Universidade Estadual de Campinas
Instituto de Computação

Tiago Martinho de Barros

Employing Transformers and Emoji to Perform Sentiment Classification of Social Media Texts

Utilizando Transformers e Emoji na Classificação de Sentimento de Textos Oriundos de Redes Sociais

CAMPINAS
2021

Tiago Martinho de Barros

Employing Transformers and Emoji to Perform Sentiment Classification of Social Media Texts

Utilizando Transformers e Emoji na Classificação de Sentimento de Textos Oriundos de Redes Sociais

Dissertação apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Dissertation presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Computer Science.

Supervisor/Orientador: Prof. Dr. Hélio Pedrini
Co-supervisor/Coorientador: Prof. Dr. Zanoni Dias

Este exemplar corresponde à versão final da Dissertação defendida por Tiago Martinho de Barros e orientada pelo Prof. Dr. Hélio Pedrini.

CAMPINAS
2021

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

B278e

Barros, Tiago Martinho de, 1986-
Employing Transformers and emoji to perform sentiment classification of social media texts / Tiago Martinho de Barros. – Campinas, SP : [s.n.], 2021.

Orientador: Hélio Pedrini.
Coorientador: Zanoni Dias.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Processamento de linguagem natural. 2. Análise de sentimentos. I. Pedrini, Hélio, 1963-. II. Dias, Zanoni, 1975-. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Utilizando Transformers e emoji na classificação de sentimento de textos oriundos de redes sociais
Palavras-chave em inglês:
Natural language processing
Sentiment analysis
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Hélio Pedrini [Orientador]
Rodrigo Frassetto Nogueira
Esther Luna Colombini
Data de defesa: 03-05-2021
Programa de Pós-Graduação: Ciência da Computação

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-8427-2673
- Currículo Lattes do autor: http://lattes.cnpq.br/5930039103393808

Universidade Estadual de Campinas
Instituto de Computação

INSTITUTO DE COMPUTAÇÃO

Tiago Martinho de Barros

Employing Transformers and Emoji to Perform Sentiment Classification of Social Media Texts

Utilizando Transformers e Emoji na Classificação de Sentimento de Textos Oriundos de Redes Sociais

Campinas, 03 de maio de 2021

Acknowledgements

First and foremost, I thank God for everything.

Then, I would like to thank my parents, Maria Terezinha and Genivaldo, for they have done their best to support me throughout my life. Among the many invaluable lessons I was given, perhaps the most important one is to be a good human being. I sincerely hope to live up to their expectations. Also, I extend my gratitude to my family, that regularly encourages me in my search for knowledge.

This work would not have been possible without the precious help of my supervisors Prof. Dr. Hélio Pedrini and Prof. Dr. Zanoni Dias, who constantly shared their hard-earned knowledge to solve the problems that cropped up during this research. I feel that I ended up trying their patience on some occasions, and for that I apologize.

I also would like to thank my colleagues from the Visual Informatics Laboratory (LIV) of the Institute of Computing (IC), especially Gabriel Bianchin, Leodécio Braz, Vinicius Teixeira, Daiane Mendes, and Marianna Severo, who helped me with many things and with whom I shared many laughs and good moments.

I should thank all my friends as well because they make my life substantially better and happier. I feel blessed to have them in my life, and I hope this continues for many more decades so we can have many more memorable moments and more stories to tell our grandchildren, should they come.

Finally, I express my gratitude to Ji-chan – my beloved bicycle – who has been with me for about 25 years. Not only she makes my life easier but also more enjoyable and adventurous. She has been a superb partner and I am grateful for that.

Last but not least, thank you, dear reader, for the interest in this modest work. I hope you find it easy to understand and maybe learn a thing or two.

Resumo

Avanços recentes na área de Processamento de Linguagem Natural trouxeram melhores soluções para uma série de tarefas interessantes como Aceitabilidade Linguística, Respostas a Perguntas, Compreensão de Leitura, Inferência de Linguagem Natural e Análise de Sentimento. Neste trabalho, focamos em Análise de Sentimento, que é um campo de pesquisa voltado ao estudo computacional de sentimentos. A Análise de Sentimento possui muitas aplicações práticas como sistemas de recomendação, monitoramento de satisfação de usuários e previsão do resultado de eleições.

As tarefas mencionadas são importantes para o avanço da Inteligência Artificial, pois são desafiadoras e podem ser aplicadas em vários problemas. A abordagem tradicional é construir um classificador específico para cada tarefa, entretanto, com a popularização do conceito de pré-treinamento seguido de ajuste fino, tornou-se muito comum a utilização de uma mesma arquitetura em diferentes problemas, por meio de ajuste fino com dados da tarefa em questão.

Métodos como ULMFiT, ELMo, BERT e seus derivados obtiveram sucesso substancial em muitas tarefas de Processamento de Linguagem Natural, no entanto, eles compartilham uma desvantagem: para pré-treinar esses modelos do zero, quantidades substanciais de dados e recursos computacionais são necessários. Nesta dissertação, propomos uma nova metodologia para classificar sentimento em textos, baseada no BERT e com foco em emoji, tratando-os como uma importante fonte de sentimento em vez de considerá-los simples *tokens* de entrada. Além disso, pode-se utilizar um modelo BERT já pré-treinado como ponto de partida para nosso modelo, reduzindo significativamente o tempo total de treinamento necessário.

Avaliamos o uso de pré-treinamento adicional com textos contendo pelo menos um emoji. Também empregamos aumentação de dados para melhorar a capacidade de generalização de nosso modelo. Experimentos em dois conjuntos de dados de *tweets* em português do Brasil – TweetSentBR e 2000-tweets-BR – mostram que nossa metodologia produz resultados competitivos em relação aos métodos publicados anteriormente e ao BERT.

Abstract

Recent advances in the Natural Language Processing field have brought better solutions to a number of interesting tasks, such as Linguistic Acceptability, Question Answering, Reading Comprehension, Natural Language Inference, and Sentiment Analysis. In this work, we focus on Sentiment Analysis, which is a research field concerned with the computational study of sentiments. Sentiment Analysis has many practical applications, such as recommender systems, user satisfaction monitoring, and election outcome prediction.

The aforementioned tasks are important to the advancement of Artificial Intelligence as they are challenging and can be used in many different scenarios. The traditional approach is to build a specific classifier for each task, but with the popularization of the concept of pre-training followed by fine tuning, it has become very common to use the same architecture to solve different problems by fine-tuning it with data from the task at hand.

Methods, such as ULMFiT, ELMo, BERT, and their derivatives, have achieved substantial success with many Natural Language Processing tasks, however they share a drawback: to pre-train these models from scratch, substantial amounts of data and computational resources are required. In this dissertation, we propose a novel methodology to classify the sentiment of texts, based on BERT and focusing on emoji, treating them as an important source of sentiment as opposed to considering them simple input tokens. Additionally, it is possible to use a previously pre-trained BERT model to warm start ours, greatly reducing the total training time required.

We evaluate the use of additional pre-training using texts which contain at least one emoji. We also employ data augmentation to improve the generalization ability of our model. Experiments on two Brazilian Portuguese tweets datasets – TweetSentBR and 2000-tweets-BR – show that our methodology produces competitive results compared to the previously published methods and to BERT.

List of Figures

List of Tables

List of Abbreviations and Acronyms

BERT	Bidirectional Encoder Representations from Transformers
BN	Batch Normalization
CNN	Convolutional Neural Network
ELMo	Embeddings from Language Models
GPT	Generative Pre-trained Transformer
GRU	Gated Recurrent Unit
LN	Layer Normalization
LR	Logistic Regression
LSTM	Long Short-Term Memory
MLM	Masked Language Modeling
MLP	Multi-Layer Perceptron
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
NSP	Next Sentence Prediction
RNN	Recurrent Neural Network
SNS	Social Networking Service
SVM	Support Vector Machines
T5	Text-To-Text Transfer Transformer
TTsBR	TweetSentBR
ULMFiT	Universal Language Model Fine-Tuning

Contents

Chapter 1

Introduction

This chapter presents the problem considered in this dissertation, the motivation for expanding the research about it, as well as our main contributions. In the latter part, the text structure of the following chapters is explained.

1.1 Problem Description and Motivation

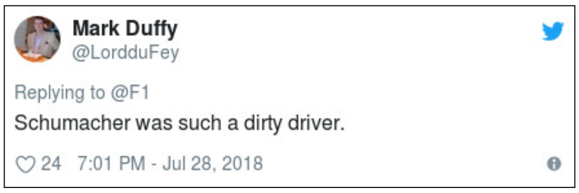
Sentiment Analysis, also referred to as Opinion Mining [41, 48], is a research field concerned with the computational study of sentiments. In this context, "sentiment" can be defined as the author's attitude, opinion, or emotion expressed on a named entity, event, or abstract concept that is mentioned in a piece of text in natural language [67].

Since early 2000s, Sentiment Analysis has grown to be one of the most active research topics in the Natural Language Processing (NLP) field. It is also widely studied in Data Mining, Web Mining, Text Mining, and Information Retrieval. In fact, it has spread from Computer Science to Management Sciences and Social Sciences such as Marketing, Finance, Political Science, Communications, Health Science, and even History, due to its importance to business and society as a whole [82].

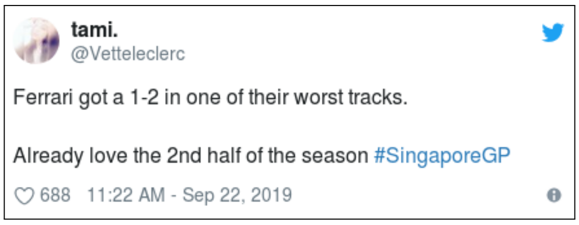
With the widespread access to the Internet in the last decades, people have gained a new and powerful medium through which voice their opinions. With an unprecedented reach, it has never been easier to make our views visible worldwide. Entities, such as companies and government agencies, are frequently interested in knowing what people think in order to make informed decisions.

Usual sources of subjective texts (texts which contain an opinion) are Social Networking Services (SNS), (micro)blogs, and websites featuring user reviews. Using data from these sources, it is possible to know the opinion of end users about a product or service [37, 45], build a recommender system [46], and even try to predict the outcome of an election [75]. Examples of user-generated content in social media – tweets in this case – are shown in Figure 1.1.

When processing user-generated content, we face some challenges such as misspellings, use of slang, and lack of punctuation, because many users, in the social media environment, tend to write in a fast, effortless, and often ungrammatical manner. However, this type of content also presents some opportunities not found in more formal texts, and one of them


Mark Duffy
@LordduFey
Replying to @F1
Schumacher was such a dirty driver.
24 7:01 PM - Jul 28, 2018

(a) Example of a negative tweet.


tami.
@Vetteleclerc
Ferrari got a 1-2 in one of their worst tracks.

Already love the 2nd half of the season #SingaporeGP
688 11:22 AM - Sep 22, 2019

(b) Example of a positive tweet.

Figure 1.1: Examples of user-generated content in social media.

is the presence of emoji[1], which can be found in many user comments and also in posts made by companies, as they try to establish a closer relationship with their customers. In fact, Huang et al. [31] found that using an emoji in casual conversation introduces feelings of enjoyment, happiness, and solidifies an overall positive impression of the interaction; and a survey from Adobe [1] has shown that 78% of emoji users say that using emoji makes you more likeable, and 74% of emoji users consider that emoji make positive news more sincere. While usage and interpretation of emoji vary according to culture and even from person to person [44], it is undeniable that they add expressiveness to a message, and sometimes actually become the entire message (e.g., replying to a question with a 👍 – thumbs up). This is the case in 27% of the time in text messaging [1].

In this work, we sought to leverage emoji and use them in the most effective manner possible in this context. Our methodology is based on Bidirectional Encoder Representations from Transformers (BERT) [16], which in itself performs well on a range of NLP tasks, including Sentiment Analysis. But when faced with an emoji, BERT will either treat it as an input token such as any other or ignore it altogether, depending on the WordPiece [79] vocabulary. We propose a methodology to handle emoji separately from the words, if present. We extract emoji from the input text and process them through the Transformer [72] encoder independently to try to obtain the maximum information from both the words sequence and the emoji sequence. Then, we combine all this information to classify the sentiment. To conduct our experiments, we adopted two datasets of tweets written in Brazilian Portuguese: TweetSentBR (TTsBR) [10] and 2000-tweets-BR [73].

1.2 Objectives

Our main goal, in this work, was to investigate and improve the sentiment classification of user-generated texts. In order to achieve our general objective, we set some specific objectives:

1. Search for suitable datasets to perform sentiment classification;
2. Evaluation of recent works on sentiment classification and on Natural Language Processing (NLP) that can be used to perform sentiment classification, mostly unsupervised language representation learning methods;

[1]We adopt the form "emoji" for both singular and plural usages, following the Unicode Consortium – `https://unicode.org/faq/emoji_dingbats.html#1.05`

3. Proposition of an original methodology to classify the sentiment of social media texts using neural networks;
4. Conduction of experiments on data augmentation;
5. Performance evaluation of the developed model.

1.3 Research Questions

The following research questions guided us through the work in this project:

1. Do emoji, considered alongside their corresponding texts, improve the sentiment classification accuracy?
2. Can further unsupervised pre-training on in-domain data improve the sentiment classification performance?
3. Considering that unsupervised language representation learning methods are pre-trained on gigabytes of textual data, does data augmentation improve the sentiment classification performance?

1.4 Contributions

The main contributions of our work in Sentiment Analysis are:

- A study of emoji occurrence and distribution for the most frequent emoji in the TweetSentBR and the 2000-tweets-BR datasets, comparing the results with general emoji usage in Twitter;
- A novel methodology to classify the sentiment of social media texts using both the expressiveness of emoji and the written text. Our model achieves a new state of the art for both datasets;
- Despite being a different model, we can reduce the training time by using a previously pre-trained $\text{BERT}_{\text{BASE}}$ model to warm start ours, thus avoiding having to pre-train it from scratch.

1.5 Publications

The following papers were written and published as results of this research work:

- T.M. Barros, H. Pedrini, Z. Dias. Leveraging Emoji to Improve Sentiment Classification of Tweets. In 36th ACM/SIGAPP Symposium on Applied Computing (SAC) - Knowledge and Language Processing (KLP) track. Gwangju, Republic of Korea, pages 845–852, 2021. Association for Computing Machinery (ACM).

- T.M. Barros, H. Pedrini, Z. Dias. Data-Augmented Emoji Approach to Sentiment Classification of Tweets. In 25th Iberoamerican Congress on Pattern Recognition (CIARP). Porto, Portugal, pages 1–10, 2021. Springer.

1.6 Text Organization

The remaining of this dissertation is structured as follows. Chapter 2 presents the background of Sentiment Analysis as well as the recent methods that are related to this research topic. In Chapter 3, we discuss the datasets used in this work and our study about emoji occurrence and distribution. Chapter 4 explains in detail our methodology for sentiment classification of social media texts. In Chapter 5, we present the results we obtained. Chapter 6 concludes the text with final remarks and directions for future work. Appendix A reports additional results obtained with this research work and Appendix B discusses overfitting and the experiments we performed to try to keep it to a minimum.

Chapter 2

Background

This chapter presents, in the first section, the most-relevant concepts to the research subject, and a discussion of the related works in the literature in the second section.

2.1 Concepts and Techniques

We review some important concepts to Sentiment Analysis and to this work, such as the usual tasks, the Unsupervised Language Representation Learning Methods, the Transformer, and emoji.

2.1.1 Sentiment Analysis Tasks

The Sentiment Analysis problem is generally tackled at three possible levels of granularity: document-level sentiment classification, sentence-level sentiment classification, and aspect-level sentiment classification [82].

Document-level sentiment classification categorizes a document as expressing an overall positive or negative opinion. It treats the entire text document as the basic unit of the process and considers that the document contain opinions about a single entity (e.g., a movie review).

Sentence-level sentiment classification categorizes individual sentences in a document. In this case, it is not reasonable to assume that every sentence contains an opinion. A traditional approach is to first classify a sentence as subjective (contains an opinion) or objective (does not contain an opinion), which is called subjectivity classification. Then, the resulting subjective sentences are classified as expressing positive or negative opinions [53]. Another possibility is to include a third class to accommodate objective sentences (e.g., a class named "neutral").

Aspect-level sentiment classification is concerned with the extraction of people's opinions expressed on entities and aspects/features of entities, which are also called targets. For example, in the sentence *"the art direction of 'Star Wars: The Force Awakens' was amazing, but the plot was uninteresting, to say the least"*, we have the entity *"Star Wars: The Force Awakens"* and the aspects *"art direction"* and *"plot"*. Aspect-level sentiment classification should classify the sentiment expressed on the art direction of the movie as positive and on the plot as negative.

Researchers are also working on other topics of Sentiment Analysis, such as Emotion Analysis [11], Cross-Domain Sentiment Classification [52], Sarcasm Detection [24], and Multilingual Sentiment Analysis [8].

Traditionally, Sentiment Analysis is cast as a classification task, either binary, with a “positive” class and a “negative” class, or as a multi-class classification problem, including, for instance, a “neutral” or “irrelevant” class to accommodate texts that do not have a sentiment associated with (i.e., objective texts). Sometimes, more classes are used to make the sentiment granularity finer, for instance the five classes: “very positive”, “positive”, “neutral”, “negative”, and “very negative”, which are equivalent to a five-star scale [54]. The number of classes varies according to the dataset considered. In this work, use utilize the TweetSentBR and the 2000-tweets-BR datasets, both using the three-class schema: “positive”, “neutral”, and “negative”.

2.1.2 Unsupervised Language Representation Learning Methods

In recent years, most of the notable progress achieved in Natural Language Processing is due to unsupervised language representation learning methods, which have established new state-of-the-art results in many tasks. The idea is to produce a one-size-fits-all model that is pre-trained on large amounts of unlabeled data, and then fine-tuned on an individual target task (downstream task), in this case Sentiment Analysis. Pre-training on large amounts of data aims to produce sentence encoders with substantial knowledge of the target language, that can be applied later on the target task. This model was first proposed by Dai and Le [14], with the basic idea being to use the parameters obtained from the pre-training as a starting point for the supervised training model.

In 2018, Peters et al. [57] introduced Embeddings from Language Models (ELMo), which employ contextualized word embedding to produce different word vectors for the same word if the context/meaning is different, using bi-directional Long Short-Term Memory (LSTM) networks [28] trained on a language modeling objective. In the same year, Howard and Ruder [29] introduced the Universal Language Model Fine-Tuning (ULMFiT), building upon the pre-training-followed-by-fine-tuning concept and addressing issues of overfitting and catastrophic forgetting. Radford et al. [58] proposed the Generative Pre-trained Transformer (GPT), that combines the unsupervised pre-training with the Transformer [72], as opposed to using LSTMs. Devlin et al. [16] introduced a method called Bidirectional Encoder Representations from Transformers (BERT), which also employs the Transformer but has a different training objective: masked language modeling, in which words in a sentence are randomly erased and replaced with a special token (“masked”) with some small probability. Then, a Transformer is used to generate a prediction for the masked word based on the unmasked words surrounding it, both to the left and right.

In 2019, Yang et al. [81] proposed the XLNet, a generalized auto-regressive pre-training method that enables learning bidirectional contexts by maximizing the expected likelihood over all permutations of the factorization order. Finally, in 2020, Raffel et al. [59] published a large-scale empirical survey to determine which Transfer Learning techniques work best and, based on these insights, developed a new model called the Text-To-Text Transfer Transformer (T5). This model employs a unified text-to-text format where the input and output are always text strings.

2.1.3 Transformer

Vaswani et al. [72] introduced the Transformer with the goal of simplifying sequence transduction, which is the process of transforming input sequences into output sequences. Many Machine Learning tasks can be expressed as sequence transduction tasks, for instance Speech Recognition, Machine Translation, Protein Secondary Structure Prediction and Text-to-Speech. The Transformer architecture was originally developed to perform Neural Machine Translation, but has since been deployed with great success to other textual tasks, and even to Computer Vision [17].

The Transformer can be considered as a modern substitute for Recurrent Neural Networks (RNNs) [18], in particular, Long Short-Term Memory (LSTM) networks [28] and Gated Recurrent Unit (GRU) networks [13], as they are used in the same type of context. One key difference is that the Transformer eschews the use of recurrence and instead makes use of self-attention mechanisms to draw global dependencies between inputs and outputs. Self-attention (sometimes called intra-attention) is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence, in other words, it decides at each step which other parts of the sequence are important. Due to this feature, the Transformer allows for much more parallelization than RNNs and, therefore, reduced training times [72]. It is used in most of the recent unsupervised language representation learning methods, such as GPT [58], BERT [16], XLNet[1] [81], and T5 [59].

The Transformer, such as most competitive neural sequence transduction models, employs an encoder-decoder structure [3, 63]. In this scheme, the encoder maps an input sequence of symbol representations $\boldsymbol{x} = (x_1, \ldots, x_n)$ to a sequence of continuous representations $\boldsymbol{z} = (z_1, \ldots, z_n)$. The decoder then uses $\boldsymbol{z}$ to generate an output sequence $\boldsymbol{y} = (y_1, \ldots, y_n)$ of symbols one element at a time [72]. In simpler terms, the encoder reads the input text and the decoder produces a prediction for the task. For instance, when translating an English text to Portuguese, the encoder will read the English text and create the representations $\boldsymbol{z}$, then the decoder will use $\boldsymbol{z}$ to output the Portuguese translation.

