



UNIVERSITAT^{DE}
BARCELONA

Bachelor's thesis

Bachelor's Degree In Computer Engineering

**Faculty of Mathematics and Computer Science
Universitat de Barcelona**

Identification of sexism on social media

Author: Rong Xing

Director: Dra. Maria Salamó Llorente

**Written in: Department of Mathematics and Computer Science
(Faculty of Mathematics and Computer Science)**

Barcelona, 10 de juny de 2023

Acknowledgements

I would like express gratitude to those who have provided unwavering support and assistance throughout this transformative journey. Firstly, I extend heartfelt appreciation to my mentor, María Salamó, and co-mentor, Alejandro Ariza, for their dedication and unwavering commitment to this project. Their attentive listening during weekly meetings, addressing concerns in a timely manner, not only about the thesis itself, but even my choice of future academic direction, and introduction to the captivating realm that I will soon embark upon is deeply valued.

I would also like to extend my gratitude to the countless researchers who have contributed to the numerous works, papers, and articles referenced throughout this thesis. Their dedication and enthusiasm for advancing the realm of science is truly commendable and serves as a valuable resource for society.

Furthermore, I express my sincere appreciation to my family and friends, who have been by my side throughout this journey. Together, we navigated through the initial stress, celebrated significant milestones, and shared the joy of obtaining positive outcomes while reaching the end of this chapter. I would also like to express my special thanks to my beloved cat, who endured my late-night writing sessions with her patient presence, despite my sometimes less than gentle touch.

Lastly, I would like to express my gratitude to all the women fighting for positions in the public sphere, particularly in male-dominated fields. I am thankful for the ordinary yet extraordinary women who have inspired and supported me on my journey of personal growth. Their strength has empowered me to fearlessly choose the passion I love and pursue my life goals. While the world may be unjust and imperfect, as women, when we support and uplift one another, we are not alone, but rather interconnected in a tight bond.

Identification of sexism on social media

Detection, Analysis, and Research on Sexism Content for Certain Languages in Social Networks

Abstract

With the rapid advancement of communication technology, smartphone usage, and sophisticated algorithms, social media has become an integral and inseparable part of modern society. Consequently, the prevalence of sexist content on these platforms has emerged as a significant and far-reaching issue. This form of online harassment not only perpetuates gender inequalities but also poses significant psychological and emotional harm to individuals targeted by such content. Thus, it is imperative to address this problem and take proactive measures to mitigate its impact.

The main goal of this work is to study, identify and analyze the process of detection of sexist content through the application of natural language processing techniques. The study utilizes two datasets from the EXIST competition, a shared task of sEXism Identification in Social neTworks from IberLeF 2021 and CLEF 2023. Five state-of-the-art language models, based on Transformers and Deep Learning, are trained and validated for subsequent comparison. The primary objective is to identify instances of online sexism and determine the optimal framework for each task, which accurately reflects real-world scenarios.

Identificación del sexismo en las redes sociales

Detección, análisis e investigación del contenido sexista de determinadas lenguas en las redes sociales

Resumen

Con el rápido avance de la tecnología de comunicación, el uso de teléfonos inteligentes y algoritmos sofisticados, las redes sociales se han convertido en una parte integral e inseparable de la sociedad moderna. En consecuencia, la prevalencia de contenido sexista en estas plataformas ha surgido como un problema significativo y de amplio alcance. Esta forma de acoso en línea no solo perpetúa las desigualdades de género, sino que también representa un daño psicológico y emocional significativo para las personas que son el objetivo de dicho contenido. Por lo tanto, es imperativo abordar este problema y tomar medidas proactivas para mitigar su impacto.

El objetivo principal de este trabajo es estudiar, identificar y analizar el proceso de detección de contenido sexista a través de la aplicación de técnicas de procesamiento del lenguaje natural. El estudio utiliza dos conjuntos de datos de la competición EXIST, una tarea compartida de identificación de sEXismo en redes sociales de IberLeF 2021 y CLEF 2023. Se entrenan y validan cinco modelos de análisis de lenguaje natural basados en Transformers y Deep Learning para su posterior comparación. El objetivo principal es identificar casos de sexismo en línea y determinar el modelo óptimo para cada tarea, que refleje con precisión escenarios del mundo real.

Contents

1	Introduction	1
1.1	Motivation: Addressing Online Sexism for Gender Equality	1
1.2	Problem to be solved: why this problem?	2
1.3	Objectives	2
1.4	Project Schedule	3
1.5	Project Organization	4
2	State-of-the-art of NLP	6
2.1	Contextualization: what is NLP and its application in social media	6
2.2	Pre-processing techniques	6
2.3	Traditional models and approaches	8
2.3.1	Bag-of-words	9
2.3.2	TF-IDF	10
2.3.3	Word2Vec: skip-gram, CBoW	10
2.3.4	Neural networks: RNNs, LSTMs	11
2.4	Deep learning approaches with Transformers	12
2.4.1	Encoder-Decoder architecture	12
2.4.2	Attention mechanism	13
2.4.3	Transformers	13
2.4.4	BERT: Bidirectional Encoder Representations from Transformers	15
2.4.5	RoBERTa: Robustly optimized BERT approach	17
2.4.6	XLNet: eXtreme Learning Network	18
2.4.7	HateBERT and DeHateBERT: Language-Specific Abusive Language Detection Models	18
2.5	Evaluations and metrics	19
3	Implementation	21
3.1	Implementation scenario	21
3.2	Libraries	22
3.2.1	LIBRARIES FOR GENERAL AND DATA SCIENCE PURPOSES	22
3.2.2	LIBRARIES FOR VISUALIZATION PURPOSES	24
3.2.3	NLP-RELATED LIBRARIES	24
3.3	Structure of Jupyter Notebook	24

3.3.1	Preparation: Install, imports and data loading	25
3.3.2	Data used for training	25
3.3.3	Statistical analysis of dataset	28
3.3.4	Dataset pre-processing procedure	33
3.3.5	Neural Network Pre-processing, Training and Evaluation Metrics	35
3.4	Model building	38
3.4.1	BERT	38
3.4.2	RoBERTa	38
3.4.3	XLNet	39
3.4.4	HateBERT	39
3.4.5	DeHateBERT	39
3.4.6	Baseline model: SVM based on TF-IDF vectorization	39
4	Results and analysis	41
4.1	Experiments Setup	41
4.2	Results of training and evaluation	42
4.2.1	EXIST 2021 Dataset	42
4.2.2	Data augmentation	44
4.2.3	Best-performing model and results for EXIST 2021	46
4.2.4	EXIST 2023 Dataset	49
4.2.5	Best-performing model and results for EXIST 2023	51
4.2.6	EXIST 2023 Task 1 submission	53
4.2.7	Comparative among all models	55
5	Conclusions and future work	57
6	Appendix	59
6.1	EXIST 2021 Tables with Translation	60
6.1.1	English	60
6.1.2	Spanish	62
6.2	EXIST 2021 Tables without Translation	64
6.2.1	English	64
6.2.2	Spanish	66
6.3	EXIST 2023 Tables with Translation	68
6.3.1	English	68
6.3.2	Spanish	70
6.4	EXIST 2023 Tables without Translation	72
6.4.1	English	72
6.4.2	Spanish	74
	Bibliography	76

Chapter 1

Introduction

This section of the Bachelor's thesis will function as an introduction, setting the stage for the remainder of the project. It will start by providing a concise overview of the motivations behind choosing this particular topic, followed by a clear statement of the problem that will be addressed. Additionally, the objectives to be accomplished and the overall structure of the document will be outlined and discussed.

1.1 Motivation: Addressing Online Sexism for Gender Equality

The issue of sexism in social media is a growing concern that requires immediate attention. As the Oxford English Dictionary defines it, sexism is the prejudice, stereotyping, or discrimination against women on the basis of sex [1]. Unfortunately, this problem is pervasive and has penetrated every facet of society, including the online space. Detecting online sexism is challenging, as it can take various forms, including subtle expressions of implicit sexist behaviors that are just as damaging as explicit misogyny. Therefore, the automatic identification of sexism in its broadest sense is critical in designing and determining the evolution of new equality policies while promoting better behavior in society.

As a female who has experienced gender discrimination firsthand, I am particularly motivated to tackle this issue. Not only from my own experience but also from the women around me, I have been observing the negative impacts of misogyny on women's lives from limiting their career opportunities to shaping their sexual image and expectations. As a student pursuing a Computer science degree, I am eager to leverage my knowledge to identify and combat sexism in real-life environments. This bachelor thesis project will enable me to contribute to the cause of gender equality by developing and applying natural language processing techniques to detect sexism in tweets. Through this project, I hope to make a tiny contribution towards creating a more equitable and fair society for

our community.

1.2 Problem to be solved: why this problem?

Social media platforms not only facilitate communication and information sharing at an unprecedented level, but also provide an anonymous environment that enables users to express aggressive attitudes towards specific individuals or groups by posting abusive language. Among the various forms of offensive speech, one that is specifically directed at female groups is sexism.

The prevalence of sexism on social media has negative real-life consequences for women. Real-life examples show how women suffer from online sexism. According to a study [2] that examined 51 countries, online harassment affects 38% of women globally, but only 25% report it to authorities, and 90% limit their online activity, worsening the gender digital divide. Furthermore, women in politics and other public figures receive sexist messages and threats, which can lead to self-censorship and avoidance of public life. Moreover, women in everyday life may face gender-based discrimination in employment, education, and personal relationships, perpetuated by online sexism. Therefore, addressing sexism on social media is crucial to promote gender equality and safeguard women's well-being.

To mitigate online sexism on social media, developing technical approaches is crucial due to the scale and complexity of the problem. Manual censorship and modification of sexist content has become difficult to manage in the past few years due to the increasing amount of user-generated content and the diversity of user behavior toward women on social media. However, academic research in dealing with automatic detection of misogynistic behavior and gender-based hate in both monolingual and multilingual contexts is rapidly increasing [3] [4]. Machine learning algorithms can help automate the identification and removal of offensive content, including sexist messages, by analyzing language patterns, sentiment, and context. Also, technical approaches can be used to improve platform design and user experience to reduce the prevalence of online sexism, such as implementing features that allow users to block or report abusive content or users. These solutions can help create a safer and more inclusive online environment, empowering women to participate in online discussions and activities without fear of harassment or discrimination.

1.3 Objectives

The main objective of the entire study, as indicated by the title of this thesis, is to analyze and develop NLP models that can accurately identify sexist content on social media messages. To achieve this goal, several interconnected subobjectives have been outlined.

1. Firstly, we will conduct a thorough assessment of sexist content identification using the EXIST dataset spanning two years, specifically 2021 [5] and 2023 [6]. Unfortunately, the 2022 dataset has not been made available to the public at this time due to internal reasons. It is important to highlight that detecting online sexism presents

a significant challenge due to the diverse forms that posts can take. They may be expressed in hateful and offensive language or appear benign and humorous, which can mislead the classification models employed for this task. Additionally, to ensure the effectiveness of the models across various contexts, it is crucial to consider implementing them in multiple languages.

2. Moreover, since this research will utilize state-of-the-art NLP models, our initial focus will be on investigating these models and delving into the fundamentals of the field. This exploration will provide a solid foundation for our subsequent work.
3. Another critical objective involves gaining a comprehensive understanding of the aforementioned datasets through statistical analysis of the available source of data. This analysis will enable us to extract valuable insights and design effective data pre-processing pipelines. By properly preparing the datasets, we can ensure their suitability for further modeling and analysis.
4. After that we will proceed to construct and evaluate different NLP models specifically designed for the task of identifying sexist content. This process will involve conducting numerous experiments for each language and dataset, where we will focus on specific metrics and loss values that serve as indicators of performance and effectiveness.
5. Finally, our final objective will be to compare the performance of the developed models using each dataset. This comparative analysis will enable us to determine the most effective model for accurately detecting sexist content within social media messages.

1.4 Project Schedule

Lastly, I would like to show the meticulous scheduling involved in the completion of this paper. Beginning with literature review in February and acquiring comprehensive knowledge of cutting-edge technologies in the field of natural language processing, followed by data compilation, model selection, actual pipeline coding, process documentation, and extensive experimentation, culminating in the final analysis and organization, these continuous efforts spanning eighteen weeks, under the guidance of my supervisors, allowed me to achieve the objectives initially established for this thesis. Attached herewith is a Gantt chart illustrating the systematic breakdown of the entire schedule into smaller components, providing approximate completion times for each segment, and more.

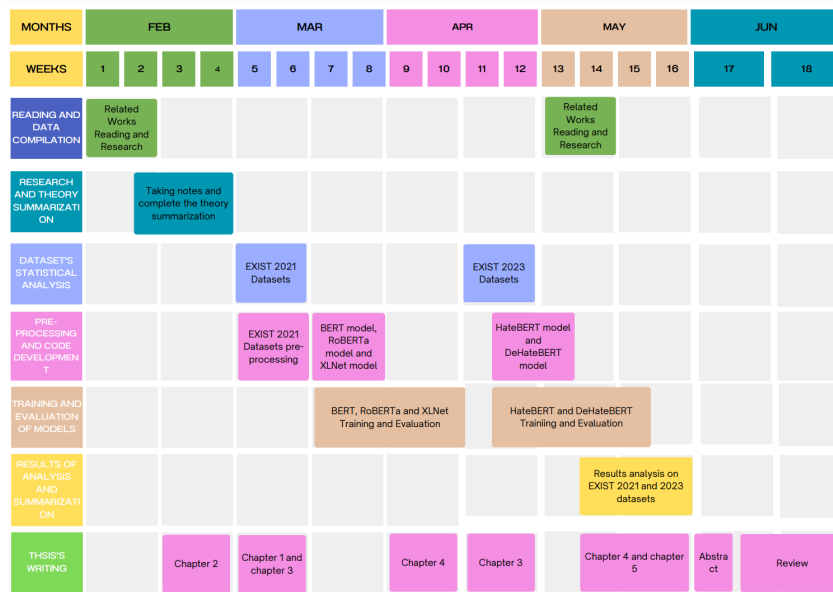


Figure 1.1: Bachelor's thesis Gantt Chart: Timeframe

1.5 Project Organization

This bachelor thesis is organized into five distinct parts. The first part is the Introduction (Chapter 1), serving as the initial chapter to provide an explanation of the problem at hand, outlining motivation, and stating the project goals. This introductory section aims to contextualize readers and provide a comprehensive overview of the project.

Following the Introduction, the second part delves into the State-of-the-Art (SOTA) section (Chapter 2). This chapter begins with a comprehensive review of various techniques, encompassing a broad range of approaches. It gradually progresses towards exploring the most advanced models available, such as transformer-based models. This section emphasizes the significance of reading and comprehending current papers, as well as presenting a detailed summary of their contents.

Subsequently, the Implementation chapter (Chapter 3) follows the literature review. This section focuses on putting into practice the knowledge acquired during the literature review phase. It encompasses various aspects, including data preparation, model selection, model building, and obtaining results with an emphasis on the selection of optimal parameters.

The fourth chapter is dedicated to presenting the gathered results (Chapter 4). It showcases the outcomes obtained from different training and evaluation sessions of the models and includes a thorough comparison of the results. This section provides a comprehensive analysis of the performance of the implemented models, enabling a deeper understanding of their effectiveness.

Lastly, the fifth and final chapter is dedicated to the conclusions and future work of the entire thesis (Chapter 5). This section reflects upon the lessons learned throughout the course of the research, offering valuable observations and evaluations of the objectives achieved. Moreover, it discusses future directions for further exploration and development. Specifically, the future work includes generalizing the trained model to encompass datasets from other fields, thereby extending its applicability. The concluding chapter aims to provide a comprehensive overview and summary of the thesis, effectively consolidating the key findings and underscoring the research's significance in advancing the field.

Chapter 2

State-of-the-art of NLP

This chapter aims to provide a comprehensive understanding of the project by presenting the fundamental concepts and techniques employed in NLP data processing. It begins with a concise introduction to these concepts, ensuring a solid foundation for the subsequent discussions. Emphasis will be placed on analyzing and comparing existing models and algorithms within the field, forming the core content of this chapter. By examining the strengths and weaknesses of these approaches, a deeper insight into the state-of-the-art methods will be attained.

2.1 Contextualization: what is NLP and its application in social media

Natural Language Processing (NLP) is an interdisciplinary field that combines computer science, artificial intelligence, linguistics, and psychology to study the complexity of human language and develop computational methods to analyze and generate it. To simplify its definition, NLP is about making computers understand and use human language.

It may not be universally acknowledged that NLP technology has become pervasive in modern society. The capability of NLP to categorize text and analyze sentiments enables its utilization in a spectrum of fields, such as chatbots and online customer service, as it requires understanding and responding to human language inputs, and email spam filtering systems, which are commonly utilized by individuals.

2.2 Pre-processing techniques

Pre-processing and data cleaning are equally important to developing an effective machine learning model. Therefore, the quality of the information has a huge impact on how reliable your model is. In order to develop effective NLP models, pre-processing techniques are essential to clean and convert raw text into a more structured and informative

format. These techniques help to reduce the dimensionality of the data, identify similar words, and standardize the language.

Tokenization

Tokenization involves breaking down a text into individual tokens or words [7]. This process segments sentences, paragraphs, or entire documents into smaller units, enabling the analysis of each word in isolation as shown in Figure 2.1. Tokenization simplifies subsequent tasks such as counting word frequencies, identifying key phrases, and analyzing syntactic structures. It forms the fundamental step in text analysis, allowing algorithms to understand and process text effectively.

Input: Friends, Romans, Countrymen, lend me your ears;
Output:

Friends	Romans	Countrymen	lend	me	your	ears
---------	--------	------------	------	----	------	------

Figure 2.1: Tokenization in Natural Language Processing

Stopwords Removal

Stopwords are commonly used words, such as articles (e.g., “a,” “an,” “the”), pronouns (e.g., “he,” “she,” “it”), and conjunctions (e.g., “and,” “but,” “or”). These words add little semantic value and can introduce noise into the analysis. Stopwords removal filters out these words from the text, reducing the dimensionality of the data and improving the accuracy of subsequent analyses. Removing stopwords is particularly useful in tasks like information retrieval and document classification.

Stemming and Lemmatization

For grammatical reasons, documents like messages always use different forms of a word. The approach of stemming and lemmatization involves the same methodologies. They both reduce the inflectional forms of each word into a common root word as depicted in Figure 2.2. However, the main difference is in the way these methods are implemented and the result they come up with. Stemming involves removing prefixes and suffixes from words to obtain their root, known as a stem. For example, stemming transforms “running” and “runner” to “run”. On the other hand, lemmatization maps words to their corresponding base form, known as a lemma, by considering the word’s part of speech. For instance, lemmatization transforms “better” to “good”. These techniques normalize word variations, consolidating similar words and reducing redundancy in the text. Stemming is computationally simpler but may result in non-grammatical stems, while lemmatization provides linguistically meaningful lemmas.

Sentiment-oriented Pre-processing Procedures

Stemming vs Lemmatization



Figure 2.2: Comparison of stemming and lemmatization [8]

The objective of sentiment analysis is to determine the underlying emotional expression in a given text. To enhance the accuracy of sentiment analysis outcomes, various pre-processing techniques can be implemented with a focus on sentiments. These techniques encompass dealing with negations, such as converting “not happy” to “not_happy”. Furthermore, it is crucial to address emoticons and **emojis**, as emojis are composed of combinations of punctuation marks and have been observed to convey significant connotations, particularly in the context of abusive language on social media platforms. Hence, in some cases, it may be appropriate to retain emojis as separate tokens (see Figure 2.3). Additionally, replacing **elongated words** and **slang**, like transforming “soooo” to “so”, is also imperative. Moreover, sentiment-specific lexicons can be employed to augment the sentiment analysis process by establishing associations between words and their corresponding sentiment polarities, such as positive, negative, or neutral.

In addition, the process of **lowercasing** involves converting every word to lowercase, as the name implies. The primary purpose of this conversion is to address the issue where words with identical meanings but different cases are treated as distinct words in the vector space if left unchanged. By converting them to lowercase, the dimensions and cost associated with these variations are reduced.

2.3 Traditional models and approaches

Language is a learned ability that takes humans a lifetime to master, and the process of learning frequently starts with understanding the meaning of the words, as language has meaning because of the way words are formed into other words. Thus we might ask: how do machines interpret language?

In fact, a computer’s comprehension of words closely resembles that of a human. When someone says, “It’s going to rain outside soon,” we already know in our heads what the words “outside”, “soon” and “rain” mean in their most general senses. We have an established neural connection between the abstract ideas of these words mentioned above in our minds that is prepared to be awakened and activated when we hear the phrase. This neural connection will then tell us that we could either remain inside or bring an

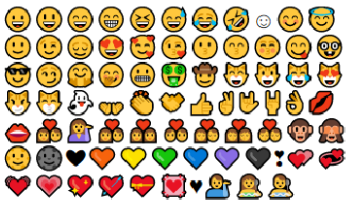





Emotion	Emojis
Happiness	
Sadness	
Fear	
Anger	
Excited	
Bored	

Figure 2.3: Emoji Classification for sentiment analysis [9]

umbrella before leaving the house. In other words, we grasp the meaning of the sentence and the meaning underlying the entire statement by comprehending the meaning of each word and by way of this process of word reorganization. At this point, we can therefore confidently assume that understanding the meaning of words is the key to language comprehension.

We are all aware that computers are capable of processing objects made up of nothing but numbers, such as all images that can be expressed using a specific integer for each pixel. Instead of being able to comprehend the pervasive complexity of human language, the computer is able to categorize the language, or rather the words, into the appropriate location. This allows it to process words and recognize the emotions hidden between the lines. Actually, the comprehension of words of machines seems to be their comprehension of place and area. This is where **Word Embedding vectors** come into play as their mathematical representations. Some techniques are Bag of Words, TF-IDF or CountVectorizer.

2.3.1 Bag-of-words

Bag-of-Words (BoW) is a popular and effective model used in NLP field which builds a representation of text that describes the occurrence of words within a document [10]. It involves two things:

1. A **vocabulary** of unique words from **corpus**.
2. A measure of the presence of known words, commonly as **vector**.

It is called a "bag" of words because any information about the order or structure of

words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document. Although it is a widely used approach in NLP due to its simplicity, BoW discards the context of words and the relationship between words in a sentence or document, which can result in loss of semantic meaning. Furthermore, it creates high dimensional data as the number of unique words in a document can be large, leading to the curse of dimensionality problem.

2.3.2 TF-IDF

TF-IDF is a statistical technique that assesses the relevance of a word to a document within a corpus, which involves computing the product of two measures: **Term Frequency (TF)** and **Inverse Document Frequency (IDF)**. Specifically, TF corresponds to the frequency of a word in a document, while IDF refers to the logarithmic value of the ratio of the total number of documents to the number of documents containing the word (see Figure 2.4). Despite being an improvement in word representation, the technique relies on the Bag-of-Words (BoW) model, which lacks the ability to capture semantic relations, word positions within documents, and co-occurrences.

$$w_{x,y} = \text{tf}_{x,y} \times \log\left(\frac{N}{\text{df}_x}\right)$$

TF-IDF
Term x within document y

$\text{tf}_{x,y}$ = frequency of x in y
 df_x = number of documents containing x
 N = total number of documents

Figure 2.4: Fig. TF-IDF formula for word representation

2.3.3 Word2Vec: skip-gram, CBoW

skip-gram

In the field of natural language processing, skip-gram is a popular Neural Network algorithm that focuses on word embeddings and contextual representations. Figure 2.5 shows its architecture. The skip-gram model, introduced by Mikolov [11], operates on the principle of using the current word to predict its neighboring words within a given context. This approach offers valuable insights into the semantic relationships between words and enhances our understanding of language structures.

By leveraging skip-gram, NLP researchers can capture the subtle nuances of word associations and syntactic patterns. The skip-gram model proves particularly useful in tasks such as language modeling, part-of-speech tagging, and sentiment analysis. It enables the extraction of rich contextual information from textual data, contributing to advancements in machine translation, information retrieval, and other related applications.

CBoW: Continuous Bag-of-Words

The continuous bag-of-words (CBoW) model is another significant algorithm employed in NLP field. Unlike skip-gram, CBoW focuses on predicting the current word based on the contexts surrounding it. The limit on the number of words in each context is determined by a parameter called "window size".

The CBoW model[11], excels in capturing the local syntactic and semantic dependencies of words. It efficiently represents the meaning of a word by considering the collective context it appears in, without relying on word order. This makes CBoW particularly suitable for tasks such as text classification and named entity recognition. The utilization of CBoW aids in uncovering important contextual information and improving the accuracy of various NLP tasks due to its simplicity and efficiency.

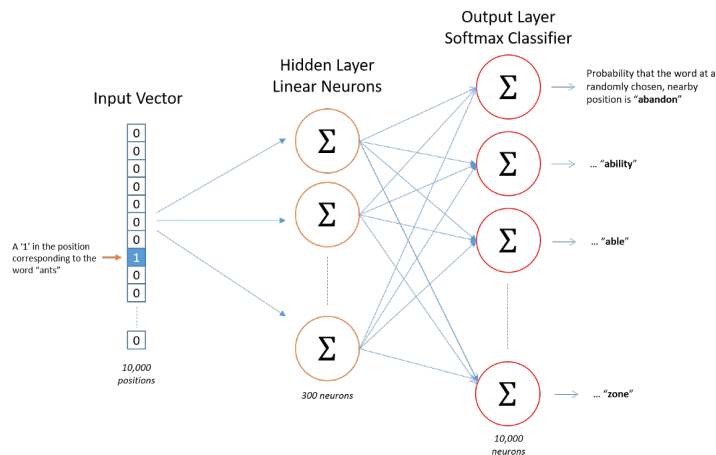


Figure 2.5: skip-gram neural network [12]

2.3.4 Neural networks: RNNs, LSTMs

The primary advantage of deep learning approaches is its "end-to-end" capability, which does not refer to the transfer of data from the client to the cloud. Rather, it denotes that in the past, developers were required to consider which features to design to extract from data. However, in the end-to-end era, this is no longer necessary, as the original input can be fed directly into a competent feature extractor, which will automatically extract useful features without human intervention.

