Rating)	5	2229 k	279 k	279 k
AG's News	4	120 k	-	7.6 k
DBPedia	14	560 k	-	70 k
Yelp (Polarity)	2	560 k	-	38 k
Yelp (Rating)	5	650 k	-	50 k
Yahoo! Answers	10	1400 k	-	60 k
Amazon (Polarity)	2	3600 k	-	400 k
Amazon (Rating)	5	3000 k	-	650 k

Table 3.3 compares the document and vocabulary sizes for each one of the datasets, accounting only for the tokens/grams occurrences that appeared more than five times in the corpus. As expected, increasing the dataset size leads to a more extensive vocabulary, except for the Buscapé, which is smaller than B2W but has longer sentences. These English datasets have richer content than the Portuguese counterparts with considerably longer sentences. This indicates that such Portuguese datasets may benefit less from more complex models than English ones.

In addition, to evaluate the similarity between different datasets, Table 3.4 shows

Table 3.3: Document length (number of tokens) and vocabulary size of Brazilian Portuguese and English datasets.

Dataset	Mean length	Median length	Vocabulary size (1 gram)	Vocabulary size (1 and 2 grams)
Olist	7	6	3.272	8.491
Buscapé	25	17	13.470	52.769
B2W	14	10	12.758	47.929
UTLC-Apps	7	5	28.283	179.227
UTLC-Movies	21	10	69.711	635.869
All combined	15	7	86.234	884.398
AG's News	21	20	24.713	96.070
DBPedia	30	30	110.755	521.403
Yelp Reviews	68	50	70.494	1.303.148
Yahoo! Answers	11	3	65.534	464.409
Amazon Reviews	38	33	176.464	3.475.911

the percentage of words/tokens shared between them. For example, 89.5% of the Olist vocabulary also integrates the Buscapé vocabulary, and 21.7% of the Buscapé vocabulary is in the Olist vocabulary. The reported rows and columns averages (avg) exclude the diagonal entries. Most of the words included in the Olist, the smallest and with the shortest sentences, are present in other datasets, on average 93.5%. In turn, the much larger UTLC-Apps and UTLC-Movies detain the highest percentage of words (an average of 69.6% and 84.4%, respectively) common to all other datasets.

Table 3.4: Percentage of words in common between datasets.

Dataset	Olist	Buscapé	B2W	UTLC Apps	UTLC Movies	Avg
Olist	100.0	21.7	25.3	10.6	4.4	15.5
Buscapé	89.5	100.0	73.5	37.1	16.3	54.1
B2W	98.5	69.6	100.0	35.6	15.8	54.9
UTLC-Apps	91.8	77.8	79.0	100.0	29.6	69.6
UTLC-Movies	94.1	84.4	86.1	72.9	100.0	84.4
Average	93.5	63.4	66.0	39.1	16.5	-

Table 3.5 demonstrates that most of the vocabulary contained in the texts integrating the datasets considered here are covered by the pre-trained word vectors from the NILC Word Embeddings Repository [92], with an adherence superior to 90%, except to UTLC-Apps.

Table 3.6 analyzes the experimental distribution of the target features: polarity and rating for these datasets. We present the original 5-point scale rate for the target value in the upper part. In the lower, we have the generated binary polarity target

Table 3.5: Word vectors coverage of NILC embeddings per dataset.

Dataset	Word vectors coverage
Olist	96.5%
Buscapé	92.9%
B2W	94.4%
UTLC-Apps	85.0%
UTLC-Movies	91.1%

feature, where 0 represents the negative cases (1 and 2 points) and 1 corresponds to the positive reviews (4 and 5 points). Despite the higher frequency of the positive label, the distribution varies significantly, which may hinder the performance of a single model to cope with all datasets.

Table 3.6: Labels distribution for each dataset (%).

Label	Olist	Buscapé	B2W	UTLC Apps	UTLC Movies	All
1	22.0	3.7	20.7	16.4	2.4	8.9
2	5.3	4.3	6.3	4.5	6.8	5.8
3	8.8	13.4	12.3	6.8	20.0	14.4
4	14.5	39.5	24.4	11.4	36.7	26.4
5	49.4	39.1	36.2	60.8	34.0	44.5
0 (1-2)	30.0	9.2	30.8	22.5	11.6	17.2
1 (3-4)	70.0	90.8	69.2	77.5	88.4	82.8

Finally, it is worth mentioning that the dataset domains significantly influence how the models understand and represent the texts [1]. In our case, four datasets (Olist, Buscapé, B2W, and UTLC-Apps) have product reviews, and the dataset UTLC-Movies contains movie reviews. The latter tends to present more nuances, and the models may face more difficulties in the sentiment classification task.

The rationale behind releasing this consolidated dataset on the Internet is to provide a useful resource for those interested in sentiment analysis of commercial textual data in Portuguese.

Chapter 4

Text Classification Pipeline

This chapter describes the experimental procedure performed to evaluate the embedding generation methods. To obtain sentiment predictions from textual excerpts, a set of sequential steps (pipeline) is required, ranging from textual preparation to the Machine Learning modeling.

The general pipeline adopted for deploying the different embeddings alternatives for text classification is depicted in Figure 4.1. The first step includes collecting annotated user reviews and cleaning them, a fundamental step when dealing with free insertion texts. After, the document content is split into textual sub-units named tokens. Then, the tokens with lower semantic (e.g. low frequency words or stop words) relevance may be removed, depending on the approach under analysis. After, a single feature vector is generated to feed the classification model. The following subsections describe these steps in detail.

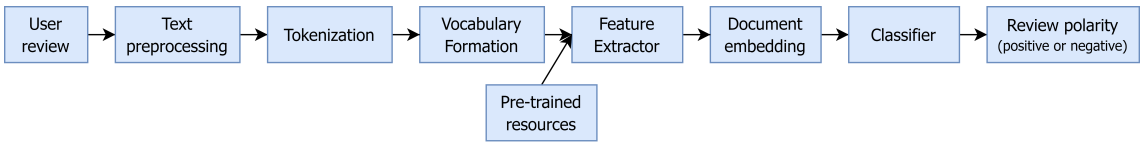

Figure 4.1: General scheme of text classification on user reviews.

Table 4.1 provides an overview of all experiments performed in this work. For didactic reasons, we divided the analyzed models into three families - Bag-of-Words, Classical Deep Learning, and Transformer-based Large Language Models - and presented the techniques in order of increasing complexity in the text. In addition, Table 4.1 also shows the variations performed in each model. The hyperparameters and configurations of the architectures that remained constant in all experiments are described throughout Subsection 4.4.

Table 4.1: Summary of the evaluated model variations.

Model Family	Model	Variations
Bag-of-Words (BoW)	TF-IDF	**Feature selection methods:** frequency, chi2, fvalue
	TF-IDF+SVD (LSA)	-
	Bag of Word Vectors Averaged (avgBoWV)	**Word vector model:** Word2vec, GloVe, FastText **Word vector dimension:** 50, 100, 300
	IDF Weighted Bag of Word Vectors (idfBoWV)	**Word vector dimension:** 50, 100, 300
	Bag of Word Vectors with first principal component removal (BoWV-PC)	**Weighting scheme:** unweighted, idf-weighted **Word vector dimension:** 50, 100, 300
Classical Deep Learning (CDL)	Convolutional neural networks (CNN)	**Filter sizes:** [2], [2,3], [2,3,4], [2,3,4,5] **Feature map size**: 50, 100, 200, 400
	Long short term memory neural networks (LSTM)	**Layers:** 1, 2 **Hidden size:** 64, 128, 256 **Pooling layer:** average pooling, max pooling, avg and max concatenated
Trasformer-based Large Language Model (TLM)	Feature-based TLM (FB TLM)	**Models:** see Table 4.2 **Aggregation types:** see Section 4.4.3
	Finetuned TLM (FT TLM)	-

4.1 Text pre-processing and tokenization

Except in the case of TLMs, the documents were lower-cased and had URLs and special characters removed. For the TF-IDF and LSA approaches, words were converted to the English alphabet, i.e., had the accents removed, and the occurrences of "ç" were replaced by "c". The models based on word vectors (Bag of Word Vectors, CNN, and LSTM) only underwent this process when a word was not found in the corresponding embedding vocabulary. This word vector match procedure is better explained in Section 4.3.

Regarding tokenization, Bag-of-words and Classical Deep Learning models considered tokens with 2 up to 30 strings of letters separated by white spaces. Transformer-based Language Models considered raw input texts after being processed by the HuggingFace's AutoTokenizer[1]. The behavior of this AutoTokenizer class varies according to the model adopted, but roughly it retains accents and capital letters, adding specific tokens whenever required by the corresponding Transformer model. No further sentence processing (e.g., normalization, spell correction, named entity recognition) is conducted in this phase.

CDL models and TLMs required sentence padding to make all sentences with the same length, truncating the texts greater than the maximum defined length and filling with zeros or padding tokens the documents shorter than the maximum length. The first ones have the sentences padded to the 90% percentile of the number of tokens of each review per database. Conversely, TLMs have the sentences padded to 60 tokens.

4.2 Vocabulary Formation

The TF-IDF experiments considered from 1 to 3-grams and different strategies to define the subset of n-grams from the corpus that should integrate the vocabulary. We did not consider n-grams for values of n larger than 3 since some preliminary tests pointed out a considerable increase in the computational time, especially for larger databases, such as the UTLC-Movies, which was not followed by any significant gain in the classification accuracy.

All remaining methods considered only one gram. Words appearing less than five times in the corpus and the stop-words were removed from the BoW experiments. In the case of CDL models, all words to which pre-trained word vectors could not be found were ignored.

[1] `https://huggingface.co/docs/transformers/v4.17.0/en/model_doc/auto`

4.3 Pre-trained Resources

For the avgBoWV model, we evaluated the pre-trained word vectors models *Word2Vec* [21], *GloVe* [22], and *FastText* [24], with dimensions 50, 100, and 300, available at NILC Word Embeddings Repository [92]. All of these word vectors were trained entirely with a Brazilian Portuguese corpus. For the other Bag of Word vectors and classical Deep Learning models, we tested only the FastText, since it was the word vector model with the best predictive performance in our initial experiments.

The process of identifying the corresponding embedding from a word was the following: the word is searched in the vocabulary list after being lower-cased and having special characters and URLs removed. If not found, it is converted to the English alphabet and searched again. If again not found, it is ignored. Table 3.5 depicts the percentage of words in each database covered by the FastText word vectors, signalizing a good embedding coverage in most cases (above 85%).

The Transformer-based Language Models considered three multilingual and seven language-specific models. We evaluated only open-source TLMs for Portuguese released by the Hugging Face [93] initiative, an open-source NLP community. Brazilian Portuguese variants of the GPT (Generative Pre-trained Transformer) on this platform do not have papers attached but were included in the experiments of this work, such as the GPT-Neo Small Portuguese[2] and GPorTuguese-2[3], representing fine-tuned versions of the GPT-Neo 125M[4] and GPT-2 Small [69], respectively. Table 4.2 summarizes the dimensionality of the related embeddings and the number of model parameters, aiming to provide some guiding information over the practical trade-offs between complexity and performance observed with these models in our experiments.

4.4 Embeddings generation

In the following, we summarize some practical aspects of the experiments reported in Table 4.1.

4.4.1 Bag-of-Words

The TF-IDF experiments with restricted-size vocabularies considered three alternatives for word selection: frequency, chi-square test statistics ("chi2"), and ANOVA F-value ("fvalue"), all available in the Scikit-Learn [94] framework. BoW models exploited word embeddings with 50, 100, and 300 dimensions, downloaded from the

[2] `www.huggingface.co/HeyLucasLeao/gpt-neo-small-portuguese`
[3] `www.huggingface.co/pierreguillou/gpt2-small-portuguese`
[4] `www.huggingface.co/EleutherAI/gpt-neo-125M`

Table 4.2: Summary of some characteristics of the Transformer-based Language Models evaluated in this work.

Model	Multilingual	Embedding length	Parameters ($\times 10^6$)
PTT5 Small	No	512	60
m-BERT	Yes	768	110
BERTimbau Base	No	768	110
GPT2 Small	No	768	117
XLM-Roberta Base	Yes	768	125
GPTNeo Small	No	768	125
PTT5 Base	No	768	220
BERTimbau Large	No	1024	345
XLM-Roberta Large	Yes	1024	355
PTT5 Large	No	1024	770

NILC Word Embeddings Repository [92], and trained over a Brazilian Portuguese corpus. We considered the arithmetic (avgBoWV) and weighted (idfBoWV) average strategies for generating document vector embeddings.

4.4.2 Classical Deep Learning

CDL models were fed with Fast Text word embeddings having a dimensionality equal to 300 and trained considering a Logistic Regression layer stacked over the top, targeting polarity prediction. After that, we adopted these models to generate vectorial representations for the documents. The experiments with the LSTM and CNN architectures are detailed in the following.

1. LSTM: the evaluated architecture is exhibited in Figure 4.2. It comprises one or two biLSTM layers, followed by a pooling layer, a dense layer with ReLU activation function (the output size is equal to the LSTM hidden size), and a linear dropout.

2. CNN: this architecture was based on [43] and is depicted in Figure 4.3. Roughly, it explores convolutional filters with different kernel sizes operating over the embeddings of the document's words. The resulting feature maps undergo a max-pooling operation, and the resulting scalars are concatenated to feed a Logistic Regression layer with dropout.

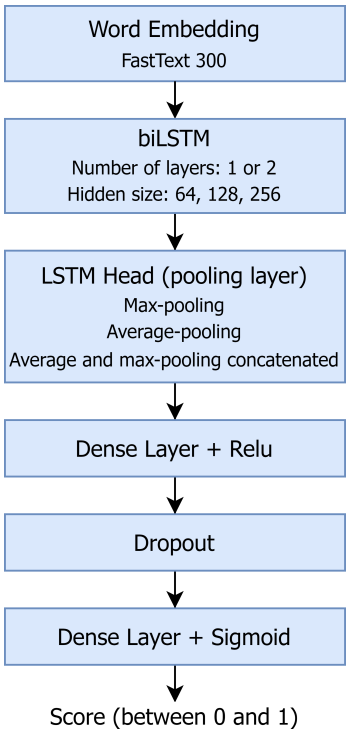

Figure 4.2: General LSTM architecture for the CDL experiments (see text).