Figure 2.1 illustrates the Transformer architecture, with the encoder on the left and the decoder on the right-hand side of the image. Both the encoder and the decoder are composed of modules that can be stacked on top of each other multiple times, which is described by "Nx" in Figure 2.1. In the original paper, the authors used $N = 6$.

The encoder module is composed of a multi-head self-attention mechanism and a position-wise fully connected feed-forward network. It also features a residual connection [26] and layer normalization [2]. The decoder module has the same elements as the encoder, plus a multi-head attention mechanism over the output of the encoder stack. Furthermore, the self-attention part in the decoder module is modified to prevent posi-

[1]XLNet uses a modified implementation of the Transformer, called Transformer-XL [15].

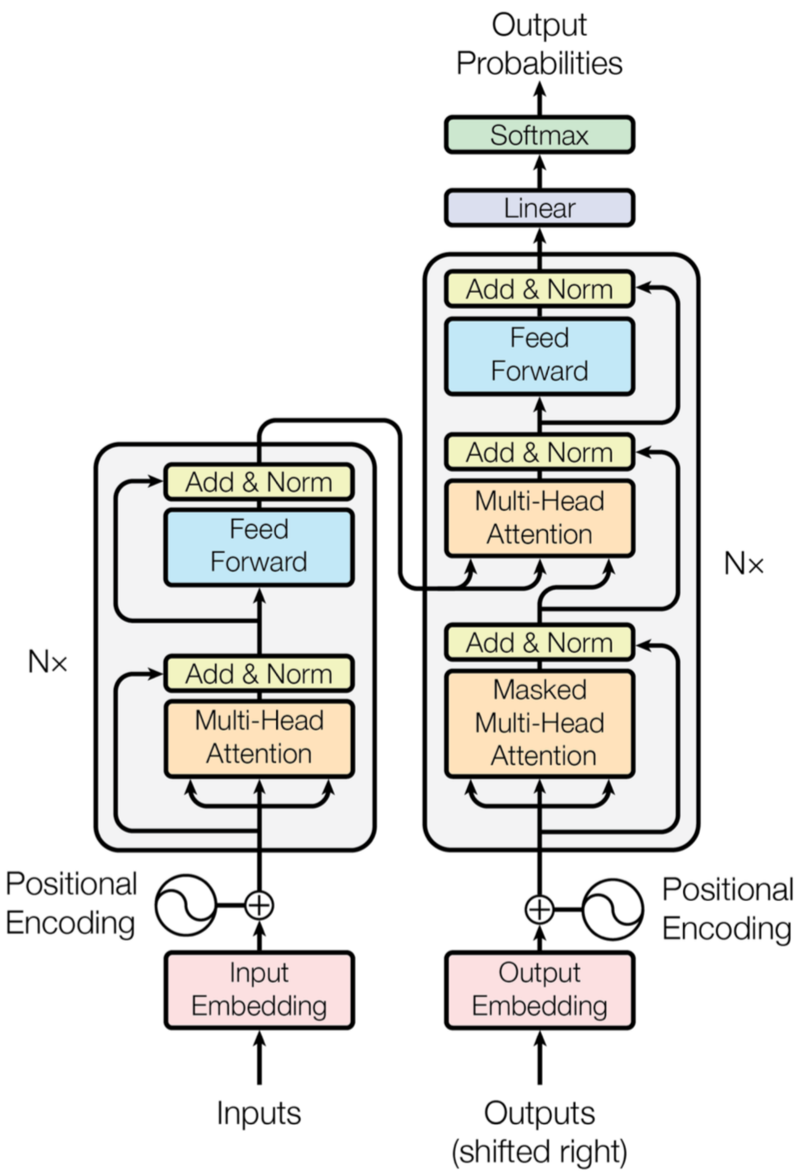
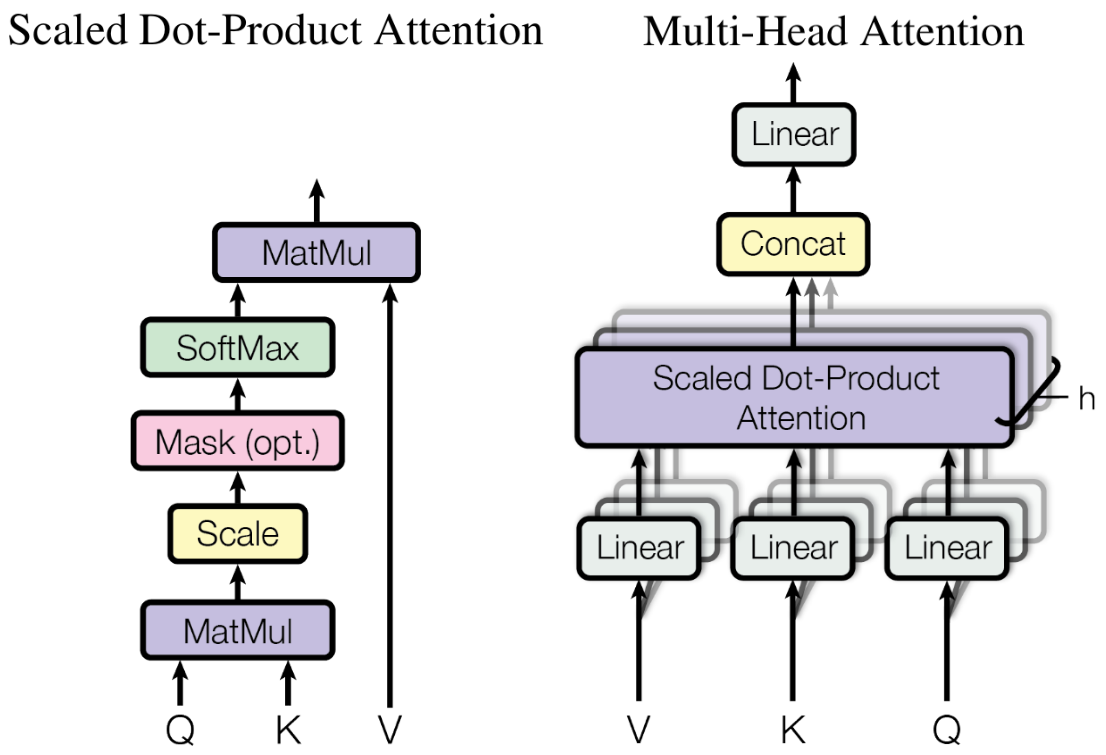


Figure 2.1: Transformer model architecture [72].

tions from attending to subsequent positions. This masking, combined with the fact that the output embeddings are offset by one position, ensures that the predictions for position i can depend only on the known outputs at positions less than i [72]. Since there are no recurrent networks that can remember how sequences are fed into a model, the positions of the words are added to the embedded representation (n-dimensional vector) of each word, since a sequence depends on the order of its elements.

Regarding the attention function, it can be seen as mapping a query and a set of key-value pairs to an output, where query, keys, values, and output are vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key [72]. The attention mechanism used in the Transformer is called *Scaled Dot-Product Attention*, and it is illustrated in Figure 2.2a.

The input consists of queries (a query is a vector representation of one word in the



(a) Scaled Dot-Product Attention. (b) Multi-head attention consists of several attention layers running in parallel.

Figure 2.2: Transformer attention computation [72].

sequence) and keys (vector representations of all the words in the sequence) of dimension d_k, and values of dimension d_v. The attention computation consists of the dot products of the query with all keys, then division of each by $\sqrt{d_k}$, and application of a softmax function to obtain the weights on the values. To be more efficient, the computation is done not over a single query but over a set of queries, packed together into a matrix Q. The keys and values are also packed together into matrices K and V, so the attention computation can be expressed as

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

We can interpret it by considering that the values in V are multiplied and summed with some attention-weights $\boldsymbol{w} = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)$. That is, these weights $\boldsymbol{w}$ are defined by how each word of the sequence (represented by Q) is influenced by all the other words in the sequence (represented by K). Additionally, the softmax function is applied to the weights $\boldsymbol{w}$ so they have a distribution between 0 and 1. The weights are then applied to all the words in the sequence that are introduced in V.

Figure 2.2b shows how the Scaled Dot-Product Attention can be parallelized into multiple mechanisms that can be used side by side. The attention mechanism is repeated h times with linear projections of Q, K and V. This allows the system to learn from different representations of Q, K and V, which the authors found beneficial to the model.

Since its introduction, there have been numerous improvements proposed by different researchers, mainly focusing on improving the time complexity, which is quadratic in the original model. Some of these improved models are: Reformer [36], reducing the overall self-attention complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log n)$ using locality-sensitive hashing (n is the sequence length); Linformer [76], which reduces the complexity to $\mathcal{O}(n)$ in both time and space by approximating the self-attention mechanism by a low-rank matrix; and Linear Transformer [33], also reducing the complexity to $\mathcal{O}(n)$, but using a kernel-based formulation of self-attention and the associative property of matrix products to calculate the self-attention weights.

2.1.4 Bidirectional Encoder Representations from Transformers (BERT)

Since our proposed method for sentiment classification builds upon BERT [16], we describe this model in more detail here. Introduced by Devlin et al. [16], one of the biggest selling points of BERT, compared with the models that came before it, is its bidirectionality. While GPT [58] uses a left-to-right Transformer and ELMo [57] uses the concatenation of independently trained left-to-right and right-to-left LSTMs to generate features for downstream tasks, BERT employs a bidirectional Transformer, producing representations that are jointly conditioned on both left and right context in all layers, which is reflected on the strong results obtained.

There are two steps in the BERT framework: *pre-training* and *fine-tuning*. During pre-training, the model is trained on unlabeled data over different pre-training tasks. For fine-tuning, the model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks. The fine-tuned models are different for each downstream task, however they are all initialized with the same pre-trained parameters. Figure 2.3 exemplifies this concept for the question-answering task.

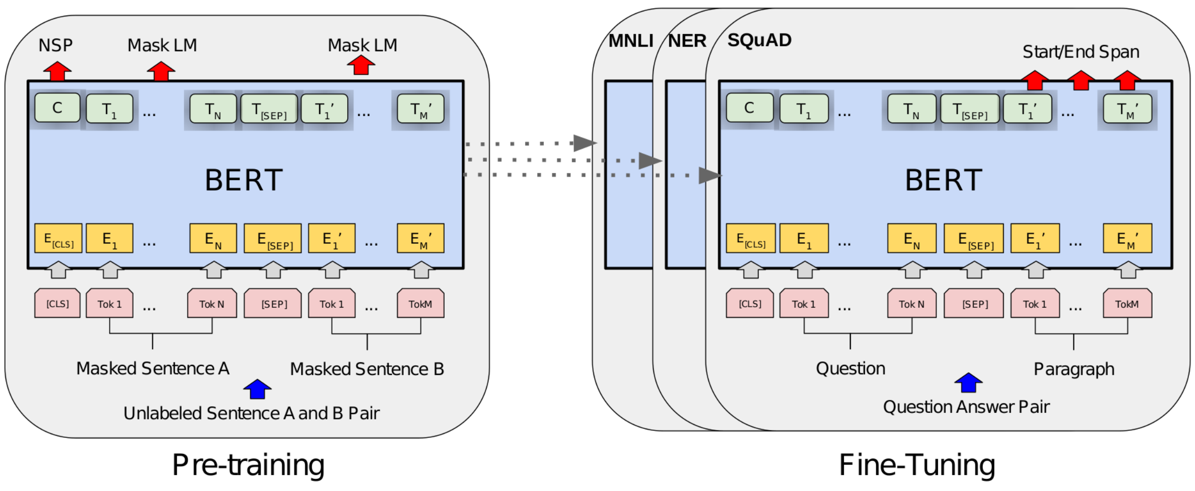


Figure 2.3: Steps in the BERT framework: pre-training and fine-tuning [16].

The model architecture of BERT is a multi-layer bidirectional Transformer encoder based on the original implementation by Vaswani et al. [72] and released in the `tensor2tensor` library[2]. As presented in Section 2.1.3, the Transformer consists of an encoder and a decoder, but since the goal of BERT is to generate a language representation model, it only needs the encoder part, to read the input text and create the continuous representations, building contextual relationships between the words. In this work, we denote the number of layers (i.e., Transformer blocks) as L, the hidden size as H, and the number of self-attention heads as A. The authors trained BERT using two model sizes: $\text{BERT}_{\text{BASE}}$ ($L = 12$, $H = 768$, $A = 12$, total parameters = 110M) and $\text{BERT}_{\text{LARGE}}$ ($L = 24$, $H = 1024$, $A = 16$, total parameters = 340M). In this dissertation, we employed only $\text{BERT}_{\text{BASE}}$ due to hardware limitations.

Since BERT is designed to handle different NLP downstream tasks, including those with two-part inputs (e.g., ⟨question, answer⟩), it uses an input representation that accommodates inputs consisting of one or two "sentences". In this context, a "sentence" is an arbitrary span of contiguous text, rather than an actual linguistic sentence [16]. A "sentence" (or two "sentences", depending on the task) forms a "sequence", which is the input token sequence to BERT. Considering Sentiment Analysis, the "sequences" consist of one "sentence", which in this work are tweets.

BERT employs WordPiece [79] embeddings to build the token vocabulary. The first token of every sequence is always a special classification token (`[CLS]`). The final hidden state corresponding to this token is used as the aggregate sequence representation for classification tasks. In the cases of two-part inputs (two "sentences"), the "sentences" are differentiated in two ways: first, they are separated with a special token (`[SEP]`); second, a learned embedding (segment embedding) is added to every token indicating whether it belongs to sentence `A` or sentence `B`. As shown in Figure 2.3, the input embeddings are denoted as E, the final hidden vector of the special `[CLS]` token as $C \in \mathbb{R}^H$, and the final hidden vector for the i-th input token as $T_i \in \mathbb{R}^H$.

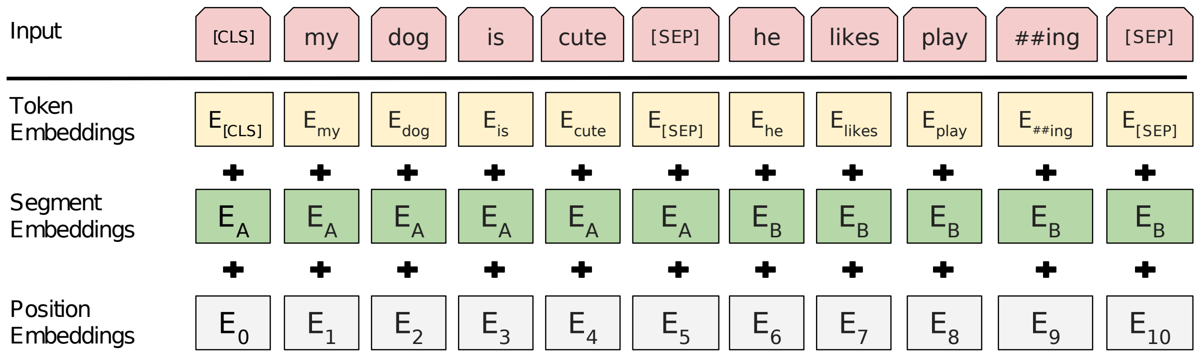

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8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

Figure 2.4: Input representation for BERT. The input embeddings are the sum of the token embeddings, the segment embeddings and the position embeddings [16].

For a given token, its input representation is constructed by summing the corresponding token, segment, and position embeddings, as illustrated in Figure 2.4. The segment embeddings are used to differentiate sentence `A` from sentence `B`, as explained earlier. The position embeddings are used to indicate the position of each token in the sentence, as required by the underlying Transformer.

[2]`https://github.com/tensorflow/tensor2tensor`

BERT is pre-trained using two unsupervised tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). MLM is the solution that the authors found to train a language model, since standard conditional language models can only be trained left-to-right or right-to-left[3]. In MLM, a percentage of the input tokens are masked at random and the model is trained to predict those masked tokens. This procedure is also known as Cloze task [65]. The NSP task objective is to build understanding of the relationship between two consecutive sentences, to be used in downstream tasks such as Question Answering and Natural Language Inference. The model is trained to predict, given two sentences `A` and `B`, if `B` follows `A` or not.

2.1.5 Emoji

"Emoji" comes from the Japanese word 絵文字, which is a compound word: 絵 (`e` ≈ picture) + 文字 (`moji` ≈ written character). They were invented in Japan in the final years of the 20th century. Most works and articles credit the creation to Shigetaka Kurita, who was working for the Japanese mobile phone operator NTT DoCoMo as an interface designer. During 1998/1999, Kurita designed a set of 176 emoji using a grid of 12×12 pixels for the "i-mode" mobile Internet software [40]. However, recently-surfaced evidence attributes the creation of emoji to NTT DoCoMo's rival SoftBank – then called J-PHONE – which released in November 1997 the SkyWalker DP-211SW mobile phone, containing a set of 90 emoji [20].

Since their inception until about 2010, they remained much more popular in Japan than elsewhere. In that year, the Unicode Version 6.0 was published, bringing with it support for the first batch of emoji, thus providing for data interchange between different mobile vendors and across the Internet [70]. In 2011, another big step was taken when Apple released the iOS 5, which popularized emoji between mobile phone users outside of Japan.

Thereafter, emoji usage skyrocketed, as shown by Figure 2.5. By mid-2015, half of all comments on Instagram included an emoji [21]. In 2020, approximately one in five tweets included at least one emoji (19.04%) [21]. And over 60 million emoji are sent on Facebook and 5 billion emoji are sent on Messenger every day, on average [19].

In total, there are 3,521 emoji in the Unicode Standard as of September 2020 [21]. Interestingly, the rise of emoji has helped Unicode's primary goal, which is that computers handle every human language. The pressure to fully support emoji has led many "lagging" implementations to flesh out their Unicode support, and stay current each year [71].

2.2 Literature Review

In this section, we briefly present some works that are related to this research. These works are grouped into three categories according to main focus: Sentiment Analysis, Unsupervised Learning, and Emoji. The works in each category are presented in chronological order.

[3]In this case, the bidirectional conditioning would allow each word to indirectly "see itself", and the model could trivially predict the target word in a multi-layered context [16].

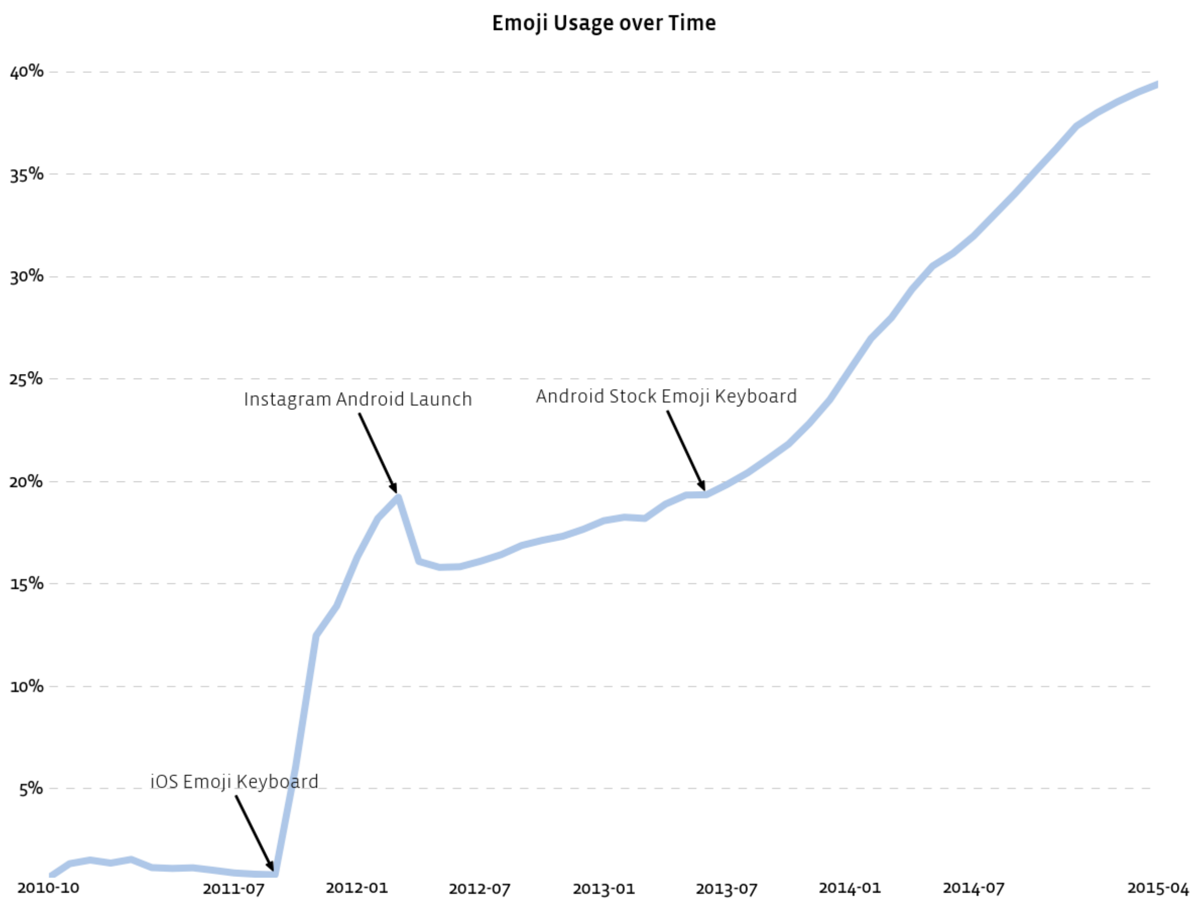


Figure 2.5: Emoji usage on Instagram over time [71].