RNN is a popular choice for NLP applications due to its ability to process linear sequences of sentences with varying lengths and capture long-range features through the inclusion of three gates in LSTM models. Despite its remarkable performance, the original RNN model faces optimization difficulties during back propagation, caused by the linear sequence structure, which results in gradient vanishing or explosion problems. To address this issue, LSTM and GRU models were introduced with intermediate state information to propagate backward directly, effectively resolving the gradient disappearance problem. These models have become the standard RNN models and achieved impressive results in various NLP tasks.

2.4 Deep learning approaches with Transformers

In order to achieve a comprehensive comprehension of the Transformer, it is necessary to acquire a prior understanding of two fundamental concepts extensively employed in NLP tasks – the Encoder-Decoder architecture [13] and the Attention mechanism.

2.4.1 Encoder-Decoder architecture

In the context of text processing, the encoder-decoder framework can be conceptualized as a general processing model appropriate for processing the creation of a new sentence (or chapter) from an existing sentence (or chapter). The target sentence **Target** is supposed to be produced by the Encoder-Decoder framework given the input sentence **Source** [14] (see Figure 2.6).

$$\mathbf{Source} = \langle x_1, x_2, \dots, x_m \rangle$$

$$\mathbf{Source} = \langle y_1, y_2, \dots, y_m \rangle$$

The Encoder, as the name implies, transforms the input sentence **Source** through a non-linear transformation into an intermediate semantic representation C :

$$C = f(x_1, x_2, \dots, x_m)$$

The decoder's job is to generate the words that will be formed at moment i based on the historical data that has already been generated in the past and the intermediate semantic representation C of the phrase **Source**:

$$y_i = g(y_1, y_2, \dots, y_{i-1})$$

After forming each y_i in sequence in this manner, it appears as if the entire system has produced the **Target** sentence, which is derived from **Source**.

The Encoder-Decoder framework is used to solve the machine translation problem when the Source is a Chinese sentence and the Target is an English sentence. When the Source is an article and the Target is a collection of general descriptive sentences, the framework is used to summarize the content.

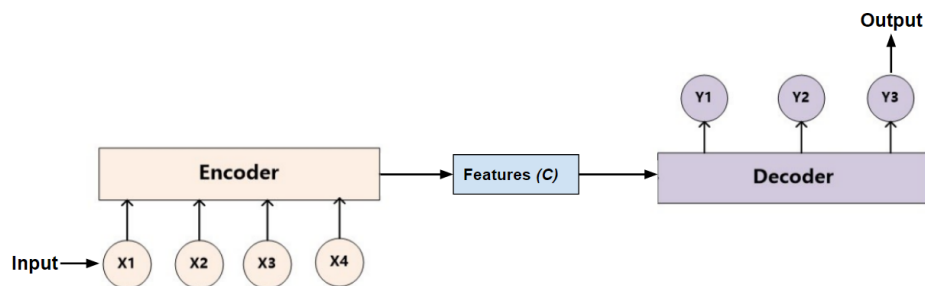


Figure 2.6: Abstract Encoder-Decoder framework for language processing [14]

2.4.2 Attention mechanism

The primary objective of attention in deep learning is to selectively focus on various features when making predictions, similar to humans' selective visual attention mechanism. To illustrate, consider the example of predicting sentiment based on the sentence:

“My mom cooked a delicious meal today.”

In this scenario, only the word *“delicious”* carries significant weight in determining the sentiment, whereas the remaining parts of the sentence are relatively less critical.

Hence, simply taking the average of the token embeddings to represent the entire sentence could introduce a significant amount of noise. The attention mechanism is proposed to overcome this limitation and assign varying weights to input tokens based on the task objective. The proposed mechanism allows the weights of the input tokens to be adjusted based on their relevance to the task. Ideally, the attention weights would assign a relatively higher weight to the tokens that carry more significant information, such as *“delicious”* and *“meal”* while assigning relatively lower weights to other tokens such as *“today”* or *“cooked”*.

The following attention mechanism is referred to as **self-attention** and it is widely used in NLP tasks. The input **Source** and output **Target** contents within the encoder-decoder framework for general tasks are inherently distinct. The nomenclature "self-attention" does not denote the attention mechanism operating between the target and the source; instead, it pertains to the attention mechanism within the constituent elements of the source or the target, which is commonly referred to as the attention mechanism in the special case where the target is equal to the source.

This mechanism is introduced due to its ability of capture **semantic properties** between words within a sequence [15]. Let us consider the following sentence as the input for a given translation task:

“The animal didn’t cross the street because it was too tired.”

Determining the referent of the pronoun **"it"** in the previous sentence may be an effortless task for humans, but can pose a challenge for an algorithmic system, ascertaining whether **"it"** pertains to the street or the animal, as shown in Figure 2.7. In this regard, self-attention enables the system to establish an association between **"it"** and **"animal"**.

2.4.3 Transformers

After comprehending the previous two concepts, we can now discuss the **Transformer**. The Transformer is a new encoder-decoder architecture based solely on attention mechanisms, which improves the most criticized drawback of slow training of RNN by using self-attention mechanism to achieve fast **parallelism**. Figure 2.8 depicts its architecture, which was introduced in 2017 by Vaswani et al. in their seminal paper titled *“Attention is all you need.”* [16]. Following its publication, the Transformer has achieved

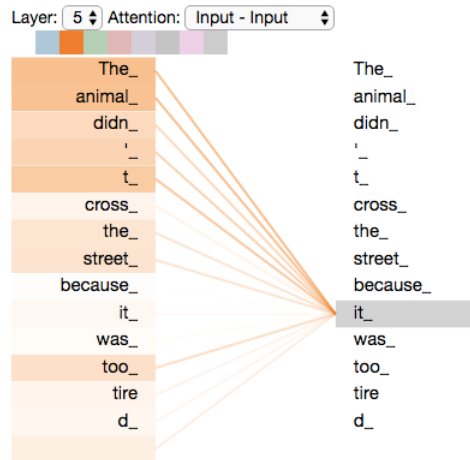


Figure 2.7: Self-attention mechanism [15]

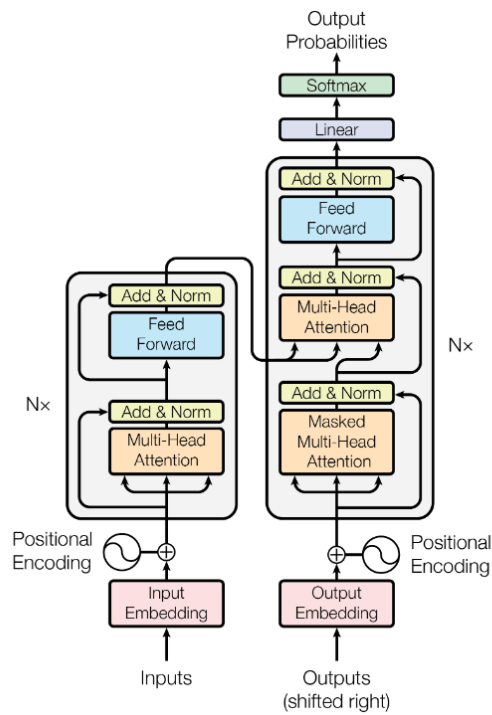


Figure 2.8: Connect the encoder and decoder through an attention mechanism—Transformer architecture [16]

state-of-the-art (SOTA) results on several benchmark datasets across various NLP tasks, marking a significant breakthrough in this field.

The Transformer architecture consists of an encoder and a decoder, both of which are composed of a stack of 6 identical layers. Each **encoder** layer has two sub-layers: the first sub-layer utilizes a multi-head self-attention mechanism that takes as input a sequence of token embeddings and computes a set of attention scores for each token, which allows the model to capture different aspects of the input and learn more complex patterns and relationships between the tokens. The second sub-layer uses a fully connected feed-forward network (MLP) transforming the output of the multi-head self-attention layer into a higher-level representation of the input sequence by applying two linear transformations with a non-linear activation function in between. The dimension of the model, d_{model} , is set to 512.

Similarly, the **decoder** consists of a stack of identical layers, each comprising three sub-layers. The first two sub-layers in the decoder mirror those in the encoder. However, the third sub-layer performs multi-head attention over the output of the encoder stack, enabling the decoder to extract the pertinent information required for the given task. Furthermore, the decoder incorporates the first masked stack, which is specifically designed to prevent positions from attending to subsequent positions. This mechanism ensures consistent behavior during both training and prediction stages.

The previously described attention mechanism in Section 2.4.2 refers to a critical component of a Transformer's NN architecture. Regarding the components of the *Source*, they can be conceptualized as a set of $\langle Key, Value \rangle$ data pairs. To determine the weight coefficient of each *Key* that corresponds to *Value* (V) for a particular element of the *Target*, the similarity or correlation between the *Query* (Q) and each *Key* (K) must be evaluated. Once the relevant *Values* have been weighted and subsequently summed, the final Attention value is attained. Essentially, the Attention mechanism is a weighted summation of the *Value* values for each component in the *Source*, wherein the *Query* and *Key* are instrumental in computing the weight coefficients associated with the relevant *Value*.

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

where $\sqrt{d_k}$ is the dimension of the key vector k and query vector q .

2.4.4 BERT: Bidirectional Encoder Representations from Transformers

The BERT model was proposed in “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding” by a group of Google AI researchers [17]. It is a bidirectional transformer pre-trained using a combination of **masked language modeling (MLM)** objective - which implies the selection and replacement of a random sample of the tokens in the input sequence to the special token [MASK]. The MLM objective is a cross-entropy loss in predicting the masked tokens. And **next sentence prediction (NSP)**, a binary classification loss for predicting whether two segments follow each other in the original text. Both are trained on a large corpus comprising the Toronto Book Corpus and Wikipedia.

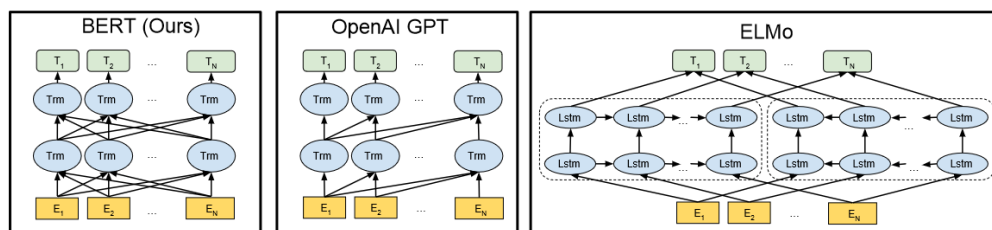


Figure 2.9: Differences in pre-training model architectures: BERT, GPT and ELMo [17]

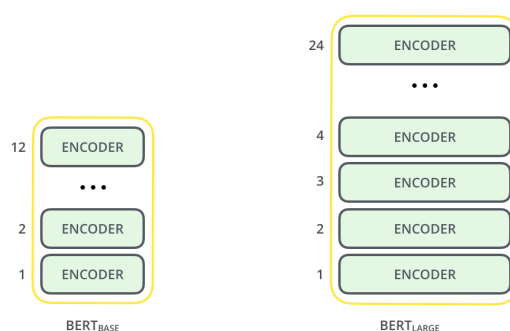


Figure 2.10: $BERT_{Base}$ VS $BERT_{Large}$ [17]

The release of BERT is an event described as marking the beginning of a new era in NLP as it broke several records for how well models can handle language-based tasks.

According to the paper, its approach enables the model to achieve a better understanding of the language context and word surroundings in comparison to other techniques, which looked at sequences either from left-to-right/right-to-left or with a combination of both single-direction language models [18]. Figure 2.9 illustrates this significant change of BERT:

BERT is fundamentally a trained Transformer Encoder stack with a multitude of encoder layers - 12 for the Base version and 24 for the Large version - that the article refers to as Transformer Blocks (see Figure 2.10).

In terms of its foundational influences and underlying concepts, the BERT model revolves around several key aspects:

1. **Semi-supervised Sequence Learning:** BERT adopts various approaches, including unsupervised feature-based methods, unsupervised fine-tuning procedures, and transfer learning from supervised data. These techniques enable the model to leverage unlabeled data effectively.
2. **Embeddings from Language Models (ELMo):** ELMo, trained to predict the next word in a sequence, generates contextualized embeddings by combining hidden states and initial embeddings in a specific manner. This approach, known as lan-

guage modeling, allows the model to learn from extensive amounts of unlabeled text input.

3. **Universal Language Model Fine-tuning for Text Classification (ULM-FiT):** ULM-FiT provides a transfer learning technique for NLP akin to the advancements witnessed in Computer Vision. It efficiently fine-tunes the language model for different tasks, expanding the applicability of transfer learning in NLP.
4. **OpenAI Transformers:** BERT leverages self-attention mechanisms, made possible by OpenAI Transformers. This utilization marks BERT as the first NLP approach to rely solely on self-attention mechanisms.

In essence, BERT is pre-trained to predict hidden or masked tokens within a sentence based on contextual cues and to comprehend the relationship between two given sentences. The framework of BERT consists of two steps: pre-training, where the model is trained on unlabeled data across multiple tasks, and fine-tuning. During fine-tuning, the model is initially initialized with the pre-training parameters and subsequently fine-tuned using labeled data. This process ensures the model's adaptation to specific downstream tasks.

2.4.5 RoBERTa: Robustly optimized BERT approach

While BERT stands as one of the leading language models in the field, it has certain limitations. Notably, BERT's training was evidently insufficient, utilizing only 16GB compared to RoBERTa's extensive 160GB training corpus [19]. Furthermore, both the pre-training and fine-tuning phases of the framework could benefit from enhancements. A primary drawback of BERT lies in its single application of masking during pre-training, resulting in a static mask that persists throughout each epoch. In response to these shortcomings, Facebook adopted the open-source BERT, originally developed by Google, and introduced their own optimization: RoBERTa [20].

While having the same architecture, BERT and RoBERTa diverge on some aspects [21]. To augment the performance of the BERT architecture, Facebook AI researchers made several straightforward design modifications in both the model's architecture and training procedure. These alterations include:

1. **Removal of the Next Sentence Prediction objective:** In BERT, the model is trained to predict whether observed document segments originate from the same or different documents, achieved through an auxiliary Next Sentence Prediction (NSP) loss. The authors conducted experiments by removing or incorporating the NSP loss in different versions and found that eliminating it either matched or slightly improved downstream task performance.
2. **Training with bigger batch sizes and longer sequences:** Initially, BERT was trained for 1 million steps with a batch size of 256 sequences. However, in this study, the authors trained the model for 125,000 steps with a batch size of 2,000 sequences, as well as for 31,000 steps with a batch size of 8,000 sequences. This approach offers two advantages: larger batches enhance perplexity on the masked language modeling

objective and improve end-task accuracy. Additionally, larger batches facilitate easier parallelization through distributed parallel training.

3. **Dynamically masking approach:** In the original BERT architecture, masking is performed once during data pre-processing, resulting in a single static mask. To overcome this limitation, the training data is duplicated and masked 10 times, each time employing a distinct mask strategy over 40 epochs, resulting in 4 epochs with the same mask. This strategy is compared with dynamic masking, where different masking is generated every time data is passed into the model.

RoBERTa has demonstrated superior performance compared to BERT and other state-of-the-art models across various natural language processing tasks, including language translation, text classification, and question answering. It has also served as a foundational model for numerous successful NLP models and has gained popularity within both research and industry applications.

2.4.6 XLNet: eXtreme Learning Network

XLNet [22], a state-of-the-art language model, combines the strengths of two successful models, Transformer-XL [23] and BERT [17]. Transformer-XL addresses the limitations of recurrent neural networks (RNNs) and LSTM networks by reusing hidden states from previous segments, enabling the model to capture longer-term dependencies and prevent context fragmentation. This auto-regressive language model outperforms traditional models by leveraging the power of vanilla Transformers.

Two innovative techniques contribute to XLNet's success in overcoming limitations in auto-regressive language modeling [24]. First, the recurrence mechanism caches the hidden state sequence generated for the previous segment during training. This extended context allows the model to utilize past knowledge effectively. Second, relative positional encoding ensures consistency in positional information when reusing hidden states. By encoding only relative positional information in the hidden states, the model obtains a temporal guide on how to obtain information and dynamically injects relative distance into each layer's attention score.

Building upon the architecture and techniques of Transformer-XL, XLNet is an auto-regressive language model that outputs the joint probability of token sequences. It learns bidirectional context by considering all possible permutations of the input sequence factorization order, combining the advantages of auto-regressive modeling (Transformer-XL) and auto-encoding modeling (BERT). Unlike BERT, XLNet does not rely on masked input or token independence, resulting in consistent training and fine-tuning.

2.4.7 HateBERT and DeHateBERT: Language-Specific Abusive Language Detection Models

In order to enhance the performance of abusive language detection across different languages, two separate models specifically trained on English and Spanish text will be chosen to utilize. The first model, HateBERT [25], is based on the BERT architecture and

BERT	HateBERT
“women”	
excluded (.075)	stu**d (.188)
encouraged (.032)	du*b (.128)
included (.027)	id***s (.075)

Figure 2.11: MLM top 3 candidates for the templates “Women are [MASK.]” [25]

has been retrained to focus on detecting abusive language in English. The second model, DeHateBERT [26], is a multilingual language model with a specific Spanish version model trained to detect the abusive language in Spanish text called DeHateBert-mono-spanish. By incorporating these language-specific models, the aim is to achieve better results and more accurate detection of abusive language in both English and Spanish contexts, as these are the languages involved in the datasets that we are going to experiment with.

HateBERT, a modified version of the English BERT base-uncased model, was retrained on the RAL-E dataset with 1,478,348 training messages. Using the Masked Language Model (MLM) objective and training for 100 epochs, HateBERT consistently filled mask tokens with profanities or abusive terms (see Figure 2.11). In comparison, the generic BERT model rarely generated such outputs. This modification allowed HateBERT to identify offensive and abusive language effectively. The results were demonstrated through a template sentence analysis, highlighting the differences between BERT and HateBERT in predicting masked tokens.

On the other hand, DeHateBERT, is a model that offers similar capabilities for detecting abusive language in Spanish, which has been trained on two datasets: Basile [27], provided multilingual hate speech dataset against immigrants and women, and Pereira [28], provided hate speech dataset for the Spanish language. Both models have been fine-tuned on extensive datasets in their respective languages to ensure optimal performance and language-specific detection capabilities.

2.5 Evaluations and metrics

In evaluating the performance of pre-trained and fine-tuned models, it is essential to consider multiple metrics. Assessing models using diverse evaluation metrics is crucial to ensure their accurate and optimal functioning. A model may excel in one metric but perform poorly in another, making the evaluation of various metrics necessary. Additionally, employing a wide range of metrics facilitates comprehensive model comparisons on a state-of-the-art (SOTA) scale. Within this context, the confusion matrix depicted in Figure 2.12 plays a vital role as a tool for evaluating models. It provides a summary of performance by presenting counts of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). These concepts are essential in assessing the required evaluation

metrics for models. The matrix enables the identification of model strengths and weaknesses, highlights errors across different classes, and assists in decision-making for model improvement.

		PREDICTED LABEL	
		NEGATIVE	POSITIVE
TRUE LABEL	NEGATIVE	TRUE NEGATIVE	FALSE POSITIVE
	POSITIVE	FALSE NEGATIVE	TRUE POSITIVE

Figure 2.12: Confusion Matrix [29]

Within this project, the key metrics to be utilized are as follows:

1. **Accuracy:** This metric measures the proportion of correct predictions or tokens made by the model. It is calculated by dividing the sum of true positives and true negatives by the total population.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.1)$$

2. **Precision:** Precision gauges the ratio of accurate positive predictions to the total number of positive predictions. In other words, it is calculated by dividing TP by the total number of positive calls.

$$Precision = \frac{TP}{TP + FP} \quad (2.2)$$

3. **Recall:** Recall assesses the proportion of correct positive predictions among all instances that are actually positive. It is determined by dividing the number of TP by the sum of TP and false negatives (FN).

$$Recall = \frac{TP}{TP + FN} \quad (2.3)$$

4. **F1-score:** The F1-score represents the harmonic mean of precision and recall. It provides a balanced measure of the model's performance, accounting for both precision and recall simultaneously.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \quad (2.4)$$

These metrics serve as essential benchmarks for evaluating the performance of models and enable meaningful comparisons between different approaches. By considering these metrics, we gain valuable insights into the accuracy, precision, recall, and overall performance of the models. The F1-score, in particular, combines precision and recall to provide a comprehensive assessment of the model's effectiveness.

Chapter 3

Implementation

This chapter is dedicated to providing a comprehensive explanation of the implementation process in this project. The discussion will begin with the project's implementation scenario, and will include the IDE environment, tools utilized, data preparation and the specific structure designed for the purpose of collecting results.

3.1 Implementation scenario

This project aligns with the prevailing trend in data science, deep learning, and the NLP field by selecting Python as the primary programming language and leveraging Jupyter Notebook for coding support. Python is widely recognized as the best language for machine learning and deep learning research due to its simplicity, consistency, and extensive library ecosystem [30]. These libraries enable efficient data access, handling, and transformation, which will be discussed in detail later. Jupyter Notebook serves as an interactive computational environment for developing Python-based data science applications. It seamlessly integrates code, images, text, and output, allowing for a step-by-step illustration of the analysis process. Additionally, it facilitates the documentation of the thought process undertaken by data scientists during analysis development.

In this project, Python 3.10 and Jupyter Notebook were selected due to their versatility and compatibility with pre-trained models. The hardware used for implementation consisted of an Huawei Matebook 14 laptop with an AMD Ryzen 7 4800H processor with Radeon Graphics processor which operates at a base frequency of 2.90 GHz, 16GB of RAM, and 512GB SSD. Considering the limitations of local resources, the idea of using client-server applications like Google Colab or Kaggle, which provide access to server-side GPUs and CPUs, was explored. However, since they are free public services, the usage time of these platforms is limited.

On the software side, the chosen IDE for this project is Jupyter Notebook, but due to resource limitations, it is supplemented by Google Colab. Google Colab is a client-server application provided by the Jupyter Project, acts as a bridge between code and explanatory

texts, allowing for code compilation and execution. Jupyter Notebook enables developers to combine code with rich text, equations, images, and multimedia renderings in one place. It also facilitates result sharing and analysis. Although resource constraints are present, they do not hinder the achievement of project goals.

3.2 Libraries

This section discusses the Python libraries used to accomplish the project goals. The purpose of each library is explained along with the functions and features applied. The following libraries were used for general and data science purposes:

3.2.1 LIBRARIES FOR GENERAL AND DATA SCIENCE PURPOSES

1. **os**: A Python module that provides functions to interact with the operating system. It was used to create directories for storing the results.
2. **re**: A Python module that contains regular expression matching operations. It was used to search for specific patterns and keywords, such as hashtags and URLs.
3. **time**: A Python module that provides time-related functions. It was used to measure the time spent during training and validation.
4. **datetime**: A Python module that provides time and date-related functions. It was used to transform time in seconds into a more readable format.
5. **enum**: A Python module that defines enumeration classes used for defining unique dataset types while saving predictions and metadata.
6. **json**: A Python module that provides an API for converting Python objects to a serialized representation called JavaScript Object Notation (JSON). It was used for loading emojis and abbreviation dictionaries.
7. **pickle**: A Python module that implements binary protocols for serializing and de-serializing data, such as metadata files.
8. **googletrans**: An open-source Python library that provides an interface for Google Translate. It enables developers to easily integrate language translation capabilities into their applications, allowing for the translation of text from one language to another.
9. **collections**: A Python module that implements specialized container data types. The used types were `namedtuple`, `defaultdict`, and `Counter`.
10. **random**: A Python module that generates pseudo-random numbers. It was used to start random numbers given an initial seed.
11. **logging**: A Python API that defines functions and classes for implementing an event logging system. It was used to disable `smote_variants` logging to avoid spam during training.

12. **unicodedata**: A Python module that provides access to the Unicode Character Database. It was used to remove accents from Unicode characters that contain them.
13. **Scikit-Learn (SkLearn)**: An open-source machine learning API that provides numerous efficient tools for statistical modeling and supervised-unsupervised learning. The following submodules were used:
 - (a) **feature_extraction.text**: A submodule for building feature vectors from text documents. It provided vectorizers (TfidfVectorizer and CountVectorizer) needed for preprocessing the datasets.
 - (b) **metrics**: A submodule that includes score functions, performance metrics, and pairwise metrics and distance computations, like accuracy scores and F1 metrics used in validation processes.
 - (c) **model_selection**: This submodule is used for cross-validation.
 - (d) **pipeline**: A submodule that helps build the pipeline used for generating the Tf-Idf matrix of one dataset.
 - (e) **preprocessing**: A submodule that includes scaling, centering, normalization, binarization methods, such as FunctionTransformer and OneHotEncoder. These are the functions used during the generation of the pipeline and training, respectively.
14. **Numpy**: A Python library that consists of multidimensional array objects and a collection of routines for processing these arrays. It was used for numerical and statistical functions, such as sum and mean, and stacking methods for arrays.
15. **Pandas**: An open-source Python library that provides multiple machine learning functions for dealing with DataFrames and Data Science tasks. It was used for loading datasets or files and concatenating and creating DataFrames from those loaded files.
16. **Scipy**: An open-source Python library that is used to solve scientific and mathematical problems involving matrices. It was used to load sparse matrices from npz files and also save them.
17. **Torch**: An open-source Python machine learning library and a scientific framework that provides a wide range of algorithms and methods for deep learning. The following submodules were used:
 - (a) **nn**: A submodule enables the creation and training of neural networks and provides Loss functions for their evaluation.
 - (b) **utils.data**: A submodule provides functions for loading, dealing, and interacting with data, along with several data samplers to specify the sequence of keys used in data loading, including DataLoader.