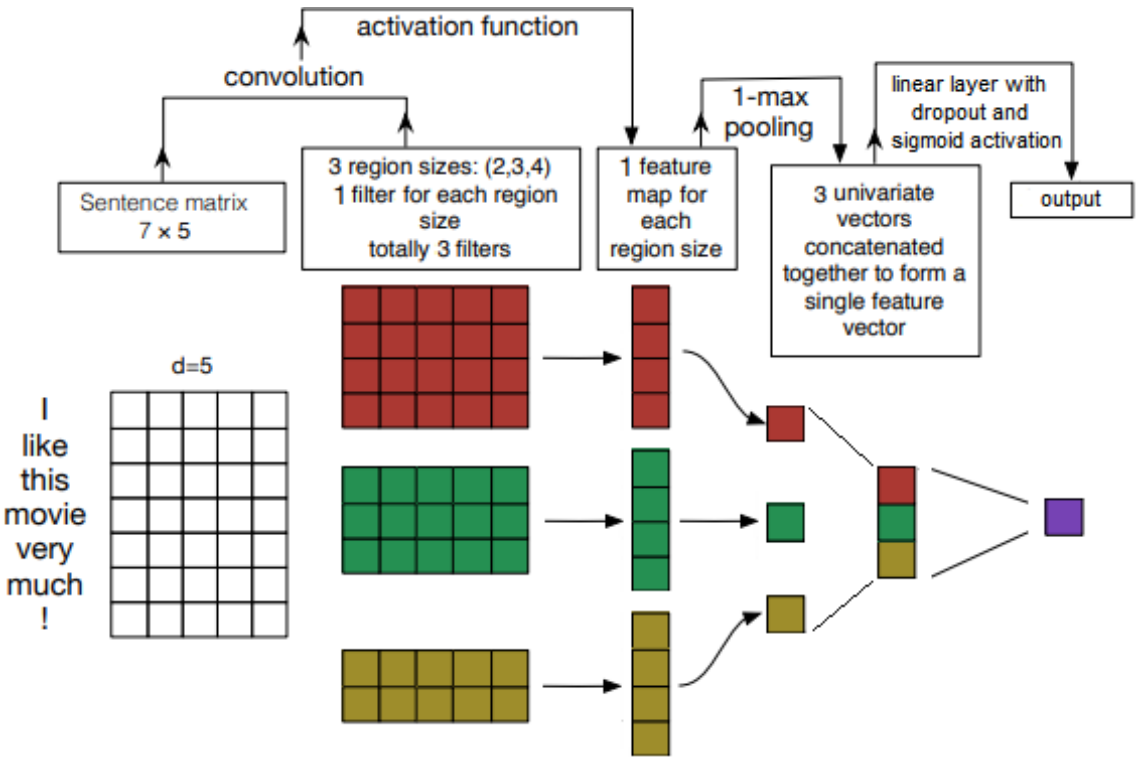

Figure 4.3: General CNN architecture for the CDL experiments (see text). Adapted from [43].

4.4.3 Transformer-based Large Language Models

The experiments involving the Transformer-based Large Language Models included two design strategies: pre-trained (or feature-based) and fine-tuned. The pre-trained solution considered just using the TLM for producing document embeddings, which were subsequently fed to a Logistic Regression classifier. Conversely, for the fine-tuned model, the TLM weights were fine-tuned to the sentiment analysis task. The experiments solely exploring pre-trained models for generating document embeddings will be referred to here as Feature-based TLM (FB TLM). In turn, those involving fine-tuning are named here as Fine-tuned TLM (FT TLM).

Feature-based Large Language Model

Previous works [71] reported that creative combinations of the token representations provided by the BERT outputs might lead to a significant performance improvement in the NER task, even without any fine-tuning of model parameters. Such findings strongly motivated the following experiments. As shown in Figure 4.4, TLMs output a 3D tensor whose number of rows is equal to the batch size, the number of columns corresponds to the number of tokens, and the embedding size defines the depth. In this way, each column of this tensor represents a sequence of embeddings corresponding to some position-specific token of the documents integrating this batch. The evaluated TLMs models assumed documents constituted by one to sixty tokens. To each token, these models produced a representation having from 512 to 1024 dimensions, as shown in Table 4.2, referred to as token embedding. In the following, we describe the approaches evaluated in this work to combine these embeddings. In parenthesis, we exhibited the number of vector concatenations of each case. Thus, the size of the vectors used for document representations has from 512 (512×1) up to 3072 (1024×3) dimensions. Due to a high computational burden, for the TLMs T5 Large, XLM-RoBERTa Base, and XLM-RoBERTa Large, we limited the aggregation types to "first", "last", "mean all", "first + mean + std", and "mean + min + max".

1. **first** (1): this embedding corresponds to the first token. For BERT models, it is equivalent to the [CLS] special token, created with the purpose of sentence classification, and considering as the default BERT document embedding;

2. **second** (1): embedding corresponding to the 2nd token;

3. **last** (1): embedding corresponding to the last token;

4. **sum all** (1): sum of all token embeddings;

5. **mean all** (1): average of all token embeddings;

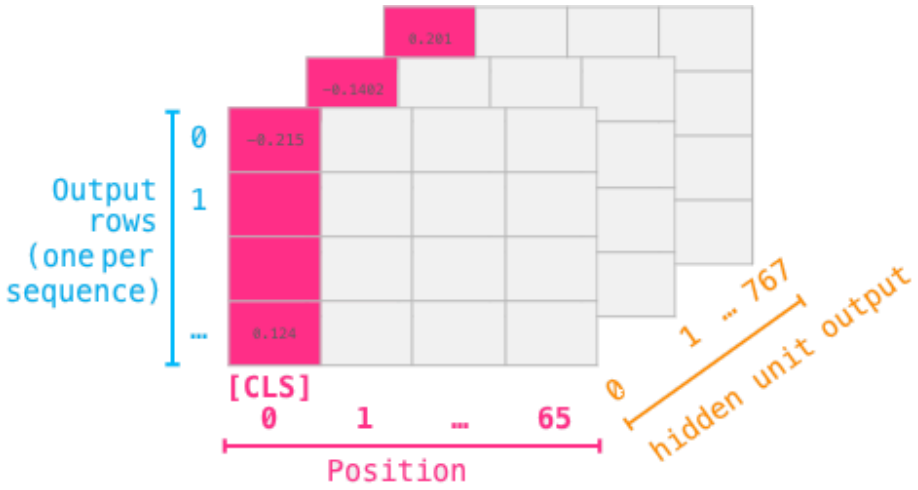

Figure 4.4: Example of BERT Base output. Adapted from [95].

6. **sum all except first** (1): sum of all token embeddings, ignoring the first one;
7. **mean all except first** (1): average of all token embeddings, ignoring the first one;
8. **sum + first** (2): concatenation of the sum of all token embeddings and the first token embedding;
9. **mean + first** (2): concatenation of the average of all token embeddings and the first token embedding;
10. **first + mean + std** (3): concatenation of the embeddings of the first token, the average, and the standard deviation of the remaining tokens;
11. **first + mean + max** (3): concatenation of the embeddings of the first token, the average, and maximum of all token embeddings;
12. **mean + min + max** (3): concatenation of the average, minimum, and maximum of all token embeddings;
13. **quantiles 25, 50, 75** (3): concatenation of the quantiles 25%, 50%, and 75% of all token embeddings.

One of the major differences between BERT and alternative TLMs is its capability to accomplish the Next Sentence Prediction (NSP) task. Besides, BERT has a token dedicated to sentence classification (CLS). It is possible to include this token when fine-tuning the other models, which would make them more suitable for text classification. However, this work restricts analyzing them in a feature-based approach.

Unlike BoW, which is based on static word embeddings, the document embeddings generated by TLMs are contextual, resulting in significant gains in the predictive performance of the models, as will be shown later.

Fine-tuned Large Language Model

The fine-tuning experiments were restricted to the BERT models (m-BERT, BERTimbau Base, and BERTimbau Large) since it would be too computationally demanding to fine-tune all TLMs. In addition, with these three BERT variants, we could verify the influence of fine-tuning on multi-lingual vs. language-specific models and on Based vs. Large size models.

Design choices, such as the training algorithm and the model hyperparameters, were based on Sun et al. [96]. A Logistic Regression network was added on the top of the layer associated with the [CLS] token, targeting to predict one of the two sentiment classification classes, i.e., assuming as target-values 0 or 1. The network training adopted the Adam optimizer with weight decay, slanted triangular learning rates with a warm-up proportion of 0.1, and a maximum number of epochs equal to 4.

4.4.4 Classifiers

All document embeddings were evaluated using Logistic Regression. In the case of BoW models and Feature-based TLMs, we adopted the *scikit-learn* implementation with default parameters (the regularization factor seemed not to affect most of our experiments significantly). Alternatively, Classical Deep Learning and Fine-tuned TLMs models have considered a neural implementation of this classifier.

The exception was the Bag-of-Words experiments, including the LightGBM from the homonymous library [97]. LightGBM is a non-linear tree-based classifier able to capture more complex patterns at a higher computational cost. In these cases, the results reported were restricted to the best-performing model.

4.4.5 Training Procedure

For the models not requiring hyperparameters' tuning, the training was conducted over fused training and validation sets, and the performance metrics were inferred over the test set. The hyperparameters of the remaining models were based on the validation set performance. After that, they were retrained with the training and validation sets concatenated.

4.4.6 Hyperparameter tuning

The hyperparameter tuning process was restricted to Classical Deep Learning and Fine-tuned TLM, focusing on design aspects with more impact on the model performance, like the learning rate and the number of training epochs.

For the Classical Deep Learning models, we performed a hyperparameter search concerning the dropout (0, 5%, 10%, 15%, 20%, 25%, 30%, 35%, and 40%) and learning rates (5e-4, 1e-3, 5e-3, 1e-2).

For the Fine-tuned TLM experiments, we varied the dropout (0 and 10%) and learning (2.5e-5 and 5e-5) rates. Experiments for hyperparameters' tuning were restricted only to the *Olist*, *Buscapé*, and *B2W* datasets, due to the expressive computational efforts that would be required for conducting this analysis in the remaining datasets. The best combination of the learning rate, dropout rate, and the number of epochs were 2.5e-5, 10%, and 1, for the BERTimbau models, whereas 2.5e-5, no dropout, and 2, for the m-BERT. As expected, the m-BERT required one more training step to learn Portuguese language patterns.

4.4.7 Computational resources

We implemented all models using the Google Colab Pro+ platform, a premium version of the Google Colab. Bag-of-Words models adopted a CPU with 16 GB of RAM, while CDLs and TLMs exploited a P100 GPU with a vRAM of 16GB. For Feature-based TLMs, as we vectorized the entire corpus of each database before going through the classifier, we used the Google Colab's high RAM environment option to store this large matrix, increasing the available RAM from 16GB to 51GB.

Chapter 5

Results and Discussion

This chapter is dedicated to analyzing the results of all experiments. First, Section 5.1 describes the performance assessment process conducted in this work. Then, we split the models' results in the Sections 5.2-5.5, stratifying them according to an increasing level of complexity/innovation regarding the models used. Then, Section 5.6 makes an overall comparison of the best configurations identified for each model family. Finally, Section 5.7 conducts a qualitative analysis of the prediction errors made by the best-evaluated model as a way to try to reason with these mistakes.

5.1 Accessing models' performance

The figure of merit to select the best model for each embedding modality was the ROC-AUC. The rationale is that it is threshold invariant, summarizing the performance of a classifier for different operational settings, which may be established according to the risk associated with a wrong prediction. Due to our focus on e-commerce, such risks often vary according to the application domain, making the ROC-AUC an attractive performance metric.

After identifying the models that have shown the best performance for each general embedding approach, we conducted a k-fold cross-validation experiment with $k = 10$ reported in Section 5.6, but restricted to these cases due to computational reasons. Besides, we computed the average and standard deviation values of some performance metrics computed over the ten (predefined) test folds. Finally, we submitted these results to statistical testing for a more rigorous analysis of the models' performance. In this overall comparison, we have also considered the Accuracy and F1-Score.

5.2 Bag-of-Words

First, we analyzed the models based on the average document embedding strategy (avgBoWV). Among all experiments involving different techniques, sizes, and datasets, the best ROC-AUC for the testing sets was achieved by the *LightGBM*, which is more powerful than LR. Table 5.1 summarizes the values obtained. Overall, there is a significant increase in the predictive performance with the growth of the embedding size. Also, FastText outperforms GloVe, which supersedes Word2Vec. The 300-dimensional FastText embedding performs better for all datasets. Thus, the embedding technique and the dimensionality of the embedding vector have a profound impact on the model performance.

Table 5.1: ROC-AUC (%) for the avgBoWV models, where "FastText 300" refers to 300-dimensional FastText word vectors. The best result of each dataset is underlined.

Model	Olist	Buscape	B2W	UTLC-Apps	UTLC-Movies	All
FastText 50	93.9	86.2	94.3	91.5	79.3	86.5
FastText 100	94.9	87.4	95.2	92.3	81.7	87.8
FastText 300	<u>95.5</u>	<u>88.5</u>	<u>96.1</u>	<u>93.2</u>	<u>83.9</u>	<u>89.0</u>
GloVe 50	93.2	85.2	93.0	90.8	78.0	85.6
GloVe 100	93.8	86.5	94.4	91.8	80.3	86.9
GloVe 300	95.1	87.7	95.6	92.6	83.1	88.2
Word2Vec 50	92.8	83.9	92.4	89.7	75.4	83.9
Word2Vec 100	94.0	85.6	93.6	91.1	78.3	85.7
Word2Vec 300	94.8	87.8	95.1	92.4	82.0	87.6

Figure 5.1 summarizes all Bag-of-Words results, including the Bag-of-Word Vectors with FastText and TF-IDF models. For all simulations, increasing the document embedding size has led to some performance improvement. Relatively to TF-IDF models, selecting words for the dictionary construction based on the frequency (standard approach) is better than using the “chi-2” and “f-value” alternatives, except for the UTLC-Movies database. In this case, the latter approach allowed a gain of 0.8% and 2.4% percentage points for 300 and 1000 dictionary words, respectively, compared to the standard approach.