2.2.1 Sentiment Analysis

Until the first years of this millennium, texts were traditionally classified by topic. However, in 2002, two seminal papers were published, that put Sentiment Analysis on the map of NLP research fields. The first one is the work of Pang et al. [56], in which they apply Machine Learning methods (Naïve Bayes, Maximum Entropy Classification, and Support Vector Machines (SVM)) to perform sentiment classification of movie reviews. The other paper is by Turney [68], in which he presents an unsupervised learning algorithm for classifying customer reviews as recommended or not recommended using the average semantic orientation of the phrases in the review that contain adjectives or adverbs.

Turney and Littman [69] introduced, in 2003, a method for inferring the semantic orientation of a word (how much positive or negative) from its statistical association with a set of positive and negative paradigm words. The authors evaluated two methods: Pointwise Mutual Information and Latent Semantic Analysis.

In 2004, Hu and Liu [30] mined and summarized the features of products on which customers have expressed their opinions and whether the opinions were positive or negative. This publication produced a dataset that is used by many works in the field. In the same year, Pang and Lee [53] proposed a novel Machine Learning method that applies text categorization techniques to just the subjective portions of the document. They presented a sentence-level graph-based formulation relying on finding minimum cuts to decide if a sentence is subjective (contains an opinion) or objective (does not contain an opinion), then, considering only the subjective sentences, a standard Machine Learning classifier is used to determine the document polarity (positive or negative).

Pang and Lee [54] considered, in 2005, the problem of classifying the sentiment of texts with respect to a multi-point scale (e.g., one to five "stars"). They also released a dataset of movie reviews that proved to be very popular amongst researchers.

In 2008, Pang and Lee [55] published a comprehensive and influential survey on Sentiment Analysis, presenting a panorama of the field at the time, covering new challenges, promising approaches, issues regarding privacy, manipulation, and economic impact of the development of opinion-oriented information-access services, and providing a discussion of available resources, benchmark datasets, and evaluation campaigns.

In 2011, Taboada et al. [64] introduced the Semantic Orientation CALculator (SO-CAL): a lexicon-based approach to extracting sentiment from text. It used sentiment dictionaries with annotations of polarity and strength of semantic orientation. The authors also described the process of dictionary creation.

Liu [41] also published an important survey, in 2012. Liu discusses the different formulations (e.g., cross-domain sentiment classification and aspect-based sentiment analysis) and approaches (e.g., dictionary-based approach and corpus-based approach) for Sentiment Analysis, as well as the problems that usually arise. Some other related tasks are also discussed, for instance, opinion spam detection.

2.2.2 Unsupervised Learning

In the realm of unsupervised learning, Mikolov et al. [43] introduced, in 2013, the Skip-Gram model to generate word vectors by training a neural network to predict words that usually occur nearby a given word. In 2014, Le and Mikolov [39] proposed the Paragraph Vector method, that applies the idea of word embedding to variable-length pieces of text, ranging from sentences to whole documents. Kiros et al. [35] abstracted the Skip-Gram model to the sentence level. Instead of using a word to predict its surrounding context, the authors encoded a sentence to predict the sentences around it. The resulting sentence encoder model, called Skip-Thoughts, was published in 2015.

Dai and Le [14] first proposed, in 2015, the supervised fine-tuning step after the unsupervised pre-training, such as predicting adjacent sentences. The basic idea is to use the parameters obtained from the pre-training as a starting point for the supervised training model. Following that, we have the unsupervised language representation learning methods, such as the Embeddings from Language Models (ELMo) [57], the Universal Language Model Fine-Tuning (ULMFiT) [29], the Generative Pre-trained Transformer (GPT) [58], the Bidirectional Encoder Representations from Transformers (BERT) [16], the XLNet [81], and the Text-To-Text Transfer Transformer (T5) [59]. For more information about these models, please refer to the Section 2.1.2.

2.2.3 Emoji

There are many interesting research works about emoji, spanning many fields such as Linguistics, Semiotics, Psychology, and Sociology. Here, we present works related to Natural Language Processing and Sentiment Analysis. One of such works was produced by Novak et al. [49] in 2015, and introduced the Emoji Sentiment Ranking[4], which is the first emoji sentiment lexicon, drawing a sentiment map of the 751 most frequently used emoji at the time. The authors used 1.6 million tweets from 13 European languages and the sentiment labels negative, neutral, and positive. They employed 83 human annotators to label the tweets.

In 2016, Barbieri et al. [4] used the Skip-Gram model [43] to generate and validate semantic vectorial models that are built over 10 million tweets by consistently mapping in the same vectorial space both words and emoji. The authors also evaluated emoji similarity (how equivalent two emoji are) and relatedness (situations in which people would use two emoji together). Plotting the emoji vectors, we can see that similar emoji get clustered together[5].

Felbo et al. [22] experimented, in 2017, using emoji occurrences from social media texts to pre-train a model – called DeepMoji – to detect sentiment, emotion, and sarcasm. The authors collected tweets which contained emoji and used them to pre-train their model by predicting which emoji were part of each tweet. Then, fine-tuned DeepMoji to perform sentiment, emotion, and sarcasm classifications. DeepMoji uses two bi-directional LSTMs [28, 63] and an attention layer [3, 80].

Also in 2017, Tian et al. [66] published a study about the relationship between emoji and the texts in which they appear. The authors argued that emoji and the linguistic texts can modify the meaning of each other and that the overall communicated meaning is not a simple sum of the two channels. They compiled 21 thousand posts from public media pages from Facebook across four countries, along with 57 million reactions and 8 million comments to conduct the research. In this work, the authors proposed that an emoji can interact with the linguistic text in six ways:

1. Replacing a word/phrase.
   E.g.: “I want to have a 🍺.”

2. Repeating a word/phrase (accenting, adding focus).
   E.g.: “Take note 📝 Sam, this is how you season food, you are almost done there babe. Like you did the chicken 🍗 the other nights.”

3. Expressing the speaker’s emotion or attitude independently.
   E.g.: (Facebook update from survivor of the Orlando nightclub shooting in 2016-06-12) “I am safely home and hoping everyone gets home safely as well 😥.”

4. Enhancing/emphasizing an emotion expressed in the text.
   E.g.: “This would probably be really good 😊.”

[4]`http://kt.ijs.si/data/Emoji_sentiment_ranking`
[5]`http://sempub.taln.upf.edu/tw/emojis`

5. Modifying the meaning of linguistic text (e.g., marking non-literal or non-serious use); implying propositional content.
   E.g.: "I bet you are enjoying your revision 😉."

6. Used for politeness.
   E.g.: "Can you please cook us something that I tag you in instead of your 4am pastas? Thanks 😊."

In order to study the meaning interplay between linguistic texts and emoji, the authors employed the Facebook reactions (Like, Love, Haha, Wow, Sad, and Angry) and emoji from the comments. They found that there is a reliable correlation between Facebook reactions and emoji usage with relation to their sentiments.

Chen et al. [12] proposed in 2018 a method to perform Sentiment Analysis of tweets with extra attention on emoji by employing bi-sense emoji embeddings under positive and negative sentimental tweets individually, and then training a sentiment classifier by attending on these bi-sense emoji embeddings with an attention-based LSTM. Different from conventional approaches, where each emoji responds to *one* embedding vector, the authors embedded each emoji into two distinct vectors (bi-sense emoji embedding): two distinct tokens were assigned to each emoji, of which one is for the particular emoji used in positive sentimental contexts and the other one is for this emoji used in negative sentimental contexts. The method was evaluated using a custom dataset built by the authors – presumably composed of tweets in English – and performed better than a simpler LSTM classifier.

In our research, we combined the special attention on emoji – akin to Chen et al. [12] – with the power of a Transformer [72]-based model, in this case BERT [16]. We utilized datasets of tweets written in Brazilian Portuguese, which is a language with far less research results than English when it comes to Natural Language Processing. However, our approach does not depend on the Portuguese specificities and can be applied to any other language.

Chapter 3

Datasets

In this chapter, we present the datasets we employed to conduct our experiments and assess our methodology: the TweetSentBR and the 2000-tweets-BR.

3.1 TweetSentBR

The TweetSentBR (TTsBR) [10] is a sentiment corpus for Brazilian Portuguese, manually annotated, with 15000 tweets on TV show domain. The tweets were labeled in three classes: positive, neutral, and negative.

Table 3.1: Outline of TweetSentBR.

Class	**Training**	**Test**	**Total**
Positive	5741 (44.2%)	907 (45.1%)	6648
Neutral	3410 (26.3%)	516 (25.7%)	3926
Negative	3839 (29.5%)	587 (29.2%)	4426
Total	12990	2010	15000

Table 3.2: Average number of words per tweet in TweetSentBR.

Class	**Training**	**Test**	**Total**
Positive	11.37 ± 5.88	11.09 ± 5.66	11.33 ± 5.85
Neutral	11.73 ± 6.12	11.84 ± 6.20	11.74 ± 6.13
Negative	12.91 ± 6.31	13.30 ± 6.32	12.96 ± 6.32
Total	11.92 ± 6.11	11.92 ± 6.07	11.92 ± 6.10

Table 3.1 presents an overview of the dataset, showing the distribution of tweets between the three classes, as well as between training and test sets, which are predetermined.

Table 3.2 shows the average number of words per tweet – along with the standard deviation – in the three classes and in the training and test sets. The longest tweet in this dataset contains 54 words. The number of words was computed using the `word_tokenize` function from NLTK [7]. We can see that all word averages are similar, meaning that the dataset is balanced in relation to sample lengths, i.e., we do not have the issue of, for example, training a classifier on short sentences and testing it on long sentences.

Since it is a corpus of tweets, it provides us a good representation of user-generated content, which is one of the primary sources of subjective texts used in Sentiment Analysis. Three example tweets (one from each class) from the dataset are:

Positive	A fátima fica mais bonita com cabelo curto[a] 💇‍♀️ 😊
Neutral	terminou a entrevista com maluma[b] 🤔
Negative	já 😱 acabouuu nãooo[c]

[a]Fátima is more beautiful with short hair.
[b]The interview with Maluma is over.
[c]Is it already over? Nooo!

As the examples show, TweetSentBR also crucially provides us with emoji usage in social media environment. While they do not occur in every tweet (as the examples may suggest), they do occur in about one fifth of the tweets, as shown in Table 3.3, which presents some statistics about emoji occurrence and distribution in the dataset.

Table 3.3: Number of tweets from TweetSentBR that have emoji and their respective percentage in relation to all tweets.

Class	**Training**	**Test**	**Total**
Positive	1688 (64.4%)	274 (66.4%)	1962 (29.5%)
Neutral	379 (14.5%)	65 (15.7%)	444 (11.3%)
Negative	552 (21.1%)	74 (17.9%)	626 (14.1%)
Total	2619 (20.2%)	413 (20.6%)	3032 (20.2%)

We can see that both training and test sets have practically the same proportion of emoji-occurring tweets: 20.2% and 20.6%, respectively, although we can also observe that the emoji distribution between classes is more unbalanced, with the positive class having almost twice as many emoji-occurring tweets than the neutral and negative classes combined. This suggests that emoji which carry positive sentiment are more frequent.

To confirm if that is indeed the case, we computed the ten most frequent emoji in the training and test sets. The result is presented in Table 3.4, which includes the code points of the characters, their visual representations[1], and their absolute frequencies in the dataset.

We can see that "positive" emoji are more frequent. That seems to be the case for tweets, in general, because the three most frequent emoji in Table 3.4 are the same as the

[1]According to the Noto Color Emoji font: `https://www.google.com/get/noto/#emoji-zsye-color`

Table 3.4: Top 10 most-frequent emoji of TweetSentBR.

Training			**Test**		
Unicode	**Emoji**	**Freq.**	**Unicode**	**Emoji**	**Freq.**
U+1F602	😂	1096	U+1F602	😂	217
U+1F60D	😍	865	U+1F60D	😍	131
U+02764	❤️	737	U+02764	❤️	97
U+1F44F	👏	518	U+1F44F	👏	62
U+1F62D	😭	282	U+1F62D	😭	46
U+1F622	😢	120	U+1F499	💙	28
U+1F631	😱	105	U+1F631	😱	25
U+1F499	💙	93	U+1F622	😢	24
U+1F3FB	🏻	89	U+1F3B6	🎶	21
U+02665	♥️	75	U+1F494	💔	13

three more popular emoji on Twitter (with the second and third positions exchanged), according to *Emojitracker*[2], a service that monitors and counts the number of emoji used on Twitter in real time. It has processed over 30 billion tweets[3], since its launch in 2013, so we can assume our results are fairly representative of the natural occurrence of emoji in social media.

Sometimes, it is not clear if an emoji has positive, negative, or no sentiment associated with it. However, this is certainly not the case with the three aforementioned emoji, as their names show: 😂 – "face with tears of joy", 😍 – "smiling face with heart-eyes", and ❤️ – "red heart". Names were obtained from The Unicode Consortium[4].

We also investigated the number of tweets containing one emoji, two emoji, and so forth. The results are presented in Figures 3.1 and 3.2 for the training and test sets, respectively. Note that the figures show only values for tweets that contain at least one emoji, since including a bar for zero emoji would eclipse all the other bars. Moreover, we considered all emoji in the tweets, as opposed to considering only unique emoji per tweet.

3.2 2000-tweets-BR

The 2000-tweets-BR [73] is a multi-domain Brazilian Portuguese corpus of tweets built to analyze the Brazilian and European varieties of the Portuguese language with respect to Sentiment Analysis. It was manually annotated and organized in four classes: positive, neutral, negative, and mixed. This last class refers to tweets having both positive and negative opinions. Originally, the corpus is organized thus: 390 positive tweets, 1040 neutral tweets, 509 negative tweets, and 61 mixed tweets, totaling 2000 tweets. The proportion of tweets in each class reflects the sentiment of the users at the time of the gathering.

[2]`https://emojitracker.com`
[3]`https://emojitracker.com/stats`
[4]`https://unicode.org/emoji/charts/full-emoji-list.html`

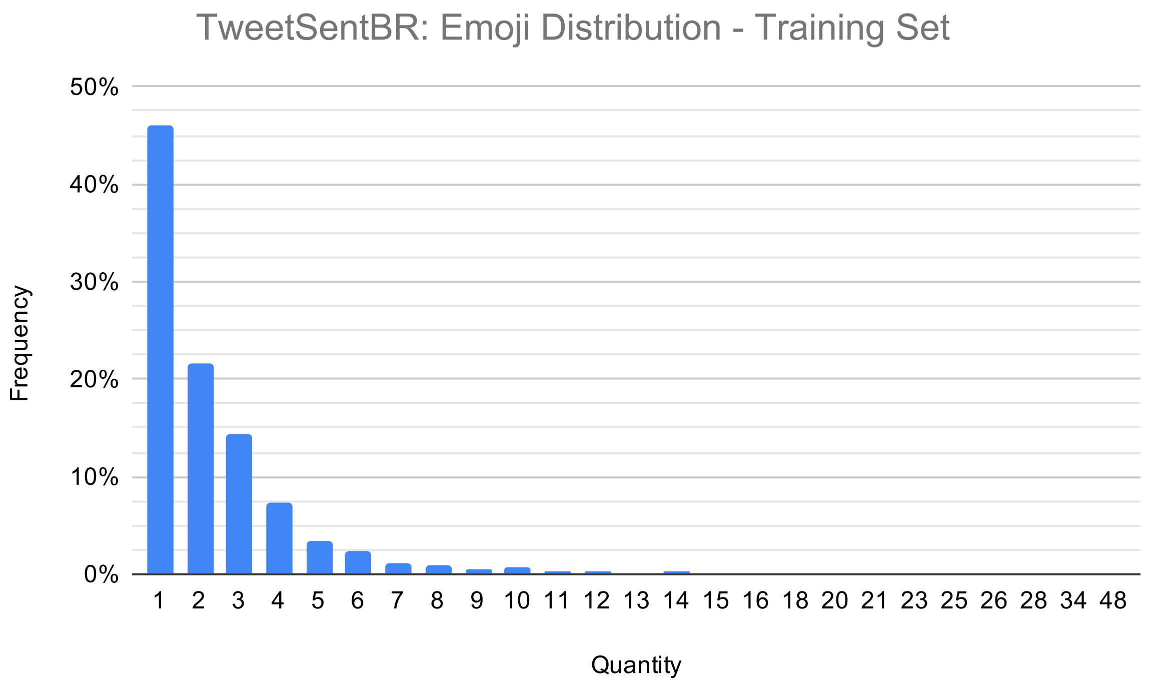


Figure 3.1: Distribution of tweets according to the quantity of emoji per tweet for TweetSentBR – training set.

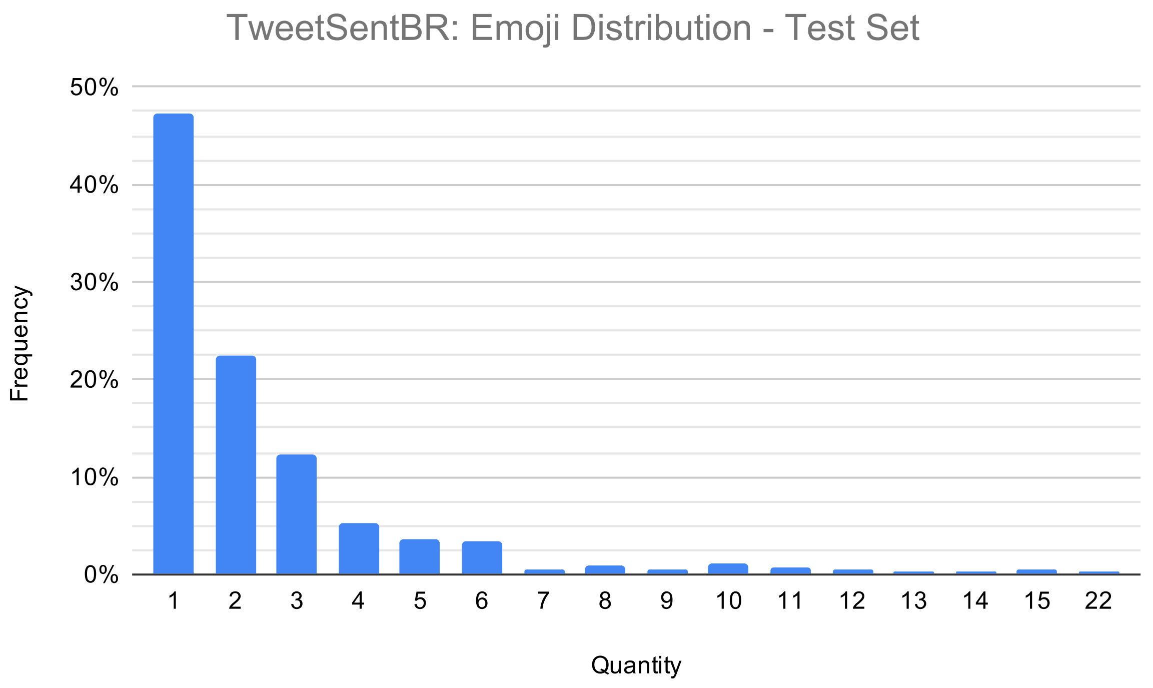


Figure 3.2: Distribution of tweets according to the quantity of emoji per tweet for TweetSentBR – test set.

Following Vitório et al. [73], who introduced the dataset, we do not use the “mixed”

Table 3.5: Outline of 2000-tweets-BR.

Class	**Training**	**Test**	**Total**
Positive	329 (20.0%)	61 (20.9%)	390
Neutral	894 (54.2%)	146 (50.2%)	1040
Negative	425 (25.8%)	84 (28.9%)	509
Total	1648	291	1939

Table 3.6: Average number of words per tweet in 2000-tweets-BR.

Class	**Training**	**Test**	**Total**
Positive	12.50 ± 6.82	10.97 ± 5.88	12.26 ± 6.70
Neutral	11.98 ± 6.79	12.21 ± 6.89	12.01 ± 6.80
Negative	12.84 ± 7.44	12.60 ± 6.85	12.80 ± 7.34
Total	12.30 ± 6.98	12.06 ± 6.68	12.27 ± 6.93

class in the classification process, so the actual number of samples is 1939. Additionally, since the dataset does not have predefined training and test sets, we split it using 15% of the samples, randomly selected, as test set. For the TweetSentBR dataset, the test set is 13.4% of the total, so we chose the nearest multiple of five here. The statistics of the resulting dataset are presented in Table 3.5.

Table 3.6 shows the average number of words per tweet – along with the standard deviation – in the three classes and in the training and test sets. The longest tweet in this dataset contains 37 words. As is the case with the TweetSentBR dataset, all word averages are similar, meaning that the 2000-tweets-BR dataset is also balanced in relation to sample lengths.

Three example tweets (one from each class) from the dataset are:

Positive	O ultimate é lindon[a] 💗 #BTS
Neutral	Quem vive de orgulho morre de saudadeee!![b] 😜💥
Negative	Não acredito[c] 😨💔💔💔

[a]Ultimate is quite handsome.
[b]He who lives with pride dies of longing.
[c]I can't believe it.

We also analyzed the emoji occurrence and distribution in the 2000-tweets-BR dataset. The results are shown in Table 3.7.