3.2.2 LIBRARIES FOR VISUALIZATION PURPOSES

1. **Matplotlib**: the open-source library is used to create static, animated, and interactive visualizations in Python.
 - (a) **pyplot**: A submodule provides a state-based interface and an implicit, MATLAB-like way of plotting, with the main goal of outputting figures for visualizing the pre-processing, training, and validation results.
2. **Seaborn**: Based on Matplotlib, this library is used for drawing attractive and informative statistical graphics to aid in the understanding of training and evaluation results.

3.2.3 NLP-RELATED LIBRARIES

1. **Transformers**: A Python library that provides thousands of pre-trained models for performing tasks such as sentiment analysis or text classification. For this thesis, the pre-trained models used were Bert, RoBERTa, and XLNet, along with their respective learning rate schedulers from this library.
2. **Spellchecker**: A Python module that allows setting a Levenshtein Distance algorithm to find permutations, within an edit distance of 2, from an original word. The goal of this module is to check for erroneous words and return their correct spelling.
3. **NLTK**: A Python package used for natural language processing tasks such as tokenization, stemming, lemmatization, parsing, tagging, and semantic reasoning. It provides several submodules, such as:
 - (a) **corpus**: This submodule provides functions for reading corpus files, which are large collections of texts used for natural language processing research. In this case, the goal is to obtain stopwords collections for languages such as Spanish and English.
 - (b) **stem.wordnet**: This submodule provides the principal function for lemmatization, which is the process of reducing words to their base or root form. This is an important step in pre-processing natural language data.
 - (c) **tokenize.regexp**: This submodule provides a regular expression tokenizer that can be used to tokenize corpus words without digits, while ignoring punctuation, except for specific cases such as Usernames and Hashtags. Tokenization is the process of splitting text into smaller units, such as words or sentences, for further processing or analysis.

3.3 Structure of Jupyter Notebook

After introducing the necessary libraries, modules, APIs, and packages, it is essential to discuss the execution of the project. The development process was carried out within Jupyter Notebooks, with each notebook dedicated to a specific dataset. Consequently,

the project consists of two separate notebooks, each encompassing the entire pipeline, from data imports to the training process of all five models, finally culminating in the evaluation of the final models. In Section 3.3, we will provide in-depth insights into these notebooks through dedicated subsections. While the overall structure remains consistent across both notebooks, there are certain distinctions in the code, such as modified functions and dataset-specific adjustments, which reflect the unique characteristics of each dataset.

Furthermore, it is important to highlight that the framework for the subsequent parts, including statistical analysis and corpus pre-processing, draws upon the previous work of María Isabel González in 2022. Her thesis titled *"Research and Analysis of Hate and Other Emotions in Social Media"* [31] focused on studying hate speech and other emotions in social media using datasets such as HatEval2019, Detoxis, Emoevent, and Universal Joy Dataset, along with various NLP techniques. Notably, we share the same mentor for our respective theses. However, it is crucial to emphasize that in adapting her framework to the domain of sexism detection, significant modifications have been made. Function by function, we have tailored and customized the code to meet the specific requirements and characteristics of the different datasets used in our study. This ensures a robust framework that aligns with the unique aspects of our field and facilitates accurate model training.

3.3.1 Preparation: Install, imports and data loading

To begin, each Jupyter Notebook will include the necessary imports for the subsequent code. In a Google Colab environment, the notebook will also handle Drive Mount and relevant pip installations. The imports are structured in sections, starting with General Purpose or Data Science libraries, followed by NLP-specific ones. In between, there is a configuration step using matplotlib and seaborn libraries to enhance data visualization and highlight the statistical results. Additionally, the datasets from the inputs directory will be loaded. These datasets mainly consist of files with **tsv** (Tab-Separated Values) and **json** (JavaScript Object Notation) extensions. Pandas library functions will read these files and save them as DataFrames, with names that identify their dataset and task.

3.3.2 Data used for training

This section focuses on the discussion of two datasets selected for training and evaluating the models. To identify sexist content, the EXIST dataset¹ is used, covering the years 2021 and 2023. Unfortunately, the 2022 dataset is currently unavailable to the public due to internal reasons. The datasets are organized into three parts: training, development, and test. The EXIST 2021 dataset lacks a separate development dataset, whereas the 2023 dataset includes one. In cases where a development dataset is absent, a portion of the training data is allocated for this purpose. Additionally, it is noteworthy that the EXIST 2021 test dataset is labeled, whereas there are no labels for the corresponding dataset in 2023. The train set is the largest, facilitating model training and classification of tasks such as hate speech detection and offensive language identification. The development set is

¹For more information about the EXIST dataset, please refer to <http://nlp.uned.es/exist2021/> and <http://nlp.uned.es/exist2023/>.

smaller and serves to evaluate the training progress. Finally, the test set is used to assess the overall performance. Each dataset follows a similar structure, ensuring consistency in the evaluation process, which will be explained as follows.

EXIST2021

EXIST2021 dataset consists of more than 11.000 short text, tweets and gabs, both in English and Spanish. In particular, the EXIST training set contains 6977 tweets while the test set contains 3386 tweets and 982 gabs [5]. Distribution between both languages has been balanced.

The training and test sets are provided in a tsv format. In particular, the training dataset contains the following columns:

1. **Test_case**: tag needed in the EvALL framework for evaluating classification tasks. In EXIST 2021, this tag is set to "EXIST2021".
2. **Id**: denotes a unique identifier of the tweet.
3. **Source**: defines the social network where the text was crawled, only "twitter" in the training set.
4. **Language**: denotes the languages of the text ("en" or "es").
5. **Text**: represents the text of the tweet.
6. **Task1**: indicates if the tweet is sexist ("sexist") or not ("non-sexist").
7. **Task2**: categorize the message according to the type of sexism. Possible categories are: "ideological-inequality", "stereotyping-dominance", "objectification", "sexual-violence" and "misogyny-non-sexual-violence".

Figure 3.1 plots a few examples of tweets in the training set.

test_case	id	source	language	text	task1	task2
EXIST2021	1	twitter	en	She calls herself "anti-feminazi" how about sh...	sexist	ideological-inequality
EXIST2021	2	twitter	en	Now, back to these women, the brave and the be...	non-sexist	non-sexist
EXIST2021	3	twitter	en	@CurvyBandida @Xalynne_B Wow, your skirt is ve...	sexist	objectification
EXIST2021	4	twitter	en	@AurelieGuiboud Incredible! Beautiful!But I l...	non-sexist	non-sexist
EXIST2021	5	twitter	en	i find it extremely hard to believe that kelly...	non-sexist	non-sexist

Figure 3.1: EXIST 2021 dataset loading preview

EXIST2023

The EXIST 2023 dataset differs from the EXIST2021 dataset in terms of data format, using JSON instead of TSV. It consists of over 10,000 labeled tweets in English and Spanish. The training set contains 6,920 tweets, the development set has 1,038 tweets, and the test set

includes 2,076 tweets. The distribution between both languages is balanced. The data sets are provided in JSON format, where each tweet is represented as a JSON object with various attributes:

1. **id_EXIST**: denotes a unique identifier of the tweet.
2. **lang**: denotes the languages of the text (“en” or “es”).
3. **tweet**: represents the text of the tweet.
4. **number_annotators**: the number of persons that have annotated the tweet.
5. **annotators**: the name of persons that have annotated the tweet, represented with anonymous format.
6. **gender_annotators**: the gender of the different annotators. Possible values are: “F” and “M”, for female and male respectively.
7. **age_annotators**: the age group of the different annotators. Possible values are: 18-22, 23-45, and 46+.
8. **labels_task1**: a set of labels (one for each of the annotators) that indicate if the tweet contains sexist expressions or refers to sexist behaviours or not. Possible values are: “YES” and “NO”.
9. **split**: subset within the dataset the tweet belongs to (“TRAIN”, “DEV”, “TEST” + “EN”/“ES”).

The two remaining tags: “labels_task2” and “labels_task3”, for source intention detection and sexism categorization respectively, are not related to the code implementation discussed in this project, thus are not included in the list presented above. Figure 3.2 depicts a comparison between the original training data in JSON format and the corresponding data loaded in a Pandas DataFrame format.

id_EXIST	lang	tweet	number_annotators	annotators	gender_annotators	age_annotators	labels_task1	split
100001	es	@TheChifflis Ignora al otro, es un capullo.El problema con este youtuber denuncio el acoso... cuando no afecta a la gente de izquierdas. Por ejemplo, en su video sobre el gasegarte presenta como \"normal\" el acoso que reciben Fisher, Anita o Zöey cuando hubo hasta amenazas de bomba.	6	[Annotator_1, Annotator_2, Annotator_3, Annota...	[F, F, F, M, M, M]	[18-22, 23-45, 46+, 46+, 23-45, 18-22]	[YES, YES, NO, YES, YES, YES]	TRAIN_ES
100002	es	@ultimonomada. Si comicgase se parece en algo...	6	[Annotator_7, Annotator_8, Annotator_9, Annota...	[F, F, F, M, M, M]	[18-22, 23-45, 46+, 46+, 23-45, 18-22]	[NO, NO, NO, NO, YES, NO]	TRAIN_ES
100003	es	@Steven2897 Lee sobre Gasegarte, y como eso ha...	6	[Annotator_7, Annotator_8, Annotator_9, Annota...	[F, F, F, M, M, M]	[18-22, 23-45, 46+, 46+, 23-45, 18-22]	[NO, NO, NO, NO, NO, NO]	TRAIN_ES
100004	es	@Luneriite7 Un retraso social bastante lamenta...	6	[Annotator_13, Annotator_14, Annotator_15, Annota...	[F, F, F, M, M, M]	[18-22, 23-45, 46+, 46+, 23-45, 18-22]	[NO, NO, YES, NO, YES, YES]	TRAIN_ES
100005	es	@novadragon21 @sep4ck @TvDannyZ Entonces como...	6	[Annotator_19, Annotator_20, Annotator_21, Annota...	[F, F, F, M, M, M]	[18-22, 23-45, 46+, 46+, 23-45, 18-22]	[YES, NO, YES, NO, YES, YES]	TRAIN_ES

Figure 3.2: EXIST 2023 original training data (json) VS training data loading preview

3.3.3 Statistical analysis of dataset

This subsection's main goal is to perform a specialized study of the imported datasets in accordance with the particular task being carried out. In order to do this, functions that offer comprehensive details about the contents of each dataset, both separately and jointly, will be defined. Additional categories for the study include:

1. Distribution of features

An important aspect to consider when analyzing a dataset is the distribution of its features, which are used to classify and categorize data into different fields. This task heavily relies on the last columns of the dataset, which have been previously encoded from text to numerical representations. In order to achieve this objective, we implemented a specific function *print_distribution_exist*. This function serves the purpose of analyzing the distribution of data points for each language and subsequently determining the proportion of text classified as sexist (1) or non-sexist (0) using the encoded column (task1_encoding) within the corpus. The analysis is conducted separately for Spanish and English texts.

In addition to our own implementation, we also refer to the work of María Isabel [31], whose code includes a similar functionality called *print_distribution_hateval* for her analysis on Hateval2019 Dataset. While our focus is on analyzing the distribution of sexist content in different languages, Maribel's function emphasizes the classification of text as hate speech or non-hate speech, along with corresponding levels of aggressiveness categorized into numerical levels. The comparison of these two functions, as shown in Figure 3.3, aims to explain our adaptation of the code to our specific domain of sexism detection, highlighting the importance of tailoring the analysis to the unique characteristics of the dataset and research objective.

The distribution of features in the two datasets, EXIST2021 and EXIST2023, provides valuable insights into the composition of the texts (see Figure 3.4). To elaborate, in the EXIST2021 Dataset, a total of 11,345 texts were analyzed. Among these, 5,701 texts were in Spanish, while 5,644 texts were in English. In terms of the distribution of sexist and non-sexist messages, the percentage of sexist messages in Spanish was found to be 50.24%, while the percentage of non-sexist messages in Spanish was 49.76%. In contrast, the percentage of sexist messages in English was 49.50%, and the percentage of non-sexist messages in English was 50.50%. Moving on to the EXIST2023 Dataset, a total of 7,958 texts were examined. Of these, 4,209 texts were in Spanish, while 3,749 texts were in English. The distribution of sexist and non-sexist messages revealed that the percentage of sexist messages in Spanish was 49.42%, while the percentage of non-sexist messages in Spanish was 50.58%. In the case of English texts, the percentage of sexist messages was 41.53%, while the percentage of non-sexist messages was 58.47%.

These distributions highlight the linguistic composition and the varying prevalence of sexist messages in both datasets. However, it is important to note that the huge decrease in the EXIST2023 dataset size is due to the availability of labeled data. In

```

def print_distribution_hateval(data):
    """
    Prints feature distribution of the dataset HatEval2019.
    TR and AG are always dependent on HS being 'True'.
    As checked on the last print (should always be 0%).

    Parameters
    -----
    data : Pandas DataFrame
           HatEval2019 Dataset

    Returns
    -----
    None

    """
    print("Distribution of features:")
    print(f"HS: {100 * np.sum(data['HS'] == True) / data.shape[0]}%")
    print(f"HS -> TR: {100 * np.sum((data['HS'] == True) & (data['TR'] == True)) / np.sum(data['HS'] == True)}%")
    print(f"HS -> AG: {100 * np.sum((data['HS'] == True) & (data['AG'] == True)) / np.sum(data['HS'] == True)}%")
    print(f"HS -> TR & AG: {100 * np.sum((data['HS'] == True) & (data['TR'] == True) & (data['AG'] == True)) / np.sum(data['HS'] == True)}%")
    print(f"HS & (TR | AG): {100 * np.sum((data['HS'] == False) & ((data['TR'] == True) | (data['AG'] == True))) / data.shape[0]}%")

```

(a) *print_distribution_hateval* function in María Isabel's hateval2019 Notebook

```

def print_distribution_exist(data):
    """
    Prints feature distribution per language of the dataset EXIST2023.

    Parameters
    -----
    data : Pandas DataFrame
           EXIST2023 Dataset

    Returns
    -----
    None

    """
    language_mapping = {'es': 'Spanish', 'en': 'English'}

    languages = ['es', 'en']

    for language_code in languages:
        language_name = language_mapping.get(language_code.lower(), language_code)
        language_data = data[data['lang'] == language_code]
        num_texts = language_data.shape[0]
        num_sexist = language_data[language_data['label1_encoded'] == 1].shape[0]
        num_non_sexist = num_texts - num_sexist
        pct_sexist = (num_sexist / num_texts) * 100 if num_texts > 0 else 0
        pct_non_sexist = (num_non_sexist / num_texts) * 100 if num_texts > 0 else 0

        print(f"Number of texts in {language_name}: {num_texts}")
        print(f"Percentage of sexist messages in {language_name}: {pct_sexist:.2f}%")
        print(f"Percentage of non-sexist messages in {language_name}: {pct_non_sexist:.2f}%")

    total_texts = data.shape[0]
    print(f"Total number of texts: {total_texts}")

```

(b) *print_distribution_exist* function in EXIST2023 Notebook

Figure 3.3: Illustration of code adaptation across datasets

the case of EXIST2021, all three sets (training, development, and test) are labeled, allowing us to include the entire dataset for this feature analysis. On the other hand, in EXIST2023, the test set is unlabeled, resulting in a smaller dataset size consisting of only the training and development sets. This distinction in dataset size should be considered when interpreting the results. The analysis of these distributions serves as a foundation for understanding the characteristics of the data and will contribute to the subsequent steps of the study.

```
EXIST2021 Dataset:  
Total number of texts: 11345  
Number of texts in Spanish: 5701  
Number of texts in English: 5644  
Percentage of sexist messages in Spanish: 50.24%  
Percentage of non-sexist messages in Spanish: 49.76%  
Percentage of sexist messages in English: 49.50%  
Percentage of non-sexist messages in English: 50.50%
```

(a) EXIST 2021

```
EXIST2023 Dataset:  
Total number of texts: 7958  
Number of texts in Spanish: 4209  
Number of texts in English: 3749  
Percentage of sexist messages in Spanish: 49.42%  
Percentage of non-sexist messages in Spanish: 50.58%  
Percentage of sexist messages in English: 41.53%  
Percentage of non-sexist messages in English: 58.47%
```

(b) EXIST 2023

Figure 3.4: Feature Distributions through Datasets

2. Patterns information of the corpus

In this analysis, the text patterns with the highest frequency of occurrence will be emphasized, including usernames, hashtags, and URLs. Each pattern will be accompanied by its count of usage within the corpus, with the most frequently used ones listed for the datasets. Here is an example of the most commonly mentioned (persons or hashtags) and its categorization in English text of EXIST 2021, as shown in Figures 3.5 and 3.6.

Most Common mentions in English (%):

	Count	non-sexist	misogyny-non-sexual-violence	sexual-violence	objectification	stereotyping-dominance	ideological-inequality
Mention							
#patriarchy	63	51.851852		3.703704	3.703704	0.925926	14.814815
#mgtow	60	31.683168		10.891089	5.940594	4.950495	11.881188
#everydaysexism	56	40.000000		7.000000	6.000000	5.000000	25.000000
#notallmen	54	50.574713		4.597701	11.494253	4.597701	3.448276
#gamergate	53	74.257426		5.940594	3.960396	4.950495	1.980198
#womensrights	50	63.529412		4.705882	3.529412	1.176471	3.529412
#harassed	46	58.227848		3.797468	16.455696	5.063291	5.063291
#womensday	37	66.666667		10.606061	3.030303	3.030303	7.575758
#feminism	25	44.444444		8.888889	2.222222	0.000000	11.111111
#horny	24	22.222222		6.666667	40.000000	6.666667	13.333333

Figure 3.5: Hashtags patterns found in EXIST2021 English.

Most Common mentions in English (%):

	Count	objectification	non-sexist	stereotyping-dominance	misogyny-non-sexual-violence	sexual-violence	ideological-inequality
Mention							
@realdonaldtrump	50	8.695652	55.072464	11.594203	11.594203	7.246377	5.797101
@	23	2.083333	66.666667	0.000000	14.583333	4.166667	12.500000
@irenemontero	20	0.000000	35.483871	3.225806	19.354839	6.451613	35.483871
@porn_click_1360	16	0.000000	50.000000	0.000000	0.000000	50.000000	0.000000
@youtube	14	5.000000	80.000000	5.000000	0.000000	5.000000	5.000000
@lopezobrador_	10	0.000000	66.666667	27.777778	5.555556	0.000000	0.000000
@aoc	9	0.000000	23.076923	23.076923	7.692308	23.076923	23.076923
@sex_porn_linkk	9	0.000000	0.000000	0.000000	0.000000	100.000000	0.000000
@superrealscott	8	13.333333	46.666667	0.000000	20.000000	6.666667	13.333333
@vlknight81	8	13.333333	46.666667	0.000000	20.000000	6.666667	13.333333

Figure 3.6: Mentions patterns found in EXIST2021 English.

Figure 3.5 provides valuable insights into the classification of tweets based on specific mentions. Notably, approximately 78% (100% - 22%) of the text associated with the hashtag #horny is classified as sexist, while 67% of tweets using #womensday are categorized as non-sexist. Similarly, as shown in Figure 3.6, tweets containing terms like “porn”, such as “@porn_click_1360” and “@sex_porn_linkk” are consistently identified as 100% sexist messages. These observations highlight the varying nature of tweets based on different hashtags and mentions, as well as the presence of explicit content associated with sexism.

3. Corpus characteristics using statistics

To gain a better understanding of the corpus, it is necessary to conduct a general review by analyzing its characteristics and statistics. This involves examining three key aspects of the datasets: the number of characters per tweet, the number of words per tweet, and the average length of a word in a tweet. However, the analysis is not limited to these three values alone. For each aspect studied, the minimum, maximum, and average values will be calculated and displayed. By performing

this simple statistical analysis, the objective is to ascertain the typical structure of a tweet. This information will be useful for determining the appropriate length for each tweet, aligning it with the maximum length supported by the models used in future implementation.

Figure 3.7 summarizes these characteristics in details. In case of EXIST2021 dataset, the tweets in this dataset vary in length. The number of characters ranges from 1 to 3520, with an average of approximately 174.6 characters per tweet. The mean number of words in the tweets is around 28.8, with a maximum of 553 words per tweet. The average word length is approximately 5.5 characters. Similarly, the EXIST2023 dataset exhibits similar characteristics. The number of characters in the tweets ranges from 12 to 3520, with an average of approximately 179.8 characters per tweet. The mean number of words is around 29.2, with a maximum of 553 words per tweet. The average word length is approximately 5.6 characters.

```
Tweets: (MIN, MEAN, MAX)
Number of characters: (1, 174.56439847577573, 3520)
Mean of Number of words: (1, 28.80468154599891, 553)
Mean of Avarage word lenght: (1.0, 5.495863752250645, 309.0)
```

(a) Corpus Characteristics of EXIST 2021

```
Tweets: (MIN, MEAN, MAX)
Number of characters: (12, 179.82763760246993, 3520)
Mean of Number of words: (1, 29.21973810284254, 553)
Mean of Avarage word lenght: (2.2, 5.554771212682303, 309.0)
```

(b) Corpus Characteristics of EXIST 2023

Figure 3.7: Corpus Characteristics of Datasets

4. Bar charts of statistics

Building on the information collected in the previous subsections, this final section presents a series of illustrative graphs. The primary goal of these visualizations is to present the calculated statistics in a clear and easily understandable format. To accomplish this, two types of bar charts are utilized. Firstly, the focus will be on the **stopwords** present in the corpus and their respective weight. Two separate bar charts will be plotted: one for the most frequently used stopwords and another for the non-stopwords. The last two bar charts will focus on the most common **bigrams** and **trigrams** present in the corpus, respectively. These charts will highlight the most frequently occurring combinations of words used in posts, providing valuable insights for further analysis.

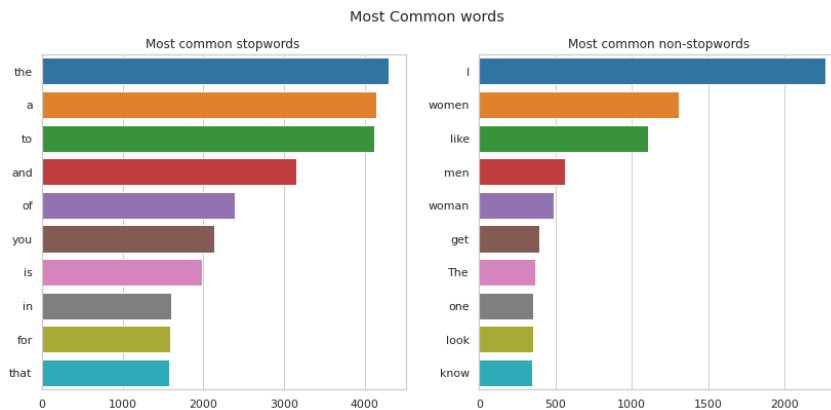


Figure 3.8: Most common bigrams in English EXIST 2021.

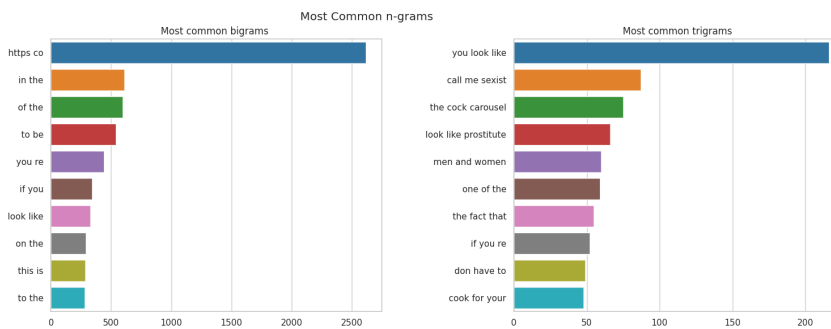


Figure 3.9: Most common trigrams in English EXIST 2023

3.3.4 Dataset pre-processing procedure

After completing the initial stage of dataset analysis and presenting its statistical information, the focus shifts towards the fundamental aspects of the notebooks: pre-processing and training. The following section will primarily address the topic of pre-processing the corpus, which is essential for its subsequent utilization in training. To accomplish this, we will employ numerous techniques and methods elaborated upon in the earlier Section 2.2, specifically highlighting pre-processing techniques.

In essence, the central concept around which the pre-processing revolves is the conversion of the corpus from text into a **Document-Term matrix** populated with **TF-IDF frequencies**. This matrix captures the frequencies of words extracted from the corpus, and before anything else, undergoes an initial pre-processing phase. The *feature_extraction.text* module of the Sklearn library provides the *TfidfVectorizer* function for this conversion, transforming nearly raw tweets into a matrix of TF-IDF features. The initial pre-processing phase comprises three techniques mentioned earlier:

1. **Tokenization:** as previously explained, involves segmenting the entire text of tweets

or posts into individual word-based units called tokens. Numbers are not considered for token formation, and accents and punctuation are ignored at this stage for subsequent treatment in the second pre-processing phase.