Interestingly, the model exploiting the SVD decomposition of the TF-IDF feature matrix has shown the best results for vocabulary sizes of 50, 100, and 300, performing even better than the unweighted average of FastText vectors. Thus, among the BoW models evaluated in this work, the TF-IDF with a vocabulary selection based on word frequency is the best approach, reaching a performance plateau of around 10,000 words. Also, if it is necessary, for reasons of computational limitation, to limit the number of input features, the SVD can be a useful resource.

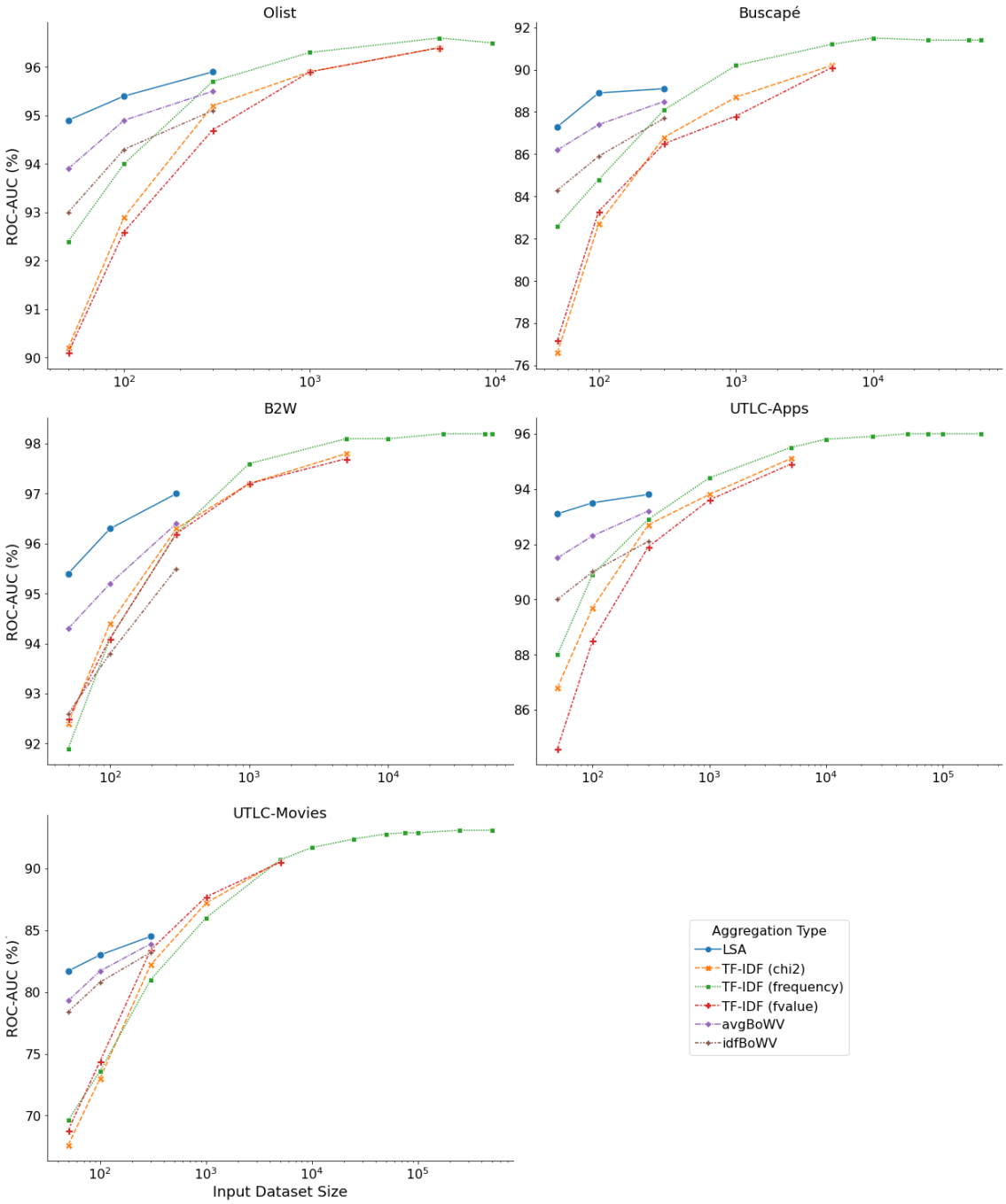

Figure 5.1: ROC-AUC (%) as a function of the dictionary size and BoW model adopted (see text).

Finally, for the best-performing word vector algorithm in our simulations (FastText), Table 5.2 shows a performance comparison between models with and without removing the first principal component. In most cases, we can quickly note that this strategy is ineffective since the observed difference is lower than 0.1%. The exception is for the weighted average in Buscapé, where such removal brings an absolute increment of 0.5%, 0.4%, and 0.3% for embeddings with 50, 100, and 300 dimensions, respectively.

Table 5.2: ROC-AUC (%) according to Embedding Size, Weighting scheme and Principal Component (PC) Removal using FastText Word Vectors.

Vector Size	Weighting Scheme	PC Removal	Olist	Buscapé	B2W	UTLC Apps	UTLC Movies
50	Unweighted	No	93.9	86.2	94.3	91.5	79.3
		Yes	93.8	86.3	94.2	91.5	79.3
		Delta	-0.1	0.1	-0.1	0.0	0.0
	IDF-weighted	No	93.0	84.3	92.6	90.0	78.4
		Yes	93.0	84.8	92.7	90.0	78.4
		Delta	0.0	0.5	0.1	0.0	0.0
100	Unweighted	No	94.9	87.4	95.2	92.3	81.7
		Yes	94.9	87.5	95.2	92.3	81.7
		Delta	0.0	0.1	0.0	0.0	0.0
	IDF-weighted	No	94.3	85.9	93.8	91.0	80.8
		Yes	94.3	86.3	93.8	91.0	80.8
		Delta	0.0	0.4	0.0	0.0	0.0
300	Unweighted	No	95.5	88.5	96.1	93.2	83.9
		Yes	95.6	88.5	96.1	93.2	84.0
		Delta	0.1	0.0	0.0	0.0	0.1
	IDF-weighted	No	95.1	87.7	95.1	92.1	83.2
		Yes	95.1	88.0	95.0	92.1	83.2
		Delta	0.0	0.3	-0.1	0.0	0.0

5.3 Classical Deep Learning

5.3.1 Convolutional Neural Network

CNN results are presented in Table 5.3. All architectures performed similarly for the datasets with short sentences, like Olist, B2W, and UTLC-Movies. However, an increase of at least one percentage point was observed for the Buscapé and UTLC-Movies when using tuned hyperparameters. We must stress that Buscapé and UTLC-Movies detain the longest sentences in this comparison.

Table 5.3: Values of ROC-AUC (%) observed for the CNN models. The delta row summarizes the gap between the best and the worst ROC-AUC performances observed in each dataset. The highest performances per database are signalized by bold, while the lowest ones are underlined.

Filter sizes	Feature map size	Olist	Buscapé	B2W	UTLC-Apps	UTLC-Movies
2	50	97.6	92.3	98.6	96.7	92.5
	100	97.7	92.2	98.6	**96.8**	93.0
	200	97.7	92.7	98.7	**96.8**	93.3
	400	**97.8**	92.8	**98.8**	**96.8**	93.5
2, 3	50	97.6	92.1	98.7	**96.8**	92.7
	100	**97.8**	92.6	98.6	**96.8**	93.3
	200	**97.8**	92.7	98.7	**96.8**	93.5
	400	**97.8**	92.9	**98.8**	**96.8**	**93.8**
2,3, 4	50	97.7	92.6	98.6	**96.8**	93.1
	100	97.7	92.8	98.7	**96.8**	93.4
	200	97.7	93.0	98.7	**96.8**	93.5
	400	**97.8**	93.0	**98.8**	96.7	93.6
2,3, 4,5	50	97.5	92.8	98.6	**96.8**	93.3
	100	97.6	92.9	98.7	**96.8**	93.5
	200	97.7	93.0	98.7	**96.8**	93.7
	400	97.7	**93.1**	98.7	96.7	**93.8**
	Delta	0.3	1.0	0.2	0.1	1.3

Only Buscapé and ULTC-Movies have shown a more significant impact of architectural issues on the ROC-AUC values. Figure 5.2 shows the influence of each architectural change for both datasets separately. For example, in the first boxplot of the upper left plot, we verified the distribution of the ROC-AUC values for all experiments that presented the feature map equal to 50.

Note that the increase in the feature map consistently raised the first, second, and third quartiles, while the increase in the size of the filters has showed a less prominent rise in the predictive power. As we can see in Table 5.3, a simple increase in the feature map may lead to a performance equivalent to adopting architectures

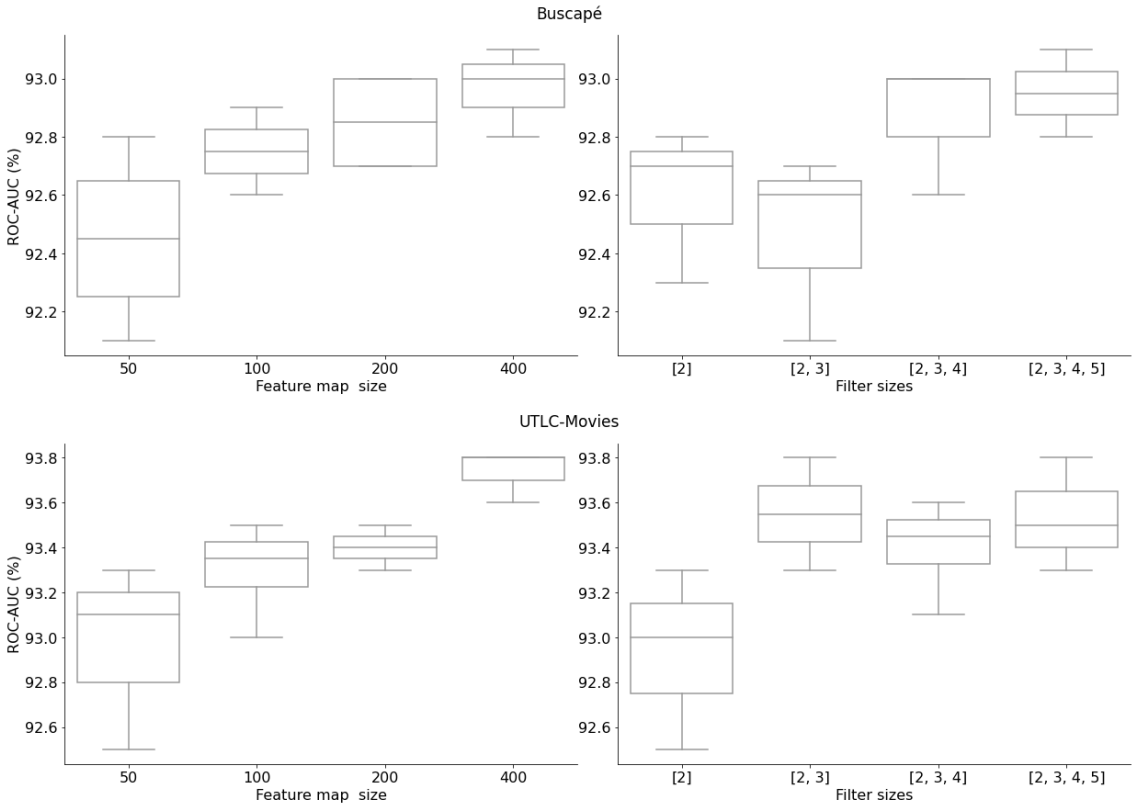

Figure 5.2: Influence of each change in the architecture of the CNN networks for the Buscapé and UTLC-Movies datasets (see text).

with more filters. This result aligns with the idea that most of the review are short and have a straightforward content, thus not benefiting from filters larger than 3-grams.

5.3.2 Recurrent Neural Networks

Table 5.4 depicts the results related to the LSTM models. Only the Buscapé dataset shows a higher impact of model hyperparameters over the performance (1.3%). All the others, including the UTLC-Movies, presented minor performance differences, around 0.2% and 0.3%.

For Buscapé, the only dataset to which the variations in the architecture have significantly impacted the ROC-AUC, we present in Figure 5.3 the influence of each of these changes, in a similar fashion than Figure 5.2. We can note that the increase in the number of layers has led to a small positive impact on the predictive performance. Similarly, regarding the hidden size of the LSTM model, the median values remained reasonably stable, but there was a relevant increase in the interquartile range when increasing the cell size. The architectural factor that has most influenced the model performance was the LSTM head, which can be associated with an increase in the median and a decrease in the interquartile range

Table 5.4: Values of ROC-AUC (%) observed for the LSTM models. The delta row summarizes the gap between the best and the worst ROC-AUC performances observed in each dataset. The highest performances per database are signalized by bold, while the lowest ones are underlined.

Number of layers	Hidden Size	Pooling	Olist	Buscapé	B2W	UTLC-Apps	UTLC-Movies
1	64	Avg	**98.0**	93.0	**98.8**	97.2	94.3
		Max	97.9	92.9	98.7	97.2	94.4
		Avg‖Max	97.8	92.8	98.7	97.2	94.4
1	128	Avg	97.7	92.2	**98.8**	97.2	94.3
		Max	97.8	92.6	**98.8**	97.2	**94.5**
		Avg‖Max	**98.0**	93.3	**98.8**	**97.3**	**94.5**
1	256	Avg	97.7	92.7	**98.8**	97.1	**94.5**
		Max	97.9	93.0	98.6	97.2	**94.5**
		Avg‖Max	97.7	**93.5**	98.7	97.1	**94.5**
2	64	Avg	**98.0**	93.0	**98.8**	97.2	94.3
		Max	97.9	92.9	98.7	97.2	94.4
		Avg‖Max	97.8	92.8	98.7	97.2	94.4
2	128	Avg	97.8	93.3	98.7	97.1	94.3
		Max	97.9	93.4	**98.8**	97.2	**94.5**
		Avg‖Max	97.8	93.1	**98.8**	**97.3**	**94.5**
2	256	Avg	97.7	92.7	**98.8**	97.1	**94.5**
		Max	97.9	93.0	98.6	97.2	**94.5**
		Avg‖Max	97.7	**93.5**	98.7	97.1	**94.5**
		Delta	0.3	1.3	0.2	0.2	0.2

observed in the plot. Notably, average pooling concatenated with max pooling is the best aggregating strategy.