For this dataset, the proportion of emoji-occurring tweets in the training and test sets is also similar: 15.5% and 14.1%, respectively, about 5% less than for TweetSentBR. We can also see that, in this case, the neutral class has more emoji-occurring tweets than the

Table 3.7: Number of tweets from 2000-tweets-BR that have emoji and their respective percentage in relation to all tweets.

Class	**Training**	**Test**	**Total**
Positive	79 (30.9%)	14 (34.2%)	93 (23.9%)
Neutral	132 (51.5%)	21 (51.2%)	153 (14.7%)
Negative	45 (17.6%)	6 (14.6%)	51 (10.0%)
Total	256 (15.5%)	41 (14.1%)	297 (15.3%)

positive class, but that is owing to the neutral class having more tweets, as 23.9% of the positive tweets have emoji, in contrast to 14.7% in the case of neutral tweets (Table 3.7).

We computed the ten most frequent emoji in the training and test sets for the 2000-tweets-BR dataset as well, as shown in Table 3.8.

Table 3.8: Top 10 most-frequent emoji of 2000-tweets-BR.

Training			**Test**		
Unicode	**Emoji**	**Freq.**	**Unicode**	**Emoji**	**Freq.**
`U+1F602`	😂	57	`U+1F602`	😂	14
`U+02764`	❤️	52	`U+1F62D`	😭	14
`U+1F60D`	😍	40	`U+02764`	❤️	7
`U+1F644`	🙄	20	`U+1F494`	💔	5
`U+1F3FB`	🏻	19	`U+1F44A`	👊	5
`U+1F62D`	😭	15	`U+1F44C`	👌	3
`U+1F499`	💙	13	`U+1F60D`	😍	2
`U+1F3B6`	🎶	12	`U+1F64F`	🙏	2
`U+1F494`	💔	11	`U+1F497`	💗	2
`U+1F44C`	👌	9	`U+1F62A`	😪	2

The three most frequent emoji in the training set are the same as the three more popular emoji on Twitter, according to *Emojitracker*, in the same order. In regard to the test set, as it is considerably small, it is more susceptible to fluctuations. For example, there is one sample[5] which has 13 occurrences of the emoji 😭 – "loudly crying face", the fourth most-popular emoji on Twitter – putting it in the top three of the most-frequent emoji in the test set of 2000-tweets-BR, otherwise it would be some positions lower, which would be more consistent with the other top 10 tables.

We computed the number of tweets per quantity of emoji for the 2000-tweets-BR dataset as well. The results are presented in Figures 3.3 and 3.4 for the training and test sets, respectively. Note that the figures show only values for tweets that contain at least one emoji, and we considered all emoji in the tweets, as opposed to considering only unique emoji per tweet.

[5]"Nãoooooooo! 😭😭😭😭😭😭😭😭😭😭😭😭😭 💔💔💔 `https://t.co/0hJpT0c5v4`"

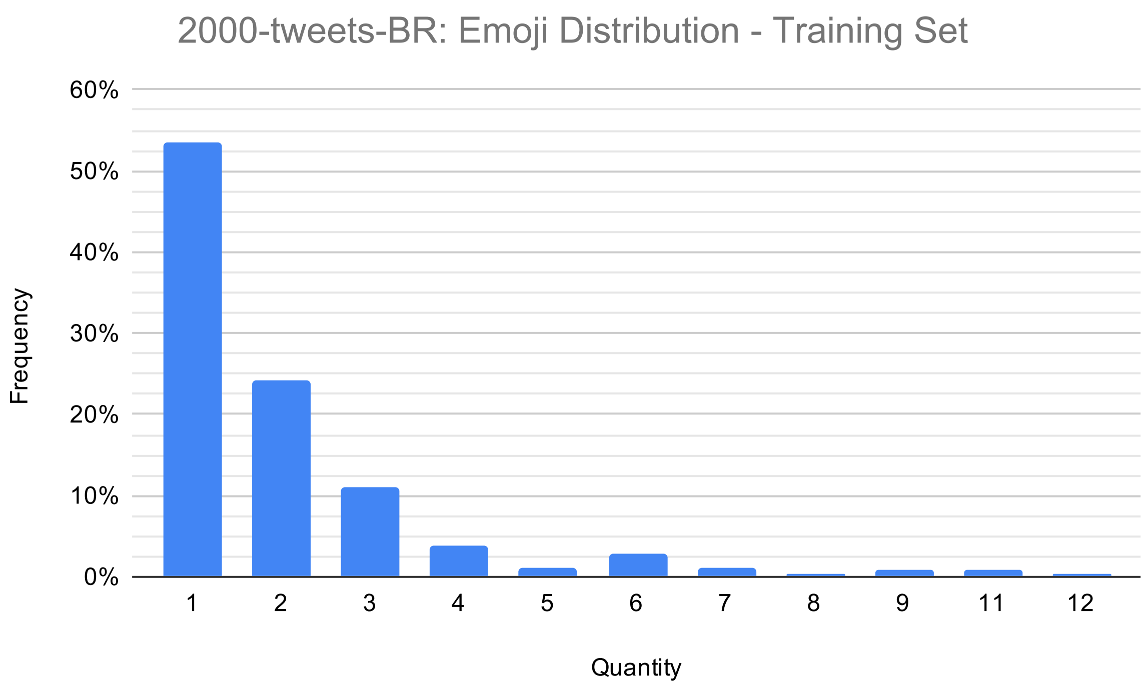


Figure 3.3: Distribution of tweets according to the quantity of emoji per tweet for 2000-tweets-BR – training set.

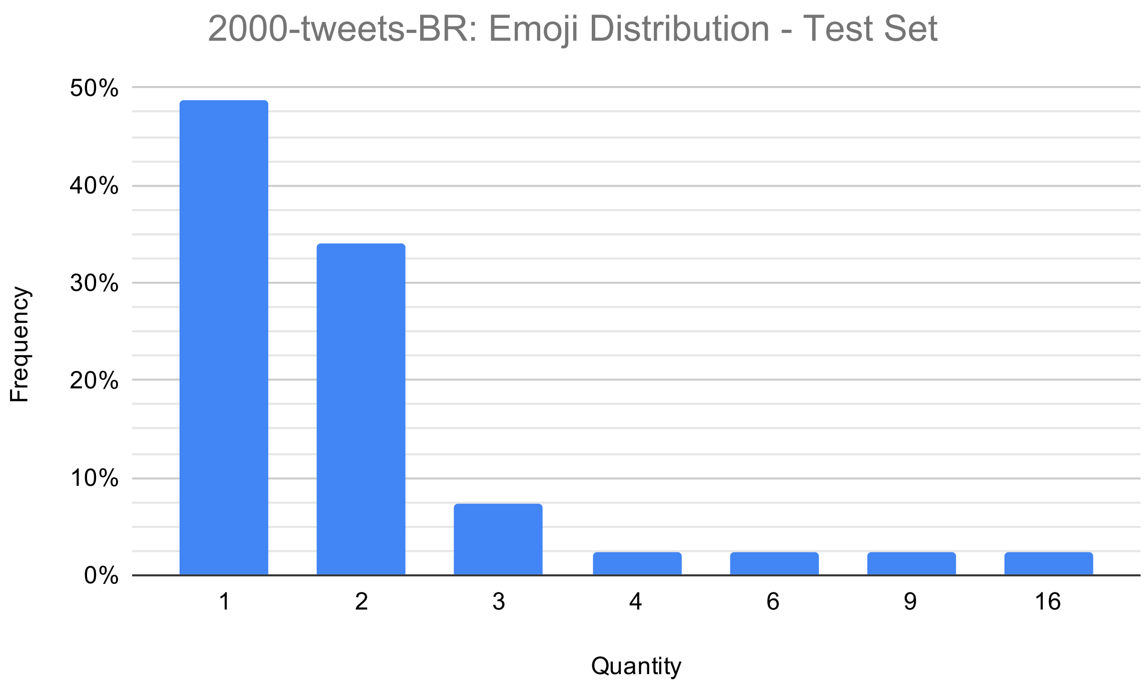


Figure 3.4: Distribution of tweets according to the quantity of emoji per tweet for 2000-tweets-BR – test set.

Chapter 4

Methodology

The core idea behind our proposed methodology is to extract the maximum information possible from emoji in order to have a richer representation of a piece of text and use that to improve the sentiment classification. The methodology comprises additional pre-training evaluation, data augmentation evaluation, emoji extraction, and fine-tuning. Figure 4.1 illustrates the process and the sequence of the steps.

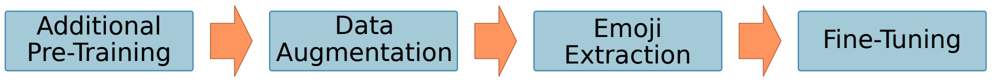


Figure 4.1: Overview of our proposed method for sentiment classification of texts.

Each step is affected by the previous steps, as the data and knowledge flow through the method. Since the starting model is already pre-trained, the pre-training we perform is additional, using similar texts to the ones we are ultimately interested in. Data augmentation is applied on both datasets. Emoji extraction is performed on the augmented datasets. Finally, fine-tuning builds upon the further pre-trained model and uses the emoji extraction on the augmented datasets. The following sections delve into these steps.

4.1 Further Pre-Training

Since pre-training a Transformer-based model from scratch is very expensive time-wise and it also requires massive amounts of data, we make use of *BERTimbau* [62], which is a BERT [16] model pre-trained on the *brWaC* corpus [74], which is composed of 2.7 billion tokens, from 120 thousand different websites. Following the general idea of Transfer Learning, we can fine-tune this model on our downstream task – sentiment classification – using a labeled dataset and obtain a trained classifier. However, according to Gururangan et al. [25], performing a second phase of pre-training, this time using in-domain documents, leads to better results in their experiments. So, we evaluated if this holds in our case as well.

To pre-train our model, we prepared a corpus of user-generated texts from social media with 89458 samples, all of which contain at least one emoji. They were obtained from social media pages related to TV shows, so the domain should be similar to the one of the TweetSentBR [10] dataset. Some examples of entries are presented in Table 4.1.

Table 4.1: Examples of samples from the pre-training corpus.

Linda a Jessica e tem senso de humor.[a] 😊 😍
Quando foi isso? A mulher não ganhou com um nhoque?[b] 🤔
Caramba, que nível.... circo de horrores[c] 😏

[a]Jessica is beautiful and has a sense of humor.
[b]When was that? Didn't she win with a gnocchi recipe?
[c]That's terrible... it's like a horror freak show.

The pre-training process is unsupervised – or, more accurately, semi-supervised, since the labels are obtained from the input samples – so we need nothing besides the corpus. Furthermore, using the same inputs, it is possible to generate different labels using different configurations (tasks). We conducted pre-training experiments using six different configurations [6] for the methodology:

- **Masked Language Modeling (MLM)**: the same task used during pre-training of BERT. Random tokens are masked with a probability of 15% and the model is trained to predict those masked tokens. For more details, please refer to Section 2.1.4. Example of this pre-training configuration:

Text	**Labels**
Alguém pede pra Jojo ⟨MASK⟩ esse vestido.[a] 🤭 🤣 🤣 ⟨MASK⟩ 🤣 🏃‍♂️ 🏃‍♂️	trocar 🤣

[a]Someone should ask Jojo to change this dress.

- **Masked Language Modeling 50% (MLM50)**: similar to the *Masked Language Modeling* configuration, but using a probability of 50% to mask a token. Example of this configuration:

Text	**Labels**
⟨MASK⟩ pede pra ⟨MASK⟩ trocar ⟨MASK⟩ vestido.[a] ⟨MASK⟩ 🤣 ⟨MASK⟩ 🤣 🤣 🏃‍♂️ ⟨MASK⟩	Alguém Jojo esse 🤭 🤣 🏃‍♂️

[a]Someone should ask Jojo to change this dress.

- **All Emoji (All)**: all emoji (and only emoji) are masked and the model is trained to predict those masked emoji. Example:

Text	**Labels**
Alguém pede pra Jojo trocar esse vestido.[a] ⟨MASK⟩ ⟨MASK⟩ ⟨MASK⟩ ⟨MASK⟩ ⟨MASK⟩ ⟨MASK⟩ ⟨MASK⟩	🤭 🤣 🤣 🤣 🤣 🏃‍♂️ 🏃‍♂️

[a]Someone should ask Jojo to change this dress.

- **First Emoji (First)**: the first occurring emoji of a text is masked and the model is trained to predict this masked emoji. If there is only one emoji in the text, it behaves like the *All Emoji* configuration. Example:

Text	Label
Alguém pede pra Jojo trocar esse vestido.[a] ⟨MASK⟩ 🤣 🤣 🤣 🤣 🏃 🏃	🤭

[a]Someone should ask Jojo to change this dress.

- **Emoji Masked Language Modeling (EMLM)**: similar to the *Masked Language Modeling* configuration, but only emoji tokens are randomly masked, with a probability of 15%. Example:

Text	Label
Alguém pede pra Jojo trocar esse vestido.[a] 🤭 🤣 🤣 🤣 🤣 ⟨MASK⟩ 🏃	🏃

[a]Someone should ask Jojo to change this dress.

- **Emoji Masked Language Modeling 50% (EMLM50)**: similar to the *Emoji Masked Language Modeling* configuration, but using a probability of 50% to mask a token. Example of this configuration:

Text	Labels
Alguém pede pra Jojo trocar esse vestido.[a] ⟨MASK⟩ 🤣 ⟨MASK⟩ 🤣 🤣 🏃 ⟨MASK⟩	🤭 🤣 🏃

[a]Someone should ask Jojo to change this dress.

In addition to these six pre-training configurations, we also evaluated the scenario without additional pre-training. In all cases, we used the original architecture of BERT.

To determine the number of pre-training epochs, we evaluated the Masked Language Modeling configuration when pre-training for one to ten epochs on the TweetSentBR dataset. As the Figure A.1 shows, five epochs seem to be sufficient for the result to stabilize, so we used this number of epochs in the experiments.

4.2 Data Augmentation

We evaluated whether data augmentation could improve the results of Sentiment Analysis of social media texts or not. The approach employed here is based on the work of Wei and Zou [77]. The central idea is to modify a piece of text using four operations:

- **Synonym Replacement:** Randomly choose n words from the sentence that are not stop words. Replace each of these words with one of its synonyms chosen at random.

- **Random Insertion:** Find a random synonym of a random word in the sentence that is not a stop word. Insert that synonym into a random position in the sentence. Do this n times.

- **Random Swap:** Randomly choose two words in the sentence and swap their positions. Do this n times.
- **Random Deletion:** For each word in the sentence, randomly remove it with probability p.

The number of words changed per sample, n, is based on the text length l and given by the formula $n = \alpha l$, where α is a parameter that indicates the percentage of words in a sample to be changed. For the random deletion operation, $p = \alpha$.

We experimented with the values of 0.1, 0.2, 0.3, 0.4, 0.5, and 0.6 for α to determine which one produces the best results for the TweetSentBR and 2000-tweets-BR datasets.

In addition to the parameter α, another important one is the n_{aug}, which determines the number of augmented samples per original sample. We experimented with the values of 1, 2, 3, 4, and 5 for n_{aug} to determine which one yields the best results for each dataset.

The original method of Wei and Zou [77] was built to work with English texts. We modified it to work with Portuguese text by employing the Open Multilingual WordNet through NLTK [7] to find synonyms of the words.

4.3 Emoji Extraction

For every tweet, during the fine-tuning phase, we perform emoji extraction to separate them from the text, so that we end up with two sequences: one with words and another with emoji [5]. Note that, in this work, we consider *emoticons* [23, 51] as emoji, because albeit not being the same thing, they essentially fulfill the same role.

Table 4.2: Emoticons considered in our method.

`:(`	`=(`	`;(`	`:-(`	`;-(`	`:)`	`=)`	`;)`	`:-)`	`;-)`	`:D`	`;D`	`<3`	`S2`

Table 4.2 contains the emoticons considered. All emoji found in the training sets are added to the vocabulary of the WordPiece embedding [79] tokenizer.

4.4 Model Architecture

Figure 4.2 illustrates the model and the sentiment classification process, from the input text to the output sentiment probabilities. Each step is explained as follows.

First, the words sequence is processed by the tokenizer, which inserts a special classification token (`[CLS]`) as the first token of the sequence and translates all input tokens to WordPiece token IDs, according to the vocabulary. These token IDs are then fed to the embedding layer, using hidden size $H = 768$. Inside the embedding layer, the token IDs and their positions in the sequence are converted to embeddings (H-dimensional vectors). Then, the embeddings for the input tokens and for the positions of the tokens are summed and then normalized [2]. The resulting embeddings then go to the multi-layer bidirectional Transformer [72] encoder. We use $L = 12$ as the number of layers (i.e., Transformer blocks) and $A = 12$ as the number of self-attention heads.

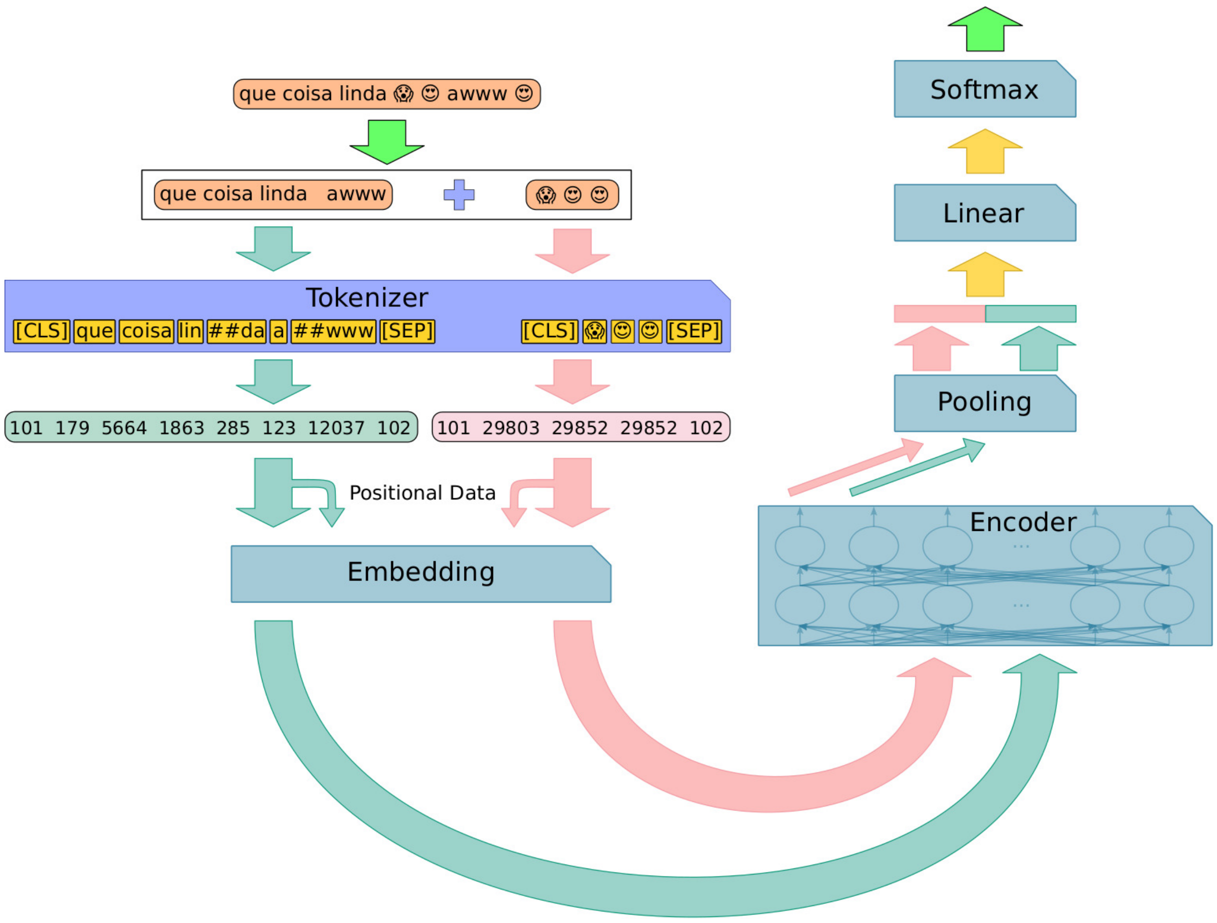


Figure 4.2: Our proposed method for sentiment classification of tweets.

The output of the encoder is then "pooled" by taking the hidden state corresponding to the special classification token, following Devlin et al. [16]. The same process is applied to the emoji sequence. The next step is to concatenate the hidden states of the two sequences and feed the result to a dropout layer and then to a fully connected classification layer, followed by the softmax function, which returns the probability of the tweet having positive, neutral, or negative sentiment.

To try to reduce the overfitting and obtain a better model, we experimented with dropout probabilities ranging from 0 (no dropout) to 0.5 (approximately half the neurons' outputs are zeroed) in steps of 0.05. We evaluated these different settings on both the TweetSentBR and 2000-tweets-BR datasets.

The results for TweetSentBR are presented in Figure A.2 and Table A.1 on page 63. The results for 2000-tweets-BR are presented in Figure A.3 and Table A.2 on page 64.

Besides the aforementioned dropout layer, dropout is also used in the self-attention computation. We evaluated the same range of values, using the value of 0.35 for the general dropout probability with TweetSentBR and 0.05 for 2000-tweets-BR, as they performed the best.

The results for self-attention dropout for TweetSentBR are presented in Figure A.4 and Table A.3 on page 65. The results for 2000-tweets-BR are presented in Figure A.5 and Table A.4 on page 66.