2. **Stopwords Removal:** two sets of stopwords will be initialized: “en_stop_set” for English stopwords and “es_stop_set” for Spanish stopwords, depending on the input text’s language, which allows the code to use the appropriate stopwords set for the given language during the stopwords removal step.
3. **Lemmatization:** in the final step of the preprocessing stage, we initialize a linguistic tool called the *WordNetLemmatizer* from the *NLTK* library. This powerful tool enables us to reduce words to their base or dictionary form, known as lemmas. The process begins by iterating through each token in the list of tokens. For each token, the lemmatizer is applied using the part-of-speech tag ‘v’ (verb), allowing us to obtain the corresponding lemma. In cases where a token is empty, it is assigned an empty string as its lemma. These lemmas are then stored in the dedicated *lemmas* list. By comparing this list of lemmas with the stopwords obtained during the previous stage, we effectively eliminate the stopwords from the lemmatized tokens. This crucial step ensures that only significant keywords remain for further analysis, enhancing the overall quality and relevance of the resulting dataset.

This whole process is being carried out by the defined function *tokenization_text* and the result of using *TfidfVectorizer* will be the vectorizer needed for the next phase of pre-processing.

The second phase involves creating a **Pipeline** necessary for training models using the generated vectorizer. This Pipeline, also from Sklearn, primarily requires two parameters: the aforementioned vectorizer and a transform function, each in a tuple with its name tag. The transform function employs the *FunctionTransformer* function from the Sklearn pre-processing module, providing the required interface for the pipeline. This interface includes standard methods like *fit* or *transform*, eliminating the need to manually implement each stage of the pipeline. This streamlined execution allows feeding the pre-processed data directly to the fit and transform methods of the Pipeline.

The *direct_replacement* function will be used in our pipeline, and prior to its execution, another secondary function will be invoked to load the necessary dictionaries in JSON format for further pre-processing steps (emojis and abbreviations). These dictionaries facilitate the replacement of emojis, both simple and complex, and abbreviations within the tokens with meaningful words. Both components of the text hold significant value for sentiment analysis since the use of certain emojis as abbreviations conveys strong emotional connotations, as shown in Figure 2.3.

Once the dictionaries are loaded with the function *load_dictionaries*, appropriate replacements for the tokens are made, including the following key steps:

1. Lower-casing every token to avoid repetition.
2. Standardizing contractions and special characters such as apostrophes, ellipses, quotation commas, and others.

3. Removing any remaining digits.
4. Replacing emojis and abbreviations with their corresponding meanings. If a meaning is not found in either dictionary, the emoji is removed from the list of tokens.
5. Removing accents.

After completing the token replacement and processing, the Pipeline required for the next section will be fully prepared to use.

For the proper functioning of this pipeline, it is crucial to provide it with appropriate data. Specifically, the pipeline requires undergoing the fitting process to acquire the necessary knowledge for text processing, feature extraction, and result modeling. To accomplish this goal, we will utilize a dedicated function called *load_Tfidf_processing* for each subdataset. This function initiates the fitting process of the pipeline. Before starting its execution, it verifies whether the extensive and time-consuming learning process has been performed previously. If it has, the function retrieves the stored information from a compressed .npz file.

3.3.5 Neural Network Pre-processing, Training and Evaluation Metrics

This subsection addresses a crucial aspect: training. After creating and training the pipeline, our focus shifts to training the different models in a comprehensive manner and pre-processing the subdatasets to ensure their readiness for the neural networks. Regarding data pre-processing, it closely resembles the process described earlier for generating the vectorizer in the pipeline. However, there is a difference. To facilitate tokenization in datasets without masked words, we remove hashtags and usernames. To achieve this, we employ the *direct_replacement()* method from the TF-IDF pre-processing section. For each subdataset, we utilize the *preprocessing_nn()* function to perform this generic pre-processing for neural networks. Additionally, an auxiliary function has been defined to sample the pre-processing performance of each subdataset, which will display the original text, followed by its tokenization and translation into IDs. After that, in order to highlight the necessity of pre-processing, a comparison which includes tokenization and translation to IDs of the pre-processed text with the original text will be shown.

The subsequent step consists of preparing the essential parameters to generate the final datasets, which will be used for training the models. These parameters include the tokenizer, the pre-processed text, the labels, and the maximum length of the tokenized text. While the tokenizer will be discussed in its specific section for each model, we will delve into the other parameters. Regarding data and labels, as mentioned previously, the data contains pre-processed texts for the neural network, and the labels are derived from the corresponding column in the set used for training. For example, in the case of the EXIST 2021 set, the labels correspond to the sexism detection column, "task1_encoding", where we have encoded sexist content as 1, and non-sexist as 0 in data loading procedures. Lastly, the maximum length of the tokenized text refers to the maximum number of tokens allowed for the inputs of the transformer model. This value is calculated by determining the minimum between the pre-established maximum length in the tokenizer

configuration and the maximum length of tokens found in the text.

Once these parameters are prepared, the final datasets will be generated using the *generate_dataset()* function. We establish the **device** for training as the first GPU with CUDA. The optimizer utilized is the *AdamW optimizer* from the *torch.optim* module. With this optimizer, we can configure the model parameters, set the desired **learning rate**, and optionally adjust **epsilon**. The remaining decisions involve selecting the **batch size** and determining the **number of epochs** for training. As with other training and evaluation functions, the train function requires the model to train, the pre-processed training and validation subdatasets, the optimizer, the batch size, the number of epochs, and the device.

This initial method acts as a prelude to the main training function, *train_with_dataloader*. Its primary purpose is to create and configure the training and validation DataLoaders and pass them to the main method, along with all the provided parameters, except for the datasets, which are replaced by the DataLoaders. Before commencing training, certain values must be established for the epoch process:

1. Sending the model to the chosen device, in this case, as we use Google Colab platform, GPU 0 with CUDA.
2. Setting the scheduler for the learning rate using *get_linear_schedule_with_warmup()*. This scheduler creates a schedule with the specified learning rate, gradually decreasing it from the initial value to 0 after a warmup period. During the warmup, the learning rate increases linearly from 0 to the initial value defined in the optimizer.
3. Setting a random seed throughout the codebase to ensure reproducibility of the training process (Torch, Numpy, etc.).
4. Initializing the timer. For each epoch, we initiate an additional timer to measure the duration of each epoch.

We reset the total training loss to 0 and switch the model to **training mode**. Subsequently, for each step of the DataLoader, we extract the batch and divide it into three parts: the **input token IDs**, the **attention masks** for the model, and the **labels**. Before evaluating the model on the training batch, it is crucial to clear any previously calculated gradients using the *zero_grad()* function. This prevents the accumulation of gradients from previous *loss.backward()* calls. The next step is the **forward propagation**, where the input data passes through the neural network's neurons, and the model determines the expected output. Afterward, we accumulate the training loss across all batches to calculate the average loss at the end. Backward pass or **backpropagation** is then performed to compute the gradients, and the norm of the gradients is clipped to 1.0 to control exploding gradients. Finally, we update the optimizer parameters and scheduler steps and repeat the process until the training with all the epoch data is complete. Progress updates are displayed on the screen every 40 steps.

Once an epoch is finished, before evaluating the results, we calculate the average training loss by dividing the total accumulated loss during the epoch steps by the size of the training DataLoader. We stop the epoch timer since its purpose has been fulfilled, and switch the model to **validation mode**.

The validation mode closely resembles the training mode. Firstly, we initialize the loss accumulator to zero. For each step of the validation DataLoader, we retrieve its batch, which is then divided into input IDs, attention masks, and labels. However, unlike in the training mode, there is no need to reset the model gradients since we evaluate the model without calculating them. In this case, forward propagation is performed to obtain the loss, which is accumulated, and the logits, representing the non-normalized predictions generated by the model.

Both the logits and labels from the batch are excluded from backpropagation, and their data is transferred to the CPU as Numpy arrays instead of remaining as tensors on the GPU. These arrays serve as predictions and ground truth values for calculating the relevant metrics. Among the metrics mentioned in Section 2.5, we have chosen accuracy and F1-score. To facilitate the calculation of these metrics, we utilize the auxiliary function `score`, which also generates a classification report. This function is used to display the training statistics immediately after completing the training.

To visualize the training data and metrics, we plot a graph depicting the training loss and validation loss across epochs. Additionally, we create a Pandas DataFrame that presents a detailed overview of the training data and metrics. Figure 3.10 illustrates the aforementioned explanation regarding the training and validation of an epoch.

```
=====  
Epoch 4 / 4  
=====  
Training...  
Batch 40 of 586. Elapsed: 0:00:18.  
Batch 80 of 586. Elapsed: 0:00:37.  
Batch 120 of 586. Elapsed: 0:00:55.  
Batch 160 of 586. Elapsed: 0:01:13.  
Batch 200 of 586. Elapsed: 0:01:32.  
Batch 240 of 586. Elapsed: 0:01:50.  
Batch 280 of 586. Elapsed: 0:02:08.  
Batch 320 of 586. Elapsed: 0:02:27.  
Batch 360 of 586. Elapsed: 0:02:45.  
Batch 400 of 586. Elapsed: 0:03:04.  
Batch 440 of 586. Elapsed: 0:03:22.  
Batch 480 of 586. Elapsed: 0:03:40.  
Batch 520 of 586. Elapsed: 0:03:59.  
Batch 560 of 586. Elapsed: 0:04:17.  
  
Average training loss: 0.47  
Training epoch took: 0:04:29  
  
Running Validation...  
Accuracy: 0.77  
F1 macro: 0.77  
Precision: 0.77  
Recall: 0.78  
Validation Loss: 0.49  
Validation took: 0:00:05  
  
Training complete!  
Total training took 0:18:15 (h:mm:ss)
```

Figure 3.10: The 4th Epoch of Training and Validation for DeHateBERT model with EXIST 2023 dataset

3.4 Model building

The remaining structure of the Jupyter notebook consists of the construction of models and the actual experimental process of training and evaluating these models to achieve our initial objectives.

While we have previously provided a technical explanation in Chapter 2, it is important to emphasize what we mentioned earlier: the code framework of current work is an extension based on María Isabel’s research, extending beyond the scope of datasets to encompass the utilization of different models as well. In María Isabel’s study, BERT, RoBERTa, and XLNet models were employed to detect hate speech. However, in this work, we not only employ these models but also introduce an additional baseline model based on SVM (see Section 3.4.6). Furthermore, recognizing the importance of language specificity, we utilize specialized models for each language: HateBERT for English and DeHateBERT for Spanish, which will be elaborated upon in detail later in this section. Thus, our approach encompasses a broader range of models to cater to the nuances of different languages and extends beyond the scope of María Isabel’s work. With this in mind, we will now shift our focus towards providing an in-depth exploration of the individual models within each category, including their respective tokenizers, and any additional configurations they may require.

3.4.1 BERT

The transformers library offers a wide range of models, including their tokenizers and configurations, which are essential for our project. Specifically, we employ the *BertForSequenceClassification* [32] model derived from the pre-trained *bert-base-uncased* model. This BERT model is specifically designed for tasks involving sequence classification, such as sentiment analysis or text categorization.

To complement the model loading process, we also initialize the corresponding tokenizer, *BertTokenizer*, from the aforementioned model. The tokenizer plays a crucial role in converting raw text into tokens that can be comprehended and processed by the BERT model. Through tokenization, the text is divided into smaller meaningful units, such as words or subwords, and each unit is assigned a unique identifier.

Given our resource constraints, it is worth mentioning that BERT has two versions: BERT Base and BERT Large (Figure 2.10). To optimize our disk space usage, we have decided to focus solely on the *bert-base-uncased* model for both the EXIST 2021 and EXIST 2023 datasets. Moreover, additional configurations such as setting hyperparameters and adjusting learning rates will be addressed in subsequent stages of our project. These configurations play a vital role in fine-tuning the model and optimizing its performance, and will be carefully considered during the experimental process.

3.4.2 RoBERTa

When it comes to RoBERTa [33], the procedure for loading the model is comparable to that of BERT. We begin by initializing the configuration using the *RobertaConfig* class, which is

based on the *roberta-base* pre-trained model. Similarly, we employ the *RobertaTokenizer* as the corresponding tokenizer. Opting for the base version of RoBERTa is advantageous as it can handle all the necessary tasks without consuming excessive disk space or compromising performance.

3.4.3 XLNet

For XLNet model, the first thing is to initialize the pre-trained model *xlnet-base-cased* [34], and its tokenizer, *XLNetTokenizer*. Besides, it is worth mentioning that, Google Colab provides a limited amount of computational resources, including memory (RAM) and GPU availability. Training models with large batch sizes requires more memory to store and process the data. If the batch size is set too high and exceeds the available memory, it can lead to out-of-memory (OOM) errors during the training process [35]. Therefore, training larger models such as XLNet demands a substantial memory capacity to ensure effective training. As the batch size is augmented, the memory requirement escalates proportionally, posing difficulties in training the model with a high batch size within the constraints of Google Colab's limited memory resources. Thus, this limitation impedes the progress of certain experiments and prevents them from being successfully performed.

3.4.4 HateBERT

As highlighted in Section 2.4.7, our approach involves employing distinct models that are specifically trained on English and Spanish text to enhance the performance of sexist content detection. For the English corpus, we have opted for the *HateBERT* [25]² model, which has demonstrated superior performance in hate speech detection compared to BERT. Access to this model will be facilitated through the Hugging Face community platform. This model will be used to process the English text part for both EXIST 2021 and EXIST 2023 datasets.

3.4.5 DeHateBERT

Following our approach to optimizing training on Spanish text, we will incorporate the DeHateBERT model, more precisely, the *dehatebert-mono-spanish* [36]³ version. This particular model has been pre-trained using the Spanish datasets [27] [28], which enhances its ability to process Spanish text effectively. We will use this model exclusively on the Spanish text portion of the EXIST 2021 and EXIST 2023 datasets. By leveraging the language-specific expertise of the DeHateBERT model, we aim to improve the accuracy and performance of hate speech detection in Spanish-language content.

3.4.6 Baseline model: SVM based on TF-IDF vectorization

In order to establish a benchmark for our experiments, a baseline model will be developed for both English and Spanish languages. The baseline model serves as a reference point

²Available at Hugging Faces: <https://huggingface.co/gronlp/hatebert>

³Available at Hugging Faces: <https://huggingface.co/Hate-speech-CNERG/dehatebert-mono-spanish>

against which we can evaluate the performance of more advanced models and techniques. The baseline model is based on a Support Vector Machine (SVM) classifier, utilizing the TF-IDF vectorization approach.

To build the baseline model, we first created separate instances of the *TfidfVectorizer* for English and Spanish text. This vectorizer allows us to convert the textual data into numerical representations based on the TF-IDF weighting scheme. Next, we filtered the training and test data based on language, creating separate datasets for English and Spanish. For the English dataset, we fitted the *TfidfVectorizer* on the training data and transformed both the training and testing data. The transformed data was then used to train an SVM classifier with a linear kernel. The same process was repeated for the Spanish dataset. The trained SVM classifiers were used to make predictions on the test data, generating predicted labels for both English and Spanish texts. The performance of the baseline model was evaluated using accuracy and F1-score metrics.

Chapter 4

Results and analysis

This chapter discusses the methodology employed to do the evaluation of the models, that is said, the experiments of each model, building upon the implementation explained in the previous chapter. Furthermore, it includes the presentation of selected model results from each dataset, facilitating meaningful comparisons. Subsequently, a comprehensive analysis of these results will be carried out, shedding light on their implications and significance.

4.1 Experiments Setup

In this section, we will provide an overview of the configuration of the models and datasets used in the experiments. It is important to note that the implementation scenario, including the selection of programming language, IDE setup and external Jupyter Notebooks have been previously explained in Section 3.1. Therefore, the following part will be focused on reproducing the experiments described in the subsequent sections. To accomplish this, we need to understand two key concepts: the distribution of the datasets and the configurable parameters of the models.

Datasets

As mentioned previously, each dataset is divided into three sets: train, dev, and test. These partitions serve specific purposes such as training, validation, and testing. In the case of EXIST 2023, the dataset is already pre-divided into these three parts for the competition, and the distribution percentages for each part have been detailed in Section 3.3.2.

However, for EXIST 2021, a portion of the training data is allocated to serve as the development data, as there is no separate dev subset available for this year's competition. So it is important to emphasize the percentage distribution of each subdataset in this case. Initially, there is a split between the train-dev and test set using the *train_test_split* function from *sklearn.model_selection* module. The test set represents 38% of the original dataset,

while the combined training and validation set account for 62%. Within this 62%, the validation set is set to 20%. Therefore, the testing set consists of 20% of the data, the training set comprises approximately 49.6%, and the validation set comprises 12.4%.

Hyperparameters

Regarding the models' parameters, there are four adjustable parameters that can be modified to specific needs. These parameters include the learning rate, epsilon, batch size, and number of epochs. Throughout the experiments, a consistent configuration was applied to the epsilon and number of epochs across all notebooks, setting them to 4 epochs and $\epsilon = 1.10^{-8}$, respectively. However, it is important to note that the learning rate and batch size can vary considerably depending on the dataset and model under consideration. To identify the appropriate values for these parameters, please consult Table 4.1 below.

Parameters of hyperparameters used per Model and Dataset in the experiments										
Models	BERT		RoBERTa		XLNet		HateBERT		DeHateBERT	
Datasets	Learning Rates	Batch Sizes	Learning Rates	Batch Sizes	Learning Rates	Batch Sizes	Learning Rates	Batch Sizes	Learning Rates	Batch Sizes
EXIST2021	2.10^{-4}		5.10^{-4}		1.10^{-4}		2.10^{-4}		1.10^{-5}	
	5.10^{-4}	8	1.10^{-5}	4	2.10^{-5}	4	5.10^{-4}	8	2.10^{-5}	4
	3.10^{-5}	16	3.10^{-5}	8	3.10^{-5}	8	3.10^{-5}	16	3.10^{-5}	8
	5.10^{-5}	32	5.10^{-5}	16	5.10^{-5}	16	5.10^{-5}	32	5.10^{-5}	16
EXIST2023	1.10^{-5}		1.10^{-5}		1.10^{-4}		1.10^{-5}		1.10^{-5}	
	2.10^{-5}	8	2.10^{-5}	4	2.10^{-5}	4	2.10^{-5}	8	2.10^{-5}	4
	3.10^{-5}	16	3.10^{-5}	8	3.10^{-5}	8	3.10^{-5}	16	3.10^{-5}	8
	5.10^{-5}	32	5.10^{-5}	16	5.10^{-5}	Not Available	5.10^{-5}	32	5.10^{-5}	16

Table 4.1: Summary of hyperparameters used per Model and Dataset in the Experiments

4.2 Results of training and evaluation

In this section, we will evaluate the overall performance of each task using the aforementioned models and compare their results as model evaluation. We will present the key metrics that provide insights into the performance of each dataset, highlighting challenges encountered during the study and the factors contributing to those challenges.

4.2.1 EXIST 2021 Dataset

In the EXIST task, the datasets for each year consist of a combination of English and Spanish information. In order to evaluate the model's performance in each language, we will train separate models for English and Spanish using the respective text. This approach allows us to gather data on the model's effectiveness in both languages.

The classification of sexist or non-sexist messages in the text field is the first task to be completed for this dataset. Focusing on the dataset reveals that it is not the most unbalanced one to be found. In reality, as shown in the classification report in Figure 3.3.(a), it shows both percentages of sexist and non-sexist messages' presence are quite balanced, with 50.24% of Spanish text is sexist and 49.50% of English is sexist.

The training and evaluation process demonstrated successful outcomes for all five models utilized in this project. As evidence of the progress made, Figure 4.1 presents an example

of the evaluation conducted on the training dataset using the HateBERT model and English text. The model was trained with a learning rate of 5.10^{-5} and a batch size of 16. While the overall process yielded positive results, it also highlighted a significant concern: the presence of **overfitting**, one of the most common problems in model training.

```

Classification Report
=====
              precision    recall  f1-score   support

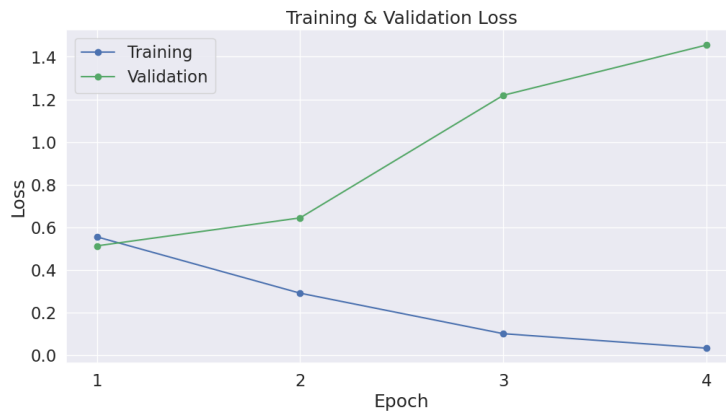
     0           1.00      1.00      1.00     2881
     1           1.00      1.00      1.00     2741

 accuracy              1.00      1.00      1.00     5622
 macro avg              1.00      1.00      1.00     5622
 weighted avg           1.00      1.00      1.00     5622

HateBERT Dev EXIST
F1 macro: 0.7569837960123744
Accuracy: 0.7571955719557195

```

(a) Classification Report with HateBERT, English



(b) Loss Graph through epochs

Figure 4.1: EXIST 2021 dataset status after training and validating

Overfitting occurs when a machine learning model becomes too closely tailored to the training data, resulting in reduced performance on new, unseen data. This phenomenon poses a challenge in the context of research, as it impacts the effectiveness of the trained models. During the evaluation process, clear indications of overfitting emerge, as the models exhibit an exceptional understanding of the training data. Figure 4.1.(a) presents the classification report, revealing precision and recall scores of 1, which further highlights the occurrence of overfitting. Additionally, Figure 4.1.(b) shows the training and validation losses, where the training loss steadily decreases with each epoch, while the validation one increases. These patterns are clear indicators of the presence of overfitting, further emphasizing the need to address this issue to ensure optimal model performance.

Taking a look at Figure 4.2, which can also be found with the rest of tables in Appendix, that relatively higher Learning Rates, such as 2.10^{-5} and 3.10^{-5} , are more prone to this particular issue. The quick way to detect the lack of learning progression, as can be seen in the red squared area in Figure 4.2, is by comparing the evaluation metrics of the training: Accuracy (Acc), F1-Score and loss values.

HateBERT with EXIST2021 English(4 epochs per per training) without translation													
Batch/LR	8				16				32				
	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
2.10 ⁻⁴	1	0.51	0.68	0.70	0.73	0.51	0.68	0.66	0.69	0.66	0.65	0.64	0.71
	2	0.51	0.68	0.69	0.69	0.65	0.64	0.59	0.68	0.70	0.70	0.52	0.65
	3	0.49	0.65	0.70	0.69	0.68	0.68	0.57	0.68	0.71	0.71	0.48	0.68
	4	0.51	0.68	0.70	0.69	0.67	0.67	0.56	0.67	0.69	0.69	0.48	0.64
5.10 ⁻⁴	1	0.51	0.68	0.73	0.69	0.51	0.68	0.74	0.76	0.51	0.68	0.77	0.78
	2	0.51	0.68	0.71	0.69	0.51	0.68	0.70	0.71	0.51	0.68	0.71	0.70
	3	0.51	0.68	0.71	0.69	0.49	0.66	0.71	0.70	0.49	0.65	0.71	0.70
	4	0.51	0.68	0.70	0.69	0.51	0.68	0.70	0.69	0.51	0.68	0.70	0.70
3.10 ⁻⁵	1	0.75	0.74	0.01	3.18	0.76	0.76	0.03	1.68	0.77	0.77	0.02	2.31
	2	0.75	0.75	0.03	2.63	0.76	0.76	0.00	1.97	0.75	0.75	0.01	2.24
	3	0.75	0.75	0.01	2.76	0.77	0.77	0.00	2.07	0.77	0.77	0.02	2.03
	4	0.76	0.76	0.00	2.79	0.78	0.78	0.00	2.08	0.77	0.77	0.00	2.03
5.10 ⁻⁵	1	0.73	0.73	0.04	2.38	0.77	0.76	0.55	0.50	0.75	0.75	0.02	2.07
	2	0.75	0.75	0.04	2.57	0.76	0.76	0.27	0.62	0.76	0.76	0.02	2.16
	3	0.75	0.75	0.01	2.54	0.78	0.78	0.09	1.08	0.77	0.77	0.00	2.15
	4	0.75	0.75	0.01	2.64	0.78	0.78	0.03	1.21	0.75	0.75	0.00	2.19

Figure 4.2: HateBERT Overfitting during EXIST2021 English text training, red squared area (without translation)

It is evident from the results that the Accuracy and F1-score have no significant changes across the four training epochs, which is highly indicative. Furthermore, the loss values, both for training and validation, demonstrate remarkable stability. Typically, during the learning process, the training loss rapidly converges towards zero, while the validation loss decreases gradually. However, in this case, both losses remain consistent, with a slight increase observed. This consistency indicates that the model has reached a plateau and is not experiencing substantial improvements or deteriorations in performance.

4.2.2 Data augmentation

One of the most used techniques in NLP field to enhance the available data and mitigate the problem of limited training samples, as well as dealing with overfitting, is **data augmentation**. These techniques were employed in code implementation in order to compare with the original training. The *Googletrans* module is a free, open-source, and widely-used Python library that provides a convenient interface for accessing the Google Translate API.