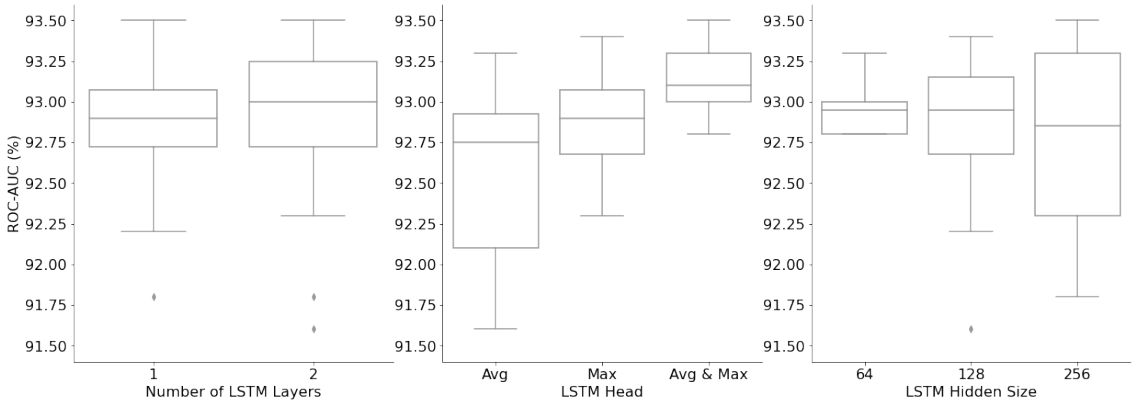

Figure 5.3: Influence of each architectural change in the LSTM networks for the Buscapé dataset.

5.4 Feature-Based Large Language Models

This analysis considered ten Transformer models, thirteen token aggregation modalities for document embedding (not all were performed for each TLM, only five were common to all), and five databases. After ranking the results for each database, an average rank for each TLM and aggregation modality was computed to provide an overall performance index, regardless of the database. Table 5.5 summarizes these results. Modalities that exploited two or more tokens usually showed better results than their single token counterparts. The quantiles-based and "second" aggregations performed poorly. Besides, for all methods and datasets, the aggregation modality "first + mean + std" have achieved the best results; thus, it represents an interesting strategy to boost the performance of TLMs that do not have dedicated tokens for classification.

Table 5.6 shows a similar analysis, but the average rankings were calculated over only the TLMs to produce a performance index, independent of the database and the aggregation modality. Here, we only considered the aggregation types common to all models ("first", "last", "mean all", "first + mean + std", "mean + min + max") to make a fair comparison. The table also reports the average ranks obtained for each model, including the average time, the reserved vRAM, and the allocated vRAM required to process a batch with a size of 128. The allocated vRAM corresponds to the portion of GPU memory currently used, while the reserved vRAM is related to the cache memory allocated.

One may readily observe that increasing the number of parameters for models from the same family results in higher predictive power. In other words, large model versions consistently outperform the corresponding base models. Among the leading models, PTT5 Large is quite competitive with BERTimbau, followed by PTT5 and XLM-Roberta (XLM-R), all representing significantly better alternatives than m-BERT, GPT2 Small, and GPTNeo Small. The GPT models seem unsuitable for embedding generation in our task due to their language model structure based only on decoder blocks and a unidirectional auto-regressive token processing, which is adequate for content generation.

Although PTT5 Large surpassed BERTimbau Large in some cases, as shown in Table 5.5, BERTimbau is still the model among TLM alternatives detaining the most attractive trade-off between performance and computational cost, since it has less than half as many parameters as PTT5 for the Large size and half as many as for the Base size, as the Table 4.2 points out.

Regarding the remaining parameters reported in Table 5.6, generally speaking, the reserved vRAM is correlated with the model size, while the allocated vRAM is with the embedding size. Therefore, these numbers shed some light on the practical

Table 5.5: LLMs and Aggregation Modalities' (AM) average rankings.

Model	AM	Rank	Model	AM	Rank
PTT5 Large	first+mean+std	1.2	GPT2	first+mean+max	51.2
BERTimbau Large	first+mean+std	1.8	PTT5 Small	mean+min+max	52.2
BERTimbau Large	first+mean	3.4	GPT2	mean+min+max	52.4
BERTimbau Large	first+mean+max	5.4	BERTimbau Base	last	52.4
PTT5 Large	mean+min+max	5.6	m-BERT	first+mean+std	53.8
PTT5 Large	mean all	6.4	XLM-R Base	first	54.8
BERTimbau Large	first+sum	7.0	PTT5 Base	second	55.6
BERTimbau Large	first	7.2	PTT5 Base	last	57.2
BERTimbau Large	sum all except first	7.6	m-BERT	mean+min+max	57.4
BERTimbau Large	sum all	7.6	XLM-R Base	last	59.8
BERTimbau Large	mean all except first	7.6	GPT2	mean all except first	60.4
BERTimbau Large	mean all	7.6	GPT2	sum all	60.4
PTT5 Base	first+mean+std	9.0	GPT2	sum all except first	60.4
BERTimbau Large	mean+min+max	10.0	GPT2	mean all	60.4
BERTimbau Large	quantiles 25,50,75	10.6	m-BERT	first+mean+max	60.8
PTT5 Base	first+mean+max	13.0	PTT5 Small	sum all	60.8
BERTimbau Base	first+mean+std	14.2	PTT5 Small	sum all except first	62.4
BERTimbau Base	first+mean	15.8	PTT5 Small	mean all	65.6
BERTimbau Base	first+mean+max	20.6	GPT2	quantiles 25,50,75	66.2
PTT5 Base	mean+min+max	21.0	m-BERT	first+mean	66.6
BERTimbau Base	first+sum	22.2	PTT5 Small	quantiles 25,50,75	66.6
PTT5 Base	mean all	23.6	XLM-R Large	first	67.2
PTT5 Base	mean all except first	24.4	PTT5 Small	mean all except first	67.8
BERTimbau Base	mean all except first	24.6	m-BERT	quantiles 25,50,75	68.8
BERTimbau Base	mean all	24.6	m-BERT	mean all	69.4
BERTimbau Base	sum all except first	24.6	m-BERT	mean all except first	69.4
PTT5 Base	sum all	25.2	m-BERT	first+sum	70.8
BERTimbau Base	sum all	25.4	m-BERT	sum all except first	71.0
BERTimbau Base	mean+min+max	25.6	m-BERT	sum all	71.0
PTT5 Base	sum all except first	25.8	GPTNeo	first+mean+max	78.8
BERTimbau Base	quantiles 25,50,75	26.4	GPTNeo	first+mean+std	80.0
PTT5 Large	first	27.2	PTT5 Small	first	80.2
PTT5 Base	quantiles 25,50,75	29.0	GPTNeo	mean+min+max	82.4
XLM-R Large	first+mean+std	30.8	GPT2	last	83.2
XLM-R Large	mean+min+max	35.4	m-BERT	first	83.2
BERTimbau Base	first	35.8	PTT5 Small	last	84.0
BERTimbau Large	second	36.0	GPTNeo	sum all except first	86.2
XLM-R Large	mean all	38.0	GPTNeo	sum all	86.2
XLM-R Base	mean+min+max	40.0	GPTNeo	mean all except first	86.2
XLM-R Large	last	40.8	GPTNeo	mean all	86.2
PTT5 Base	first	41.2	PTT5 Small	second	87.2
XLM-R Base	first+mean+std	41.6	GPTNeo	quantiles 25,50,75	88.0
BERTimbau Large	last	42.8	m-BERT	last	88.6
PTT5 Large	last	42.8	m-BERT	second	90.0
XLM-R Base	mean all	45.8	GPTNeo	last	94.2
PTT5 Small	first+mean+std	46.8	GPT2	second	94.8
GPT2	first+mean+std	47.6	GPTNeo	second	96.0
BERTimbau Base	second	49.4	GPT2	first	97.0
PTT5 Small	first+mean+max	50.0	GPTNeo	first	98.0

Table 5.6: Average ranks for the Transformer-based Language Models, and the average time, the reserved vRAM (GB), the allocated vRAM (GB) required to process a batch with size equal to 128.

Model	Average Rank	Time (s)	Reserved vRAM	Allocated vRAM
BERTimbau Large	9.4	1.0	1.7	1.3
PTT5 Large	9.7	1.0	3.3	2.8
BERTimbau Base	16.8	0.5	0.8	0.4
PTT5 Base	17.2	0.5	1.3	0.8
XLM-R Large	22.7	1.5	10.3	2.1
XLM-R Base	26.6	0.9	8.8	1.0
PTT5 Small	36.3	0.5	0.5	0.2
GPT2 Small	37.0	0.6	3.6	0.5
m-BERT	38.0	0.5	1.1	0.7
GPTNeo Small	45.4	0.6	3.4	0.5

compromises established between predictive performance, computational time, and computational requirements of each model. Compared to the corresponding Base versions, the Large models usually spend up to twice as much processing time and memory space. The XLM-Roberta obtained interesting results despite being multilingual, thanks to a cumbersome use of computational resources. PTT5 large is quite competitive with BERTimbau in terms of performance and processing time, but it is significantly more memory hungry (almost twice). Therefore, BERTimbau Base is the model with the most attractive performance and computational trade-off.

Figure 5.4 depicts the predictive performance for each dataset, model, and aggregation modality, restricting only to the top-seven TLMs and the five aggregation modalities common to all models for a better content visualization. Unlike BERT, other models, like XLM-Roberta Large, obtained competitive results when used the "last" token. This is somewhat expected, since one of the major differences from BERT to the remaining TLMs is the existence of a preferential token for classification (CLS).

The results discussed in this section bring us some important practical guidelines for dealing with operational scenarios with limited computational resources in this and similar NLP problems. For instance, applications with more straight restrictions on the availability of vRAM GPUs may hinder the use of Large models. In this case, a simple aggregation of the type "first + mean + std" or even "first + mean" can significantly boost the Base model performance, reducing the gap between both. Additionally, all embeddings evaluated here used numbers in 16 bits (dtype="float16" in NumPy) precision, instead of 32 bits (the default). A quick analysis of the influence of using 16 and 32 bits in model training for the BERTimbau Base and Large models with an aggregation of the type "first" has shown no

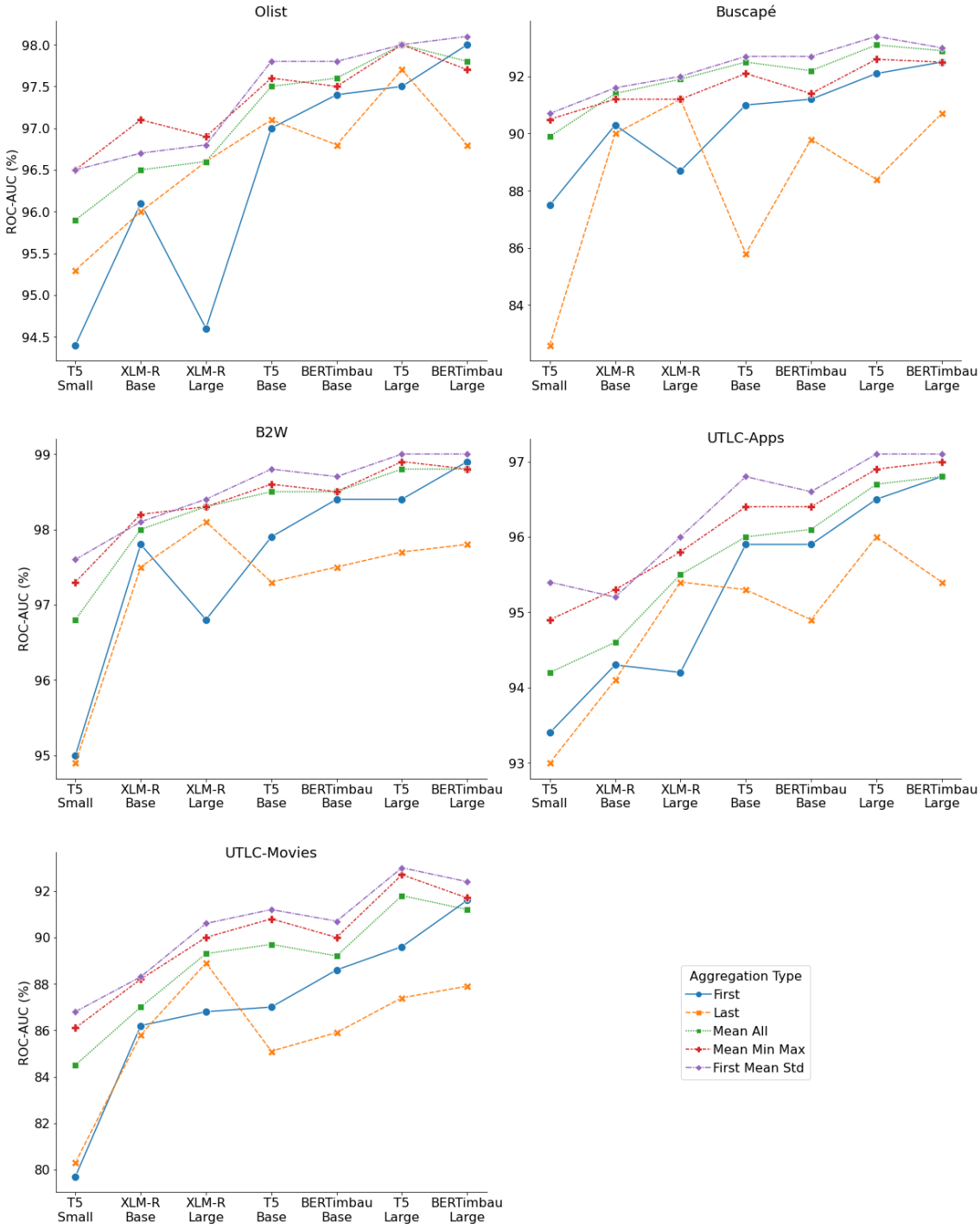

Figure 5.4: Values of the ROC-AUC (%) per Large Language Model, Embedding Type, and Dataset (see text).

significant impact on model performance. Nonetheless, the computational time and RAM usage reduction with 16 bits of precision is impressive. Therefore, a practical lesson learned here is to adopt numbers with 16 bits of precision when dealing with pre-trained BERT embedding for text classification.