To summarize the results we obtained in our dropout experiments, the best dropout settings for the TweetSentBR dataset are general dropout rate of 35% and self-attention dropout rate of 5%. As for the 2000-tweets-BR dataset, the best settings are general dropout rate of 5% and self-attention dropout rate of 15%.

4.5 Training Protocol

Starting with a pre-trained BERTimbau [62] model, we evaluated additional pre-training using the original architecture of BERT, according to Section 4.1. We then fine-tuned our model on the datasets using maximum sequence length of 128 tokens with padding, which means that sequences longer than 128 tokens are truncated, and sequences shorter than 128 tokens are padded so that every sequence in the batch has the same length. As analyzed in Chapter 3, the datasets we use in this work do not contain any sample longer than 128 words[1], so we did not have any truncated token.

We utilized AdamW [42] optimizer with initial learning rate of 1×10^{-5}, weight decay of 0.01, $\beta_1 = 0.9$, $\beta_2 = 0.999$, batch size of 32, and maximum number of epochs of 20 for TweetSentBR and 100 for 2000-tweets-BR, since it is smaller and the model was still learning in the 20th epoch in some cases. To determine the best values for the training parameters, we used a stratified 5-fold cross-validation schema for both datasets, on their training sets. With this setting, we get five values per run (one per fold), and the result reported as “validation” is the arithmetic mean of these five values. We opted to adopt cross-validation over a fixed training/validation split due to its robustness, at the expense of taking longer to execute. The test sets (holdout sets) are only used in the final fine-tuning runs, to obtain the final results.

4.6 Loss Function

To train our model we used the cross-entropy loss function. Since there are more than two classes, we must compute the loss for each class. Considering an input sample i, the model produces a vector x, whose length is C, the number of classes, containing the logits for each class. Let y_i be the target class of sample i, then the loss can be expressed by:

$$\ell(x, y) = -\log\left(\frac{\exp(x_y)}{\sum_{j=1}^{C}(\exp(x_j))}\right) = -x_y + \log\left(\sum_{j=1}^{C}(\exp(x_j))\right) \tag{4.1}$$

Since, in our case, the classes are unbalanced, we used weights to ensure that no class was neglected during the training process. For a dataset with a total of T samples, set of classes $\mathcal{C}$, and T_c samples of the class $c, \forall c \in \mathcal{C}$, the weight assigned to each class is given by:

$$w_c = \frac{T}{T_c},\ \forall c \in \mathcal{C} \tag{4.2}$$

[1]The longest sample in TweetSentBR has 54 words, and the longest one in 2000-tweets-BR has 37 words.

Considering these weights, the equation of the loss per class becomes:

$$\ell(x, y) = w_y \left(-x_y + \log \left(\sum_{j=1}^{C} (\exp(x_j)) \right) \right) \tag{4.3}$$

The losses are averaged across observations for each batch. For a batch size of N samples, the cross-entropy loss is given by:

$$\mathcal{L} = \frac{\sum_{i=1}^{N} (\ell(i, y_i))}{\sum_{i=1}^{N} (w_{y_i})} \tag{4.4}$$

4.7 Computational Resources

To develop our methodology we used the Python[2] programming language, PyTorch[3], the 🤗 Transformers[4] library from Hugging Face, BERTimbau[5], NumPy[6], Pandas[7], the Natural Language Toolkit[8], and scikit-learn[9]. To obtain the texts used to perform the additional pre-training we used the Odysci Media Analyzer[10]. Most of the experiments were executed on Google Colaboratory[11], with the following hardware specifications: 2.2 GHz Intel Xeon processors, 12 GB of RAM memory, and NVidia Tesla P100 graphics cards, with 16 GB of memory HBM2 and 3584 CUDA cores, running Linux operating system.

[2]`https://www.python.org`
[3]`https://pytorch.org`
[4]`https://huggingface.co/transformers`
[5]`https://github.com/neuralmind-ai/portuguese-bert`
[6]`https://numpy.org`
[7]`https://pandas.pydata.org`
[8]`https://www.nltk.org`
[9]`https://scikit-learn.org`
[10]`https://www.odysci.com`
[11]`https://colab.research.google.com`

Chapter 5

Experimental Results

We evaluate our methodology and compare the obtained results with the published results for the TweetSentBR and 2000-tweets-BR datasets. Since our model is based on BERT and we could not find published results for it on these datasets, we performed the evaluation of a standard BERT model using the same training protocol as our model. In all cases, the inputs are the full tweets, including emoji, if present.

5.1 Evaluation Metrics

The evaluation metrics traditionally used for the TweetSentBR and 2000-tweets-BR datasets are accuracy and F_1 score. We also present the results for precision and recall to offer a better representation of the effectiveness of the classifiers. Additionally, since the datasets used in this work are imbalanced, we computed the balanced accuracy and the balanced F_1 score. In order to keep the results as clear and unambiguous as possible, we include the definitions we used.

Accuracy is defined as:

$$Accuracy(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}(\mathbb{1}(\hat{y}_i = y_i)) \tag{5.1}$$

where $\hat{y}_i$ is the predicted label of the i-th sample, y_i is the corresponding true label, and N is the number of samples.

Balanced accuracy is defined as:

$$Balanced\ Accuracy = \frac{1}{|C|}\sum_{c \in C}\left(\frac{r_c}{n_c}\right) \tag{5.2}$$

where C is the set of classes, n_c is the number of samples from class c, and r_c is the number of samples from class c that were predicted correctly.

Precision and **recall** are defined as:

$$Precision(c) = \frac{TP_c}{TP_c + FP_c} \tag{5.3}$$

$$Recall(c) = \frac{TP_c}{TP_c + FN_c} \tag{5.4}$$

where TP_c, FP_c, and FN_c are the number of "true positives", "false positives", and "false negatives", respectively, regarding the class c. Since we are working with multi-class datasets, precision and recall are computed for each class.

F_1 score, also per class, is defined as:

$$F_1\ score(c) = 2 \times \frac{Precision(c) \times Recall(c)}{Precision(c) + Recall(c)} \tag{5.5}$$

To obtain one final value of F_1 score for a classifier, there are different options, such as macro-averaged F_1 score (macro-F_1), weighted-average F_1 score (weighted-F_1), and micro-averaged F_1 score (micro-F_1). We adopt the first option – macro-F_1 – which is likely the most-used option. We are aware of the existence of two methods to obtain the macro-F_1 [50], and we use the arithmetic mean over harmonic means version, as it is reported to be significantly more robust and, again, likely the most-used version. The macro-F_1 score is, then, defined as:

$$Macro\text{-}F_1\ score = \frac{1}{|C|} \sum_{c \in C} (F_1\ score(c)) \tag{5.6}$$

where C is the set of classes.

Balanced F_1 score is defined as:

$$Balanced\ F_1\ score = \frac{1}{\sum_{c \in C} |S_c|} \sum_{c \in C} (|S_c| \times F_1\ score(c)) \tag{5.7}$$

where S_c is the subset of S (the set of input samples) for the class c.

All the metrics have their values in the interval $[0, 1]$, and the higher the better.

5.2 Pre-Training Results

One of the first aspects we have to verify is whether additional pre-training improves the results of the final model or not. We experimented with six different pre-training configurations (tasks), presented in Section 4.1, namely Masked Language Modeling (MLM), Masked Language Modeling 50% (MLM50), All Emoji (All), First Emoji (First), Emoji Masked Language Modeling (EMLM), and Emoji Masked Language Modeling 50% (EMLM50), in addition to no further pre-training at all (None). The results for the Tweet-SentBR dataset are presented in Table 5.1. "Bal. Acc." and "Bal. F_1" stand for balanced accuracy and balanced F_1 score, respectively.

These results are from the sentiment classification fine-tuning phase using each of the pre-trained configurations. As we can see in the table, the best results were obtained with the Masked Language Modeling 50% configuration, which yielded the best results for all metrics. When considering only emoji-based configurations, the First Emoji (First) and Emoji Masked Language Modeling 50% (EMLM50) had similar performance rates –

Table 5.1: Pre-training experiment results for TweetSentBR – validation.

Config.	**Accuracy**	**Bal. Acc.**	**F_1 score**	**Bal. F_1**	**Precision**	**Recall**
None	0.7592	0.7476	0.7441	0.7621	0.7425	0.7476
MLM	0.7647	0.7531	0.7495	0.7670	0.7466	0.7531
MLM50	**0.7706**	**0.7576**	**0.7552**	**0.7727**	**0.7532**	**0.7576**
All	0.7567	0.7383	0.7389	0.7589	0.7432	0.7383
First	0.7627	0.7528	0.7487	0.7668	0.7489	0.7528
EMLM	0.7582	0.7445	0.7423	0.7607	0.7431	0.7445
EMLM50	0.7637	0.7500	0.7484	0.7659	0.7482	0.7500

difference of 0.10 percentage points (p.p.) in accuracy and of 0.03 p.p. in F_1 score – and were the best options, while the All Emoji (All) configuration produced the worst results of all configurations.

Table 5.2: Pre-training experiment results for 2000-tweets-BR – validation.

Config.	**Accuracy**	**Bal. Acc.**	**F_1 score**	**Bal. F_1**	**Precision**	**Recall**
None	**0.8144**	0.7665	**0.7939**	**0.8102**	**0.8416**	0.7665
MLM	0.7972	0.7680	0.7796	0.7952	0.7992	0.7680
MLM50	0.7938	0.7627	0.7758	0.7911	0.7992	0.7627
All	0.7938	0.7414	0.7729	0.7892	0.8310	0.7414
First	0.8041	**0.7871**	0.7896	0.8033	0.7981	**0.7871**
EMLM	0.7938	0.7477	0.7765	0.7900	0.8274	0.7477
EMLM50	0.8041	0.7641	0.7881	0.8013	0.8270	0.7641

Table 5.2 shows the results for the 2000-tweets-BR dataset. Except for the balanced accuracy and the recall metrics, the best results were obtained with no further pre-training. One possible reason for that is the nature of the pre-training data, which is not exactly from the same domain as the 2000-tweets-BR dataset. When considering only emoji-based configurations, the First Emoji (First) and Emoji Masked Language Modeling 50% (EMLM50) had similar performance rates – difference of 0.00 p.p. in accuracy and of 0.15 p.p. in F_1 score – and were the best options, similar to the TweetSentBR case. Also similar is the fact that the All Emoji (All) configuration produced the worst results of all configurations. This can be caused by the imbalance of the number of masked tokens, since each input sample has a different number of emoji.

5.3 Data Augmentation Results

Using an adaptation for the Portuguese language of the approach by Wei and Zou [77], we evaluated data augmentation for Sentiment Analysis. There are two important parameters in this method: α and n_{aug}. The former indicates the percentage of words in a sample to be changed in the augmented samples and the latter determines the number of augmented samples per original sample.

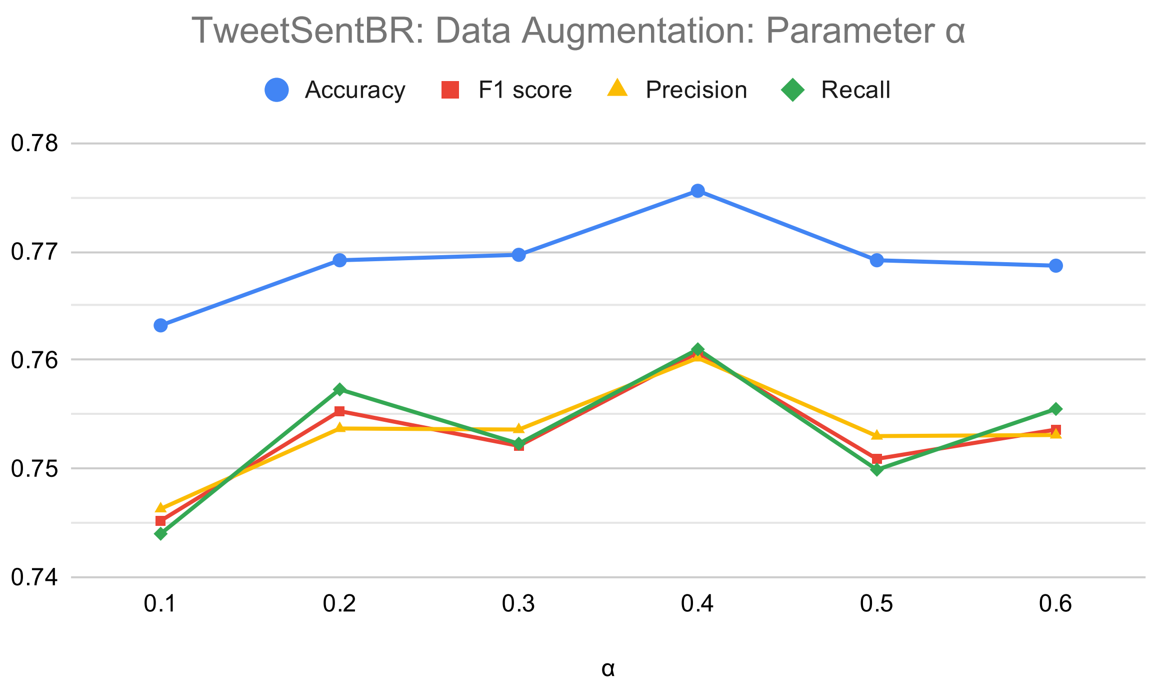


Figure 5.1: Data augmentation results for TweetSentBR – parameter α – validation.

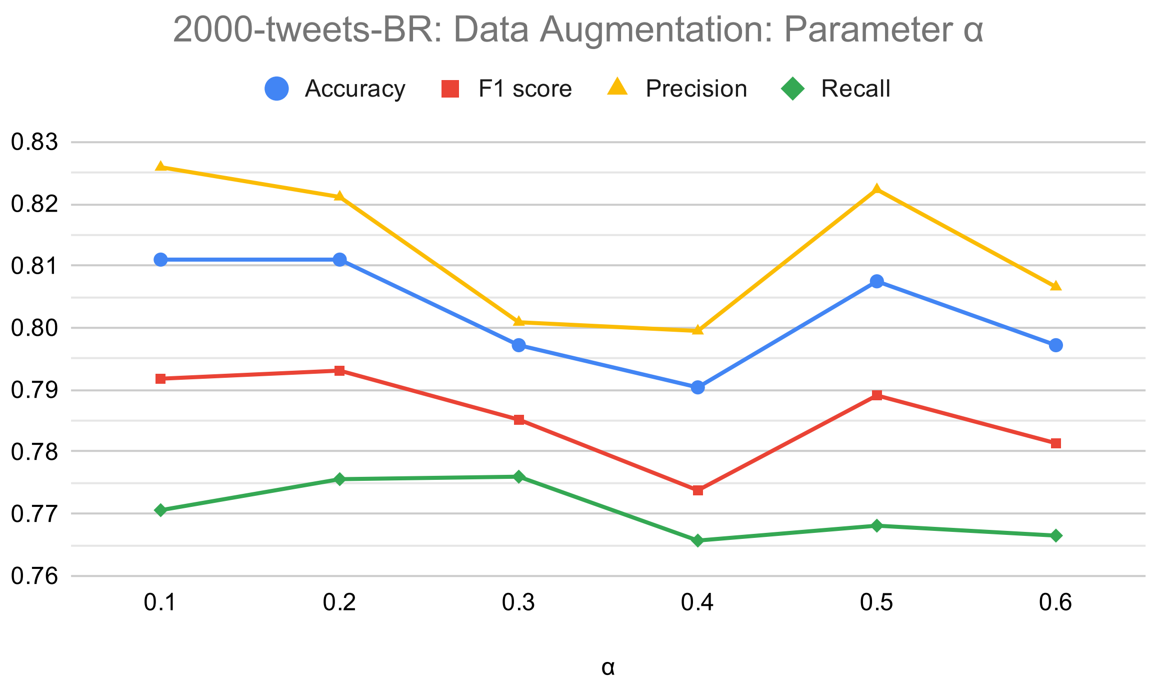


Figure 5.2: Data augmentation results for 2000-tweets-BR – parameter α – validation.

Figure 5.1 shows the results for the experiment to determine the best value for α considering the TweetSentBR dataset. To execute this experiment, we used $n_{aug} = 1$. These results are from the sentiment classification fine-tuning phase. We found 0.4 to be the best value, producing accuracy of 0.7756 and F$_1$ score of 0.7607. The complete results can be found in Table A.5 on page 67.

The results of the experiment to determine the best value for α considering the 2000-tweets-BR dataset are presented in Figure 5.2. Again we fixed $n_{aug} = 1$ and varied only α. The best value for α is not as evident as it is in the case of TweetSentBR. We considered it to be 0.2 because it yielded the best results for F$_1$ score, balanced F$_1$ score (not shown in Figure 5.2), and accuracy (tied with $\alpha = 0.1$). For the complete results, please refer to Table A.6 on page 67.

As for the parameter n_{aug}, the results are presented in Figure 5.3 for the TweetSentBR dataset. We used $\alpha = 0.4$, according to the previous experiment, and found $n_{aug} = 3$ to be the best value, producing accuracy of 0.7762 and F$_1$ score of 0.7625. The complete results can be found in Table A.7 on page 67.

The results of the experiment to determine the best value for n_{aug} considering the 2000-tweets-BR dataset are presented in Figure 5.4. We used $\alpha = 0.2$, according to the previous experiment. Similar to the results for TweetSentBR, we found $n_{aug} = 3$ to be the best value for 2000-tweets-BR as well, yielding accuracy of 0.8245 and F$_1$ score of 0.8037. For the complete results, please refer to Table A.8 on page 68.

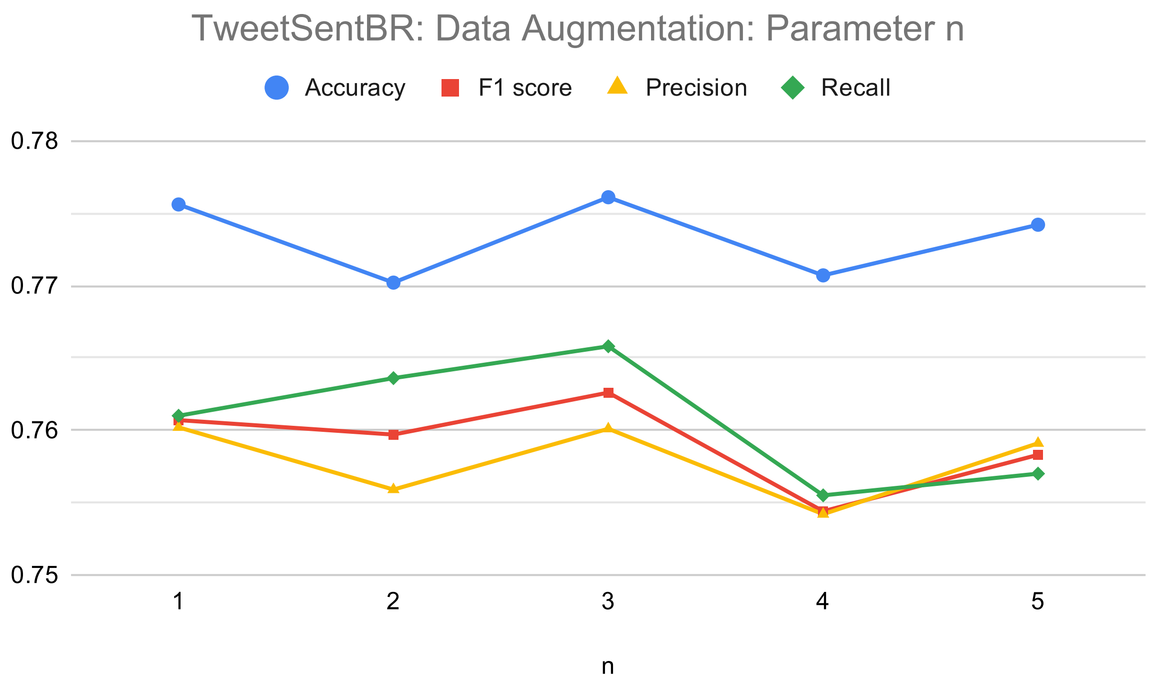


Figure 5.3: Data augmentation results for TweetSentBR – parameter n_{aug} – validation.

Summarizing the results we obtained in our data augmentation experiments, the best data augmentation schema for the TweetSentBR dataset is 3 augmented samples per original sample, with 40% of the words changed. For the 2000-tweets-BR dataset, the best schema is 3 augmented samples per original sample, with 20% of the words changed.