Data augmentation through translation involves the conversion of text messages from one language to another. In our case, we focused on translating text from English to Spanish and vice versa. Leveraging the capabilities of the *Googletrans* library, we developed an auxiliary function called *translate_text*, which streamlined the translation process. This function accepts a text message as input, automatically detects its language, and performs the translation only if the target language differs from the detected language. The Google Translate API is utilized to carry out the translation, and the resulting translated text is

saved as a TSV file at the specified file path. To optimize efficiency, the function checks if the translation has already been performed and saved at the specified path, and if so, skips the translation process to save time and computational resources.

By incorporating data translation as part of our augmentation strategy, we aimed to increase the volume of available training data. The augmented dataset allows us to train single-language models with a larger and more diverse set of examples, enabling them to capture a wider range of language patterns and nuances. This approach is particularly valuable in situations where obtaining a substantial amount of labeled data in a single language is challenging or costly.

To visually demonstrate the impact of data translation on the dataset, we have included Figure 4.3, which illustrates the change in dataset size before and after applying the data translation technique to the training dataset.

```
train.shape before translation: (6977, 7)
```

	test_case	id	source	language	text
0	EXIST2021	1	twitter	en	She calls herself "anti-feminazi" how about sh...
1	EXIST2021	2	twitter	en	Now, back to these women, the brave and the be...
2	EXIST2021	3	twitter	en	@CurvyBandida @Xalynne_B Wow, your skirt is ve...
3	EXIST2021	4	twitter	en	@AurelieGuiboud Incredible! Beautiful!But I l...
4	EXIST2021	5	twitter	en	i find it extremely hard to believe that kelly...
...
6972	EXIST2021	6973	twitter	es	Estamos igual sin pareja, pero puedes besar a ...
6973	EXIST2021	6974	twitter	es	2020 hijo de re mil putas
6974	EXIST2021	6975	twitter	es	SEGURAMENTE ESTA CHICA NO COBRA EL DINERO QUE ...
6975	EXIST2021	6976	twitter	es	@safetyaitana mi madre dice q va fea y i agree
6976	EXIST2021	6977	twitter	es	¿En vuestras casas también tenéis esa tradició...

6977 rows x 7 columns

(a) Size of Training Dataset before applied data translation

```
train.shape after translation: (13954, 7)
```

	test_case	id	source	language	text
0	EXIST2021	1	twitter	en	She calls herself "anti-feminazi" how about sh...
1	EXIST2021	2	twitter	en	Now, back to these women, the brave and the be...
2	EXIST2021	3	twitter	en	@CurvyBandida @Xalynne_B Wow, your skirt is ve...
3	EXIST2021	4	twitter	en	@AurelieGuiboud Incredible! Beautiful!But I l...
4	EXIST2021	5	twitter	en	i find it extremely hard to believe that kelly...
...
13949	EXIST2021	6973	twitter	en	We are the same without a partner, but you can...
13950	EXIST2021	6974	twitter	en	2020 Son of Re Mil Putas
13951	EXIST2021	6975	twitter	en	Surely this girl does not charge the money I w...
13952	EXIST2021	6976	twitter	en	@safetyaitana my mother says that he goes ugly...
13953	EXIST2021	6977	twitter	en	In your houses you also have that shit traditi...

13954 rows x 7 columns

(b) Size of Training Dataset after applied data translation

Figure 4.3: Application of data translation to the Training Dataset of EXIST2021

Before applying data translation, the original dataset had a shape of 6,977 instances. After applying the technique, the dataset size increased to 13,954 instances. This indicates a significant augmentation, effectively doubling the size of the training data. The percentage

of augmentation can be calculated as $(\text{New Size} - \text{Original Size}) / \text{Original Size} * 100$, which in this case would be $((13,954 - 6,977) / 6,977) * 100 = 99.96\%$. This demonstrates the substantial impact of data translation in expanding the dataset and providing more diverse training examples for our models.

However, it is important to note that while data translation serves as a valuable tool for data augmentation, it also comes with certain limitations. Translating text from one language to another can sometimes result in a **loss of meaning** or **subtle changes** in the original message. These variations may impact the performance of the models, particularly when dealing with tasks that require precise semantic understanding or cultural context.

4.2.3 Best-performing model and results for EXIST 2021

Despite the potential challenges associated with translation, we decided to explore this strategy in order to compare the results. Therefore, the next step is to conduct exhaustive experiments using all the parameters specified in Table 4.1. These experiments will be performed on both the training set with translation and the training set without translation.

Upon analyzing the best results achieved by each model, we observed from Figure 4.4 that most models exhibited only slight variations in performance for the binary classification task of sexist content detection. However, it is interesting to note that the baseline model, as mentioned earlier, showed distinct performance metrics. In terms of Accuracy, the baseline model achieved 0.71 for English and 0.70 for Spanish. Similarly, its F1 Score for English was 0.71, while for Spanish it was 0.70.

To provide a comprehensive overview of the results, we have categorized them by language and whether or not translation was applied to the training dataset. For detailed results, please refer to the Appendix, where tables listing the results per language and dataset are provided. These tables will provide a more comprehensive analysis of the performance of each model, taking into account the presence or absence of translation during training.

After analyzing the best results achieved by each model, we observed from Figure 4.4 that most models exhibited only slight variations in performance for the binary classification task with English text. In contrast, we observed slightly greater fluctuations in performance when dealing with Spanish. Notably, the baseline model, as previously mentioned, exhibited distinct performance metrics. Specifically, the baseline model achieved an accuracy score of 0.71 for English and 0.70 for Spanish, with corresponding F1 scores of 0.71 for English and 0.70 for Spanish.

This outcome is comprehensible, as it aligns with the prevailing trend in the field of NLP, wherein models are predominantly trained on English as the preferred language. Consequently, the availability of highly precise models for less commonly spoken languages remains limited.

To offer a comprehensive overview of the results, we have categorized them based on lan-

guage and the application of translation to the training dataset. For a detailed breakdown of the findings, we refer to the Appendix, which includes tables enumerating the results for each language and dataset. These tables will provide a thorough analysis of each model’s performance, taking into account the presence or absence of translation during the training process.

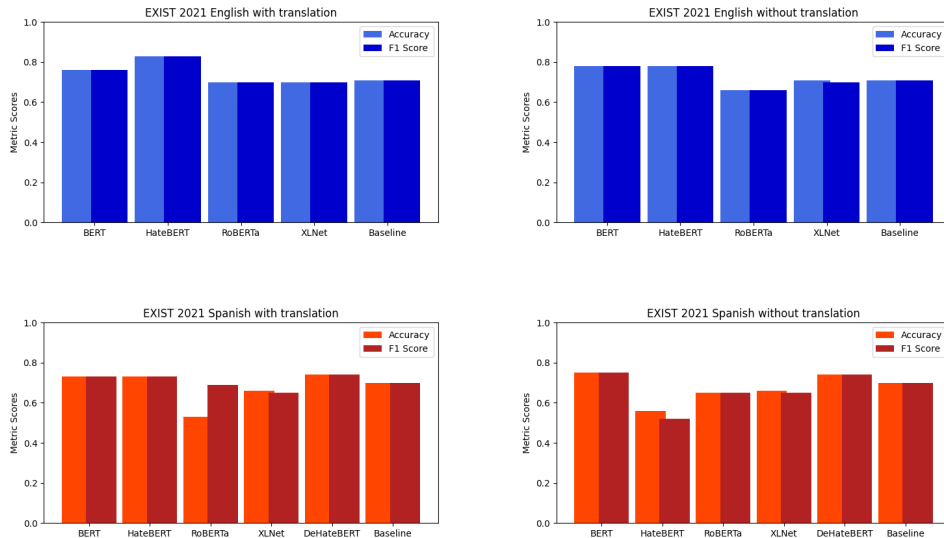


Figure 4.4: Best results per model comparison of EXIST 2021 Dataset

To analyze and summarize the results shown above, we refer to the Table 4.2. The best results are bolded in each language, with or without translation.

Table 4.2: Results obtained in the EXIST 2021 Identification Task

Model/Dataset	Accuracy	F1-Score
English with Translation		
HateBERT (Best)	0.83	0.83
RoBERTa and XLNet (Worst)	0.70	0.70
English without Translation		
BERT and HateBERT (Best)	0.78	0.78
RoBERTa (Worst)	0.66	0.66
Spanish with Translation		
DeHateBERT (Best)	0.74	0.74
RoBERTa (Worst)	0.53	0.69
Spanish without Translation		
BERT (Best)	0.75	0.75
HateBERT (Worst)	0.56	0.52

In all cases, the **HateBERT** model achieved the highest performance for the EXIST 2021 identification task in English, attaining an Accuracy and F1-Score of 0.83 each, using the

dataset extended with Translation. BERT also performs well in English without translation, with accuracy and F1-score of 0.78. Conversely, in the case of Spanish text, the BERT model demonstrated the best performance without translation, achieving an Accuracy and F1-Score of 0.75 each. DeHateBERT achieves the highest accuracy of 0.74 and F1-score of 0.74 in Spanish with translation. Overall, HateBERT and BERT emerge as the top-performing models, with HateBERT being particularly effective in English, while BERT demonstrates strong performance in both English and Spanish.

However, it is important to mention that even though we made efforts to enhance the training process by increasing the size of the training data and implementing extensive augmentation techniques, these measures do not significantly improve the model's performance. While the best-performing model for English text is indeed one trained with translation (BERT), it does not guarantee consistent outcomes in all cases. This highlights the fact that certain models have the ability to effectively grasp the language-specific patterns and nuances without relying on translation, as exemplified by the case of Spanish. In fact, in this scenario, the worst performing model's accuracy even experienced a greater decrease when translation was employed.

By analyzing the decrease in performance of models with adding translation text, some important factors can be pointed out. Firstly, the translation quality, particularly when using a free API like Googletrans, is not guaranteed. Machine translation can introduce errors, inaccuracies, and changes in meaning that directly impact the training data. Consequently, these distortions undermine the model's ability to accurately identify sexist content and can even result in mislabeled examples. To illustrate this, a comparison is presented below between two tweets, before and after using Googletrans translation service:

Original text: *"Ella se hace llamar antifeminazi ¿Qué tal si callan en su vil comentario sobre un ciudadano responsable de ancianos tu sach muuch ghani baawri-bewdi hai bey <https://t.co/zmætdwsy5d>"*

Translated text: *"She calls herself anti-feminazi how about shut the fucking up on your vile commentary on an elderly responsible citizen tu sach muuch ghani baawri-bewdi hai bey <https://t.co/ZMæTDwsY5D>"*

Original text: *"Realmente solo quiero ser rico pero no de una esposa trofeo rica, rica en la mía"*

Translated text: *"I really just want to be rich but not trophy wife rich, rich with my own"*

Another factor to consider is the influence of cultural and linguistic nuances. Sexist content heavily relies on subtle nuances that vary across languages. Unfortunately, machine translation may struggle to capture these nuances accurately, leading to a loss of context and inhibiting the model's recognition of more nuanced forms of sexism.

Moreover, machine translation can inadvertently introduce additional noise or imbalances into the translated training dataset. This, in turn, can give rise to grammatical inconsistencies and syntactic errors. The presence of such noise can confuse the model and impede its

capacity to effectively learn the underlying patterns and characteristics of sexist content.

4.2.4 EXIST 2023 Dataset

Moving on to the dataset of this year's competition, the data consists of both English and Spanish texts. Interestingly, their structures exhibit considerable similarity, except that the datasets for 2023 are provided in JSON format (for a detailed description, see Section 3.3.2).

Unlike the EXIST 2021 task, where Machine Translation was used for data augmentation, a different approach was adopted this time. Instead, the data was augmented by incorporating the complete dataset from the previous task, which includes the labeled training, development, and test sets, into this year's training dataset. This integration of datasets aims to enhance the training data size and diversify the content for improved model performance.

The resulting changes in the size of each dataset after implementing this approach are depicted in Figure 4.6. Before the integration, the dataset had a shape of 6,920 records. After incorporating the previous task's dataset, the size of the training data expanded to 13,954. This corresponds to an augmentation of approximately $((13,954 - 6,920) / 6,920) * 100 = 101.01\%$. This significant increase in the amount of training data contributes to a richer and more diverse dataset, thereby enhancing the effectiveness of the models trained on it.

The first challenge we faced when conducting experiments with the EXIST 2023 dataset was the encoding of the task1 labels. To facilitate the training process, it was necessary to assign numerical values to the *labels_task1* column. In the example from 2021, this encoding step was relatively straightforward, as we simply transformed "sexist" tags to 1 and "non-sexist" to 0. However, this year, the organizing committee introduced a more detailed classification system, which involved multiple opinions from each rater, along with age range and gender information, all stored as lists (refer to Section 3.3.2 for more details). Consequently, numerically encoding this year's labels required additional steps. Specifically, we followed these two steps:

1. **Step 1:** We determined the majority opinion for each instance. If the number of "YES" classification exceeded the number of "NO", we assigned the code 1; otherwise, it was assigned as 0. However, since there were six annotators in total, situations where there was a tie, with three "YES" and three "NO" responses, could occur. In such cases, we temporarily assigned the special code -1. Consequently, we obtained 3367 instances labeled as absolute "NO," 2697 instances labeled as absolute "YES," and 856 instances labeled as -1.
2. **Step 2:** We proceeded to process the data corresponding to the -1 label, which couldn't be resolved in the previous step. Considering that the nature of this task involves detecting gender discrimination, we decided to give more weight to the opinions of **female annotators** while considering their viewpoints. Since there were three male annotators and three female annotators in total, after this step, we ob-

```
original train.shape: (6920, 11)
```

id_EXIST	lang	tweet	number_annotators	annotators	gender_annotators	age_annotators	labels_task1
100001	100001	es @TheChiffis Ignora al otro, es un capullo.El p...	6	[Annotator_1, Annotator_2, Annotator_3, Annota...	[F, F, F, M, M, M]	[18-22, 23-45, 46+, 46+, 23-45, 18-22]	[YES, YES, NO, YES, YES, YES]
100002	100002	es @ultimonomada Si comicsgate se parece en algo...	6	[Annotator_7, Annotator_8, Annotator_9, Annota...	[F, F, F, M, M, M]	[18-22, 23-45, 46+, 46+, 23-45, 18-22]	[NO, NO, NO, NO, YES, NO]
100003	100003	es @Steven2897 Lee sobre Gamergate, y como eso ha...	6	[Annotator_7, Annotator_8, Annotator_9, Annota...	[F, F, F, M, M, M]	[18-22, 23-45, 46+, 46+, 23-45, 18-22]	[NO, NO, NO, NO, NO, NO]
100004	100004	es @Lunariita7 Un retraso social bastante lamenta...	6	[Annotator_13, Annotator_14, Annotator_15, Ann...	[F, F, F, M, M, M]	[18-22, 23-45, 46+, 46+, 23-45, 18-22]	[NO, NO, YES, NO, YES, YES]
100005	100005	es @novadragon21 @icp4ck @TvDannyZ Entonces como...	6	[Annotator_19, Annotator_20, Annotator_21, Ann...	[F, F, F, M, M, M]	[18-22, 23-45, 46+, 46+, 23-45, 18-22]	[YES, NO, YES, NO, YES, YES]

(a) Size of Training Dataset before adding 2021 data

```
train.shape after adding 2021 data: (18265, 5)
```

id_EXIST	lang	tweet	label1_encoded
0	6563	es @Lunamj4216 @RaeiouESC @Almu_DS Y te estás co...	0
1	5862	es Alejandro Vargas discriminó en razón de género...	1
2	1011	en @ArvindKejriwal @msisodia @dmeastdelhi Pending...	0
3	3708	es 2020 será histórico, Golpe a la democracia en ...	0
4	932	en @sonelinus I call literally every single incon...	0
...
18260	203256	en idk why y' all bitches think having half your a...	1
18261	203257	en This has been a part of an experiment with @Wo...	1
18262	203258	en "Take me already" "Not yet. You gotta be ready...	1
18263	203259	en @clintneedcoffee why do you look like a whore?...	1
18264	203260	en ik when mandy says "you look like a whore" i l...	1

18265 rows x 4 columns

(b) Size of Training Dataset after adding 2021 data

Figure 4.5: Comparison of the Training dataset size

tained the definitive numerical encoding for all instances, ensuring there were no more tied opinions. Finally, we had 3792 instances labeled as absolute non-sexist messages and 3128 instances labeled as “YES” representing sexist messages.

We repeated this encoding process for both the training and development sets, and stored the results in the new column *label1_encoded*. After that, our experiments proceeded smoothly. Figure 4.7 shows an example of the evaluation conducted on the training dataset using the DeHateBERT model and Spanish text. The model was trained with a learning rate of $1.10 \cdot 10^{-5}$ and a batch size of 16.

```

Classification Report
=====

```

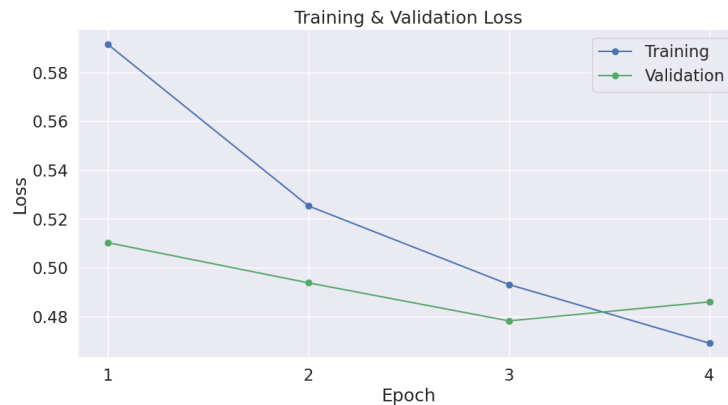
	precision	recall	f1-score	support
0	0.85	0.76	0.80	4710
1	0.78	0.87	0.82	4651
accuracy			0.81	9361
macro avg	0.82	0.81	0.81	9361
weighted avg	0.82	0.81	0.81	9361

```

DeHateBERT Dev EXIST
F1 macro: 0.7867040794301748
Accuracy: 0.7868852459016393

```

(a) Classification Report with DeHateBERT, Spanish



(b) Loss Graph through epochs

Figure 4.6: EXIST 2023 dataset Status after training and validating

4.2.5 Best-performing model and results for EXIST 2023

We evaluated each model on both extended English and Spanish datasets, with and without translation, as what we did in case of EXIST 2021. Figure 4.8 summarizes the best-achieved accuracy and F1-score for each model and dataset. In this year, the baseline model exhibits an Accuracy on English of 0.75 and an F1 Score on English of 0.75, while

its Accuracy on Spanish is 0.72 and its F1-Score, the same value.

For the English with Translation dataset, both **BERT** and **HateBERT** are the best-performing model, reaching an accuracy and F1-score of 0.83. On the other hand, **RoBERTa** had the lowest performance among the English models, with an accuracy of 0.69 and an F1-score of 0.70. When came to without translation, **HateBERT** emerged as the only best model, with accuracy and F1-score of 0.81. **RoBERTa** exhibited the lowest performance among the English models, attaining an accuracy of 0.69 and an F1-score of 0.70. Similarly, **XLNet** faced a similar situation, with both accuracy and F1-score of 0.70. These results demonstrate that both RoBERTa and XLNet fall below the baseline model in terms of performance.

Here arises the question: why do RoBERTa and XLNet underperform the baseline model? To address this issue, several factors could be considered. Firstly, it is essential to evaluate the suitability of the dataset for RoBERTa and XLNet. Not all models perform equally well on every dataset, and both RoBERTa and XLNet may not be optimally suited to the specific characteristics of the EXIST2023 dataset. The training sets used for these models might not have adequately captured the nuances and complexities present in the evaluation set, leading to a mismatch between the training and evaluation data. Additionally, the process of feature engineering should be taken into consideration. The quality of the features extracted from the dataset and fed into RoBERTa and XLNet can significantly impact their ability to capture relevant information. If the features did not adequately represent the data, it could have hindered the performance of both models. Moreover, the complexity of RoBERTa and XLNet should not be overlooked. Although both models are sophisticated with a large number of parameters, superior performance is not guaranteed in all cases, especially when the dataset is small or the task is relatively simple. In such instances, a simpler baseline model may outperform more complex models like RoBERTa or XLNet.

Moving on to the Spanish with Translation dataset, a noteworthy aspect is that the majority of models exhibit a certain level of stability both with and without translation, except for HateBERT. Interestingly, when no translated data is added, HateBERT experiences a significant drop in accuracy. **DeHateBERT** showed the highest performance, achieving accuracy and F1-score of 0.79. In contrast, **XLNet** and **RoBERTa** had the lowest performance, with an accuracy of 0.73 and an F1-score of 0.73.

Finally, for the Spanish dataset without translation, **DeHateBERT** once again performed the best, with accuracy and F1-score of 0.78. On the other hand, **HateBERT** had the lowest performance among the Spanish models, as we mentioned before, with an accuracy of 0.53 and an F1-score of 0.67, also falling below the baseline model. This significant drop could be attributed to the nature of the training data as the HateBERT model had been specifically retrained on English corpus to focus on detecting abusive language in English [25]. The retraining process may have fine-tuned its' parameters and features to align with the characteristics of English hate speech, making it more effective in that specific domain. When no translated data is included, HateBERT lacks exposure to hate speech examples in the original language. As a result, it may struggle to effectively identify and understand the contextual nuances in the original language.

A summary is presented in Table 4.3, among all the models evaluated, the best model

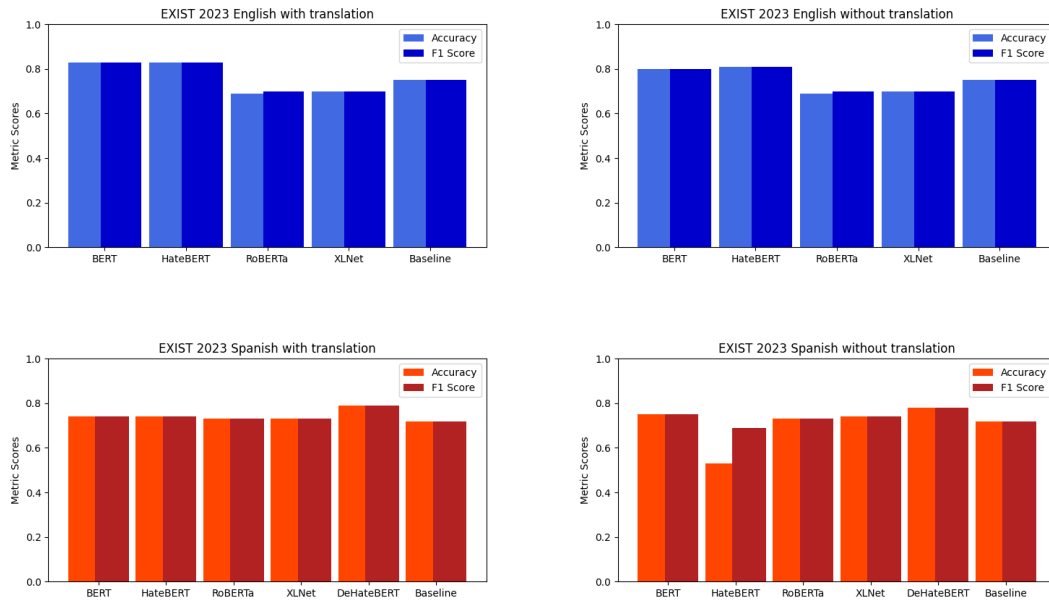


Figure 4.7: Best results per model comparison of EXIST 2023 Dataset

for the English dataset was both **BERT** and **HateBERT**, while **DeHateBERT** emerged as the top performer for the Spanish dataset. **RoBERTa** consistently showed the lowest performance across both languages.

4.2.6 EXIST 2023 Task 1 submission

In fact, as of May this year, EXIST 2023 remained as an active competition, and fortunately, we managed to align our research and experimentation timeline of the **Task 1: Sexism Identification**¹ with the submission deadline for the EXIST 2023 task, so as to submit results for this specific binary classification task. The systems must decide whether a given tweet contains sexist expressions or behaviors (i.e., it is sexist itself, describes a sexist situation, or criticizes a sexist behavior) and classify it according to two categories: YES and NO. To achieve that, we employed the top-performing models during the stages of experiments in task 2021 on this year's dataset and obtained the best and worst-performing models for this year's data, separately, as depicted in Table 4.3. Specifically, for English, we used BERT and HateBERT, while for Spanish, we employed DeHateBERT. We organized the outputs generated by these models and conscientiously followed the submission guidelines provided in the competition. It is important to note that our primary objective in participating was not solely to win a reward or emerge as the top team among all participants. Rather, our intention was to assess the performance of our models on this year's test dataset and evaluate their effectiveness in tackling the given task.

¹EXIST 2023 competition: <http://nlp.uned.es/exist2023/>

Table 4.3: Results obtained in the EXIST 2023 Identification Task

Model/Dataset	Accuracy	F1-Score
English with Translation		
BERT and HateBERT (Best)	0.83	0.83
RoBERTa (Worst)	0.69	0.70
English without Translation		
HateBERT (Best)	0.81	0.81
RoBERTa (Worst)	0.69	0.70
Spanish with Translation		
DeHateBERT (Best)	0.79	0.79
RoBERTa and XLNet (Worst)	0.73	0.73
Spanish without Translation		
DeHateBERT (Best)	0.78	0.78
HateBERT (Worst)	0.53	0.69

Hard labels VS Soft labels

According to the guidelines provided, participants were instructed to format their runs for Task 1 in JSON format, with each tweet represented as a JSON object containing specific attributes. These attributes included *id_exist*, *hard_label*, and *soft_label*. The *hard_label* attribute allowed for values of either “YES” or “NO”, following the conventional practice in machine learning. On the other hand, the *soft_label* attribute required participants to assign a probability to each of the two possible labels (“YES” and “NO”), ensuring that the sum of the probabilities equaled 1.0. Participants had the flexibility to provide hard labels, soft labels, or a combination of both.