5.5 Fine-tuned Large Language Models

Table 5.7 exhibits the results for the fine-tuned BERT models, reproducing those from the feature-based BERT with the best aggregation type ("first + mean + std") to make easy the subsequent analysis. The fine-tuning process allowed the new models to surpass all the pre-trained and baseline alternatives for all datasets. The *UTLC-Movies* dataset was the one that most benefited from the fine-tuning process. Moreover, for this dataset, the BERT models can only surpass the TF-IDF baseline when fine-tuning is adopted, as further discussed in the next section. The BERT model that most increased its performance with the fine-tuning was the m-BERT, although it did not surpass the BERTimbau performance after retraining. Therefore, the large dataset used to pre-train BERTimbau seems to contribute to its significant predictive power.

Table 5.7: Values of the ROC-AUC (%) for the Fine-tuned BERT models, compared to the Feature-based BERT with the aggregation modality "first+mean+std".

LLM	Configuration	Olist	Buscapé	B2W	UTLC Apps	UTLC Movies
m-BERT	first+mean+std	97.0	90.0	97.3	94.7	84.9
	finetuned	97.6	91.8	98.6	97.4	94.1
	delta	0.6	1.8	1.3	2.7	9.2
BERTimbau Base	first+mean+std	97.8	92.7	98.7	96.6	90.7
	finetuned	98.5	93.4	99.2	97.9	95.6
	delta	0.7	0.7	0.5	1.3	4.9
BERTimbau Large	first+mean+std	98.1	93.0	99.0	97.1	92.4
	finetuned	98.6	94.1	99.3	97.9	95.8
	delta	0.5	1.1	0.3	0.8	3.4

We have also evaluated how these fine-tuned models behave in other contextual settings. For this analysis, the five datasets considered in this work were explored in a cross-predictive performance experiment, i.e., each model fine-tuned with a particular dataset was evaluated with instances from another one. The cross-model comparisons were restricted to the BERTimbau Large model, considering the "first + mean + std" embedding. Table 5.8 summarizes the results. For all cases, fine-tuning the models for each dataset resulted in higher performance, as expected. The most extensive and complex content dataset, the *UTLC-Movies*, exhibited the highest gains with retraining. The general fine-tuned model obtained with the All-Combined dataset performed better than the pre-trained alternative but worse than the specialized models. Some datasets also seemed to benefit from fine-tuning, even if this procedure is conducted with another dataset. A notable example is again

the *UTLC-Movies*, to which a retraining considering the *B2W* dataset resulted in a gain of 1.3% percentage points (92.4% vs. 93.7%) as compared to the pre-trained model. However, the gains observed with the retraining process for most datasets were inferior to 1%, which might not be cost-effective for some applications.

Table 5.8: Cross-comparison of the ROC-AUC (%) values obtained with the BERTimbau pre-trained and fine-tuned models, considering different datasets. The highest performances per database are signalized by bold, while the lowest ones are underlined.

Model		Olist	Buscapé	B2W	UTLC Apps	UTLC Movies
BERTimbau fine-tuned with	Olist	**98.6**	93.4	99.1	97.4	93.2
	Buscapé	98.3	**94.1**	<u>99.0</u>	97.3	93.3
	B2W	98.2	93.4	**99.3**	97.5	93.7
	UTLC-Apps	98.4	93.8	99.1	**97.9**	92.7
	UTLC-Movies	98.2	93.7	99.1	97.4	**95.8**
	All-Combined	98.4	93.7	<u>99.0</u>	97.5	95.3
Pre-trained BERTimbau		<u>98.1</u>	<u>93.0</u>	<u>99.0</u>	<u>97.1</u>	<u>92.4</u>

5.6 Statistical Models' Comparison

To allow a more rigorous comparison of the models, we conducted a 10-fold cross-validation experiment involving the best setups per algorithm family, shown in Table 5.9, using folds already defined for each database.

Table 5.10 reports the mean and standard deviation values for Accuracy, F1-Score, and ROC-AUC derived from the ten test folds to each model and database. UTLC-Movies presents the lowest performance metrics concerning the other datasets. This may occur because it is the only dataset that does not analyze any product but movies, which tend to have greater subjectivities and nuances.

Besides, over the ROC-AUC values, we performed Friedman's Chi-Square Test [98] to assess if the differences observed in methods' performance are statistically significant, assuming a significance level of 5%. Then, we conducted a subsequent Posthoc Tukey test to identify which pairs of methods performed statistically differently (only in the cases wherein the first test signalized an overall difference). According to the first test, we have $\chi^2(4) > 33.5$ and $p < 0.001$ for all datasets, confirming that the differences observed in the methods' performance are meaningful. Based on the Posthoc Tukey test, we can establish the following relations between the methods' performance and databases in order of increasing performance. In Appendix B, we provide more details about the test results.

1. **Olist**: BoW $<$ CNN $\equiv$ LSTM $<$ FB TLM $<$ FT TLM
2. **Buscapé**: BoW $<$ CNN $=$ LSTM $\equiv$ FB TLM $<$ FT TLM
3. **B2W**: BoW $<$ CNN $\equiv$ LSTM $<$ FB TLM $<$ FT TLM
4. **UTLC-Apps**: BoW $<$ CNN $<$ LSTM $\equiv$ FB TLM $<$ FT TLM
5. **UTLC-Movies**: BoW $\equiv$ FB TLM $<$ CNN $<$ LSTM $<$ FT TLM

As expected, the Bag-of-Words is the worst, while the Fine-tuned BERT is the best for all cases. The CNN and LSTM models performed similarly for the Olist, Buscapé, and B2W databases, while the LSTM surpassed the CNN in both UTLC databases. The FB-TLM is generally equivalent or superior to the classic Deep Learning models, except to UTLC-Movies, a database with more samples, longer, and more complex sentences, to which models involving some training performed better. Considering practical applications, the BoW models, despite a lower predictive performance, are still attractive due to their implementation simplicity and low computational burden, not requiring the use of high-end GPUs. If one focuses on predictive performance, FB TLM represents an excellent intermediate alternative, as it generally performs better than the classic Deep Learning models and has a simple implementation when open-source pre-trained models are used.

Table 5.9: Hyperparameters for the best model setups for each database, in terms of the Vocabulary Size (VS), Learning Rate (LR), Dropout Rate (DR), Hidden Size (HS), and Agg Type (Aggregation Type), as other factors.

Model	Parameter	Olist	Buscapé	B2W	UTLC-Apps	UTLC-Movies
TF IDF	Feature selection	Frequency				
	VS ($\times 10^3$)	5	25	25	50	250
CNN	Filters	2,3	2,3,4,5	2,3	2,3	2,3
	Feature size	400				
	LR	10^{-3}				
	DR	0.0	0.0	0.1	0.0	0.0
	Epochs	16	7	14	4	5
LSTM	Layers	1				
	HS	128	256	128	128	128
	Pooling	Average ‖ Maximum				
	LR ($\times 10^{-3}$)	5	5	10	5	5
	DR	0.3	0.2	0.3	0.3	0.3
	Epochs	8	5	5	3	3
FB TLM	Model	BERTimbau Large				
	Agg Type	First + mean + std				
FT TLM	Model	BERTimbau Large				
	LR	2×10^{-5}				
	DR	0.1				
	Epochs	1				

Table 5.10: Mean and standard deviation of the ROC-AUC (%), Accuracy (%), and F1-Score (%) values obtained with each model and database. The highest average values are signalized in bold.

Metric	Model Family	Olist	Buscapé	B2W	UTLC Apps	UTLC Movies
Accuracy	BoW	91.8 ± 0.2	94.8 ± 0.2	94.0 ± 0.3	92.3 ± 0.1	93.1 ± 0.0
	CNN	93.3 ± 0.6	95.7 ± 0.2	94.7 ± 0.6	93.1 ± 0.6	93.7 ± 0.1
	LSTM	93.4 ± 0.6	95.5 ± 0.2	94.4 ± 1.0	93.6 ± 0.1	94.0 ± 0.1
	FB TLM	94.7 ± 0.4	95.6 ± 0.2	96.1 ± 0.2	93.6 ± 0.1	93.2 ± 0.1
	FT TLM	**95.3 ± 0.3**	**96.0 ± 0.1**	**97.0 ± 0.1**	**94.9 ± 0.1**	**95.2 ± 0.1**
F1-Score	BoW	94.2 ± 0.2	97.2 ± 0.1	95.7 ± 0.2	95.1 ± 0.0	96.2 ± 0.0
	CNN	95.2 ± 0.4	97.7 ± 0.1	96.2 ± 0.4	95.6 ± 0.4	96.5 ± 0.0
	LSTM	95.3 ± 0.4	97.6 ± 0.1	96.0 ± 0.6	95.9 ± 0.1	96.6 ± 0.1
	FB TLM	96.2 ± 0.3	97.6 ± 0.1	97.2 ± 0.1	95.9 ± 0.0	96.2 ± 0.0
	FT TLM	**96.6 ± 0.2**	**97.8 ± 0.1**	**97.8 ± 0.1**	**96.7 ± 0.1**	**97.3 ± 0.0**
ROC-AUC	BoW	96.6 ± 0.3	91.9 ± 0.6	98.1 ± 0.1	96.1 ± 0.1	92.8 ± 0.1
	CNN	97.7 ± 0.1	93.3 ± 0.6	98.8 ± 0.1	97.0 ± 0.1	93.7 ± 0.1
	LSTM	97.6 ± 0.2	93.2 ± 0.6	98.8 ± 0.1	97.3 ± 0.1	94.5 ± 0.1
	FB TLM	98.0 ± 0.1	93.0 ± 0.6	99.0 ± 0.1	97.2 ± 0.1	92.9 ± 0.1
	FT TLM	**98.4 ± 0.2**	**94.3 ± 0.4**	**99.4 ± 0.1**	**98.1 ± 0.1**	**96.1 ± 0.1**

5.7 Qualitative Analysis

In order to understand the reasons for prediction errors, we performed a qualitative analysis of positive texts predicted as negative and negative texts predicted as positive. We limited the analysis to the best model, Fine-tuned Transformer-based Language Model, and the datasets Olist (shorter sentences and product reviews) and UTLC-Movies (longer sentences and movie reviews) datasets. In the following two sections, we make specific comments about each sampled review, and, in the end, we make a general conclusion about these comments.

5.7.1 Olist

Positive reviews predicted as negative

- **Review:** "Insatisfação com a descrição do produto."

  Comment: Target feature is apparently incorrect.

- **Review:** "MEU PRODUTO VEIO ERRADO."

  Comment: Target feature is apparently incorrect.

- **Review:** "O produto chegou muito antes do previsto, fato louvável. No entanto, veio com defeito. Como devo proceder?"

  Comment: The product arrived much earlier than expected, and the customer scored the reviews as positive even though it was damaged. This duality between delivery service and product integrity may have led to the prediction error.

- **Review:** "Comprei dois produtos. Este foi entregue no prazo. O outro ainda não foi entregue e o prazo já acabou."

  Comment: The user purchased two products, where the first one arrived on time and the second one did not. This difference may have confused the model.

- **Review:** "Sem nada a acrescentar."

  Comment: The person said there is nothing to add, so a neutral review.

Negative reviews predicted as positive

- **Review:** "Recebi tido certinho, no prazo certo. A Loja esta de Parabéns em relação a logistica, porém o produto não é de qualidade."

  Comment: A situation similar to the case of the previous subsection, where the customer praises the speed of delivery, but complains about the quality of

the product. However, in this case, it seems that the quality of the product weighed more negatively on the score.

- **Review:** "achei a cor fraca"

  Comment: Target feature is apparently incorrect.

- **Review:** "O prazo de entrega foi de 28 dias. Chegou antes do prazo, mas demorou 3 semanas entre a compra e a chegada. "

  Comment: The customer praises the fast delivery, but complains about the delay of a certain stage of the logistics. This difference may have confused the model.

- **Review:** "Com um mix da Walita Philips e veio aberto. Fiz o teste e está funcionando. Achei estranho e não gostei do produto vir aberto"

  Comment: The person pointed out that the product is working, but he didn't like that the package came opened. Again, the duality may have impaired the prediction.

- **Review:** "Achei q fosse mais resistente"

  Comment: The user expected a more resistant product. The model did not get that nuance and must have focused on the word "resistente" ("resistant") as something positive.

5.7.2 UTLC-Movies

Positive reviews predicted as negative

- **Review:** *"Uma das poucas adaptações de games para telona que deram certo. (Acredito que a segunda, a primeira foi Mortal Kombat). O resto que saiba são muito ruim.Quem jogou Tomb Raider (2013), deve familiarizar com as cenas de ação, ótima aventura.Aguardando a continuação. Bom filme para você!!!!!!"*

  Comment: The user praises the movie but criticizes other movies in the same subgenre, which may have confused the model.

- **Review:** *"Apesar de frio e monótono, o filme nos convida a viver experiências que não fazem parte de um cotidiano saudável mas todos tem ciência de que elas são reais."*

  Comment: The person's overall impression is positive, but the negative caveat at the beginning of the sentence may have caused the model to mispredict.

- **Review:** *"muito terrivel, tomei um susto blaster kkkkkk recomendo"*

  Comment: The user may have used the word "terrível" ("terrible"), often used in negative contexts, to praise a horror movie.

- **Review:** "O filme trouxe uma proposta bastante boa, no entanto a execução fica a desejar a partir da segunda metade do filme. O terceiro ato, a meu ver, ficou a desejar, assemelhando-se mais a um filme de terror trash e destoando do estilo inicial do filme. Caso contrario, teria dado um 4.5, a experiência sensorial foi muito imersiva"

  Comment: The overall impression is positive, but the various negative points raised may have led the model to a negative prediction.

- **Review:** "Filme chacoalhão."

  Comment: The model may not have identified the word "chacoalhão" ("shaking"). Moreover, this word can be dubious since the user does not give more details about his impressions.

Negative reviews predicted as positive

- **Review:** "95 minutos mais demorados :("

  Comment: The model probably did not identify the emoji, which is crucial in expressing the person's feelings in this case.

- **Review:** "Filme pra assistir com os filhos, tios, avós, pais, cachorros, passarinhos e gatos.A sensação foi a mesma de ver um filme de cinema em casa, sessão da tarde. Não empolgou nada, muito fraco, o final até que foi legalzinho mas nada além disso. Clichê ao extremo mas que até vale 1 hora e pouquinho perdidas se não houver nada melhor pra fazer e você não esperar nenhuma grande reviravolta ou algo marcante."