Having found the best settings for data augmentation, we compared those results with those obtained without data augmentation to determine whether it is beneficial to the

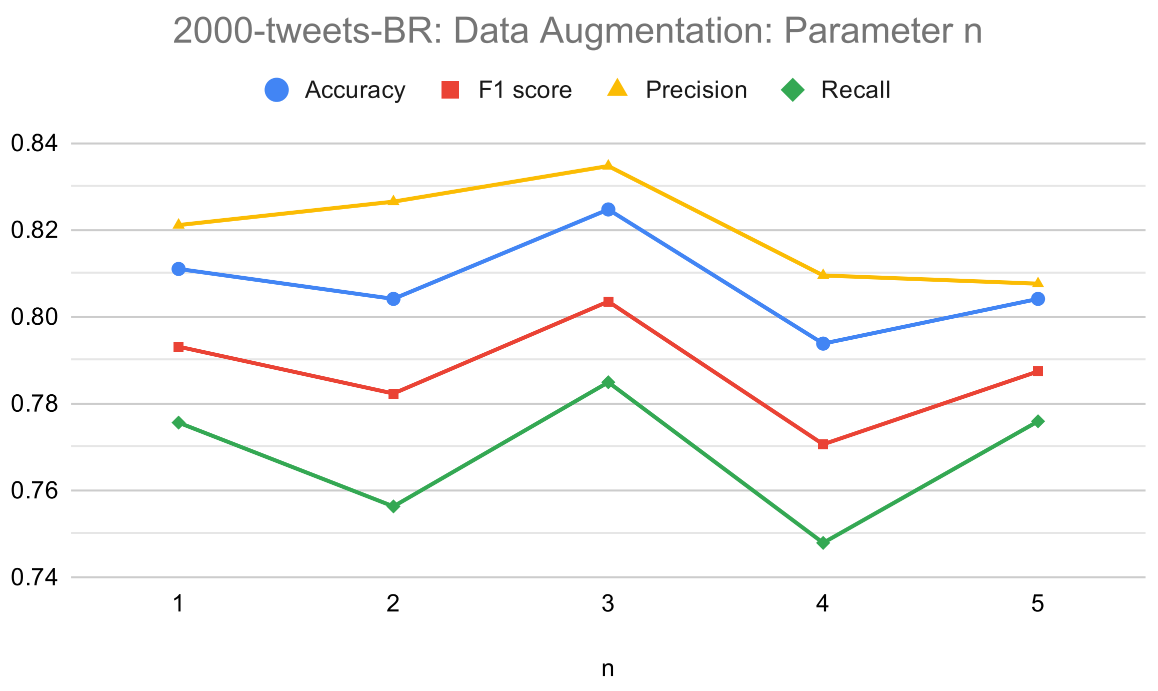


Figure 5.4: Data augmentation results for 2000-tweets-BR – parameter n_{aug} – validation.

method or not. Table 5.3 shows this comparison for the TweetSentBR dataset. "Bal. Acc." and "Bal. F_1" stand for balanced accuracy and balanced F_1 score, respectively. From the table, we can see that the classifier using data augmentation yielded better results for all the metrics considered.

Table 5.3: Data augmentation experiment results for TweetSentBR – validation.

Aug.	**Accuracy**	**Bal. Acc.**	**F_1 score**	**Bal. F_1**	**Precision**	**Recall**
No	0.7751	0.7648	0.7611	0.7776	0.7591	0.7648
Yes	**0.7762**	**0.7657**	**0.7625**	**0.7792**	**0.7602**	**0.7657**

Table 5.4: Data augmentation experiment results for 2000-tweets-BR – validation.

Aug.	**Accuracy**	**Bal. Acc.**	**F_1 score**	**Bal. F_1**	**Precision**	**Recall**
No	0.8213	**0.7856**	**0.8049**	0.8186	**0.8353**	**0.7856**
Yes	**0.8245**	0.7851	0.8037	**0.8212**	0.8346	0.7851

Table 5.4 shows the comparison for the 2000-tweets-BR dataset. The best option is not evident since the results do not differ much from each other. We performed the Wilcoxon signed-rank test [78] for all six metrics and found no statistical difference between the augmented and non-augmented classifiers. We also performed the same test with respect to the TweetSentBR dataset (cf. Table 5.3), which yielded a similar result. Since, for both datasets, we did not perceive statistically meaningful differences between the augmented

and non-augmented classifiers and the augmented version produced numerically superior results for TweetSentBR, we opted to adopt data augmentation for the 2000-tweets-BR dataset as well.

5.4 Fine-Tuning Results

Table 5.5 lists the experimental results for the TweetSentBR dataset. Note that the values for precision and recall are the arithmetic means of their per-class values, so as not to clutter the table with one entry for each class.

Brum and Nunes [10], who introduced the TweetSentBR dataset, used the Naïve Bayes [60] and Support Vector Machine (SVM) [27] classifiers to classify the sentiment of the tweets. The best results were obtained with the Naïve Bayes classifier, so we consider these to be the baseline results. Brum and Nunes [9], in a subsequent work, used Naïve Bayes, SVM, Logistic Regression (LR), Multi-Layer Perceptron (MLP), Decision Trees, and Random Forest classifiers. In this case, Multi-Layer Perceptron was the best-performing classifier, as listed in the table.

Sakiyama et al. [61] developed a breaking news event detector based on the time series of the number of positive, neutral, and negative tweets obtained from a Sentiment Analysis classifier, which was based on a Convolutional Neural Network (CNN) [34]. Nascimento [47] employed ensembles of classifiers in his work. The best-performing ensemble is composed of Multi-Layer Perceptron, Logistic Regression, and Gaussian Naïve Bayes classifiers.

When building the model, we used the same size parameters as $\text{BERT}_{\text{BASE}}$ ($L = 12$, $H = 768$, and $A = 12$), enabling us to make a fair comparison against it. The entry "*Our model*" shows the results obtained when using only our model (cf. Figure 4.2) in fine-tuning, while the entry "*Our model* + PT + DA" shows the results obtained when using our model in tandem with additional pre-training and data augmentation. Our methodology achieves absolute improvements of 6.6 percentage points (p.p.) in accuracy over the ensemble classifier and 2.9 p.p. over BERT; and 10.6 p.p. in F_1 score over TextCNN and 3.3 p.p. over BERT.

Regarding the 2000-tweets-BR dataset, the experimental results are presented in Table 5.6. Vitório et al. [73], who introduced the dataset, used Support Vector Machines (SVM) [27] with linear kernel to classify the sentiment of the tweets. Nascimento [47] obtained the best results using Stochastic Gradient Descent. For the 2000-tweets-BR dataset, using only our model produced the best results, nonetheless our full methodology achieved good results in all metrics, improving the accuracy by 14.5 p.p. over the Stochastic Gradient Descent classifier and 1.4 p.p. over BERT; and improving the F_1 score by 23.4 p.p. over the Stochastic Gradient Descent classifier and 1.0 p.p. over BERT.

Since our approach focuses on emoji, we tested on a subset of tweets that have one or more emoji, which are about 20% of the TweetSentBR dataset, as seen in Table 3.3. With this subset, we can compare the effectiveness of $\text{BERT}_{\text{BASE}}$ and our model when dealing with emoji-occurring tweets. As can be seen in Table 5.7, in this scenario, our model excelled $\text{BERT}_{\text{BASE}}$ by 4.9 p.p. in accuracy and by 17.9 p.p. in F_1 score, indicating

Table 5.5: Fine-tuning results for TweetSentBR – test.

Authors	**Classifier**	**Accuracy**	**Bal. Acc.**	**F_1 score**	**Bal. F_1**	**Precision**	**Recall**
Brum and Nunes [10]	Naïve Bayes	0.6462	–	0.5985	–	–	–
Brum and Nunes [9]	Multi-Layer Perceptron	–	–	0.6214	–	–	–
Sakiyama et al. [61]	TextCNN	0.6840	–	0.6560	–	–	–
Nascimento [47]	Ensemble MLP + LR + GNB[a]	0.7100	–	0.5000	–	–	–
–	$\text{BERT}_{\text{BASE}}$	0.7468	0.7297	0.7292	0.7546	0.7287	0.7297
–	*Our model*	0.7577	0.7396	0.7395	0.7569	0.7394	0.7396
–	*Our model* + PT + DA[b]	**0.7761**	**0.7658**	**0.7626**	**0.7790**	**0.7601**	**0.7658**

[a]Multi-Layer Perceptron + Logistic Regression + Gaussian Naïve Bayes.
[b]Pre-Training + Data Augmentation

Table 5.6: Fine-tuning results for 2000-tweets-BR – test.

Authors	**Classifier**	**Accuracy**	**Bal. Acc.**	**F_1 score**	**Bal. F_1**	**Precision**	**Recall**
Vitório et al. [73]	Linear SVM	0.6451	–	–	–	–	–
Nascimento [47]	Stochastic Gradient Descent	0.6800	–	0.5700	–	–	–
–	$\text{BERT}_{\text{BASE}}$	0.8110	0.7807	0.7937	0.8086	0.8135	0.7807
–	*Our model*	**0.8316**	**0.8005**	**0.8151**	**0.8297**	**0.8358**	**0.8005**
–	*Our model* + PT + DA[c]	0.8247	0.7849	0.8035	0.8211	0.8347	0.7849

[c]Pre-Training + Data Augmentation

that our model can actually extract more information from emoji and use that to perform a better sentiment classification.

Table 5.7: Fine-tuning results for TweetSentBR – emoji subset – test.

Classifier	**Accuracy**	**Bal. Acc.**	**F_1 score**	**Bal. F_1**	**Precision**	**Recall**
$\text{BERT}_{\text{BASE}}$	0.7724	0.5747	0.5631	0.7170	**0.8342**	0.5747
Our model	**0.8208**	**0.7607**	**0.7425**	**0.8193**	0.7311	**0.7607**

Table 5.8: Fine-tuning results for 2000-tweets-BR – emoji subset – test.

Classifier	**Accuracy**	**Bal. Acc.**	**F_1 score**	**Bal. F_1**	**Precision**	**Recall**
$\text{BERT}_{\text{BASE}}$	0.7073	**0.6587**	**0.6779**	0.7045	0.7146	**0.6587**
Our model	**0.7317**	0.6270	0.6652	**0.7162**	**0.8323**	0.6270

For the 2000-tweets-BR dataset, tweets that have one or more emoji represent about 15% of the dataset, as can be seen in Table 3.7. Considering the subset formed by such tweets, the results are listed in Table 5.8. We can see that our model outperforms BERT in the accuracy metric by 2.4 p.p., but is surpassed in the F_1 score by 1.3 p.p., since the lower recall value pulls the F_1 score down, even though the precision is high. It may be caused by having emoji which contradict the text in tweet or even by inconsistent labels in the dataset.

Interestingly, the balanced metrics gave opposing results with relation to their non-balanced counterparts and the comparison between the two classifiers. Since we have six metrics and each classifier performed best in three of them, one could say that they are equally competent in this scenario. We performed the Wilcoxon signed-rank test [78] for all six metrics and found no statistical difference between the two classifiers, so we can consider that they performed equally well.

Chapter 6

Conclusion and Future Work

Sentiment Analysis is currently a relevant research field within Natural Language Processing and likely to increase in importance as more and more user-generated content is created and becomes available. In our competitive society, having a better understanding of what people think can represent a huge advantage for established companies and startups alike, for instance evaluating user reception of a proposed new product.

Deep Learning methods usually produce competitive results but often require copious amounts of labeled data, which, for many languages, are hard to acquire because it is very time-consuming to build a large manually-annotated dataset – the gold standard in Machine Learning. Unsupervised language representation learning methods alleviate this issue by means of Transfer Learning, pre-training a language representation model using a large amount of unlabeled data and then fine-tuning it on a labeled dataset, which does not need to be too large.

One prominent such model is the Bidirectional Encoder Representations from Transformers (BERT) [16], used as a basis to develop our method. We focused on extracting information not only from text but also from emoji, which frequently are used in the social media environment to add expressiveness and set the tone of a message. We extract emoji from the input text and process them through the Transformer [72] encoder independently, then combine the hidden states corresponding to the text and emoji before sending them to the classification layer of the model.

Our experiments with two datasets of tweets in Brazilian Portuguese – TweetSentBR (TTsBR) [10] and 2000-tweets-BR [73] – produced compelling results, surpassing the previously published results for both datasets and establishing new state-of-the-art results on TweetSentBR with accuracy of 0.7761 (improvement of 6.6 percentage points (p.p.)) and F_1 score of 0.7626 (improvement of 10.6 p.p.); and on 2000-tweets-BR with accuracy of 0.8247 (improvement of 14.5 p.p.) and F_1 score of 0.8035 (improvement of 23.4 p.p.). It is possible to use a previously pre-trained $\text{BERT}_{\text{BASE}}$ model to warm start ours, greatly reducing the total training time.

We also present a study of emoji occurrence and distribution for the most frequent emoji in the TweetSentBR and 2000-tweets-BR datasets, and compare the results with general emoji usage in Twitter obtained via Emojitracker. We found that, in general, emoji occurrence in these datasets is similar to the overall emoji occurrence in Twitter, although we observed fluctuations when the number of samples is small, because tweets with many occurrences of the same emoji can distort the results. Also, we found that emoji are more frequent in positive contexts.

Regarding the research questions we posed in Section 1.3, we found that emoji, considered alongside their corresponding texts, can improve the sentiment classification. Also, additional pre-training using in-domain data improved the results for the TweetSentBR dataset but not for the 2000-tweets-BR dataset. Considering that the latter is a multi-domain dataset – so it is not trivial to obtain great quantity of similar data – and the additional pre-training was performed using data similar to TweetSentBR, we think that further research on this topic may improve the outcome. Finally, data augmentation produced marginally better results for the TweetSentBR dataset but not for 2000-tweets-BR, and in both cases the difference was not statistically significant.

As future work, we intend to investigate the use of a more sophisticated method of data augmentation, such as the one proposed by Kumar et al. [38]. Additionally, we would like to apply our methodology to datasets in other languages, in particular English, since resources in this language are more plentiful. We also plan to evaluate the application of other Transformer models other than BERT as the basis for our approach. In addition, we intend to evaluate the model using positional data for emoji according to the original text, instead of the positional data obtained after the split. Finally, we think that exploring different forms of text preprocessing may yield further gains.

Bibliography

[1] Adobe. Emoji Trend Report 2019. `https://www.slideshare.net/adobe/adobe-emoji-trend-report-2019`, 2019. Accessed: 2020-10-31.

[2] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer Normalization. *Computing Research Repository*, pages 1–14, 2016.

[3] D. Bahdanau, K. Cho, and Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations (ICLR)*, pages 1–15, San Diego, USA, 2015. International Conference on Learning Representations (ICLR), ICLR.

[4] F. Barbieri, F. Ronzano, and H. Saggion. What does this Emoji Mean? A Vector Space Skip-Gram Model for Twitter Emojis. In *10th International Conference on Language Resources and Evaluation (LREC)*, pages 3967–3972, Portorož, Slovenia, 2016. European Language Resources Association (ELRA), ELRA.

[5] T. M. Barros, H. Pedrini, and Z. Dias. Leveraging Emoji to Improve Sentiment Classification of Tweets. In *36th ACM/SIGAPP Symposium on Applied Computing (SAC)*, pages 845–852, Gwangju, Republic of Korea, 2021. Association for Computing Machinery (ACM), ACM.

[6] T. M. Barros, H. Pedrini, and Z. Dias. Data-Augmented Emoji Approach to Sentiment Classification of Tweets. In *25th Iberoamerican Congress on Pattern Recognition (CIARP)*, pages 1–10, Porto, Portugal, 2021. Iberoamerican Congress on Pattern Recognition (CIARP), Springer.

[7] S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media Inc., Sebastopol, USA, 2009.

[8] E. Boiy and M.-F. Moens. A Machine Learning Approach to Sentiment Analysis in Multilingual Web Texts. *Information Retrieval*, 12(5):526–558, 2009.

[9] H. B. Brum and M. d. G. V. Nunes. Semi-supervised Sentiment Annotation of Large Corpora. In *13th International Conference on Computational Processing of the Portuguese Language (PROPOR)*, pages 385–395, Canela, Brazil, 2018. Organization Committee of the International Conference on Computational Processing of the Portuguese Language (OC-PROPOR), Springer.

[10] H. B. Brum and M. G. V. Nunes. Building a Sentiment Corpus of Tweets in Brazilian Portuguese. In *11th International Conference on Language Resources and Evaluation (LREC)*, pages 4167–4172, Miyazaki, Japan, 2018. European Language Resources Association (ELRA), ELRA.

[11] E. Cambria, B. Schuller, Y. Xia, and C. Havasi. New Avenues in Opinion Mining and Sentiment Analysis. *IEEE Intelligent Systems*, 28(2):15–21, 2013.

[12] Y. Chen, J. Yuan, Q. You, and J. Luo. Twitter Sentiment Analysis via Bi-sense Emoji Embedding and Attention-based LSTM. In *26th ACM International Conference on Multimedia (MM)*, pages 117–125, Seoul, Republic of Korea, 2018. Association for Computing Machinery (ACM), ACM.

[13] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *Computing Research Repository*, pages 1–9, 2014.

[14] A. M. Dai and Q. V. Le. Semi-Supervised Sequence Learning. In *29th Conference on Neural Information Processing Systems (NIPS)*, pages 3079–3087, Montréal, Canada, 2015. Neural Information Processing Systems (NIPS) Foundation, Curran Associates, Inc.

[15] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. Le, and R. Salakhutdinov. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. In *57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2978–2988, Florence, Italy, 2019. Association for Computational Linguistics (ACL), ACL.

[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *20th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT): Long and Short Papers - Volume 1*, pages 4171–4186, Minneapolis, USA, 2019. Association for Computational Linguistics (ACL), ACL.

[17] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *Computing Research Repository*, pages 1–21, 2020.

[18] J. L. Elman. Finding Structure in Time. *Cognitive Science*, 14(2):179–211, 1990.

[19] Emojipedia. 5 Billion Emoji Sent Daily on Messenger. `https://blog.emojipedia.org/5-billion-emojis-sent-daily-on-messenger`, 2017. Accessed: 2020-10-30.

[20] Emojipedia. Correcting the Record on the First Emoji Set. `https://blog.emojipedia.org/correcting-the-record-on-the-first-emoji-set/`, 2019. Accessed: 2020-10-31.

[21] Emojipedia. Emoji Statistics [Updated September 2020]. `https://emojipedia.org/stats`, 2020. Accessed: 2020-10-30.

[22] B. Felbo, A. Mislove, A. Søgaard, I. Rahwan, and S. Lehmann. Using Millions of Emoji Occurrences to Learn Any-domain Representations for Detecting Sentiment, Emotion and Sarcasm. In *22nd Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1615–1625, Copenhagen, Denmark, 2017. Association for Computational Linguistics (ACL).

[23] A. Go, R. Bhayani, and L. Huang. Twitter Sentiment Classification using Distant Supervision. *CS224N project report, Stanford*, 1(12):1–6, 2009.

[24] R. González-Ibáñez, S. Muresan, and N. Wacholder. Identifying Sarcasm in Twitter: A Closer Look. In *49th Annual Meeting of the Association for Computational Linguistics (ACL): Human Language Technologies (HLT): Short Papers - Volume 2*, pages 581–586. Association for Computational Linguistics (ACL), 2011.

[25] S. Gururangan, A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, and N. A. Smith. Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. *Computing Research Repository*, pages 1–19, 2020.

[26] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *29th Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Las Vegas, USA, 2016. Institute of Electrical and Electronics Engineers (IEEE), IEEE.

[27] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. Support Vector Machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28, 1998.

[28] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[29] J. Howard and S. Ruder. Universal Language Model Fine-Tuning for Text Classification. In *56th Annual Meeting of the Association for Computational Linguistics (ACL): Long Papers - Volume 1*, pages 328–339, Melbourne, Australia, 2018. Association for Computational Linguistics (ACL), ACL.

[30] M. Hu and B. Liu. Mining and Summarizing Customer Reviews. In *10th International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 168–177. Association for Computing Machinery (ACM), 2004.

[31] A. H. Huang, D. C. Yen, and X. Zhang. Exploring the Potential Effects of Emoticons. *Information and Management*, 45(7):466–473, 2008.

[32] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *32nd International Conference on Machine Learning (ICML)*, pages 448–456, Lille, France, 2015. International Conference on Machine Learning (ICML), Proceedings of Machine Learning Research (PMLR).

[33] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. In *37th International Conference on Machine Learning (ICML)*, pages 5156–5165, Vienna, Austria, 2020. International Conference on Machine Learning (ICML), Proceedings of Machine Learning Research (PMLR).

[34] Y. Kim. Convolutional Neural Networks for Sentence Classification. In *19th Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, 2014. Association for Computational Linguistics (ACL).

[35] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-Thought Vectors. In *29th Conference on Neural Information Processing Systems (NIPS)*, pages 3294–3302. Neural Information Processing Systems (NIPS) Foundation, 2015.

[36] N. Kitaev, L. Kaiser, and A. Levskaya. Reformer: The Efficient Transformer. In *7th International Conference on Learning Representations (ICLR)*, pages 1–11, New Orleans, USA, 2019. International Conference on Learning Representations (ICLR), ICLR.

[37] N. Kobayashi, K. Inui, and Y. Matsumoto. Opinion Mining from Web Documents: Extraction and Structurization. *Information and Media Technologies*, 2(1):326–337, 2007.

[38] V. Kumar, A. Choudhary, and E. Cho. Data Augmentation using Pre-trained Transformer Models. In *2nd Workshop on Life-long Learning for Spoken Language Systems at 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (AACL)*, pages 18–26, Suzhou, China, 2020. Asia-Pacific Chapter of the Association for Computational Linguistics (AACL), AACL.

[39] Q. Le and T. Mikolov. Distributed Representations of Sentences and Documents. In *31st International Conference on Machine Learning (ICML)*, pages 1188–1196. International Conference on Machine Learning (ICML), 2014.

[40] S. Lee. Emoji at MoMA: Considering the ‘original emoji’ as art. *First Monday*, 23 (9), 2018.

[41] B. Liu. Sentiment Analysis and Opinion Mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167, 2012.

[42] I. Loshchilov and F. Hutter. Decoupled Weight Decay Regularization. In *7th International Conference on Learning Representations (ICLR)*, pages 1–10, New Orleans, USA, 2019. International Conference on Learning Representations (ICLR), ICLR.