It is worth noting that, up until now, the outcomes we have obtained have consistently comprised hard labels. In other words, we have been exclusively using hard labels for evaluation purposes. As an initial approach, we organized two submission runs using only hard labels. The first run employed BERT for English and DeHateBERT for Spanish, while the second run utilized HateBERT for English and DeHateBERT for Spanish, as we mentioned earlier, both BERT and HateBERT archived best results for English.

Therefore, we decided to make changes to our implementation in order to obtain results in the form of soft labels. This modification aimed to assess whether the introduction of soft labels would have any impact on the evaluation metrics. Specifically, we made slight modifications to the evaluation code, particularly the *evaluate_with_dataloader* method. Within the prediction loop, after calculating the logits using the model, we incorporated the *torch.softmax* function to obtain the class probabilities. These probabilities were then returned for each predicted class.

Figure 4.9 provides a clear comparison between the two submission formats for EXIST 2023 competition. One example shows the hard label format, while the other demonstrates the use of soft labels.

Unfortunately, despite transitioning from hard labels to soft labels as the output method,

<pre>"500001": { "hard_label": "NO" }, "500002": { "hard_label": "NO" }, "500003": { "hard_label": "NO" },</pre> <p>(a) Hard labeled test set</p>	<pre>"500001": { "soft_label": { "YES": "2.09%", "NO": "97.91%" } }, "500002": { "soft_label": { "YES": "25.25%", "NO": "74.75%" } },</pre> <p>(b) Soft labeled test set</p>
---	--

Figure 4.8: Different submission formats comparison

the best results achieved by the model did not surpass the performance observed during the period of hard label evaluation. In other words, the highest Accuracy and F1-Score achieved for the English part remained at 0.83, while the Accuracy for Spanish was 0.77, accompanied by the same F1-Score.

While this transition from hard labels to soft labels did not result in improved overall performance, exploring the use of soft labels is still worthwhile. Soft labels provide a more nuanced representation of prediction probabilities, capturing the model’s confidence in its predictions. They allow for a more detailed understanding of the content’s offensiveness spectrum.

After careful consideration, we made the decision to stick with the use of hard labels, as we had implemented initially. The submission of our two runs resulted in rankings of 32 and 38, out of a total of 70 teams, in the hard label leaderboard [37]. The first run achieved an F1-Score of 0.76, while the second run demonstrated the same metric of 0.75. The rankings and scores presented above provide insights into the effectiveness of our approach in accurately identifying and classifying sexist content. While we may not have reached the top positions on the leaderboard, our results demonstrate a solid performance.

4.2.7 Comparative among all models

Once all five models and their characteristics have been seen, several key observations emerge.

Firstly, when considering the best performing models based on the evaluation results, **BERT and HateBERT** stand out as the optimal choices for the English dataset, while **De-HateBERT** proves to be the recommended model for the Spanish dataset, considering both the EXIST 2021 and EXIST 2023 datasets. Secondly, it becomes evident that achieving consistent progress requires a greater number of epochs and higher batch sizes. However, considering our study period, four epochs can already provide valuable insights. In addition, when examining the optimal results achieved for each model and dataset, an interesting pattern emerges. It becomes apparent that this occurrence is more pronounced

when using lower learning rates, such as $3e-05$ or $5e-05$, in comparison to higher learning rates like $1e-04$ or $2e-04$. This observation is supported by our comprehensive analysis of various parameter settings, which can be found in the tables provided in the Appendix. Additionally, overfitting remains as a significant challenge to tackle with. While data augmentation serves as a viable approach in addressing this issue, it is important to note that there are numerous alternative methods beyond machine translation, such as leveraging synthetic data generation techniques, exploring transfer learning from related tasks, or employing ensemble models to combine diverse sources of information.

Chapter 5

Conclusions and future work

Having reached this point, a brief summary of what has been learned during this Bachelor's thesis will be made. Furthermore, we will delve into the future prospects and objectives that lie ahead.

What has been learned so far

Reflecting upon the journey, this research conducted in this project spanned a five-month period, entailing dedication and countless hours invested in comprehending the fundamental principles of Natural Language Processing (NLP) up to the implementation of state-of-the-art models, and further to the experimental process and results' analysis.

The extensive exploration of numerous papers not only restored a fluent understanding of technical information, but also introduced a neglected field of computer science that I did not delve into in four years of college. As the interest for this work grew, the implementation phase started, initially analyzing the two important datasets: EXIST 2021 and EXIST 2023. From being completely puzzled by the unfamiliar dataset at the beginning, to gradually becoming familiar through meticulous statistical analysis, and to implementing actual code for models that initially seemed complex and difficult to comprehend based on the knowledge gained from papers, our models were able to successfully run on the dataset! Consequently, we obtained a large amount of experimental data, which allowed us to gain insights into the characteristics of each model. However, along with these advancements, challenges emerged during the training process. We also explored possible solutions to these issues, although due to time constraints, we regrettably could not delve deeply into them. Nonetheless, the process of problem-solving was highly rewarding.

Finally, with lots of satisfaction and conviction, we affirm that all the objectives outlined in Section 1.3 of the Introduction have been successfully achieved. The most valuable lesson learned from this Bachelor's thesis is the significant impact even minor details, such as learning rates and batch sizes, can have on the overall results. Moreover, the captivating field of natural language processing harbors endless intrigue, beckoning further exploration and immersion. This has also sparked my interest in pursuing a related master's

program in the following semester.

Besides, as mentioned in the initial sections of this project (see Section 1.1), being a woman who has personally experienced gender discrimination, I am well aware of how misogynistic remarks on the internet can impact the living environment for women. Therefore, I am delighted to contribute, even a tiny step, to fostering a fair and just society through this work. During the completion of this thesis, I have come to realize that the road ahead is still long and challenging. For instance, a major obstacle in addressing online gender discrimination is the lack of high-quality datasets, which are essential for model training. Nevertheless, every journey has its highlights, and while this bachelor's thesis may mark a temporary conclusion, the project itself continues to thrive and evolve.

What the future holds for this project

As I mentioned earlier, there is currently a scarcity of datasets specifically targeting online sexism detection. Many researchers resort to supplementing their training with other hate speech datasets, which may not capture the nuances of gender discrimination adequately. Therefore, further research in this area could involve conducting more in-depth studies, such as incorporating additional languages, especially those for which state-of-the-art models have limited support, and employing enhanced preprocessing techniques.

After delving deeper into the study of gender-based hate speech detection and understanding its complexity, it is time to expand beyond the scope of the same theme. Looking ahead, the focus will shift towards multi-class analysis and classification, while leveraging the vast opportunities presented by social media as an inexhaustible source of data. This ever-evolving landscape offers tremendous potential. The next step is to embark on this exciting new phase.

Chapter 6

Appendix

Training-Evaluation Tables

In this first chapter of the Appendix, one will find all the tables with the training outcomes mentioned above in chapters 3 and 4.

1. **The Batch sizes:** it could be from 4 to 32, (4, 8, 16, 32).
2. **The Learning Rates:** it can be found in the left side column.
3. **Metrics and Losses:** they can be found below the batch sizes and they will be:
 - Accuracy (shown as ACC)
 - F1-score (shown as F1)
 - Training Loss
 - Validation Loss (shown as Valid. Loss)

It should be noted that every table will have two epochs in different color. If the epoch is highlighted in electric blue, this will be the epoch with the best metric values in the whole experiment or table. Notice that we have not been able to obtain results for the XLNet model with batch sizes higher than 8 in the EXIST 2023 dataset due to GPU limitations.

6.1 EXIST 2021 Tables with Translation

6.1.1 English

BERT with EXIST2021 English(4 epochs per per training) with translation													
Batch/LR	8				16				32				
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
2.10 ⁻⁴	1	0.53	0.69	0.71	0.69	0.53	0.69	0.71	0.70	0.47	0.64	0.70	0.70
	2	0.53	0.69	0.71	0.69	0.47	0.64	0.71	0.70	0.47	0.64	0.70	0.70
	3	0.47	0.64	0.70	0.69	0.47	0.64	0.71	0.69	0.53	0.69	0.70	0.69
	4	0.53	0.69	0.70	0.69	0.53	0.69	0.70	0.69	0.53	0.69	0.70	0.69
5.10 ⁻⁴	1	0.47	0.64	0.72	0.71	0.53	0.69	0.71	0.77	0.53	0.69	0.71	0.69
	2	0.47	0.64	0.72	0.72	0.53	0.69	0.71	0.69	0.47	0.64	0.70	0.70
	3	0.47	0.64	0.71	0.70	0.53	0.69	0.71	0.69	0.47	0.64	0.70	0.70
	4	0.53	0.69	0.70	0.69	0.53	0.69	0.70	0.69	0.47	0.64	0.70	0.70
3.10 ⁻⁵	1	0.75	0.74	0.13	1.69	0.76	0.76	0.07	1.62	0.75	0.74	0.58	0.51
	2	0.74	0.74	0.07	2.02	0.75	0.75	0.04	1.92	0.75	0.74	0.40	0.55
	3	0.75	0.75	0.03	2.16	0.75	0.75	0.01	1.98	0.76	0.76	0.24	0.64
	4	0.74	0.74	0.01	2.34	0.75	0.75	0.01	2.05	0.76	0.75	0.14	0.75
5.10 ⁻⁵	1	0.53	0.69	0.68	0.70	0.75	0.75	0.18	1.29	0.75	0.75	0.28	1.36
	2	0.47	0.64	0.69	0.69	0.74	0.74	0.12	1.41	0.76	0.76	0.19	1.56
	3	0.53	0.69	0.69	0.69	0.75	0.75	0.05	1.77	0.76	0.76	0.12	1.66
	4	0.53	0.69	0.70	0.69	0.74	0.74	0.02	1.83	0.76	0.76	0.07	1.75

RoBERTa with EXIST2021 English (4 epochs per per training) with translation													
Batch/LR	4				8				16				
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
5.10 ⁻⁴	1	0.47	0.64	0.70	0.75	0.47	0.64	0.70	0.75	0.53	0.69	0.75	0.74
	2	0.47	0.64	0.68	0.72	0.47	0.64	0.68	0.72	0.53	0.69	0.71	0.69
	3	0.47	0.64	0.71	0.70	0.47	0.64	0.71	0.70	0.53	0.69	0.71	0.69
	4	0.53	0.69	0.70	0.69	0.53	0.69	0.70	0.69	0.53	0.69	0.70	0.69
1.10 ⁻⁵	1	0.47	0.64	0.70	0.71	0.53	0.69	0.70	0.71	0.68	0.68	0.56	0.62
	2	0.53	0.46	0.70	0.78	0.53	0.46	0.70	0.74	0.67	0.66	0.51	0.68
	3	0.64	0.62	0.60	0.64	0.67	0.67	0.62	0.63	0.67	0.67	0.47	0.67
	4	0.67	0.67	0.57	0.70	0.67	0.67	0.55	0.66	0.69	0.69	0.42	0.67
3.10 ⁻⁵	1	0.61	0.59	0.69	0.65	0.62	0.60	0.57	0.66	0.53	0.69	0.49	0.69
	2	0.59	0.55	0.64	0.73	0.65	0.64	0.48	0.71	0.48	0.33	0.69	0.70
	3	0.68	0.68	0.58	0.62	0.68	0.68	0.41	0.70	0.53	0.69	0.70	0.69
	4	0.68	0.68	0.52	0.69	0.70	0.70	0.37	0.86	0.53	0.69	0.69	0.69
5.10 ⁻⁵	1	0.47	0.64	0.71	0.70	0.47	0.64	0.69	0.72	0.53	0.69	0.69	0.71
	2	0.47	0.64	0.71	0.70	0.47	0.64	0.69	0.71	0.46	0.32	0.69	0.71
	3	0.53	0.69	0.70	0.69	0.53	0.69	0.70	0.69	0.53	0.69	0.71	0.69
	4	0.53	0.69	0.70	0.69	0.53	0.69	0.70	0.69	0.53	0.69	0.70	0.69

XLNet with EXIST2021 English(4 epochs per per training) with translation													
Batch/LR	4				8				16				
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
1.10 ⁻⁴	1	0.52	0.69	0.71	0.73	0.47	0.64	0.70	0.75	0.53	0.69	0.72	0.73
	2	0.53	0.69	0.71	0.68	0.47	0.64	0.68	0.72	0.53	0.69	0.71	0.69
	3	0.53	0.69	0.70	0.69	0.47	0.64	0.71	0.70	0.53	0.69	0.70	0.69
	4	0.47	0.64	0.71	0.69	0.53	0.69	0.70	0.69	0.47	0.64	0.70	0.69
2.10 ⁻⁵	1	0.70	0.69	0.53	1.24	0.65	0.65	0.62	0.68	0.70	0.70	0.43	0.67
	2	0.66	0.66	0.45	1.59	0.64	0.63	0.56	0.71	0.70	0.70	0.35	0.70
	3	0.70	0.70	0.35	1.45	0.70	0.70	0.47	0.61	0.70	0.70	0.30	0.73
	4	0.69	0.69	0.27	1.66	0.70	0.70	0.39	0.73	0.70	0.70	0.25	0.89
3.10 ⁻⁵	1	0.53	0.69	0.69	0.70	0.53	0.69	0.68	0.72	0.53	0.69	0.68	0.69
	2	0.47	0.64	0.70	0.69	0.53	0.69	0.68	0.69	0.53	0.69	0.69	0.69
	3	0.53	0.69	0.70	0.69	0.53	0.69	0.69	0.69	0.53	0.69	0.69	0.69
	4	0.53	0.69	0.70	0.69	0.53	0.69	0.70	0.69	0.53	0.69	0.69	0.69
5.10 ⁻⁵	1	0.60	0.53	0.65	0.70	0.55	0.39	0.68	0.71	0.53	0.69	0.70	0.69
	2	0.47	0.64	0.66	0.69	0.53	0.35	0.69	0.69	0.47	0.64	0.70	0.69
	3	0.63	0.58	0.65	0.69	0.53	0.35	0.70	0.69	0.53	0.69	0.70	0.69
	4	0.64	0.62	0.62	0.73	0.53	0.35	0.70	0.69	0.53	0.69	0.70	0.69

HateBERT with EXIST2021 English(4 epochs per per training) with translation													
Batch/LR	8				16				32				
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
2.10 ⁻⁴	1	0.53	0.69	0.69	0.70	0.53	0.69	0.70	0.70	0.53	0.69	0.70	0.69
	2	0.47	0.64	0.70	0.69	0.47	0.64	0.70	0.69	0.47	0.64	0.69	0.70
	3	0.47	0.64	0.70	0.69	0.53	0.69	0.70	0.69	0.53	0.69	0.70	0.69
	4	0.47	0.64	0.70	0.69	0.53	0.69	0.70	0.69	0.53	0.69	0.70	0.69
5.10 ⁻⁴	1	0.47	0.64	0.72	0.71	0.53	0.69	0.71	0.74	0.53	0.69	0.71	0.69
	2	0.47	0.64	0.72	0.72	0.53	0.69	0.71	0.69	0.47	0.64	0.70	0.70
	3	0.47	0.64	0.71	0.70	0.53	0.69	0.70	0.69	0.47	0.64	0.70	0.69
	4	0.53	0.69	0.70	0.69	0.53	0.69	0.70	0.69	0.53	0.69	0.70	0.69
3.10 ⁻⁵	1	0.74	0.74	0.05	2.29	0.81	0.80	0.03	1.23	0.82	0.82	0.03	2.44
	2	0.74	0.74	0.02	2.37	0.83	0.83	0.02	1.22	0.81	0.80	0.01	2.06
	3	0.74	0.74	0.01	2.46	0.83	0.82	0.01	1.31	0.82	0.81	0.00	2.00
	4	0.75	0.75	0.01	2.44	0.83	0.83	0.00	1.33	0.82	0.82	0.00	2.00
5.10 ⁻⁵	1	0.76	0.76	0.09	1.67	0.81	0.81	0.50	0.43	0.79	0.79	0.03	1.65
	2	0.74	0.74	0.07	2.03	0.82	0.82	0.28	0.45	0.74	0.72	0.03	2.12
	3	0.74	0.74	0.01	2.32	0.79	0.78	0.12	0.80	0.82	0.81	0.01	1.72
	4	0.74	0.74	0.01	2.37	0.83	0.83	0.04	0.80	0.81	0.81	0.00	1.75

6.1.2 Spanish

BERT with EXIST2021 Spanish(4 epochs per per training) with translation													
Batch/LR	8				16				32				
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
2.10 ⁻⁴	1	0.53	0.67	0.60	0.72	0.68	0.68	0.17	1.66	0.47	0.64	0.71	0.70
	2	0.61	0.61	0.45	1.20	0.69	0.69	0.11	1.67	0.47	0.64	0.71	0.70
	3	0.61	0.61	0.45	1.20	0.70	0.70	0.03	2.16	0.53	0.69	0.70	0.69
	4	0.61	0.61	0.23	1.76	0.71	0.71	0.02	2.14	0.53	0.69	0.70	0.69
5.10 ⁻⁴	1	0.61	0.61	0.68	0.80	0.47	0.64	0.72	0.71	0.47	0.64	0.72	0.73
	2	0.63	0.63	0.53	1.29	0.47	0.64	0.71	0.72	0.47	0.64	0.71	0.70
	3	0.68	0.68	0.27	1.45	0.47	0.64	0.71	0.71	0.53	0.69	0.71	0.69
	4	0.68	0.68	0.05	1.89	0.53	0.69	0.70	0.69	0.53	0.69	0.70	0.69
3.10 ⁻⁵	1	0.71	0.71	0.06	2.24	0.71	0.71	0.02	2.41	0.70	0.69	0.01	2.55
	2	0.71	0.71	0.05	2.06	0.72	0.72	0.02	2.38	0.72	0.72	0.01	2.50
	3	0.71	0.71	0.03	2.26	0.72	0.72	0.00	2.73	0.72	0.72	0.01	2.56
	4	0.73	0.73	0.01	2.28	0.72	0.72	0.01	2.56	0.72	0.72	0.00	2.60
5.10 ⁻⁵	1	0.70	0.70	0.44	0.73	0.63	0.61	0.67	0.66	0.72	0.72	0.02	2.41
	2	0.70	0.70	0.33	1.19	0.71	0.71	0.57	0.56	0.71	0.70	0.01	2.30
	3	0.71	0.71	0.20	1.31	0.71	0.71	0.44	0.62	0.71	0.71	0.01	2.40
	4	0.71	0.71	0.07	1.82	0.73	0.73	0.29	0.68	0.73	0.73	0.00	2.33

RoBERTa with EXIST2021 Spanish (4 epochs per per training) with translation													
Batch/LR	4				8				16				
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
5.10 ⁻⁴	1	0.47	0.64	0.75	0.69	0.53	0.69	0.74	0.69	0.47	0.64	0.74	0.71
	2	0.53	0.69	0.72	0.69	0.47	0.64	0.71	0.70	0.47	0.64	0.71	0.72
	3	0.47	0.64	0.71	0.74	0.47	0.64	0.71	0.75	0.47	0.64	0.71	0.71
	4	0.53	0.69	0.71	0.69	0.53	0.69	0.70	0.69	0.53	0.69	0.70	0.69
1.10 ⁻⁵	1	0.47	0.68	0.69	0.71	0.53	0.69	0.69	0.69	0.53	0.69	0.68	0.69
	2	0.53	0.69	0.68	0.69	0.53	0.69	0.69	0.69	0.53	0.69	0.69	0.69
	3	0.53	0.69	0.69	0.69	0.53	0.69	0.69	0.69	0.53	0.69	0.69	0.69
	4	0.53	0.69	0.70	0.69	0.53	0.69	0.69	0.69	0.53	0.69	0.69	0.69
3.10 ⁻⁵	1	0.47	0.64	0.60	0.70	0.47	0.64	0.68	0.70	0.53	0.69	0.68	0.69
	2	0.47	0.64	0.68	0.70	0.47	0.64	0.68	0.69	0.53	0.69	0.69	0.69
	3	0.47	0.64	0.69	0.69	0.53	0.69	0.69	0.69	0.53	0.69	0.69	0.69
	4	0.53	0.69	0.70	0.69	0.53	0.69	0.70	0.69	0.53	0.69	0.69	0.69
5.10 ⁻⁵	1	0.47	0.68	0.70	0.47	0.47	0.64	0.68	0.69	0.47	0.64	0.72	0.71
	2	0.47	0.64	0.68	0.70	0.47	0.64	0.68	0.71	0.47	0.64	0.71	0.72
	3	0.47	0.64	0.69	0.70	0.47	0.64	0.69	0.70	0.47	0.64	0.71	0.69
	4	0.53	0.69	0.70	0.69	0.53	0.69	0.70	0.69	0.53	0.69	0.70	0.69

XLNet with EXIST2021 Spanish(4 epochs per per training) with translation													
Batch/LR	4				8				16				
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
1.10 ⁻⁴	1	0.53	0.69	0.67	0.69	0.53	0.69	0.69	0.69	0.47	0.64	0.72	0.72
	2	0.61	0.60	0.69	0.67	0.47	0.64	0.70	0.71	0.47	0.64	0.71	0.72
	3	0.50	0.39	0.69	0.73	0.47	0.64	0.70	0.70	0.47	0.64	0.71	0.72
	4	0.63	0.62	0.67	0.67	0.53	0.69	0.70	0.69	0.53	0.69	0.70	0.69
2.10 ⁻⁵	1	0.64	0.64	0.66	0.65	0.53	0.69	0.69	0.69	0.47	0.64	0.69	0.69
	2	0.61	0.60	0.61	0.70	0.47	0.64	0.69	0.69	0.53	0.69	0.69	0.69
	3	0.64	0.63	0.61	0.66	0.53	0.35	0.69	0.69	0.53	0.69	0.69	0.69
	4	0.66	0.65	0.58	0.66	0.53	0.35	0.69	0.69	0.53	0.69	0.69	0.69
3.10 ⁻⁵	1	0.49	0.38	0.67	0.69	0.47	0.32	0.68	0.70	0.47	0.64	0.68	0.70
	2	0.65	0.65	0.66	0.72	0.47	0.64	0.69	0.71	0.47	0.64	0.68	0.69
	3	0.64	0.64	0.64	0.69	0.47	0.64	0.69	0.69	0.53	0.69	0.69	0.69
	4	0.65	0.64	0.62	0.65	0.53	0.69	0.70	0.69	0.53	0.69	0.69	0.69
5.10 ⁻⁵	1	0.47	0.64	0.73	0.70	0.53	0.69	0.67	0.69	0.47	0.64	0.62	0.71
	2	0.47	0.64	0.71	0.70	0.47	0.64	0.68	0.71	0.47	0.64	0.66	0.70
	3	0.47	0.64	0.71	0.72	0.47	0.64	0.69	0.71	0.53	0.69	0.68	0.69
	4	0.53	0.69	0.70	0.69	0.53	0.69	0.70	0.69	0.53	0.69	0.69	0.69

HateBERT with EXIST2021 Spanish(4 epochs per per training) with translation													
Batch/LR	8				16				32				
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
2.10 ⁻⁴	1	0.53	0.69	0.69	0.69	0.47	0.64	0.69	0.72	0.47	0.64	0.70	0.70
	2	0.47	0.64	0.70	0.72	0.47	0.64	0.70	0.70	0.47	0.64	0.70	0.69
	3	0.47	0.64	0.71	0.70	0.53	0.69	0.70	0.69	0.53	0.69	0.70	0.69
	4	0.53	0.69	0.70	0.69	0.53	0.69	0.70	0.69	0.53	0.69	0.69	0.69
5.10 ⁻⁴	1	0.53	0.69	0.72	0.69	0.47	0.64	0.71	0.71	0.47	0.64	0.71	0.73
	2	0.47	0.64	0.71	0.70	0.47	0.64	0.71	0.74	0.47	0.64	0.70	0.71
	3	0.47	0.64	0.71	0.74	0.47	0.64	0.71	0.71	0.53	0.69	0.70	0.69
	4	0.53	0.69	0.70	0.69	0.53	0.69	0.70	0.69	0.53	0.69	0.70	0.69
3.10 ⁻⁵	1	0.69	0.69	0.08	2.48	0.72	0.72	0.25	1.13	0.65	0.63	0.20	1.34
	2	0.71	0.71	0.06	2.27	0.70	0.70	0.19	1.41	0.69	0.68	0.09	1.52
	3	0.72	0.72	0.03	2.50	0.70	0.70	0.12	1.96	0.69	0.69	0.05	1.81
	4	0.71	0.71	0.02	2.63	0.70	0.70	0.06	2.12	0.70	0.70	0.07	1.74
5.10 ⁻⁵	1	0.72	0.72	0.33	1.32	0.69	0.69	0.18	2.04	0.64	0.62	0.65	0.63
	2	0.71	0.71	0.19	1.60	0.67	0.67	0.17	2.29	0.72	0.72	0.52	0.57
	3	0.71	0.71	0.11	2.01	0.71	0.71	0.10	2.02	0.72	0.71	0.38	0.63
	4	0.73	0.73	0.04	2.09	0.72	0.72	0.04	2.23	0.72	0.72	0.28	0.68