  Comment: The person's opinion is that the film is not very good in terms of artistic quality, but it is light and good to watch with the family. This nuance may have led to the prediction error.

- **Review:** "Referências muito boas! Apesar de que achei o filme muito away."

  Comment: The general impression is that the person liked the film, despite the negative polarity. The use of an English word ("away") may have harmed the model.

- **Review:** "caaaara, toca Cold Water do Damien Rice!!!!!!! e ah sim, eu gostei do filme principalmente pq era tudo delirio mesmo, tava achando bem chatinho ate que vi o final e ai sim, fez o filme valer"

Comment: Despite the negative caveat, it seems that the user liked the film in general, leading to the belief that the negative polarity attributed to the film is not consistent with the text.

- **Review:** "É lindo, mas é um porre."

  Comment: The text contains two contrasting clauses, but the grammatical construction lets us notice that the most important opinion is present in the second sentence. The model did not capture this nuance and placed much emphasis on the word "lindo" ("beautiful"), very frequent in the context of user reviews.

5.7.3 Overall Comments

Most prediction errors come from situations where the person emphasizes positive and negative points in the same sentence. In some cases, the general impression of the user is implied. In others, there is no indication of which point is more relevant for the user to give a negative or positive rating.

The UTLC-Movies dataset, because it contains longer sentences and deeper analysis, has more of these nuances. In the Olist dataset, we found reviews that contradict the label entirely.

Finally, this qualitative analysis elucidates the limitation of reducing sentiment analysis to a binary text classification task. It is common for people to express multiple opinions in texts, and it is not always explicit what is the general feeling, positive or negative, of a text.

Chapter 6

Conclusions and Next Steps

6.1 Conclusions

In recent years, there has been a revolution in NLP, with the increasing size of models and their predictive powers. However, the Portuguese language disposes of only limited linguistic resources. In addition, previous works aiming for text classification evaluate a wide variety of datasets or considers different targets. Also, the subsets considered for model evaluation often differ, hindering comparing the results.

In this way, this work collected five open-source corpora in Brazilian Portuguese focused on sentiment analysis, cleaned and pre-processed the databases, and added columns with pre-defined partitions, making the final dataset available in a public repository. Such columns allow the reproducibility of the results in future works, encouraging researchers to use these data to evaluate new techniques.

Furthermore, this work represents a comprehensive experimental study of document embedding strategies targeting text classification, including from classical to recently proposed Transformed-based models, the latter exploiting a transfer-learning paradigm.

Regarding the state-of-the-art, Transformer-based Language Models, we analyzed three multilingual models, and seven focused on Brazilian Portuguese, a broad study that we did not find in previous works for our language. In addition, we proposed different ways to generate document embeddings using TLMs under a feature-based approach, demonstrating a significant influence on predictive performance.

Concerning the experimental results, the main paper findings can be summarized as follows. The simple TF-IDF approach outperformed more complex word vector aggregation strategies, representing an attractive compromise between complexity and performance, especially for simpler classification tasks over low-complex semantic texts.

Classical deep learning models, such as CNN and LSTM, seem to be more significantly affected by model hyperparameters only when dealing with databases having more complex and longer sentences, such as Buscapé and UTLC-Movies. As pointed out by the statistical tests, their performance tends to be superior to BoW models and inferior to Transformer-based Language Models.

Regarding the Transformer-based Language Models, increasing the model complexity usually leads to higher performance for the same architecture (Large vs. Base). Models exclusively trained with Portuguese corpora like BERTimbau and PTT5 obtained the best results. Surprisingly, the multilingual XLM-Roberta has shown to be competitive with BERTimbau and PTT5.

Moreover, all Transformer-based models have benefited from aggregating tokens when generating document embeddings. From a practical point of view, the BoW model is convenient due to implementation simplicity and reduced computational cost. In turn, the Feature-Based TLM represents a solid alternative to intermediate performance when a higher predictive performance is required.

Finally, it is worth mentioning that this master's dissertation generated two works published in international conferences and one submission to a journal in the final review phase, as described in Appendix A.

6.2 Next Steps

In future work, it would be interesting to deepen the fine-tuning studies of the TLMs, expand the results to models beyond BERT, and consider different schemas. In this work, the fine-tuning considered of all weights, inspired by the reference work of Sun et al. [96]. For instance, we could have evaluated the impact of gradually unfreezing the Transformer weights on the models' predictive performance.

In addition, this work considered four datasets with product reviews and one with movie reviews, that is, only two different domains. The addition of corpora from more varied domains, such as *ReLi* [99] (books), *TweetSentBR* [100] (posts on the social network Twitter), and *Corpus TripAdvisor* [101] (hotels), would enrich the analyzes and allow a better understanding of how the models perform in texts on different datasets.

Furthermore, this work considered a model-centric approach, analyzing models with different complexities targeting to achieve superior predictive performance. However, as we saw in the qualitative analysis section, several prediction errors involved either a clear misattribution of the label or a nuance that did not make it clear whether the review was positive or negative. A consistent target feature is fundamental for the model quality, and the work of Xu et al. [102] presents techniques to improve the quality of ground-truth labels. As most studies nowadays are model-

centric, and in the NLP field, there is a large amount of low-quality corpus available, such a study is of particular relevance.

Appendix A

Academic Publications

This appendix presents the publications produced during the development of this master's dissertation.

- Souza, F.D., Filho, J.B.O. (2021). "Sentiment Analysis on Brazilian Portuguese User Reviews". In: 2021 IEEE Latin American Conference on Computational Intelligence (LA-CCI). https://doi.org/10.1109/LA-CCI48322.2021.9769838

  Abstract: Sentiment Analysis is one of the most classical and primarily studied natural language processing tasks. This problem had a notable advance with the proposition of more complex and scalable machine learning models. Despite this progress, the Brazilian Portuguese language still disposes only of limited linguistic resources, such as datasets dedicated to sentiment classification, especially when considering the existence of predefined partitions in training, testing, and validation sets that would allow a more fair comparison of different algorithm alternatives. Motivated by these issues, this work analyzes the predictive performance of a range of document embedding strategies, assuming the polarity as the system outcome. This analysis includes five sentiment analysis datasets in Brazilian Portuguese, unified in a single dataset, and a reference partitioning in training, testing, and validation sets, both made publicly available through a digital repository. A cross-evaluation of dataset-specific models over different contexts is conducted to evaluate their generalization capabilities and the feasibility of adopting a unique model for addressing all scenarios.

- Souza, F.D., Filho, J.B.O. (2022). "BERT for Sentiment Analysis: Pre-trained and Fine-Tuned Alternatives". In: Computational Processing of the Portuguese Language. PROPOR 2022. Lecture Notes in Computer Science, vol 13208. Springer, Cham. https://doi.org/10.1007/978-3-030-98305-5_20

Abstract: BERT has revolutionized the NLP field by enabling transfer learning with large language models that can capture complex textual patterns, reaching the state-of-the-art for an expressive number of NLP applications. For text classification tasks, BERT has already been extensively explored. However, aspects like how to better cope with the different embeddings provided by the BERT output layer and the usage of language-specific instead of multilingual models are not well studied in the literature, especially for the Brazilian Portuguese language. The purpose of this article is to conduct an extensive experimental study regarding different strategies for aggregating the features produced in the BERT output layer, with a focus on the sentiment analysis task. The experiments include BERT models trained with Brazilian Portuguese corpora and the multilingual version, contemplating multiple aggregation strategies and open-source datasets with predefined training, validation, and test partitions to facilitate the reproducibility of the results. BERT achieved the highest ROC-AUC values for the majority of cases as compared to TF-IDF. Nonetheless, TF-IDF represents a good trade-off between the predictive performance and computational cost.

- Souza, F.D., Filho, J.B.O. (2022). Embedding generation for text classification of Brazilian Portuguese user reviews: from bag-of-words to transformers. Neural Computing & Applications. `https://doi.org/10.1007/s00521-022-08068-6`.

  Abstract: Text classification is a Natural Language Processing (NLP) task relevant to many commercial applications, like e-commerce and customer service. Naturally, classifying such excerpts accurately often represents a challenge, due to intrinsic language aspects, like irony and nuance. To accomplish this task, one must provide a robust numerical representation for documents, a process known as embedding. Embedding represents a key NLP field nowadays, having faced a significant advance in the last decade, especially after the introduction of the word-to-vector concept and the popularization of Deep Learning models for solving NLP tasks, including Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Transformer-based Language Models (TLMs). Despite the impressive achievements in this field, the literature coverage regarding generating embeddings for Brazilian Portuguese texts is scarce, especially when considering commercial user reviews. Therefore, this work aims to provide a comprehensive experimental study of embedding approaches targeting a binary sentiment classification of user reviews in Brazilian Portuguese. This study includes from classical (Bag-of-Words) to state-of-the-art (Transformer-based) NLP models. The methods

are evaluated with five open-source databases with predefined data partitions made available in an open digital repository to encourage reproducibility. The Fine-tuned TLMs achieved the best results for all cases, being followed by the Feature-based TLM, LSTM, and CNN, with alternate ranks, depending on the database under analysis.

Appendix B

Statistical Models' Comparison

Regarding the results in Section 5.6, Table B.1 presents the ROC-AUC values for each dataset, model, and fold. Over these data, we performed Friedman's Chi-Square Test using the implementation from *Python*'s library *scipy*[1]. Table B.2 depicts the Friedman test statistics and p-values for each dataset.

Assuming a significance level of 5%, we verified that there was a significant difference among the ROC-AUC values obtained with each model in all datasets. Then, we conducted a Posthoc Tukey test, using the implementation from *Python*'s library *statsmodels*[2], to identify which pairs of methods performed statistically differently. Tables B.3, B.4, B.5, B.6, and B.7 show the mean difference of ROC-AUC between each pair of models and the adjusted p-value. With these results, we obtained the predictive performance ordering for each dataset.

[1] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.friedmanchisquare.html`
[2] `https://statsmodels.org/dev/generated/statsmodels.stats.multicomp.pairwise_tukeyhsd.html`

Table B.1: ROC-AUC (%) values obtained with each model, database and partition (from fold 1 to 10).

Model	Dataset	1	2	3	4	5	6	7	8	9	10
Olist	TF-IDF	96.6	96.3	96.1	96.6	97.0	96.5	96.9	96.8	96.3	96.5
	CNN	97.6	97.6	97.5	97.8	97.7	97.7	97.8	97.7	97.4	97.8
	LSTM	97.5	97.9	97.7	97.8	97.6	97.6	97.4	97.7	97.4	98.0
	FB TLM	98.2	98.1	97.8	98.2	98.0	98.2	97.9	98.0	97.8	98.1
	FT TLM	98.4	98.6	98.4	98.1	98.4	98.3	98.2	98.6	98.3	98.6
Buscapé	TF-IDF	91.3	92.5	92.7	91.1	91.8	91.9	91.0	92.6	92.3	91.6
	CNN	92.2	93.7	93.9	92.6	93.5	93.3	92.5	94.1	93.8	93.1
	LSTM	92.1	93.7	93.8	92.6	93.1	93.2	92.8	93.9	93.8	93.3
	FB TLM	92.1	93.1	93.0	92.2	93.3	92.9	92.6	94.6	92.8	93.2
	FT TLM	93.6	94.2	94.6	93.7	94.5	94.0	94.0	95.0	94.7	94.4
B2W	TF-IDF	98.1	98.3	97.9	98.2	98.2	98.0	98.0	98.3	98.2	98.1
	CNN	98.8	98.9	98.7	98.8	98.9	98.7	98.7	98.9	98.9	98.8
	LSTM	98.9	98.9	98.7	98.9	98.7	98.7	98.7	98.7	98.8	98.8
	FB TLM	99.0	99.1	98.9	99.1	99.2	99.0	99.0	99.0	99.0	99.0
	FT TLM	99.3	99.4	99.3	99.4	99.5	99.3	99.3	99.4	99.4	99.3
UTLC-Apps	TF-IDF	96.1	96.1	96.1	96.3	96.1	96.0	96.0	96.2	96.1	95.9
	CNN	96.9	97.1	96.9	97.1	97.1	97.0	96.9	97.1	97.0	96.8
	LSTM	97.3	97.3	97.2	97.4	97.3	97.2	97.2	97.3	97.2	97.2
	FB TLM	97.1	97.3	97.1	97.3	97.3	97.2	97.1	97.2	97.2	97.1
	FT TLM	98.1	98.1	98.0	98.1	98.1	98.0	98.0	98.1	98.1	98.0
UTLC-Movies	TF-IDF	92.8	92.8	92.9	93.0	92.8	92.7	92.8	92.8	92.8	92.9
	CNN	93.7	93.7	93.9	93.7	93.7	93.6	93.6	93.8	93.8	93.8
	LSTM	94.5	94.6	94.7	94.6	94.6	94.4	94.4	94.5	94.6	94.5
	FB TLM	92.9	93.0	93.1	92.8	93.0	92.6	92.8	92.9	93.0	92.8
	FT TLM	96.2	96.1	96.2	96.2	96.1	96.0	96.1	96.1	96.2	96.2

Table B.2: Friedman Test statistics and p-values obtained with each dataset.

Dataset	Test statistic (χ^2)	P-value
Olist	37.36	1.5e-07
Buscapé	33.52	9.3e-07
B2W	38.00	1.1-07
UTLC-Apps	40.00	4.3-08
UTLC-Movies	38.32	9.6-08

Table B.3: Mean difference in ROC-AUC (%) and Adjusted p-value (%) obtained through Tukey Test for the dataset Olist and each pairs of models.

Dataset	Models	Mean difference ROC-AUC	Adjusted p-value
Olist	FB TLM - CNN	0.4	0.1
	FT TLM - CNN	0.7	0.1
	LSTM - CNN	-0.0	90.0
	TF-IDF - CNN	-1.1	0.1
	FT TLM - FB TLM	0.4	0.1
	LSTM - FB TLM	-0.4	0.1
	TF-IDF - FB TLM	-1.5	0.1
	LSTM - FT TLM	-0.7	0.1
	TF-IDF - FT TLM	-1.8	0.1
	TF-IDF - LSTM	-1.1	0.1

Table B.4: Mean difference in ROC-AUC (%) and Adjusted p-value (%) obtained through Tukey Test for the dataset Buscapé and each pairs of models.