[43] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. *Computing Research Repository, abs/1301.3781*, pages 1–15, 2013.

[44] H. Miller, J. Thebault-Spieker, S. Chang, I. Johnson, L. Terveen, and B. Hecht. "Blissfully Happy" or "Ready to Fight": Varying Interpretations of Emoji. In *10th International Conference on Web and Social Media (ICWSM)*, pages 259–268, Cologne, Germany, 2016. Association for the Advancement of Artificial Intelligence (AAAI), AAAI Press.

[45] S. Morinaga, K. Yamanishi, K. Tateishi, and T. Fukushima. Mining Product Reputations on the Web. In *8th International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 341–349, Edmonton, Canada, 2002. Association for Computing Machinery (ACM), ACM.

[46] M. Nakatsuji and Y. Fujiwara. Linked Taxonomies to Capture Users' Subjective Assessments of Items to Facilitate Accurate Collaborative Filtering. *Artificial Intelligence*, 207:52–68, 2014.

[47] P. d. A. Nascimento. Aplicando Ensemble para Classificação de Textos Curtos em Português do Brasil. Master's thesis, Universidade Federal de Pernambuco, Recife, Brazil, 2019.

[48] T. Nasukawa and J. Yi. Sentiment Analysis: Capturing Favorability Using Natural Language Processing. In *2nd International Conference on Knowledge Capture (K-CAP)*, pages 70–77, Florida, USA, 2003. Association for Computing Machinery (ACM), ACM.

[49] P. K. Novak, J. Smailović, B. Sluban, and I. Mozetič. Sentiment of Emojis. *PLOS One*, 10(12):1–22, 2015.

[50] J. Opitz and S. Burst. Macro F1 and Macro F1. *Computing Research Repository*, pages 1–12, 2019.

[51] A. Pak and P. Paroubek. Twitter as a Corpus for Sentiment Analysis and Opinion Mining. In *7th International Conference on Language Resources and Evaluation (LREC)*, pages 1320–1326, Valletta, Malta, 2010. European Language Resources Association (ELRA), ELRA.

[52] S. J. Pan, X. Ni, J.-T. Sun, Q. Yang, and Z. Chen. Cross-Domain Sentiment Classification via Spectral Feature Alignment. In *19th International Conference on World Wide Web (WWW)*, pages 751–760. Association for Computing Machinery (ACM), 2010.

[53] B. Pang and L. Lee. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. In *42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 271–278, Barcelona, Spain, 2004. Association for Computational Linguistics (ACL), ACL.

[54] B. Pang and L. Lee. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. In *43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 115–124, Ann Arbor, USA, 2005. Association for Computational Linguistics (ACL), ACL.

[55] B. Pang and L. Lee. Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval*, 2(1–2):1–135, 2008.

[56] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs Up? Sentiment Classification using Machine Learning Techniques. In *7th Conference on Empirical Methods in Natural Language Processing (EMNLP)*, volume 10, pages 79–86. Association for Computational Linguistics (ACL), 2002.

[57] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep Contextualized Word Representations. In *16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT): Long Papers - Volume 1*, pages 2227–2237, New Orleans, USA, 2018. Association for Computational Linguistics (ACL), ACL.

[58] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving Language Understanding by Generative Pre-Training. Technical report, OpenAI, 2018.

[59] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[60] I. Rish. An Empirical Study of the Naive Bayes Classifier. In *17th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 41–46, Seattle, USA, 2001. International Joint Conferences on Artificial Intelligence (IJCAI), IJCAI.

[61] K. M. Sakiyama, A. Q. B. Silva, and E. T. Matsubara. Twitter Breaking News Detector in the 2018 Brazilian Presidential Election using Word Embeddings and Convolutional Neural Networks. In *37th International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, Budapest, Hungary, 2019. Institute of Electrical and Electronics Engineers (IEEE), IEEE.

[62] F. Souza, R. Nogueira, and R. Lotufo. BERTimbau: Pretrained BERT Models for Brazilian Portuguese. In *9th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 403–417, Rio Grande, Brazil, 2020. Sociedade Brasileira de Computação (SBC), Springer.

[63] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to Sequence Learning with Neural Networks. In *28th Conference on Neural Information Processing Systems (NIPS)*, volume 27, pages 3104–3112. Neural Information Processing Systems (NIPS) Foundation, 2014.

[64] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede. Lexicon-Based Methods for Sentiment Analysis. *Computational Linguistics*, 37(2):267–307, 2011.

[65] W. L. Taylor. "Cloze Procedure": A New Tool for Measuring Readability. *Journalism Quarterly*, 30(4):415–433, 1953.

[66] Y. Tian, T. Galery, G. Dulcinati, E. Molimpakis, and C. Sun. Facebook Sentiment: Reactions and Emojis. In *5th International Workshop on Natural Language Processing for Social Media (SocialNLP)*, pages 11–16, Valencia, Spain, 2017. Association for Computational Linguistics (ACL), ACL.

[67] M. Tsytsarau and T. Palpanas. Survey on Mining Subjective Data on the Web. *Data Mining and Knowledge Discovery*, 24(3):478–514, 2012.

[68] P. D. Turney. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. In *40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 417–424. Association for Computational Linguistics (ACL), 2002.

[69] P. D. Turney and M. L. Littman. Measuring Praise and Criticism: Inference of Semantic Orientation from Association. *ACM Transactions on Information Systems*, 21(4):315–346, 2003.

[70] Unicode. Unicode Version 6.0: Support for Popular Symbols in Asia. `http://blog.unicode.org/2010/10/unicode-version-60-support-for-popular.html`, 2010. Accessed: 2020-12-11.

[71] Unicode. Unicode & Emoji. `https://unicode.org/emoji/slides.html`, 2017. Accessed: 2020-12-12.

[72] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is All You Need. In *31st Conference on Neural Information Processing Systems (NIPS)*, pages 5998–6008, Long Beach, USA, 2017. Neural Information Processing Systems (NIPS) Foundation, Curran Associates, Inc.

[73] D. Vitório, E. Souza, I. Teles, and A. L. Oliveira. Investigating Opinion Mining through Language Varieties: a Case Study of Brazilian and European Portuguese tweets. In *11th Brazilian Symposium in Information and Human Language Technology (STIL)*, pages 43–52, Uberlândia, Brazil, 2017. Sociedade Brasileira de Computação (SBC), SBC.

[74] J. A. Wagner Filho, R. Wilkens, M. Idiart, and A. Villavicencio. The brWaC Corpus: A New Open Resource for Brazilian Portuguese. In *11th International Conference on Language Resources and Evaluation (LREC)*, pages 4339–4344, Miyazaki, Japan, 2018. European Language Resources Association (ELRA), ELRA.

[75] H. Wang, D. Can, A. Kazemzadeh, F. Bar, and S. Narayanan. A System for Real-time Twitter Sentiment Analysis of 2012 U.S. Presidential Election Cycle. In *50th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 115–120, Jeju, Republic of Korea, 2012. Association for Computational Linguistics (ACL), ACL.

[76] S. Wang, B. Li, M. Khabsa, H. Fang, and H. Ma. Linformer: Self-Attention with Linear Complexity. *Computing Research Repository*, pages 1–12, 2020.

[77] J. Wei and K. Zou. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. In *24th Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6383–6389, Hong Kong, China, 2019. Association for Computational Linguistics (ACL), ACL.

[78] F. Wilcoxon. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1 (6):80–83, 1945.

[79] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *Computing Research Repository*, pages 1–23, 2016.

[80] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. Hierarchical Attention Networks for Document Classification. In *15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1480–1489, San Diego, USA, 2016. Association for Computational Linguistics (ACL), ACL.

[81] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *33rd Conference on Neural Information Processing Systems (NeurIPS)*, pages 5753–5763, Vancouver, Canada, 2019. Neural Information Processing Systems (NeurIPS) Foundation, Curran Associates, Inc.

[82] L. Zhang, S. Wang, and B. Liu. Deep Learning for Sentiment Analysis: A Survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253, 2018.

Appendix A

Additional Results

In this chapter, we include graphs and tables with all the numeric results that do not appear on the main text. "Bal. Acc." and "Bal. F_1" stand for balanced accuracy and balanced F_1 score, respectively. The definition of every evaluation metric used can be found in Section 5.1. Entries highlighted in bold face in the tables are the best values for the corresponding evaluation metrics.

A.1 Pre-Training Experiments

Figure A.1 shows the F_1 score per pre-training epoch for the TweetSentBR dataset using the Masked Language Modeling configuration. It was used to determine the number of pre-training epochs for the additional pre-training.

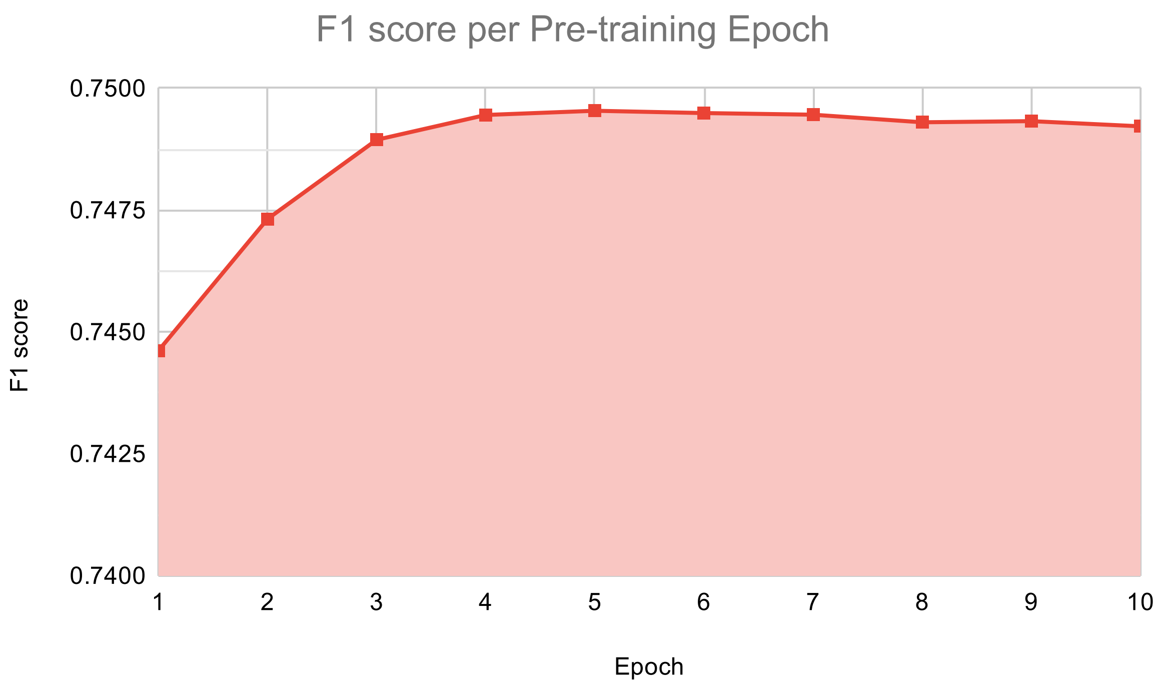


Figure A.1: F_1 score per pre-training epoch for TweetSentBR – validation.

A.2 Dropout Experiments

Figure A.2 and Table A.1 present the results of general dropout for the TweetSentBR dataset.

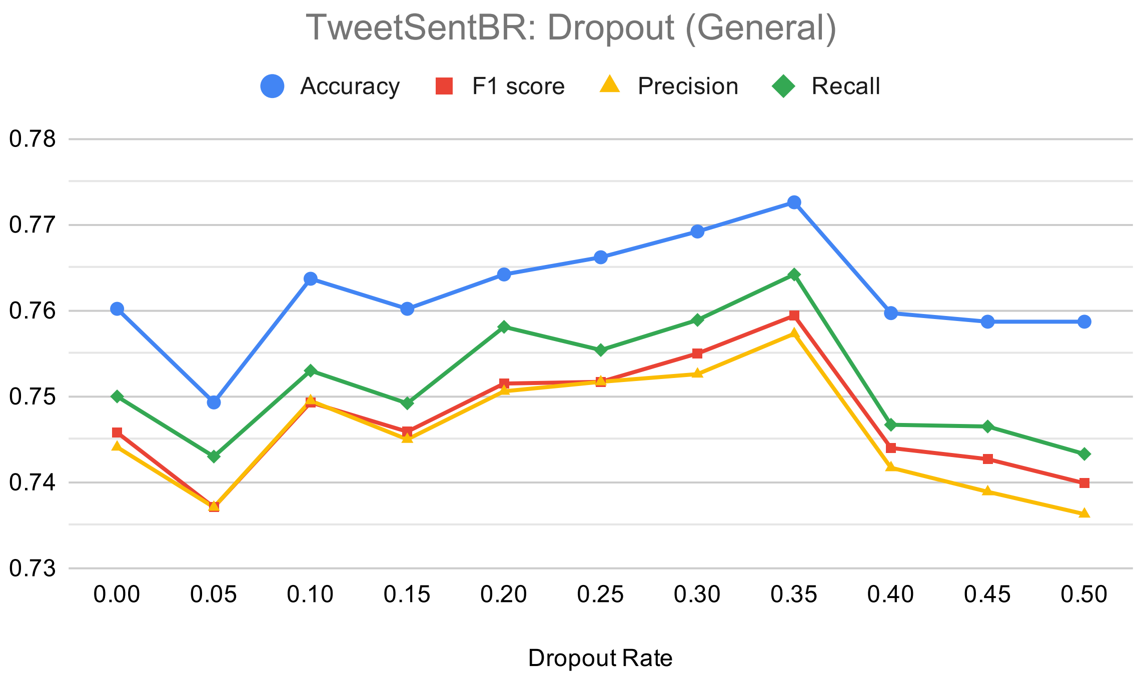


Figure A.2: Dropout experiments results for TweetSentBR – validation.

Table A.1: Dropout experiments results for TweetSentBR – validation.

Rate	Accuracy	Bal. Acc.	F_1 score	Bal. F_1	Precision	Recall
0.00	0.7602	0.7500	0.7458	0.7635	0.7441	0.7500
0.05	0.7493	0.7430	0.7371	0.7538	0.7371	0.7430
0.10	0.7637	0.7530	0.7493	0.7668	0.7495	0.7530
0.15	0.7602	0.7492	0.7459	0.7629	0.7450	0.7492
0.20	0.7642	0.7581	0.7515	0.7687	0.7506	0.7581
0.25	0.7662	0.7554	0.7517	0.7694	0.7517	0.7554
0.30	0.7692	0.7589	0.7550	0.7719	0.7526	0.7589
0.35	**0.7726**	**0.7642**	**0.7594**	**0.7757**	**0.7573**	**0.7642**
0.40	0.7597	0.7467	0.7440	0.7618	0.7417	0.7467
0.45	0.7587	0.7465	0.7427	0.7605	0.7389	0.7465
0.50	0.7587	0.7433	0.7399	0.7596	0.7363	0.7433

For all metrics considered, the best values were obtained with 35% of general dropout.

Figure A.3 and Table A.2 present the results of general dropout for the 2000-tweets-BR dataset.

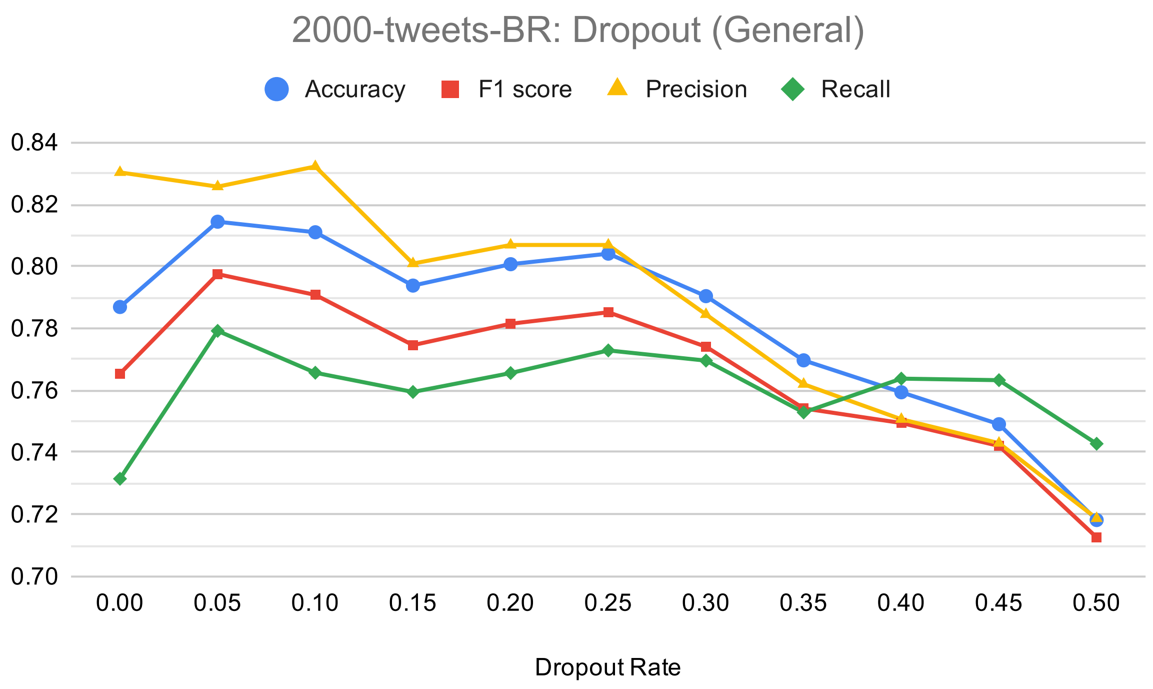


Figure A.3: Dropout experiments results for 2000-tweets-BR – validation.

Table A.2: Dropout experiments results for 2000-tweets-BR – validation.

Rate	Accuracy	Bal. Acc.	F_1 score	Bal. F_1	Precision	Recall
0.00	0.7869	0.7315	0.7654	0.7816	0.8303	0.7315
0.05	**0.8144**	**0.7792**	**0.7975**	**0.8118**	0.8257	**0.7792**
0.10	0.8110	0.7657	0.7908	0.8075	**0.8322**	0.7657
0.15	0.7938	0.7595	0.7746	0.7906	0.8009	0.7595
0.20	0.8007	0.7656	0.7815	0.7978	0.8069	0.7656
0.25	0.8041	0.7729	0.7852	0.8012	0.8069	0.7729
0.30	0.7904	0.7696	0.7741	0.7895	0.7845	0.7696
0.35	0.7697	0.7529	0.7542	0.7691	0.7620	0.7529
0.40	0.7594	0.7638	0.7495	0.7598	0.7507	0.7638
0.45	0.7491	0.7633	0.7421	0.7490	0.7430	0.7633
0.50	0.7182	0.7428	0.7126	0.7189	0.7187	0.7428

Apart from precision, the best results were obtained with 5% of general dropout. The best value for precision was obtained with 10% of general dropout, and it was 0.65 percentage point better than the value obtained with 5% of general dropout.

Figure A.4 and Table A.3 present the results of self-attention computation dropout for the TweetSentBR dataset.

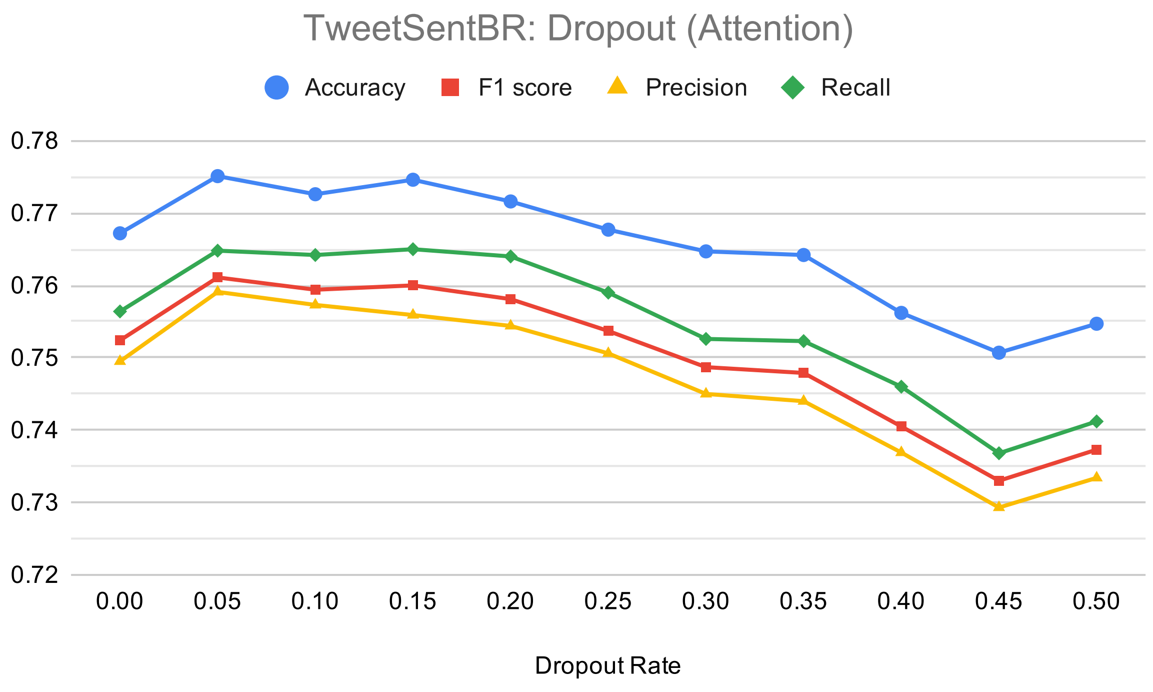


Figure A.4: Dropout experiments results for TweetSentBR – self-attention – validation.