DeHateBERT with EXIST2021 Spanish(4 epochs per per training) with translation													
Batch/LR	4				8				16				
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
1.10 ⁻⁵	1	0.71	0.71	0.64	1.23	0.72	0.72	0.01	3.05	0.69	0.69	0.62	0.60
	2	0.71	0.71	0.50	1.67	0.71	0.71	0.01	3.27	0.70	0.70	0.56	0.57
	3	0.71	0.71	0.35	2.12	0.71	0.71	0.01	3.31	0.73	0.73	0.52	0.57
	4	0.71	0.71	0.21	2.43	0.70	0.70	0.05	3.41	0.74	0.74	0.51	0.55
2.10 ⁻⁵	1	0.70	0.70	0.45	1.23	0.72	0.71	0.06	2.37	0.71	0.71	0.31	0.91
	2	0.71	0.71	0.27	1.68	0.71	0.71	0.05	2.58	0.72	0.71	0.29	0.77
	3	0.71	0.71	0.13	1.97	0.71	0.70	0.04	2.74	0.72	0.72	0.30	0.80
	4	0.71	0.71	0.07	2.03	0.71	0.70	0.09	2.74	0.72	0.72	0.39	0.71
3.10 ⁻⁵	1	0.71	0.71	0.23	1.88	0.72	0.71	0.20	1.57	0.70	0.70	0.12	1.92
	2	0.71	0.71	0.15	1.98	0.72	0.71	0.18	1.53	0.71	0.71	0.17	1.75
	3	0.71	0.71	0.07	2.33	0.72	0.71	0.15	1.89	0.72	0.72	0.19	1.45
	4	0.71	0.71	0.08	2.33	0.70	0.70	0.15	2.00	0.72	0.72	0.35	1.17
5.10 ⁻⁵	1	0.71	0.70	0.64	1.39	0.74	0.73	0.54	0.60	0.69	0.69	0.21	1.36
	2	0.72	0.71	0.47	1.30	0.71	0.70	0.45	0.80	0.71	0.71	0.21	1.58
	3	0.71	0.71	0.33	1.81	0.72	0.72	0.39	1.04	0.72	0.72	0.22	1.65
	4	0.72	0.72	0.20	1.99	0.72	0.72	0.31	1.19	0.70	0.69	0.30	1.19

6.2 EXIST 2021 Tables without Translation

6.2.1 English

BERT with EXIST2021 English(4 epochs per per training) without translation													
Batch/LR	8				16				32				
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
2.10 ⁻⁴	1	0.51	0.68	0.71	0.72	0.78	0.78	0.59	0.50	0.74	0.74	0.19	1.13
	2	0.51	0.68	0.70	0.69	0.76	0.76	0.36	0.57	0.73	0.73	0.07	1.37
	3	0.49	0.65	0.70	0.70	0.76	0.76	0.16	1.10	0.75	0.75	0.04	1.45
	4	0.51	0.68	0.70	0.69	0.75	0.75	0.04	1.32	0.76	0.76	0.01	1.63
5.10 ⁻⁴	1	0.51	0.68	0.71	0.71	0.51	0.68	0.73	0.77	0.49	0.65	0.85	0.69
	2	0.51	0.68	0.70	0.69	0.51	0.68	0.71	0.71	0.51	0.68	0.70	0.71
	3	0.51	0.68	0.70	0.69	0.49	0.65	0.71	0.70	0.49	0.65	0.70	0.70
	4	0.51	0.68	0.70	0.69	0.51	0.68	0.69	0.69	0.51	0.68	0.70	0.69
3.10 ⁻⁵	1	0.51	0.68	0.68	0.69	0.51	0.68	0.68	0.69	0.51	0.68	0.69	0.69
	2	0.51	0.68	0.68	0.69	0.51	0.68	0.68	0.69	0.51	0.68	0.69	0.69
	3	0.51	0.68	0.68	0.69	0.51	0.68	0.69	0.69	0.51	0.68	0.69	0.69
	4	0.51	0.68	0.68	0.69	0.51	0.68	0.69	0.69	0.51	0.68	0.69	0.69
5.10 ⁻⁵	1	0.51	0.68	0.67	0.70	0.77	0.77	0.58	0.50	0.51	0.68	0.68	0.69
	2	0.51	0.68	0.67	0.69	0.75	0.75	0.34	0.60	0.51	0.68	0.69	0.69
	3	0.49	0.65	0.70	0.70	0.74	0.74	0.15	1.06	0.49	0.65	0.69	0.69
	4	0.51	0.68	0.70	0.69	0.76	0.76	0.06	1.16	0.51	0.68	0.69	0.69

RoBERTa with EXIST2021 English (4 epochs per per training) without translation													
Batch/LR	4				8				16				
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
5.10 ⁻⁴	1	0.51	0.68	0.74	0.70	0.51	0.68	0.76	0.69	0.51	0.68	0.73	0.76
	2	0.51	0.68	0.72	0.69	0.51	0.68	0.72	0.69	0.51	0.68	0.72	0.71
	3	0.51	0.68	0.71	0.69	0.51	0.68	0.71	0.69	0.49	0.65	0.71	0.70
	4	0.51	0.68	0.70	0.69	0.51	0.68	0.70	0.69	0.51	0.68	0.70	0.69
1.10 ⁻⁵	1	0.51	0.68	0.68	0.69	0.51	0.68	0.68	0.70	0.51	0.68	0.72	0.69
	2	0.51	0.68	0.68	0.69	0.51	0.68	0.69	0.69	0.51	0.68	0.70	0.70
	3	0.51	0.68	0.69	0.69	0.64	0.63	0.62	0.70	0.50	0.37	0.47	0.69
	4	0.51	0.68	0.70	0.69	0.66	0.66	0.49	0.69	0.56	0.47	0.68	0.69
3.10 ⁻⁵	1	0.51	0.68	0.68	0.70	0.51	0.68	0.69	0.71	0.51	0.68	0.74	0.70
	2	0.51	0.68	0.68	0.69	0.51	0.68	0.69	0.69	0.51	0.68	0.70	0.70
	3	0.49	0.65	0.69	0.69	0.49	0.65	0.69	0.69	0.49	0.65	0.70	0.70
	4	0.51	0.68	0.69	0.69	0.51	0.68	0.70	0.69	0.51	0.68	0.70	0.69
5.10 ⁻⁵	1	0.51	0.68	0.68	0.72	0.51	0.68	0.70	0.72	0.51	0.68	0.73	0.71
	2	0.51	0.68	0.68	0.69	0.51	0.68	0.71	0.69	0.51	0.68	0.71	0.70
	3	0.51	0.68	0.69	0.69	0.51	0.68	0.71	0.69	0.49	0.65	0.70	0.70
	4	0.51	0.68	0.69	0.69	0.51	0.68	0.70	0.69	0.51	0.68	0.70	0.69

XLNet with EXIST2021 English(4 epochs per per training) without translation													
Batch/LR	4				8				16				
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
1.10 ⁻⁴	1	0.51	0.68	0.74	0.72	0.53	0.68	0.70	0.83	0.53	0.68	0.70	0.83
	2	0.51	0.68	0.71	0.69	0.53	0.68	0.68	0.80	0.53	0.68	0.68	0.80
	3	0.51	0.68	0.70	0.69	0.61	0.61	0.62	1.12	0.61	0.61	0.62	1.12
	4	0.51	0.68	0.70	0.69	0.61	0.61	0.62	1.23	0.61	0.61	0.62	1.23
2.10 ⁻⁵	1	0.66	0.66	0.70	0.63	0.51	0.64	0.70	0.63	0.51	0.64	0.70	0.63
	2	0.70	0.70	0.58	0.60	0.51	0.64	0.70	0.69	0.51	0.64	0.70	0.69
	3	0.69	0.69	0.52	1.11	0.69	0.69	0.69	0.68	0.69	0.69	0.69	0.68
	4	0.70	0.70	0.46	1.25	0.71	0.70	0.68	0.69	0.71	0.70	0.68	0.69
3.10 ⁻⁵	1	0.67	0.66	0.33	1.84	0.51	0.64	0.70	0.80	0.51	0.64	0.70	0.80
	2	0.67	0.67	0.24	1.81	0.51	0.64	0.53	1.02	0.51	0.64	0.53	1.02
	3	0.67	0.67	0.19	1.74	0.63	0.61	0.28	1.23	0.63	0.61	0.28	1.23
	4	0.67	0.67	0.15	1.94	0.63	0.61	0.06	1.57	0.63	0.61	0.06	1.57
5.10 ⁻⁵	1	0.67	0.67	0.63	1.29	0.67	0.67	0.63	1.29	0.67	0.67	0.60	0.66
	2	0.66	0.65	0.54	1.14	0.66	0.65	0.54	1.14	0.66	0.65	0.54	0.66
	3	0.68	0.68	0.40	1.55	0.68	0.68	0.40	1.55	0.68	0.68	0.32	0.98
	4	0.69	0.69	0.28	1.54	0.69	0.69	0.30	1.55	0.71	0.70	0.17	1.21

HateBERT with EXIST2021 English(4 epochs per per training) without translation													
Batch/LR	8				16				32				
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
2.10 ⁻⁴	1	0.51	0.68	0.70	0.73	0.51	0.68	0.66	0.69	0.66	0.65	0.64	0.71
	2	0.51	0.68	0.69	0.69	0.65	0.64	0.59	0.68	0.70	0.70	0.52	0.65
	3	0.49	0.65	0.70	0.69	0.68	0.68	0.57	0.68	0.71	0.71	0.48	0.68
	4	0.51	0.68	0.70	0.69	0.67	0.67	0.56	0.67	0.69	0.69	0.48	0.64
5.10 ⁻⁴	1	0.51	0.68	0.73	0.69	0.51	0.68	0.74	0.76	0.51	0.68	0.77	0.78
	2	0.51	0.68	0.71	0.69	0.51	0.68	0.70	0.71	0.51	0.68	0.71	0.70
	3	0.51	0.68	0.71	0.69	0.49	0.66	0.71	0.70	0.49	0.65	0.71	0.70
	4	0.51	0.68	0.70	0.69	0.51	0.68	0.70	0.69	0.51	0.68	0.70	0.70
3.10 ⁻⁵	1	0.75	0.74	0.01	3.18	0.76	0.76	0.03	1.68	0.77	0.77	0.02	2.31
	2	0.75	0.75	0.03	2.63	0.76	0.76	0.00	1.97	0.75	0.75	0.01	2.24
	3	0.75	0.75	0.01	2.76	0.77	0.77	0.00	2.07	0.77	0.77	0.02	2.03
	4	0.76	0.76	0.00	2.79	0.78	0.78	0.00	2.08	0.77	0.77	0.00	2.03
5.10 ⁻⁵	1	0.73	0.73	0.04	2.38	0.77	0.76	0.55	0.50	0.75	0.75	0.02	2.07
	2	0.75	0.75	0.04	2.57	0.76	0.76	0.27	0.62	0.76	0.76	0.02	2.16
	3	0.75	0.75	0.01	2.54	0.78	0.78	0.09	1.08	0.77	0.77	0.00	2.15
	4	0.75	0.75	0.01	2.64	0.78	0.78	0.03	1.21	0.75	0.75	0.00	2.19

6.2.2 Spanish

BERT with EXIST2021 Spanish(4 epochs per per training) without translation													
Batch/LR	8				16				32				
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
2.10 ⁻⁴	1	0.50	0.67	0.70	0.69	0.50	0.67	0.72	0.70	0.50	0.67	0.69	0.71
	2	0.50	0.67	0.70	0.69	0.50	0.67	0.71	0.69	0.50	0.67	0.69	0.69
	3	0.50	0.67	0.70	0.70	0.50	0.67	0.71	0.73	0.50	0.67	0.69	0.69
	4	0.50	0.67	0.70	0.69	0.50	0.67	0.70	0.69	0.50	0.67	0.69	0.69
5.10 ⁻⁴	1	0.50	0.67	0.74	0.70	0.50	0.67	0.72	0.71	0.50	0.67	0.74	0.70
	2	0.50	0.67	0.73	0.70	0.50	0.67	0.71	0.69	0.50	0.67	0.70	0.69
	3	0.50	0.67	0.71	0.71	0.50	0.67	0.71	0.73	0.50	0.67	0.70	0.70
	4	0.50	0.67	0.71	0.70	0.50	0.67	0.70	0.69	0.50	0.67	0.70	0.69
3.10 ⁻⁵	1	0.50	0.67	0.70	0.70	0.70	0.70	0.66	0.59	0.70	0.70	0.61	0.62
	2	0.50	0.67	0.71	0.69	0.73	0.73	0.54	0.54	0.70	0.70	0.53	0.65
	3	0.50	0.67	0.69	0.69	0.73	0.72	0.38	0.63	0.70	0.69	0.48	0.64
	4	0.50	0.67	0.69	0.69	0.73	0.73	0.25	0.72	0.73	0.73	0.40	0.65
5.10 ⁻⁵	1	0.50	0.67	0.69	0.69	0.70	0.70	0.40	0.66	0.50	0.67	0.71	0.71
	2	0.74	0.74	0.64	0.57	0.70	0.70	0.19	1.12	0.50	0.67	0.70	0.69
	3	0.73	0.73	0.49	0.58	0.72	0.72	0.10	1.42	0.56	0.56	0.69	0.69
	4	0.75	0.75	0.38	0.65	0.73	0.73	0.04	1.60	0.68	0.68	0.66	0.63

RoBERTa with EXIST2021 Spanish (4 epochs per per training) without translation													
Batch/LR	4				8				16				
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
5.10 ⁻⁴	1	0.50	0.67	0.74	0.88	0.50	0.67	0.73	0.70	0.50	0.67	0.72	0.70
	2	0.50	0.67	0.73	0.72	0.50	0.67	0.73	0.71	0.50	0.67	0.72	0.69
	3	0.50	0.67	0.71	0.75	0.50	0.67	0.71	0.72	0.50	0.67	0.71	0.73
	4	0.50	0.67	0.71	0.69	0.50	0.67	0.71	0.70	0.50	0.67	0.71	0.69
1.10 ⁻⁵	1	0.50	0.67	0.70	0.69	0.50	0.67	0.68	0.69	0.50	0.67	0.69	0.69
	2	0.50	0.67	0.70	0.69	0.50	0.67	0.68	0.69	0.50	0.67	0.69	0.69
	3	0.50	0.67	0.70	0.69	0.50	0.67	0.69	0.69	0.50	0.67	0.69	0.69
	4	0.50	0.67	0.70	0.69	0.50	0.67	0.70	0.69	0.50	0.67	0.70	0.69
3.10 ⁻⁵	1	0.50	0.67	0.70	0.69	0.50	0.67	0.68	0.70	0.50	0.67	0.72	0.69
	2	0.50	0.67	0.70	0.69	0.50	0.67	0.68	0.69	0.50	0.67	0.71	0.69
	3	0.50	0.67	0.70	0.69	0.50	0.67	0.69	0.69	0.64	0.63	0.69	0.67
	4	0.50	0.67	0.70	0.69	0.50	0.67	0.70	0.69	0.65	0.65	0.62	0.63
5.10 ⁻⁵	1	0.50	0.67	0.70	0.69	0.50	0.67	0.68	0.70	0.50	0.67	0.70	0.71
	2	0.50	0.67	0.70	0.70	0.50	0.67	0.68	0.69	0.50	0.67	0.70	0.69
	3	0.50	0.67	0.70	0.70	0.50	0.67	0.69	0.69	0.50	0.67	0.69	0.73
	4	0.50	0.67	0.70	0.69	0.50	0.67	0.70	0.69	0.50	0.67	0.70	0.69

XLNet with EXIST2021 Spanish (4 epochs per per training) without translation													
Batch/LR	4				8				16				
	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
1.10 ⁻⁴	1	0.50	0.67	0.74	0.88	0.50	0.67	0.73	0.70	0.50	0.67	0.72	0.70
	2	0.50	0.67	0.73	0.72	0.50	0.67	0.73	0.71	0.50	0.67	0.72	0.69
	3	0.50	0.67	0.71	0.75	0.50	0.67	0.71	0.72	0.50	0.67	0.71	0.73
	4	0.50	0.67	0.71	0.69	0.50	0.67	0.71	0.70	0.50	0.67	0.71	0.69
2.10 ⁻⁵	1	0.50	0.67	0.70	0.69	0.50	0.67	0.68	0.69	0.50	0.67	0.69	0.69
	2	0.50	0.67	0.70	0.69	0.50	0.67	0.68	0.69	0.50	0.67	0.69	0.69
	3	0.50	0.67	0.70	0.69	0.50	0.67	0.69	0.69	0.50	0.67	0.69	0.69
	4	0.50	0.67	0.70	0.69	0.50	0.67	0.70	0.69	0.50	0.67	0.70	0.69
3.10 ⁻⁵	1	0.50	0.67	0.70	0.69	0.50	0.67	0.68	0.70	0.50	0.67	0.75	0.69
	2	0.50	0.67	0.70	0.69	0.50	0.67	0.68	0.69	0.50	0.67	0.71	0.69
	3	0.50	0.67	0.70	0.69	0.50	0.67	0.69	0.69	0.66	0.65	0.70	0.66
	4	0.50	0.67	0.70	0.69	0.50	0.67	0.70	0.69	0.65	0.65	0.62	0.63
5.10 ⁻⁵	1	0.50	0.67	0.70	0.69	0.50	0.67	0.68	0.70	0.50	0.67	0.70	0.71
	2	0.50	0.67	0.70	0.70	0.50	0.67	0.68	0.69	0.50	0.67	0.70	0.69
	3	0.50	0.67	0.70	0.70	0.50	0.67	0.69	0.69	0.50	0.67	0.69	0.73
	4	0.50	0.67	0.70	0.69	0.50	0.67	0.70	0.69	0.50	0.67	0.70	0.69

HateBERT with EXIST2021 Spanish(4 epochs per per training) without translation													
Batch/LR	8				16				32				
	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
2.10 ⁻⁴	1	0.50	0.67	0.72	0.69	0.50	0.67	0.71	0.70	0.50	0.67	0.70	0.71
	2	0.50	0.67	0.71	0.70	0.50	0.67	0.71	0.69	0.50	0.67	0.70	0.69
	3	0.50	0.67	0.70	0.71	0.50	0.67	0.70	0.69	0.50	0.67	0.70	0.69
	4	0.50	0.67	0.70	0.69	0.50	0.67	0.70	0.69	0.50	0.67	0.70	0.69
5.10 ⁻⁴	1	0.50	0.67	0.73	0.70	0.50	0.67	0.72	0.50	0.67	0.71	0.74	0.70
	2	0.50	0.67	0.72	0.69	0.50	0.67	0.71	0.69	0.50	0.67	0.70	0.69
	3	0.50	0.67	0.70	0.71	0.50	0.67	0.70	0.73	0.50	0.67	0.70	0.70
	4	0.50	0.67	0.70	0.70	0.50	0.67	0.70	0.69	0.50	0.67	0.70	0.69
3.10 ⁻⁵	1	0.50	0.67	0.67	0.70	0.50	0.67	0.71	0.70	0.50	0.67	0.69	0.69
	2	0.50	0.67	0.67	0.70	0.50	0.67	0.70	0.69	0.50	0.67	0.69	0.69
	3	0.50	0.67	0.67	0.71	0.50	0.67	0.70	0.69	0.50	0.67	0.69	0.69
	4	0.50	0.67	0.70	0.70	0.50	0.67	0.70	0.69	0.50	0.67	0.70	0.69
5.10 ⁻⁵	1	0.50	0.67	0.68	0.70	0.50	0.67	0.70	0.70	0.50	0.67	0.68	0.69
	2	0.50	0.67	0.69	0.69	0.50	0.67	0.69	0.69	0.50	0.67	0.68	0.69
	3	0.50	0.67	0.69	0.69	0.50	0.67	0.70	0.69	0.50	0.67	0.69	0.69
	4	0.50	0.67	0.69	0.69	0.56	0.52	0.69	0.71	0.50	0.67	0.69	0.69

DeHateBERT with EXIST2021 Spanish(4 epochs per per training) without translation													
Batch/LR	4				8				16				
	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
1.10 ⁻⁵	1	0.71	0.71	0.64	1.23	0.72	0.72	0.01	3.05	0.69	0.69	0.62	0.60
	2	0.71	0.71	0.50	1.67	0.71	0.71	0.01	3.27	0.70	0.70	0.56	0.57
	3	0.71	0.71	0.35	2.12	0.71	0.71	0.01	3.31	0.73	0.73	0.52	0.57
	4	0.71	0.71	0.21	2.43	0.70	0.70	0.05	3.41	0.74	0.74	0.51	0.55
2.10 ⁻⁵	1	0.70	0.70	0.45	1.23	0.72	0.71	0.06	2.37	0.71	0.71	0.31	0.91
	2	0.71	0.71	0.27	1.68	0.71	0.71	0.05	2.58	0.72	0.71	0.29	0.77
	3	0.71	0.71	0.13	1.97	0.71	0.70	0.04	2.74	0.72	0.72	0.30	0.80
	4	0.71	0.71	0.07	2.03	0.71	0.70	0.09	2.74	0.72	0.72	0.39	0.71
3.10 ⁻⁵	1	0.71	0.71	0.23	1.88	0.72	0.71	0.20	1.57	0.70	0.70	0.12	1.92
	2	0.71	0.71	0.15	1.98	0.72	0.71	0.18	1.53	0.71	0.71	0.17	1.75
	3	0.71	0.71	0.07	2.33	0.72	0.71	0.15	1.89	0.72	0.72	0.19	1.45
	4	0.71	0.71	0.08	2.33	0.70	0.70	0.15	2.00	0.72	0.72	0.35	1.17
5.10 ⁻⁵	1	0.71	0.70	0.64	1.39	0.74	0.73	0.54	0.60	0.69	0.69	0.21	1.36
	2	0.72	0.71	0.47	1.30	0.71	0.70	0.45	0.80	0.71	0.71	0.21	1.58
	3	0.71	0.71	0.33	1.81	0.72	0.72	0.39	1.04	0.72	0.72	0.22	1.65
	4	0.72	0.72	0.20	1.99	0.72	0.72	0.31	1.19	0.70	0.69	0.30	1.19

6.3 EXIST 2023 Tables with Translation

6.3.1 English

BERT with EXIST2023 English(4 epochs per per training) with translation													
Batch/LR	8				16				32				
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
1.10 ⁻⁵	1	0.74	0.74	0.70	0.70	0.78	0.78	0.28	0.74	0.81	0.81	0.60	0.42
	2	0.76	0.76	0.45	1.27	0.78	0.78	0.15	1.11	0.82	0.83	0.43	0.41
	3	0.76	0.76	0.05	1.83	0.78	0.78	0.07	1.46	0.82	0.83	0.20	0.49
	4	0.76	0.76	0.02	2.13	0.80	0.80	0.04	1.56	0.82	0.81	0.08	0.65
2.10 ⁻⁵	1	0.76	0.76	0.13	1.96	0.78	0.78	0.28	0.74	0.53	0.69	0.71	0.69
	2	0.76	0.76	0.07	2.02	0.78	0.78	0.15	1.11	0.47	0.64	0.70	0.70
	3	0.76	0.76	0.03	2.16	0.78	0.78	0.07	1.46	0.47	0.64	0.70	0.70
	4	0.76	0.76	0.01	2.34	0.80	0.80	0.04	1.56	0.47	0.64	0.70	0.70
3.10 ⁻⁵	1	0.76	0.76	0.13	1.74	0.81	0.81	0.07	1.25	0.82	0.81	0.58	0.42
	2	0.76	0.76	0.05	2.27	0.81	0.81	0.05	1.31	0.83	0.83	0.36	0.41
	3	0.74	0.75	0.02	2.36	0.82	0.81	0.02	1.45	0.81	0.81	0.20	0.49
	4	0.74	0.75	0.02	2.31	0.81	0.80	0.00	1.52	0.82	0.81	0.10	0.65
5.10 ⁻⁵	1	0.76	0.76	0.13	1.96	0.81	0.81	0.08	1.25	0.82	0.81	0.58	0.42
	2	0.76	0.76	0.07	2.02	0.81	0.81	0.05	1.31	0.83	0.83	0.36	0.41
	3	0.78	0.78	0.03	2.16	0.82	0.81	0.02	1.56	0.81	0.81	0.20	0.49
	4	0.78	0.78	0.01	2.34	0.81	0.80	0.00	1.78	0.82	0.81	0.10	0.65

RoBERTa with EXIST2023 English (4 epochs per per training) with translation													
Batch/LR	4				8				16				
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
1.10 ⁻⁵	1	0.44	0.61	0.80	0.80	0.44	0.61	0.70	0.70	0.56	0.72	0.68	0.69
	2	0.44	0.64	0.80	0.80	0.56	0.72	0.69	0.69	0.56	0.72	0.68	0.69
	3	0.56	0.67	0.69	0.69	0.65	0.65	0.69	0.69	0.65	0.65	0.68	0.65
	4	0.56	0.67	0.69	0.69	0.56	0.72	0.68	0.69	0.64	0.64	0.68	0.69
2.10 ⁻⁵	1	0.49	0.67	0.70	0.70	0.49	0.67	0.70	0.53	0.69	0.70	0.70	0.62
	2	0.56	0.72	0.69	0.69	0.56	0.72	0.69	0.69	0.56	0.72	0.69	0.69
	3	0.65	0.65	0.69	0.69	0.65	0.65	0.69	0.69	0.65	0.65	0.69	0.69
	4	0.63	0.62	0.68	0.69	0.63	0.62	0.68	0.69	0.56	0.72	0.68	0.69
3.10 ⁻⁵	1	0.44	0.61	0.78	0.70	0.49	0.67	0.70	0.70	0.44	0.61	0.70	0.70
	2	0.61	0.64	0.70	0.73	0.49	0.67	0.69	0.69	0.56	0.72	0.69	0.69
	3	0.65	0.65	0.69	0.89	0.53	0.67	0.69	0.69	0.64	0.64	0.69	0.69
	4	0.64	0.63	0.68	0.96	0.63	0.62	0.68	0.69	0.56	0.72	0.68	0.69
5.10 ⁻⁵	1	0.44	0.61	0.68	0.70	0.47	0.64	0.70	0.70	0.44	0.61	0.78	0.70
	2	0.56	0.75	0.69	0.69	0.53	0.59	0.69	0.69	0.50	0.72	0.70	0.69
	3	0.56	0.75	0.69	0.69	0.53	0.69	0.56	0.86	0.65	0.65	0.70	0.69
	4	0.56	0.75	0.68	0.69	0.63	0.62	0.52	1.01	0.65	0.65	0.69	0.69