Dataset	Models	Mean difference ROC-AUC	Adjusted p-value
Buscapé	FB TLM - CNN	-0.3	78.4
	FT TLM - CNN	1.0	0.4
	LSTM - CNN	-0.0	90.0
	TF-IDF - CNN	-1.4	0.1
	FT TLM - FB TLM	1.3	0.1
	LSTM - FB TLM	0.3	86.5
	TF-IDF - FB TLM	-1.1	0.2
	LSTM - FT TLM	-1.1	0.3
	TF-IDF - FT TLM	-2.4	0.1
	TF-IDF - LSTM	-1.4	0.1

Table B.5: Mean difference in ROC-AUC (%) and Adjusted p-value (%) obtained through Tukey Test for the dataset B2W and each pairs of models.

Dataset	Models	Mean difference ROC-AUC	Adjusted p-value
B2W	FB TLM - CNN	0.2	0.1
	FT TLM - CNN	0.5	0.1
	LSTM - CNN	-0.0	75.5
	TF-IDF - CNN	-0.7	0.1
	FT TLM - FB TLM	0.3	0.1
	LSTM - FB TLM	-0.3	0.1
	TF-IDF - FB TLM	-0.9	0.1
	LSTM - FT TLM	-0.6	0.1
	TF-IDF - FT TLM	-1.2	0.1
	TF-IDF - LSTM	-0.6	0.1

Table B.6: Mean difference in ROC-AUC (%) and Adjusted p-value (%) obtained through Tukey Test for the dataset UTLC-Apps and each pairs of models.

Dataset	Models	Mean difference ROC-AUC	Adjusted p-value
UTLC-Apps	FB TLM - CNN	0.2	0.1
	FT TLM - CNN	1.1	0.1
	LSTM - CNN	0.3	0.1
	TF-IDF - CNN	-0.9	0.1
	FT TLM - FB TLM	0.9	0.1
	LSTM - FB TLM	0.1	34.0
	TF-IDF - FB TLM	-1.1	0.1
	LSTM - FT TLM	-0.8	0.1
	TF-IDF - FT TLM	-2.0	0.1
	TF-IDF - LSTM	-1.2	0.1

Table B.7: Mean difference in ROC-AUC (%) and Adjusted p-value (%) obtained through Tukey Test for the dataset UTLC-Movies and each pairs of models.

Dataset	Models	Mean difference ROC-AUC	Adjusted p-value
UTLC-Movies	FB TLM - CNN	-0.8	0.1
	FT TLM - CNN	2.4	0.1
	LSTM - CNN	0.8	0.1
	TF-IDF - CNN	-0.9	0.1
	FT TLM - FB TLM	3.3	0.1
	LSTM - FB TLM	1.6	0.1
	TF-IDF - FB TLM	-0.1	53.8
	LSTM - FT TLM	-1.6	0.1
	TF-IDF - FT TLM	-3.3	0.1
	TF-IDF - LSTM	-1.7	0.1

References

[1] LIU, B. "Sentiment Analysis and Opinion Mining", *Synthesis Lectures on Human Language Technologies*, v. 5, n. 1, 2012. Available at: <http://dx.doi.org/10.2200/S00416ED1V01Y201204HLT016>.

[2] RAO, D., MCMAHAN, B. *Natural Language Processing with PyTorch: Build Intelligent Language Applications Using Deep Learning*. USA, O'Reilly Media, 2019. ISBN: 9781491978184. Available at: <https://books.google.com.br/books?id=NsuEDwAAQBAJ>.

[3] EISENSTEIN, J. *Introduction to Natural Language Processing*. Adaptive Computation and Machine Learning series. USA, MIT Press, 2019. ISBN: 9780262042840.

[4] GOLDBERG, Y. *Neural Network Methods for Natural Language Processing*, v. 37, *Synthesis Lectures on Human Language Technologies*. San Rafael, CA, Morgan & Claypool, 2017. ISBN: 978-1-62705-298-6. doi: 10.2200/S00762ED1V01Y201703HLT037.

[5] MINAEE, S., KALCHBRENNER, N., CAMBRIA, E., et al. "Deep Learning Based Text Classification: A Comprehensive Review". 2020. Available at: <https://arxiv.org/abs/2004.03705>.

[6] LI, Q., PENG, H., LI, J., et al. "A Survey on Text Classification: From Shallow to Deep Learning". 2020. Available at: <https://arxiv.org/abs/2008.00364>.

[7] KOWSARI, MEIMANDI, J., HEIDARYSAFA, et al. "Text Classification Algorithms: A Survey", *Information*, v. 10, n. 4, 2019. ISSN: 2078-2489. doi: 10.3390/info10040150.

[8] PEREIRA, D. A. "A survey of sentiment analysis in the Portuguese language", *Artificial Intelligence Review*, v. 54, n. 2, 2020. doi: 10.1007/s10462-020-09870-1.

[9] ICMC-USP. "Opinion Mining for Portuguese". 2019. Available at: <https://sites.google.com/icmc.usp.br/opinando/pÃągina-inicial>.

[10] VALDIVIA, A., OTHERS. "Consensus vote models for detecting and filtering neutrality in Sentiment Analysis", *Information Fusion*, v. 44, 2018. ISSN: 1566-2535. doi: https://doi.org/10.1016/j.inffus.2018.03.007. Available at: <https://www.sciencedirect.com/science/article/pii/S1566253517306590>.

[11] KOPPEL, M., SCHLER, J. "The importance of neutral examples for learning sentiment", *Workshop on the Analysis of Informal and Formal Information Exchange During Negotiations (FINEXIN)*, 2005.

[12] MEDHAT, W., HASSAN, A., KORASHY, H. "Sentiment analysis algorithms and applications: A survey", *Ain Shams Engineering Journal*, v. 5, n. 4, pp. 1093–1113, 2014. ISSN: 2090-4479. doi: https://doi.org/10.1016/j.asej.2014.04.011. Available at: <https://www.sciencedirect.com/science/article/pii/S2090447914000550>.

[13] KRIZHEVSKY, A., SUTSKEVER, I., HINTON, G. E. "ImageNet Classification with Deep Convolutional Neural Networks". In: Pereira, F., Burges, C. J. C., Bottou, L., et al. (Eds.), *Advances in Neural Information Processing Systems 25*, Curran Associates, Inc., pp. 1097–1105, 2012.

[14] SPARCK JONES, K. "A Statistical Interpretation of Term Specificity and Its Application in Retrieval". In: *Document Retrieval Systems*, p. 132–142, GBR, Taylor Graham Publishing, 1988. ISBN: 0947568212.

[15] RAJARAMAN, A., ULLMAN, J. D. "Data Mining", *Mining of Massive Datasets*, p. 1–17, 2011. doi: 10.1017/CBO9781139058452.002.

[16] MIHI, S., OTHERS. "A Comparative Study of Feature Selection Methods for Informal Arabic", *Innovation in Information Systems and Technologies to Support Learning Research*, 2020.

[17] LANDAUER, T. K., FOLTZ, P. W., LAHAM, D. "An introduction to latent semantic analysis", *Discourse Processes*, v. 25, n. 2-3, pp. 259–284, 1998. Available at: <https://doi.org/10.1080/01638539809545028>.

[18] ZHANG, X., ZHAO, J., LECUN, Y. "Character-level Convolutional Networks for Text Classification", *Advances in Neural Information Processing Systems*, 2016. Available at: <https://proceedings.neurips.cc/paper/2015/file/250cf8b51c773f3f8dc8b4be867a9a02-Paper.pdf>.

[19] CHAUBARD, F., FANG, M., GENTHIAL, G., et al. "Word Vectors I: Introduction, SVD and Word2Vec". 2019. Available at: <http://web.stanford.edu/class/cs224n/readings/cs224n-2019-notes01-wordvecs1.pdf>.

[20] COLLOBERT, R., WESTON, J., BOTTOU, L., et al. "Natural Language Processing (Almost) from Scratch", *J. Mach. Learn. Res.*, v. 999888, pp. 2493–2537, nov 2011. ISSN: 1532-4435. Available at: <http://dl.acm.org/citation.cfm?id=2078183.2078186>.

[21] MIKOLOV, T., CHEN, K., CORRADO, G., et al. "Efficient Estimation of Word Representations in Vector Space", *1st International Conference on Learning Representations, ICLR*, 2013.

[22] PENNINGTON, J., SOCHER, R., MANNING, C. D. "Glove: Global vectors for word representation", *EMNLP*, 2014.

[23] MUNDRA, R., PENG, E., SOCHER, R., et al. "Word Vectors II: GloVe, Evaluation and Training". 2019. Available at: <http://web.stanford.edu/class/cs224n/readings/cs224n-2019-notes02-wordvecs2.pdf>.

[24] BOJANOWSKI, P., GRAVE, E., JOULIN, A., et al. "Enriching Word Vectors with Subword Information", *CoRR*, v. abs/1607.04606, 2016. Available at: <http://arxiv.org/abs/1607.04606>.

[25] LEVY, O., GOLDBERG, Y. "Neural Word Embedding as Implicit Matrix Factorization", *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, 2014.

[26] SINGH, P., MUKERJEE, A. "Words are not Equal: Graded Weighting Model for Building Composite Document Vectors". In: *Proceedings of the 12th International Conference on Natural Language Processing*, pp. 11–19, Trivandrum, India, dec 2015. NLP Association of India. Available at: <https://aclanthology.org/W15-5903>.

[27] ARORA, S., LIANG, Y., MA, T. "A Simple but Tough-to-Beat Baseline for Sentence Embeddings". In: *International Conference on Learning Representations*, 2017. Available at: <https://openreview.net/forum?id=SyK00v5xx>.

[28] GUPTA, V., KARNICK, H., BANSAL, A., et al. "Product Classification in E-Commerce using Distributional Semantics". In: *Proceedings of COLING 2016, the 26th International Conference on Computational*

Linguistics: Technical Papers, pp. 536–546, Osaka, Japan, dec 2016. The COLING 2016 Organizing Committee. Available at: <https://aclanthology.org/C16-1052>.

[29] MEKALA, D., GUPTA, V., PARANJAPE, B., et al. "SCDV : Sparse Composite Document Vectors using soft clustering over distributional representations". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 659–669, Copenhagen, Denmark, sep 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1069. Available at: <https://aclanthology.org/D17-1069>.

[30] GUPTA, V., SAW, A., NOKHIZ, P., et al. "P-SIF: Document Embeddings Using Partition Averaging", *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020. Available at: <https://aaai.org/ojs/index.php/AAAI/article/view/6292>.

[31] LE, Q., MIKOLOV, T. "Distributed Representations of Sentences and Documents". In: Xing, E. P., Jebara, T. (Eds.), *Proceedings of the 31st International Conference on Machine Learning*, Proceedings of Machine Learning Research, pp. 1188–1196, Bejing, China, 22–24 Jun 2014. PMLR. Available at: <https://proceedings.mlr.press/v32/le14.html>.

[32] LIU, P., QIU, X., HUANG, X. "Learning Context-Sensitive Word Embeddings with Neural Tensor Skip-Gram Model", *Proceedings of the 24th International Conference on Artificial Intelligence*, 2015.

[33] ELMAN, J. L. "Finding structure in time", *Cognitive Science*, v. 14, pp. 213–252, 1990.

[34] MOHAMMADI, M., MUNDRA, R., SOCHER, R., et al. "Language Models, RNN, GRU and LSTM". 2019. Available at: <http://web.stanford.edu/class/cs224n/readings/cs224n-2019-notes05-LM_RNN.pdf>.

[35] CHOLLET, F. *Deep Learning with Python*. 1st ed. USA, Manning Publications Co., 2017. ISBN: 1617294438.

[36] HOCHREITER, S., SCHMIDHUBER, J. "Long Short-Term Memory", *Neural Computation*, v. 9, n. 8, pp. 1735–1780, 1997.

[37] LIU, P., QIU, X., HUANG, X. "Recurrent Neural Network for Text Classification with Multi-Task Learning", *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 2016.

[38] NOWAK, J., TASPINAR, A., SCHERER, R. "LSTM Recurrent Neural Networks for Short Text and Sentiment Classification", *Artificial Intelligence and Soft Computing*, 2017.

[39] WANG, J.-H., LIU, T.-W., LUO, X., et al. "An LSTM Approach to Short Text Sentiment Classification with Word Embeddings". In: *Proceedings of the 30th Conference on Computational Linguistics and Speech Processing (ROCLING 2018)*, pp. 214–223, Hsinchu, Taiwan, oct 2018. The Association for Computational Linguistics and Chinese Language Processing (ACLCLP). Available at: <https://aclanthology.org/O18-1021>.

[40] LECUN, Y., BOTTOU, L., BENGIO, Y., et al. "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, v. 86, n. 11, pp. 2278–2324, 1998. doi: 10.1109/5.726791.

[41] MANNING, C. "Lecture 16: ConvNets for NLP and Tree Recursive Neural Networks". 2022. Available at: <http://web.stanford.edu/class/cs224n/slides/cs224n-2022-lecture16-CNN-TreeRNN.pdf>.

[42] KIM, Y. "Convolutional Neural Networks for sentence classification". In: *Proceedings of the 2014 EMNLP*, Doha, Qatar, oct 2014. Association for Computational Linguistics. Available at: <https://aclanthology.org/D14-1181>.

[43] ZHANG, Y., WALLACE, B. "A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification". In: *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 253–263, Taipei, Taiwan, nov 2017. Asian Federation of Natural Language Processing. Available at: <https://aclanthology.org/I17-1026>.

[44] CONNEAU, A., SCHWENK, H., BARRAULT, L., et al. "Very Deep Convolutional Networks for Text Classification", *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, apr 2017. Available at: <https://aclanthology.org/E17-1104>.

[45] ZHOU, C., SUN, C., LIU, Z., et al. "A C-LSTM Neural Network for Text Classification". 2015. Available at: <https://arxiv.org/abs/1511.08630>.

[46] LEE, J. Y., DERNONCOURT, F. "Sequential Short-Text Classification with Recurrent and Convolutional Neural Networks", *CoRR*,

v. abs/1603.03827, 2016. Available at: <http://arxiv.org/abs/1603.03827>.

[47] BRADBURY, J., MERITY, S., XIONG, C., et al. "Quasi-Recurrent Neural Networks", *International Conference on Learning Representations*, 2017. Available at: <https://openreview.net/forum?id=H1zJ-v5xl>.