Table A.3: Dropout experiments results for TweetSentBR – self-attention – validation.

Rate	Accuracy	Bal. Acc.	F_1 score	Bal. F_1	Precision	Recall
0.00	0.7672	0.7564	0.7524	0.7696	0.7495	0.7564
0.05	**0.7751**	0.7648	**0.7611**	**0.7776**	**0.7591**	0.7648
0.10	0.7726	0.7642	0.7594	0.7757	0.7573	0.7642
0.15	0.7746	**0.7650**	0.7600	0.7769	0.7559	**0.7650**
0.20	0.7716	0.7640	0.7581	0.7746	0.7544	0.7640
0.25	0.7677	0.7590	0.7537	0.7707	0.7506	0.7590
0.30	0.7647	0.7526	0.7487	0.7667	0.7450	0.7526
0.35	0.7642	0.7523	0.7479	0.7663	0.7440	0.7523
0.40	0.7562	0.7460	0.7405	0.7593	0.7369	0.7460
0.45	0.7507	0.7368	0.7330	0.7528	0.7293	0.7368
0.50	0.7547	0.7412	0.7373	0.7558	0.7334	0.7412

Barring balanced accuracy and recall, all other metrics achieved their best values with 5% of self-attention dropout. As for the balanced accuracy and recall, the best value was 0.7650, obtained with 15% of self-attention dropout, but it was very close to 0.7648, obtained with 5% of self-attention dropout.

Figure A.5 and Table A.4 present the results of self-attention computation dropout for the 2000-tweets-BR dataset.

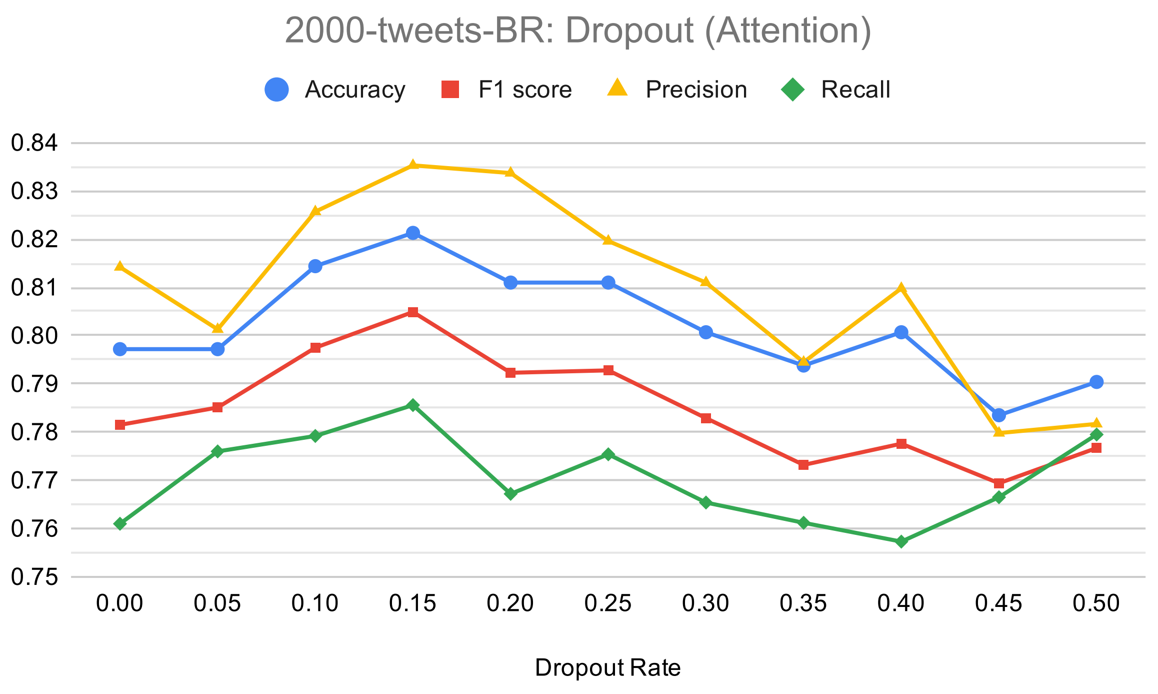


Figure A.5: Dropout experiments results for 2000-tweets-BR – self-attention – validation.

Table A.4: Dropout experiments results for 2000-tweets-BR – self-attention – validation.

Rate	**Accuracy**	**Bal. Acc.**	**F_1 score**	**Bal. F_1**	**Precision**	**Recall**
0.00	0.7972	0.7610	0.7815	0.7947	0.8142	0.7610
0.05	0.7972	0.7760	0.7851	0.7957	0.8013	0.7760
0.10	0.8144	0.7792	0.7975	0.8118	0.8257	0.7792
0.15	**0.8213**	**0.7856**	**0.8049**	**0.8186**	**0.8353**	**0.7856**
0.20	0.8110	0.7672	0.7923	0.8074	0.8337	0.7672
0.25	0.8110	0.7754	0.7928	0.8083	0.8196	0.7754
0.30	0.8007	0.7654	0.7829	0.7979	0.8110	0.7654
0.35	0.7938	0.7612	0.7732	0.7907	0.7945	0.7612
0.40	0.8007	0.7573	0.7776	0.7973	0.8098	0.7573
0.45	0.7835	0.7665	0.7694	0.7832	0.7798	0.7665
0.50	0.7904	0.7795	0.7767	0.7901	0.7817	0.7795

All the metrics considered peaked at 15% of self-attention dropout, as can be seen in the graph and in the table.

A.3 Data Augmentation Experiments

Tables A.5 and A.6 present the results of the experiments on data augmentation regarding parameter α for the TweetSentBR and 2000-tweets-BR datasets, respectively. For more information about the data augmentation approach employed, please refer to Section 4.2.

Table A.5: Data augmentation results for TweetSentBR – parameter α – validation.

$\boldsymbol{\alpha}$	**Accuracy**	**Bal. Acc.**	**$\mathbf{F_1}$ score**	**Bal. $\mathbf{F_1}$**	**Precision**	**Recall**
0.1	0.7632	0.7440	0.7452	0.7638	0.7463	0.7440
0.2	0.7692	0.7573	0.7553	0.7713	0.7537	0.7573
0.3	0.7697	0.7523	0.7521	0.7699	0.7536	0.7523
0.4	**0.7756**	**0.7610**	**0.7607**	**0.7772**	**0.7602**	**0.7610**
0.5	0.7692	0.7499	0.7509	0.7692	0.7530	0.7499
0.6	0.7687	0.7555	0.7536	0.7701	0.7531	0.7555

Table A.6: Data augmentation results for 2000-tweets-BR – parameter α – validation.

$\boldsymbol{\alpha}$	**Accuracy**	**Bal. Acc.**	**$\mathbf{F_1}$ score**	**Bal. $\mathbf{F_1}$**	**Precision**	**Recall**
0.1	**0.8110**	0.7706	0.7918	0.8077	**0.8259**	0.7706
0.2	**0.8110**	0.7756	**0.7931**	**0.8081**	0.8211	0.7756
0.3	0.7972	**0.7760**	0.7852	0.7957	0.8009	**0.7760**
0.4	0.7904	0.7657	0.7738	0.7873	0.7995	0.7657
0.5	0.8075	0.7681	0.7891	0.8044	0.8223	0.7681
0.6	0.7972	0.7665	0.7814	0.7947	0.8066	0.7665

Tables A.7 and A.8 present the results of the experiments on data augmentation regarding parameter n_{aug} for the TweetSentBR and 2000-tweets-BR datasets, respectively.

Table A.7: Data augmentation results for TweetSentBR – parameter n_{aug} – validation.

$\boldsymbol{n_{aug}}$	**Accuracy**	**Bal. Acc.**	**$\mathbf{F_1}$ score**	**Bal. $\mathbf{F_1}$**	**Precision**	**Recall**
1	0.7756	0.7610	0.7607	0.7772	**0.7602**	0.7610
2	0.7702	0.7636	0.7597	0.7742	0.7559	0.7636
3	**0.7762**	**0.7657**	**0.7625**	**0.7792**	**0.7602**	**0.7657**
4	0.7707	0.7555	0.7544	0.7717	0.7542	0.7555
5	0.7742	0.7570	0.7583	0.7751	0.7591	0.7570

Table A.8: Data augmentation results for 2000-tweets-BR – parameter n_{aug} – validation.

$\boldsymbol{n_{aug}}$	**Accuracy**	**Bal. Acc.**	**$\mathbf{F_1}$ score**	**Bal. $\mathbf{F_1}$**	**Precision**	**Recall**
1	0.8110	0.7756	0.7931	0.8081	0.8211	0.7756
2	0.8041	0.7563	0.7823	0.7999	0.8265	0.7563
3	**0.8245**	**0.7851**	**0.8037**	**0.8212**	**0.8346**	**0.7851**
4	0.7938	0.7479	0.7706	0.7895	0.8095	0.7479
5	0.8041	0.7759	0.7874	0.8020	0.8076	0.7759

Appendix B

Overfitting

During our research, we noticed that the models we obtained were consistently producing better results on the training sets than on the evaluation sets, suggesting that overfitting was occurring.

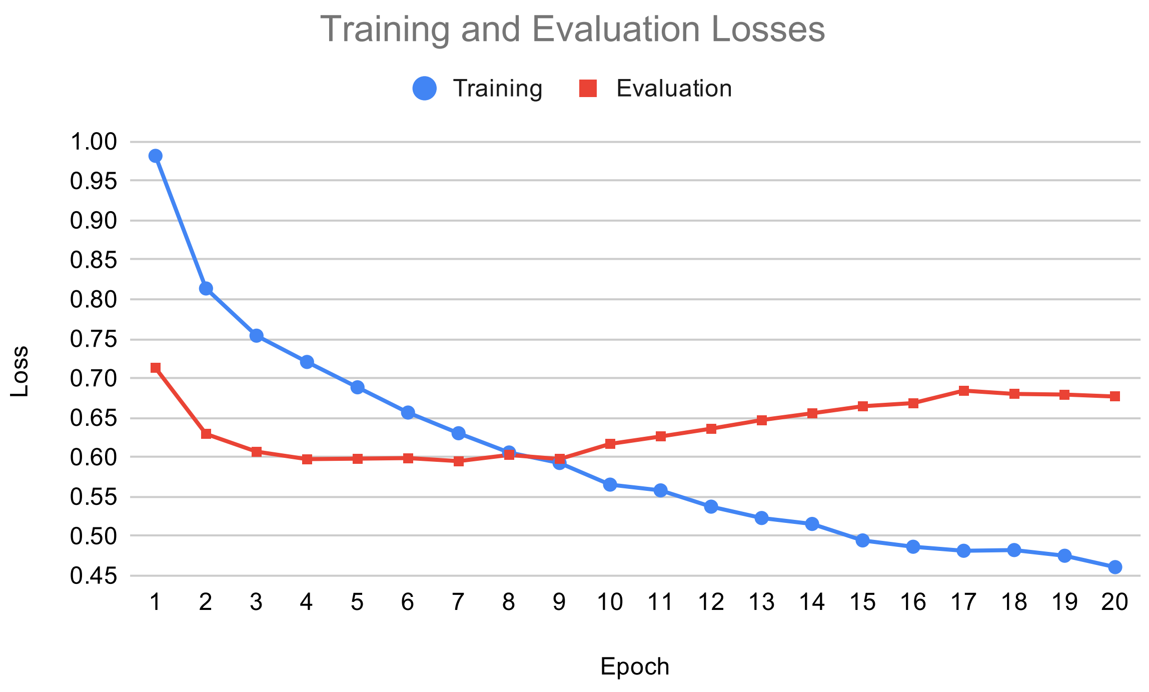


Figure B.1: Training and evaluation losses for TweetSentBR – validation

.

Figure B.1 illustrates the behavior of the training and evaluation losses for the TweetSentBR dataset during all the training epochs. It was obtained with the best-performing model for this dataset. As expected, the training loss maintains a downwards trajectory. The evaluation loss also drops for the first few epochs, then it stabilizes and, from the 10th epoch, starts to rise, until it stabilizes again.

One measure to try to overcome the overfitting is using dropout. However, despite the results having improved, the values obtained for the training sets were still considerably better than the ones for the evaluation sets. We then experimented with other possible strategies: layer normalization and evaluation of different values of weight decay and learning rate.

B.1 Layer Normalization

Layer Normalization (LN) [2] is similar to Batch Normalization (BN) [32] in that they both are techniques to normalize activations in intermediate layers of deep neural networks, leading to faster and more stable training. While batch normalization makes use of batch statistics to compute the mean and variance which are then used to normalize the summed input to a neuron on each training sample, layer normalization uses all of the summed inputs to the neurons in a layer on a single training sample to compute the mean and variance used for normalization, according to the formula:

$$y = \frac{x - E[x]}{\sqrt{Var[x] + \epsilon}} \times \gamma + \beta \tag{B.1}$$

where γ and β are learnable affine transform parameters, and ϵ is a value added to the denominator for numerical stability.

Layer normalization is utilized in the embedding layer, in the encoder, and in the Masked Language Modeling head during pre-training.

All of our experiments on overfitting were performed using the TweetSentBR dataset. Table B.1 presents the results of our experiments on layer normalization, varying the ϵ parameter. Unlike the tables from other sections, here we include the results for the training set to show how the models perform in each set.

Table B.1: Layer normalization experiment results for TweetSentBR.

Set	ϵ	**Accuracy**	**Bal. Acc.**	**F_1 score**	**Bal. F_1**	**Precision**	**Recall**
Train	10^{-12}	0.8391	0.8335	0.8294	0.8408	0.8270	0.8335
	10^{-09}	0.8301	0.8246	0.8205	0.8318	0.8181	0.8246
	10^{-07}	0.8293	0.8239	0.8198	0.8311	0.8174	0.8239
	$\mathbf{10^{-05}}$	**0.8504**	**0.8477**	**0.8428**	**0.8523**	**0.8404**	**0.8477**
	10^{-03}	0.8297	0.8240	0.8202	0.8314	0.8181	0.8240
	10^{-01}	0.7548	0.7405	0.7373	0.7546	0.7385	0.7405
Eval	$\mathbf{10^{-12}}$	**0.7752**	**0.7630**	**0.7604**	**0.7775**	**0.7590**	**0.7630**
	10^{-09}	0.7746	0.7621	0.7595	0.7768	0.7582	0.7621
	10^{-07}	0.7746	0.7622	0.7596	0.7769	0.7584	0.7622
	10^{-05}	0.7728	0.7621	0.7590	0.7758	0.7582	0.7621
	10^{-03}	0.7741	0.7614	0.7590	0.7762	0.7577	0.7614
	10^{-01}	0.7440	0.7253	0.7229	0.7442	0.7239	0.7253

Since BERTimbau [62] was pre-trained using $\epsilon = 10^{-12}$, we consider this value as the baseline. Also, we used weight decay of 10^{-02} and learning rate of 10^{-05}. In Table B.1, we can see that the best value for ϵ, when considering the training set, is 10^{-05}, but if we consider the evaluation set, $\epsilon = 10^{-12}$ yielded the best results. For $\epsilon = 10^{-05}$, the difference between the results of the training and evaluation sets is around 8.2 percentage points (p.p.), considering the metrics presented; for $\epsilon = 10^{-12}$, this difference is around 6.8 p.p. So, the baseline not only yielded the best results for the evaluation set but also resulted in less overfitting than $\epsilon = 10^{-05}$.

Another experiment we performed was to add an extra layer normalization just before the final linear layer (cf. Figure 4.2 on page 40) to check its impact on overfitting. Table B.2 presents the results obtained. The "No Extra" entry represents the same baseline as the previous experiment: no additional layer normalization and $\epsilon = 10^{-12}$.

Table B.2: Layer normalization additional layer experiment results for TweetSentBR.

Set	ϵ	**Accuracy**	**Bal. Acc.**	$\mathbf{F_1}$ **score**	**Bal.** $\mathbf{F_1}$	**Precision**	**Recall**
Train	No Extra	0.8391	0.8335	0.8294	0.8408	0.8270	0.8335
	10^{-12}	0.8534	0.8504	0.8466	0.8557	0.8407	0.8504
	$\mathbf{10^{-09}}$	**0.8613**	**0.8590**	**0.8548**	**0.8632**	**0.8483**	**0.8590**
	10^{-07}	0.8434	0.8396	0.8359	0.8460	0.8303	0.8396
	10^{-05}	0.8578	0.8546	0.8513	0.8601	0.8459	0.8546
	10^{-03}	0.8552	0.8510	0.8479	0.8573	0.8421	0.8510
	10^{-01}	0.6901	0.6556	0.6414	0.6723	0.6762	0.6556
Eval	**No Extra**	**0.7752**	**0.7630**	**0.7604**	**0.7775**	**0.7590**	**0.7630**
	10^{-12}	0.7675	0.7598	0.7551	0.7724	0.7513	0.7598
	10^{-09}	0.7684	0.7611	0.7561	0.7730	0.7514	0.7611
	10^{-07}	0.7660	0.7584	0.7542	0.7704	0.7503	0.7584
	10^{-05}	0.7689	0.7614	0.7571	0.7737	0.7538	0.7614
	10^{-03}	0.7699	0.7616	0.7572	0.7749	0.7537	0.7616
	10^{-01}	0.6824	0.6459	0.6348	0.6671	0.6623	0.6459

The results show that the extra layer normalization did not improve the values for the evaluation set, for every ϵ we tested. For the training set, $\epsilon = 10^{-09}$ did improve the metrics, but also increased the overfitting, with the difference between the results of the training and evaluation sets being around 9.6 p.p.

B.2 Weight Decay

Another approach we experimented with was evaluating different values of weight decay [42] for the Adam optimizer. The results are presented in Table B.3.

For both the training and evaluation sets, the best values were obtained with a weight decay of 10^{-02}. The difference between the results of the training and evaluation sets is around 6.8 p.p. Interestingly, the smaller decay values yielded exactly the same results, suggesting that some kind of saturation is occurring.

B.3 Learning Rate

The last strategy we evaluated to cope with overfitting is experimenting with different learning rates.

Table B.3: Weight decay experiment results for TweetSentBR.

Set	**Decay**	**Accuracy**	**Bal. Acc.**	$\mathbf{F_1}$ **score**	**Bal.** $\mathbf{F_1}$	**Precision**	**Recall**
Train	10^{-05}	0.8301	0.8247	0.8205	0.8318	0.8180	0.8247
	10^{-04}	0.8301	0.8247	0.8205	0.8318	0.8180	0.8247
	10^{-03}	0.8301	0.8247	0.8205	0.8318	0.8180	0.8247
	$\mathbf{10^{-02}}$	**0.8391**	**0.8335**	**0.8294**	**0.8408**	**0.8270**	**0.8335**
	10^{-01}	0.8297	0.8244	0.8202	0.8314	0.8177	0.8244
Eval	10^{-05}	0.7710	0.7590	0.7562	0.7731	0.7547	0.7590
	10^{-04}	0.7710	0.7590	0.7562	0.7731	0.7547	0.7590
	10^{-03}	0.7710	0.7590	0.7562	0.7731	0.7547	0.7590
	$\mathbf{10^{-02}}$	**0.7752**	**0.7630**	**0.7604**	**0.7775**	**0.7590**	**0.7630**
	10^{-01}	0.7702	0.7585	0.7556	0.7724	0.7541	0.7585

Table B.4: Learning rate experiment results for TweetSentBR.

Set	**Rate**	**Accuracy**	**Bal. Acc.**	$\mathbf{F_1}$ **score**	**Bal.** $\mathbf{F_1}$	**Precision**	**Recall**
Train	10^{-07}	0.4267	0.4354	0.4223	0.4330	0.4596	0.4354
	10^{-06}	0.7077	0.6923	0.6874	0.7072	0.6879	0.6923
	10^{-05}	0.8391	0.8335	0.8294	0.8408	0.8270	0.8335
	$\mathbf{10^{-04}}$	**0.8965**	**0.8861**	**0.8879**	**0.8950**	**0.8939**	**0.8861**
	10^{-03}	0.4420	0.3333	0.2043	0.2709	0.1473	0.3333
Eval	10^{-07}	0.4825	0.4914	0.4786	0.4890	0.5142	0.4914
	10^{-06}	0.7363	0.7214	0.7166	0.7362	0.7172	0.7214
	$\mathbf{10^{-05}}$	**0.7752**	**0.7630**	**0.7604**	**0.7775**	**0.7590**	**0.7630**
	10^{-04}	0.7540	0.7288	0.7275	0.7493	0.7322	0.7288
	10^{-03}	0.4921	0.3833	0.2544	0.3210	0.1974	0.3833

As Table B.4 shows, a learning rate of 10^{-04} produced the best results on the training set. The difference between the results of the training and evaluation sets for this value of learning rate is around 15.4 p.p. Considering now the evaluation set, the best results were obtained with a learning rate of 10^{-05}. In this case, the difference between the results of the training and evaluation sets is around 6.8 p.p. We can see that using a learning rate of 10^{-04} substantially increases the overfitting.

Considering everything we tried, unfortunately we were not able to eliminate the overfitting.