[48] WU, Z., PAN, S., CHEN, F., et al. "A Comprehensive Survey on Graph Neural Networks", *IEEE Transactions on Neural Networks and Learning Systems*, v. 32, n. 1, pp. 4–24, Jan 2021. ISSN: 2162-2388. doi: 10.1109/tnnls.2020.2978386. Available at: <http://dx.doi.org/10.1109/TNNLS.2020.2978386>.

[49] GORI, M., MONFARDINI, G., SCARSELLI, F. "A new model for learning in graph domains", *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, 2005. doi: 10.1109/IJCNN.2005.1555942.

[50] SCARSELLI, F., GORI, M., TSOI, A. C., et al. "The Graph Neural Network Model", *IEEE Transactions on Neural Networks*, v. 20, n. 1, pp. 61–80, 2009. doi: 10.1109/TNN.2008.2005605.

[51] PENG, H., LI, J., HE, Y., et al. "Large-Scale Hierarchical Text Classification with Recursively Regularized Deep Graph-CNN", *Proceedings of the 2018 World Wide Web Conference*, 2018. doi: 10.1145/3178876.3186005.

[52] YAO, L., MAO, C., LUO, Y. "Graph Convolutional Networks for Text Classification", *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, 2019. Available at: <https://doi.org/10.1609/aaai.v33i01.33017370>.

[53] GENTHIAL, G., LIU, L., OSHRI, B., et al. "Neural Machine Translation, Seq2seq and Attention". 2019. Available at: <http://web.stanford.edu/class/cs224n/readings/cs224n-2019-notes06-NMT_seq2seq_attention.pdf>.

[54] BAHDANAU, D., CHO, K., BENGIO, Y. "Neural Machine Translation by Jointly Learning to Align and Translate", *3rd International Conference on Learning Representations, ICLR 2015*, 2015. Available at: <http://arxiv.org/abs/1409.0473>.

[55] CHAUDHARI, S., MITHAL, V., POLATKAN, G., et al. "An Attentive Survey of Attention Models". 2019. Available at: <https://arxiv.org/abs/1904.02874>.

[56] MANNING, C. "Lecture 7: Machine Translation, Sequence-to-Sequence and Attention". 2021. Available at: <http://web.stanford.edu/class/cs224n/slides/cs224n-2021-lecture07-nmt.pdf>.

[57] ARIK, S. O., PFISTER, T. "TabNet: Attentive Interpretable Tabular Learning", *Proceedings of the AAAI Conference on Artificial Intelligence*, v. 35, n. 8, 2021. doi: 10.1609/aaai.v35i8.16826. Available at: <https://ojs.aaai.org/index.php/AAAI/article/view/16826>.

[58] WANG, Y., HUANG, M., ZHU, X., et al. "Attention-based LSTM for Aspect-level Sentiment Classification". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 606–615, nov 2016. Available at: <https://aclanthology.org/D16-1058>.

[59] LIU, G., GUO, J. "Bidirectional LSTM with attention mechanism and convolutional layer for text classification", *Neurocomputing*, v. 337, pp. 325–338, 2019. Available at: <https://www.sciencedirect.com/science/article/pii/S0925231219301067>.

[60] YANG, Z., YANG, D., DYER, C., et al. "Hierarchical Attention Networks for Document Classification". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1480–1489, San Diego, California, jun 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1174. Available at: <https://aclanthology.org/N16-1174>.

[61] MIN, B., ROSS, H., SULEM, E., et al. "Recent Advances in Natural Language Processing via Large Pre-Trained Language Models: A Survey", 2021. Available at: <https://arxiv.org/abs/2111.01243>.

[62] VASWANI, A., SHAZEER, N., PARMAR, N., et al. "Attention is All You Need", *Proceedings of the 31st International Conference on Neural Information Processing Systems*, p. 6000–6010, 2017.

[63] RUDER, S. "NLP's ImageNet moment has arrived". https://ruder.io/nlp-imagenet/, 2018. Acesso em: November 8th, 2022.

[64] ROTHMAN, D. *Transformers for Natural Language Processing: Build Innovative Deep Neural Network Architectures for NLP with Python, PyTorch, TensorFlow, BERT, RoBERTa, and More*. Packt Publishing, 2021. ISBN: 9781800565791. Available at: <https://books.google.com.br/books?id=Ua03zgEACAAJ>.

[65] ALAMMAR, J. "The Illustrated Transformer". 2018. Available at: <http://jalammar.github.io/illustrated-transformer/>.

[66] PETERS, M. E., NEUMANN, M., IYYER, M., et al. "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)*, New Orleans, Louisiana, jun 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. Available at: <https://aclanthology.org/N18-1202>.

[67] KHAN, S., NASEER, M., HAYAT, M., et al. "Transformers in Vision: A Survey", *ACM Computing Surveys*, v. 54, n. 10s, pp. 1–41, 2022. Available at: <https://doi.org/10.1145%2F3505244>.

[68] RADFORD, A., NARASIMHAN, K. "Improving Language Understanding by Generative Pre-Training", 2018. Available at: <https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf>.

[69] RADFORD, A., WU, J., CHILD, R., et al. "Language Models are Unsupervised Multitask Learners", 2019. Available at: <https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf>.

[70] BROWN, T., MANN, B., RYDER, N., et al. "Language Models are Few-Shot Learners", *Advances in Neural Information Processing Systems*, pp. 1877–1901, 2020. Available at: <https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf>.

[71] DEVLIN, J., CHANG, M.-W., LEE, K., et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, jun 2019. Available at: <https://aclanthology.org/N19-1423>.

[72] LEWIS, M., LIU, Y., GOYAL, N., et al. "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7871–7880, Online, jul 2020. Association for Computational Linguistics. Available at: <https://aclanthology.org/2020.acl-main.703>.

[73] LIU, Y., OTT, M., GOYAL, N., et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". 2019. Available at: <https://arxiv.org/abs/1907.11692>.

[74] RAFFEL, C., SHAZEER, N., ROBERTS, A., et al. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer", *Journal of Machine Learning Research*, v. 21, n. 140, pp. 1–67, 2020. Available at: <http://jmlr.org/papers/v21/20-074.html>.

[75] ALVI, A., KHARYA, P. "Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, the World's Largest and Most Powerful Generative Language Model". https://www.microsoft.com/en-us/research/blog/using-deepspeed-and-megatron-to-train-megatron-turing-nlg-530b-the-worlds-largest-and-most-powerful-generative-language-model/, Oct 2021.

[76] CONNEAU, A., KHANDELWAL, K., GOYAL, N., et al. "Unsupervised Cross-lingual Representation Learning at Scale". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8440–8451, jul 2020. Available at: <https://aclanthology.org/2020.acl-main.747>.

[77] BENDER, E. M., GEBRU, T., MCMILLAN-MAJOR, A., et al. "On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?" In: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, p. 610–623, New York, NY, USA, 2021. Association for Computing Machinery. ISBN: 9781450383097. doi: 10.1145/3442188.3445922. Available at: <https://doi.org/10.1145/3442188.3445922>.

[78] SOUZA, F., NOGUEIRA, R., LOTUFO, R. "BERTimbau: pretrained BERT models for Brazilian Portuguese", *9th Brazilian Conference on Intelligent Systems, BRACIS, Rio Grande do Sul, Brazil, October 20-23 (to appear)*, 2020.

[79] WAGNER FILHO, J. A., WILKENS, R., IDIART, M., et al. "The brWaC Corpus: A New Open Resource for Brazilian Portuguese". In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, may 2018. European Language Resources Association (ELRA). Available at: <https://aclanthology.org/L18-1686>.

[80] LOPES, E., CORREA, U., FREITAS, L. "Exploring BERT for Aspect Extraction in Portuguese Language", *The International FLAIRS Conference Proceedings*, v. 34, Apr. 2021. doi: 10.32473/flairs.v34i1.128357. Available at: <https://journals.flvc.org/FLAIRS/article/view/128357>.

[81] LEITE, J. A., SILVA, D. F., BONTCHEVA, K., et al. "Toxic Language Detection in Social Media for Brazilian Portuguese: New Dataset and Multilingual Analysis", *CoRR*, v. abs/2010.04543, 2020. Available at: <https://arxiv.org/abs/2010.04543>.

[82] JIANG, S., CHEN, C., LIN, N., et al. "Irony Detection in the Portuguese Language using BERT", *Iberian Languages Evaluation Forum 2021*, 2021.

[83] NETO, A., OSTI, B., AZEVEDO, C., et al. "SiDi-NLP-Team at IDPT2021: Irony Detection in Portuguese 2021", *Iberian Languages Evaluation Forum 2021*, 2021. Available at: <http://ceur-ws.org/Vol-2943/idpt_paper6.pdf>.

[84] CARRICO, N., QUARESMA, P. "Sentence Embeddings and Sentence Similarity for Portuguese FAQs". In: *IberSPEECH 2021*, pp. 200–204, 03 2021.

[85] FEIJO, D. D. V., MOREIRA, V. P. "Mono vs Multilingual Transformer-based Models: a Comparison across Several Language Tasks". 2020. Available at: <https://arxiv.org/abs/2007.09757>.

[86] FINARDI, P., VIEGAS, J. D., FERREIRA, G. T., et al. "BERTaú: Itaú BERT for digital customer service". 2021. Available at: <https://arxiv.org/abs/2101.12015>.

[87] CARMO, D., PIAU, M., CAMPIOTTI, I., et al. "PTT5: Pretraining and validating the T5 model on Brazilian Portuguese data". 2020. Available at: <https://arxiv.org/abs/2008.09144>.

[88] OLIST. "Brazilian E-Commerce Public Dataset by Olist". Nov 2018. Available at: <https://www.kaggle.com/olistbr/brazilian-ecommerce>.

[89] REAL, L., OSHIRO, M., MAFRA, A. "B2W-Reviews01 - An open product reviews corpus", *STIL - Symposium in Information and Human Language Technology*, 2019. Available at: <https://github.com/b2wdigital/b2w-reviews01>.

[90] HARTMANN, N., AVANÇO, L., BALAGE, P., et al. "A Large Corpus of Product Reviews in Portuguese: Tackling Out-Of-Vocabulary Words". In: *Proceedings of the Ninth International Conference on Language Resources*

and Evaluation (LREC'14), Reykjavik, Iceland, may 2014. European Language Resources Association (ELRA).

[91] SOUSA, R. F. D., BRUM, H. B., NUNES, M. D. G. V. "A bunch of helpfulness and sentiment corpora in brazilian portuguese", *Symposium in Information and Human Language Technology - STIL*, 2019.

[92] Núcleo Interinstitucional de Linguística Computacional. "Repositório de Word Embeddings do NILC". 2017. Available at: <`http://www.nilc.icmc.usp.br/embeddings`>.

[93] WOLF, T., DEBUT, L., SANH, V., et al. "Transformers: State-of-the-Art Natural Language Processing". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, oct 2020. Available at: <`https://aclanthology.org/2020.emnlp-demos.6`>.

[94] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., et al. "Scikit-learn: Machine Learning in Python", *Journal of Machine Learning Research*, v. 12, pp. 2825–2830, 2011.

[95] JAY ALAMMAR. "A visual guide to using BERT for the first time". 2019. Available at: <`https://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/`>.

[96] SUN, C., QIU, X., XU, Y., et al. "How to Fine-Tune BERT for Text Classification?" In: Sun, M., Huang, X., Ji, H., et al. (Eds.), *Chinese Computational Linguistics*, pp. 194–206. Springer International Publishing, 2019.

[97] KE, G., MENG, Q., FINLEY, T., et al. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree", *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.

[98] DEMŠAR, J. "Statistical Comparisons of Classifiers over Multiple Data Sets", *JMLR*, v. 7, pp. 1–30, dec 2006. ISSN: 1532-4435.

[99] FREITAS, C., MOTTA, E., MILIDIÚ, R., et al. "Sparkling Vampire... lol! Annotating Opinions in a Book Review Corpus". pp. 128–146, 01 2014. ISBN: 978-1443853774.

[100] BRUM, H., DAS GRAÇAS VOLPE NUNES, M. "Building a Sentiment Corpus of Tweets in Brazilian Portuguese". In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC*

2018), Miyazaki, Japan, May 7-12, 2018 2018. European Language Resources Association (ELRA). ISBN: 979-10-95546-00-9.

[101] DE SOUZA, J. G. R., DE PAIVA OLIVEIRA, A., MOREIRA, A. "Development of a Brazilian Portuguese Hotel's Reviews Corpus". In: *Computational Processing of the Portuguese Language*, pp. 353–361, Cham, 2018. Springer International Publishing. ISBN: 978-3-319-99722-3.

[102] XU, L., LIU, J., PAN, X., et al. "DataCLUE: A Benchmark Suite for Data-centric NLP". 2021. Available at: <https://arxiv.org/abs/2111.08647>.

[103] Google Research. "BERT". https://github.com/google-research/bert, 2019.

[104] 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)*, pp. 328–339, Melbourne, Australia, jul 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1031. Available at: <https://aclanthology.org/P18-1031>.

[105] IYYER, M., MANJUNATHA, V., BOYD-GRABER, J., et al. "Deep Unordered Composition Rivals Syntactic Methods for Text Classification". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1681–1691, Beijing, China, jul 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1162.

[106] ZHOU, P., QI, Z., ZHENG, S., et al. "Text Classification Improved by Integrating Bidirectional LSTM with Two-dimensional Max Pooling." *COLING*, 2016.

[107] LO, S. L., CAMBRIA, E., CHIONG, R., et al. "Multilingual sentiment analysis: from formal to informal and scarce resource languages", *Artificial Intelligence Review*, v. 48, n. 4, pp. 499–527, 2016. doi: 10.1007/s10462-016-9508-4.

[108] HUGGING FACE. "Pretrained models". 2019. Available at: <https://huggingface.co/transformers/v2.4.0/pretrained_models.html>.

[109] SOUZA, F. D., SOUZA FILHO, J. B. O. "Sentiment Analysis on Brazilian Portuguese User Reviews". In: *IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, 2021. doi: 10.1109/LA-CCI48322.2021.9769838.

[110] SOUZA, F. D., SOUZA FILHO, J. B. O. "BERT for Sentiment Analysis: Pre-Trained and Fine-Tuned Alternatives". In: *Computational Processing of the Portuguese Language (PROPOR)*, 2022. doi: 10.1007/978-3-030-98305-5_20.