XLNet with EXIST2023 English(4 epochs per training) with translation											
Batch/LR	4				8				16		
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Not available due to GPU's limitation		
1.10 ⁻⁴	1	0.51	0.68	0.74	0.72	0.51	0.68	0.98	0.82		
	2	0.51	0.68	0.71	0.69	0.51	0.68	0.91	0.82		
	3	0.51	0.68	0.70	0.69	0.51	0.68	0.91	0.82		
	4	0.51	0.68	0.70	0.69	0.51	0.68	0.91	0.82		
2.10 ⁻⁵	1	0.66	0.66	0.78	0.63	0.67	0.66	0.65	0.63		
	2	0.70	0.70	0.65	0.57	0.64	0.63	0.56	0.71		
	3	0.69	0.69	0.52	1.09	0.70	0.70	0.51	1.11		
	4	0.70	0.70	0.39	1.35	0.70	0.70	0.39	1.23		
3.10 ⁻⁵	1	0.67	0.66	0.33	1.86	0.67	0.67	0.32	1.65		
	2	0.67	0.67	0.24	1.79	0.67	0.67	0.12	1.89		
	3	0.67	0.67	0.19	1.64	0.67	0.67	0.09	2.01		
	4	0.67	0.67	0.15	1.86	0.67	0.67	0.03	2.17		
5.10 ⁻⁵	1	0.67	0.67	0.67	1.29	0.68	0.68	0.65	1.07		
	2	0.66	0.65	0.55	1.27	0.66	0.66	0.54	1.27		
	3	0.68	0.68	0.42	1.64	0.68	0.68	0.42	1.44		
	4	0.68	0.68	0.18	1.84	0.68	0.68	0.09	1.86		

HateBERT with EXIST2023 English(4 epochs per per training) with translation													
Batch/LR	8				16				32				
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
1.10 ⁻⁵	1	0.81	0.80	0.00	2.43	0.82	0.82	0.00	2.31	0.53	0.69	0.70	0.69
	2	0.80	0.80	0.00	2.49	0.83	0.83	0.00	2.37	0.47	0.64	0.69	0.70
	3	0.81	0.80	0.00	2.43	0.81	0.81	0.01	2.54	0.53	0.69	0.70	0.69
	4	0.81	0.80	0.00	2.42	0.82	0.82	0.00	2.38	0.53	0.69	0.70	0.69
2.10 ⁻⁵	1	0.80	0.80	0.07	1.32	0.82	0.82	0.00	1.92	0.53	0.69	0.71	0.69
	2	0.81	0.81	0.03	1.44	0.82	0.82	0.00	1.98	0.47	0.64	0.70	0.70
	3	0.79	0.79	0.01	1.89	0.81	0.81	0.00	1.89	0.47	0.64	0.70	0.69
	4	0.80	0.80	0.00	1.92	0.82	0.82	0.01	1.84	0.53	0.69	0.70	0.69
3.10 ⁻⁵	1	0.80	0.80	0.04	1.42	0.81	0.80	0.03	1.23	0.82	0.82	0.03	2.44
	2	0.80	0.80	0.01	1.64	0.83	0.83	0.02	1.22	0.81	0.80	0.01	2.06
	3	0.79	0.79	0.00	1.79	0.83	0.82	0.01	1.31	0.82	0.81	0.00	2.00
	4	0.80	0.80	0.01	1.76	0.83	0.83	0.00	1.33	0.82	0.82	0.00	2.00
5.10 ⁻⁵	1	0.80	0.80	0.51	0.46	0.81	0.81	0.50	0.43	0.79	0.79	0.03	1.65
	2	0.80	0.79	0.31	0.69	0.82	0.82	0.28	0.45	0.74	0.72	0.03	2.12
	3	0.83	0.82	0.12	0.80	0.79	0.78	0.12	0.80	0.82	0.81	0.01	1.72
	4	0.83	0.83	0.04	0.96	0.83	0.83	0.04	0.80	0.81	0.81	0.00	1.75

6.3.2 Spanish

BERT with EXIST2023 Spanish(4 epochs per per training) with translation													
Batch/LR	8				16				32				
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
1.10 ⁻⁵	1	0.70	0.69	0.23	1.56	0.70	0.70	0.00	3.56	0.71	0.71	0.14	1.23
	2	0.70	0.70	0.17	1.90	0.70	0.70	0.03	2.62	0.68	0.67	0.09	2.03
	3	0.71	0.71	0.08	2.36	0.71	0.71	0.01	2.55	0.69	0.69	0.03	2.36
	4	0.72	0.72	0.01	2.60	0.71	0.71	0.01	2.62	0.70	0.70	0.01	2.23
2.10 ⁻⁵	1	0.70	0.69	0.37	1.95	0.71	0.71	0.04	2.28	0.71	0.71	0.02	2.22
	2	0.70	0.70	0.07	2.24	0.70	0.70	0.03	2.29	0.73	0.73	0.01	2.04
	3	0.71	0.71	0.04	2.53	0.71	0.71	0.01	2.56	0.72	0.72	0.01	2.19
	4	0.72	0.72	0.01	2.60	0.71	0.71	0.01	2.53	0.72	0.72	0.00	2.33
3.10 ⁻⁵	1	0.70	0.69	0.14	1.94	0.71	0.71	0.24	1.00	0.74	0.73	0.32	0.65
	2	0.70	0.70	0.07	2.24	0.72	0.72	0.16	1.41	0.71	0.71	0.18	0.89
	3	0.71	0.71	0.04	2.53	0.70	0.70	0.09	1.60	0.71	0.71	0.11	1.28
	4	0.72	0.72	0.01	2.60	0.73	0.73	0.07	1.70	0.71	0.71	0.10	1.16
5.10 ⁻⁵	1	0.71	0.71	0.18	1.94	0.71	0.71	0.18	1.47	0.64	0.63	0.68	0.64
	2	0.70	0.69	0.12	1.90	0.70	0.70	0.16	1.51	0.72	0.72	0.57	0.53
	3	0.73	0.73	0.04	2.48	0.69	0.69	0.08	2.06	0.74	0.74	0.46	0.55
	4	0.72	0.72	0.01	2.55	0.71	0.71	0.03	2.04	0.74	0.74	0.34	0.56

RoBERTa with EXIST2023 Spanish (4 epochs per per training) with translation													
Batch/LR	4				8				16				
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
1.10 ⁻⁵	1	0.72	0.72	0.63	0.58	0.70	0.70	0.21	1.24	0.73	0.73	0.00	3.53
	2	0.72	0.72	0.49	0.60	0.71	0.71	0.07	2.04	0.72	0.72	0.01	3.46
	3	0.71	0.71	0.40	0.60	0.72	0.72	0.06	2.13	0.73	0.73	0.00	3.31
	4	0.72	0.72	0.33	0.65	0.72	0.72	0.08	2.10	0.72	0.72	0.01	3.33
2.10 ⁻⁵	1	0.72	0.72	0.63	0.58	0.70	0.70	0.21	1.24	0.71	0.70	0.05	2.11
	2	0.72	0.72	0.49	0.60	0.71	0.71	0.07	2.04	0.72	0.72	0.05	2.61
	3	0.71	0.71	0.49	0.60	0.72	0.72	0.06	2.13	0.74	0.74	0.01	2.69
	4	0.71	0.71	0.43	0.65	0.72	0.72	0.08	2.10	0.74	0.74	0.02	2.74
3.10 ⁻⁵	1	0.72	0.72	0.63	0.58	0.70	0.70	0.21	1.24	0.70	0.70	0.21	1.48
	2	0.72	0.72	0.50	0.60	0.71	0.71	0.07	2.04	0.72	0.72	0.19	1.39
	3	0.71	0.71	0.37	0.68	0.72	0.72	0.06	2.13	0.72	0.72	0.09	1.96
	4	0.72	0.72	0.25	0.69	0.72	0.72	0.08	2.10	0.72	0.72	0.06	2.28
5.10 ⁻⁵	1	0.72	0.72	0.63	0.58	0.70	0.70	0.81	0.59	0.71	0.71	0.19	1.75
	2	0.72	0.72	0.49	0.60	0.72	0.72	0.52	0.57	0.72	0.72	0.14	1.70
	3	0.71	0.71	0.40	0.60	0.72	0.72	0.36	0.67	0.73	0.73	0.07	2.10
	4	0.72	0.72	0.33	0.65	0.73	0.73	0.24	0.75	0.73	0.73	0.05	2.22

XLNet with EXIST2023 English(4 epochs per training) with translation											
Batch/LR	4				8				16		
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Not available due to GPU's limitation		
1.10 ⁻⁴	1	0.73	0.73	0.00	3.53	0.54	0.67	0.70	0.70		
	2	0.72	0.72	0.01	3.46	0.54	0.71	0.69	0.70		
	3	0.73	0.73	0.00	3.31	0.66	0.66	0.69	0.69		
	4	0.72	0.72	0.01	3.33	0.66	0.66	0.69	0.69		
2.10 ⁻⁵	1	0.71	0.70	0.05	2.11	0.70	0.70	0.21	1.24		
	2	0.72	0.72	0.05	2.31	0.71	0.71	0.07	2.04		
	3	0.73	0.73	0.03	2.49	0.72	0.72	0.06	2.13		
	4	0.74	0.74	0.02	2.74	0.72	0.72	0.08	2.10		
3.10 ⁻⁵	1	0.70	0.70	0.21	1.48	0.70	0.70	0.21	1.24		
	2	0.72	0.72	0.19	1.39	0.71	0.71	0.07	2.04		
	3	0.72	0.72	0.09	1.96	0.72	0.72	0.06	2.13		
	4	0.72	0.72	0.06	2.28	0.72	0.72	0.08	2.10		
5.10 ⁻⁵	1	0.71	0.71	0.19	1.75	0.70	0.70	0.81	0.59		
	2	0.72	0.72	0.14	1.70	0.72	0.72	0.52	0.57		
	3	0.73	0.73	0.07	2.10	0.72	0.72	0.36	0.67		
	4	0.73	0.73	0.05	2.22	0.73	0.73	0.24	0.75		

HateBERT with EXIST2023 Spanish(4 epochs per per training) with translation													
Batch/LR	8				16				32				
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
1.10 ⁻⁵	1	0.70	0.70	0.21	1.24	0.47	0.73	0.73	0.00	0.70	0.70	0.21	1.24
	2	0.71	0.71	0.07	2.04	0.72	0.72	0.01	3.46	0.71	0.71	0.07	2.04
	3	0.72	0.72	0.06	2.16	0.73	0.73	0.00	3.31	0.72	0.72	0.08	2.10
	4	0.72	0.72	0.08	2.10	0.72	0.72	0.00	3.33	0.72	0.72	0.08	2.10
2.10 ⁻⁵	1	0.72	0.72	0.63	0.58	0.71	0.70	0.05	2.11	0.70	0.70	0.21	1.24
	2	0.72	0.72	0.49	0.60	0.72	0.72	0.05	2.61	0.71	0.71	0.07	2.04
	3	0.72	0.72	0.06	2.13	0.74	0.74	0.01	2.69	0.53	0.69	0.70	0.69
	4	0.72	0.72	0.33	0.65	0.74	0.74	0.02	2.74	0.72	0.72	0.08	2.10
3.10 ⁻⁵	1	0.70	0.70	0.21	1.48	0.70	0.70	0.21	1.48	0.70	0.70	0.21	1.24
	2	0.72	0.72	0.19	1.39	0.72	0.72	0.19	1.39	0.71	0.71	0.07	2.04
	3	0.72	0.72	0.09	1.96	0.72	0.72	0.09	1.96	0.72	0.72	0.06	2.13
	4	0.72	0.72	0.06	2.28	0.72	0.72	0.06	2.28	0.72	0.72	0.08	2.10
5.10 ⁻⁵	1	0.70	0.70	0.81	0.59	0.71	0.71	0.19	1.75	0.70	0.70	0.81	0.59
	2	0.72	0.72	0.52	0.57	0.72	0.72	0.14	1.70	0.72	0.72	0.52	0.57
	3	0.72	0.72	0.36	0.67	0.73	0.73	0.07	2.10	0.72	0.72	0.36	0.67
	4	0.73	0.73	0.24	0.75	0.73	0.73	0.05	2.22	0.73	0.73	0.24	0.75

DeHateBERT with EXIST2023 Spanish(4 epochs per per training) with translation													
Batch/LR	4				8				16				
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
1.10 ⁻⁵	1	0.72	0.72	0.59	0.50	0.77	0.77	0.59	0.56	0.75	0.75	0.59	0.50
	2	0.72	0.72	0.50	0.50	0.74	0.74	0.51	0.66	0.77	0.77	0.52	0.49
	3	0.73	0.73	0.49	0.48	0.77	0.77	0.45	0.59	0.79	0.79	0.49	0.48
	4	0.73	0.73	0.47	0.49	0.77	0.77	0.40	0.64	0.79	0.79	0.47	0.49
2.10 ⁻⁵	1	0.75	0.75	0.59	0.50	0.77	0.77	0.59	0.56	0.77	0.77	0.58	0.50
	2	0.75	0.75	0.50	0.49	0.74	0.74	0.51	0.66	0.77	0.77	0.50	0.51
	3	0.79	0.79	0.49	0.48	0.77	0.77	0.45	0.59	0.77	0.77	0.46	0.51
	4	0.79	0.79	0.47	0.49	0.77	0.77	0.40	0.64	0.78	0.78	0.42	0.51
3.10 ⁻⁵	1	0.75	0.75	0.50	0.49	0.77	0.77	0.59	0.56	0.76	0.75	0.59	0.51
	2	0.77	0.77	0.52	0.49	0.74	0.74	0.51	0.66	0.77	0.77	0.53	0.49
	3	0.79	0.79	0.49	0.48	0.77	0.77	0.45	0.59	0.78	0.78	0.49	0.48
	4	0.79	0.79	0.47	0.49	0.77	0.77	0.40	0.64	0.77	0.77	0.47	0.49
5.10 ⁻⁵	1	0.75	0.75	0.60	0.70	0.75	0.75	0.60	0.55	0.76	0.75	0.59	0.51
	2	0.77	0.77	0.60	0.79	0.72	0.72	0.51	0.62	0.77	0.77	0.53	0.49
	3	0.78	0.78	0.52	0.79	0.77	0.77	0.46	0.58	0.78	0.78	0.49	0.48
	4	0.78	0.78	0.47	0.86	0.76	0.76	0.41	0.60	0.77	0.77	0.47	0.49

6.4 EXIST 2023 Tables without Translation

6.4.1 English

BERT with EXIST2023 English(4 epochs per per training) without translation													
Batch/LR		8				16				32			
Metrics		Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss
1.10 ⁻⁵	1	0.76	0.76	0.13	1.96	0.79	0.79	0.08	1.25	0.80	0.80	0.08	0.74
	2	0.76	0.76	0.07	2.02	0.79	0.79	0.05	1.31	0.80	0.79	0.03	1.01
	3	0.76	0.76	0.03	2.16	0.79	0.80	0.02	1.56	0.79	0.80	0.02	1.10
	4	0.76	0.76	0.01	2.34	0.79	0.80	0.00	1.78	0.80	0.80	0.01	1.14
2.10 ⁻⁵	1	0.76	0.76	0.13	1.74	0.79	0.79	0.07	1.25	0.80	0.80	0.58	0.42
	2	0.76	0.76	0.05	2.27	0.79	0.79	0.05	1.31	0.80	0.80	0.36	0.41
	3	0.74	0.75	0.02	2.36	0.80	0.80	0.02	1.56	0.79	0.79	0.20	0.49
	4	0.74	0.75	0.02	2.31	0.79	0.80	0.00	1.52	0.79	0.79	0.10	0.65
3.10 ⁻⁵	1	0.76	0.76	0.13	1.96	0.78	0.78	0.28	0.74	0.78	0.78	0.60	0.42
	2	0.76	0.76	0.07	2.02	0.78	0.78	0.15	1.11	0.80	0.80	0.43	0.41
	3	0.76	0.76	0.03	2.16	0.78	0.78	0.07	1.46	0.78	0.78	0.20	0.49
	4	0.76	0.76	0.01	2.34	0.80	0.80	0.04	1.56	0.78	0.78	0.08	0.65
5.10 ⁻⁵	1	0.74	0.74	0.70	0.70	0.79	0.80	0.28	0.74	0.76	0.77	0.36	0.41
	2	0.76	0.76	0.45	1.27	0.78	0.78	0.15	1.11	0.80	0.80	0.20	0.49
	3	0.76	0.76	0.05	1.83	0.78	0.78	0.07	1.46	0.80	0.80	0.10	0.49
	4	0.76	0.76	0.02	2.13	0.80	0.80	0.04	1.56	0.80	0.80	0.10	0.65

RoBERTa with EXIST2023 English (4 epochs per per training) without translation													
Batch/LR		4				8				16			
Metrics		Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss
1.10 ⁻⁵	1	0.44	0.61	0.80	0.80	0.44	0.61	0.70	0.70	0.56	0.72	0.68	0.69
	2	0.44	0.64	0.80	0.80	0.56	0.72	0.69	0.69	0.56	0.72	0.68	0.69
	3	0.56	0.67	0.69	0.69	0.65	0.65	0.69	0.69	0.65	0.65	0.68	0.65
	4	0.56	0.67	0.69	0.69	0.56	0.72	0.68	0.69	0.64	0.64	0.68	0.69
2.10 ⁻⁵	1	0.49	0.67	0.70	0.70	0.49	0.67	0.70	0.53	0.69	0.70	0.70	0.62
	2	0.56	0.72	0.69	0.69	0.56	0.72	0.69	0.69	0.56	0.72	0.69	0.69
	3	0.65	0.65	0.69	0.69	0.65	0.65	0.69	0.69	0.65	0.65	0.69	0.69
	4	0.63	0.62	0.68	0.69	0.63	0.62	0.68	0.69	0.56	0.72	0.68	0.69
3.10 ⁻⁵	1	0.44	0.61	0.78	0.70	0.49	0.67	0.70	0.70	0.44	0.61	0.70	0.70
	2	0.61	0.64	0.70	0.73	0.49	0.67	0.69	0.69	0.56	0.72	0.69	0.69
	3	0.65	0.65	0.69	0.89	0.53	0.67	0.69	0.69	0.64	0.64	0.69	0.69
	4	0.64	0.63	0.68	0.96	0.63	0.62	0.68	0.69	0.56	0.72	0.68	0.69
5.10 ⁻⁵	1	0.44	0.61	0.68	0.70	0.47	0.64	0.70	0.70	0.44	0.61	0.78	0.70
	2	0.56	0.75	0.69	0.69	0.53	0.59	0.69	0.69	0.50	0.72	0.70	0.69
	3	0.56	0.75	0.69	0.69	0.53	0.69	0.56	0.86	0.65	0.65	0.70	0.69
	4	0.56	0.75	0.68	0.69	0.63	0.62	0.52	1.01	0.65	0.65	0.69	0.69

XLNet with EXIST2023 English(4 epochs per training) without translation											
Batch/LR	4				8				16		
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Not available due to GPU's limitation		
1.10 ⁻⁴	1	0.51	0.68	0.74	0.72	0.51	0.68	0.98	0.82		
	2	0.51	0.68	0.71	0.69	0.51	0.68	0.91	0.82		
	3	0.51	0.68	0.70	0.69	0.51	0.68	0.91	0.82		
	4	0.51	0.68	0.70	0.69	0.51	0.68	0.91	0.82		
2.10 ⁻⁵	1	0.66	0.66	0.78	0.63	0.67	0.66	0.65	0.63		
	2	0.70	0.70	0.65	0.57	0.64	0.63	0.56	0.71		
	3	0.69	0.69	0.52	1.09	0.70	0.70	0.51	1.11		
	4	0.70	0.70	0.39	1.35	0.70	0.70	0.39	1.23		
3.10 ⁻⁵	1	0.67	0.66	0.33	1.86	0.67	0.67	0.32	1.65		
	2	0.67	0.67	0.24	1.79	0.67	0.67	0.12	1.89		
	3	0.67	0.67	0.19	1.64	0.67	0.67	0.09	2.01		
	4	0.67	0.67	0.15	1.86	0.67	0.67	0.03	2.17		
5.10 ⁻⁵	1	0.67	0.67	0.67	1.29	0.68	0.68	0.65	1.07		
	2	0.66	0.65	0.55	1.27	0.66	0.66	0.54	1.27		
	3	0.68	0.68	0.42	1.64	0.68	0.68	0.42	1.44		
	4	0.68	0.68	0.18	1.84	0.68	0.68	0.09	1.86		

HateBERT with EXIST2023 English(4 epochs per per training) without translation													
Batch/LR	8				16				32				
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
1.10 ⁻⁵	1	0.80	0.79	0.00	2.51	0.82	0.82	0.00	2.31	0.53	0.69	0.70	0.69
	2	0.80	0.80	0.00	2.49	0.83	0.83	0.00	2.37	0.47	0.64	0.69	0.70
	3	0.81	0.80	0.00	2.43	0.81	0.81	0.01	2.54	0.53	0.69	0.70	0.69
	4	0.81	0.80	0.00	2.42	0.82	0.82	0.00	2.38	0.53	0.69	0.70	0.69
2.10 ⁻⁵	1	0.80	0.80	0.07	1.32	0.82	0.82	0.00	1.92	0.53	0.69	0.71	0.69
	2	0.81	0.81	0.03	1.44	0.82	0.82	0.00	1.98	0.47	0.64	0.70	0.70
	3	0.79	0.79	0.01	1.89	0.81	0.81	0.00	1.89	0.47	0.64	0.70	0.69
	4	0.80	0.80	0.00	1.92	0.82	0.82	0.01	1.84	0.53	0.69	0.70	0.69
3.10 ⁻⁵	1	0.80	0.80	0.04	1.42	0.78	0.78	0.51	0.40	0.78	0.78	0.06	2.14
	2	0.80	0.80	0.01	1.64	0.78	0.78	0.33	0.41	0.81	0.80	0.01	2.06
	3	0.79	0.79	0.00	1.79	0.79	0.78	0.18	0.71	0.78	0.81	0.00	2.00
	4	0.80	0.80	0.01	1.76	0.80	0.79	0.10	0.79	0.78	0.78	0.00	2.00
5.10 ⁻⁵	1	0.80	0.80	0.51	0.46	0.81	0.81	0.50	0.43	0.79	0.79	0.03	1.65
	2	0.80	0.79	0.31	0.69	0.82	0.82	0.28	0.45	0.74	0.72	0.03	2.12
	3	0.81	0.82	0.12	0.80	0.79	0.78	0.12	0.80	0.78	0.81	0.01	1.72
	4	0.81	0.83	0.04	0.96	0.81	0.81	0.00	1.75	0.81	0.81	0.00	1.75

6.4.2 Spanish

BERT with EXIST2023 Spanish(4 epochs per per training) without translation													
Batch/LR	8				16				32				
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
1.10 ⁻⁵	1	0.70	0.69	0.23	1.56	0.69	0.69	0.15	1.58	0.71	0.71	0.14	1.23
	2	0.70	0.70	0.17	1.90	0.68	0.68	0.09	1.75	0.68	0.67	0.09	2.03
	3	0.71	0.71	0.08	2.36	0.71	0.71	0.05	1.65	0.69	0.69	0.03	2.36
	4	0.72	0.72	0.01	2.60	0.72	0.72	0.02	1.79	0.70	0.70	0.01	2.23
2.10 ⁻⁵	1	0.70	0.69	0.37	1.95	0.71	0.71	0.04	2.28	0.71	0.71	0.02	2.22
	2	0.70	0.70	0.07	2.24	0.70	0.70	0.03	2.29	0.73	0.73	0.01	2.04
	3	0.71	0.71	0.04	2.53	0.71	0.71	0.01	2.56	0.72	0.72	0.01	2.19
	4	0.72	0.72	0.01	2.60	0.71	0.71	0.01	2.53	0.72	0.72	0.00	2.33
3.10 ⁻⁵	1	0.70	0.69	0.14	1.94	0.71	0.71	0.24	1.00	0.74	0.73	0.32	0.65
	2	0.70	0.70	0.07	2.24	0.72	0.72	0.16	1.41	0.71	0.71	0.18	0.89
	3	0.71	0.71	0.04	2.53	0.70	0.70	0.09	1.60	0.71	0.71	0.11	1.28
	4	0.72	0.72	0.01	2.60	0.73	0.73	0.07	1.70	0.71	0.71	0.10	1.16
5.10 ⁻⁵	1	0.71	0.71	0.18	1.94	0.71	0.71	0.18	1.47	0.64	0.63	0.68	0.64
	2	0.70	0.69	0.12	1.90	0.70	0.70	0.16	1.51	0.72	0.72	0.57	0.53
	3	0.73	0.73	0.04	2.48	0.69	0.69	0.08	2.06	0.74	0.74	0.46	0.55
	4	0.72	0.72	0.01	2.55	0.71	0.71	0.03	2.04	0.75	0.75	0.03	0.89

RoBERTa with EXIST2023 Spanish (4 epochs per per training) without translation													
Batch/LR	4				8				16				
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
1.10 ⁻⁵	1	0.72	0.72	0.63	0.58	0.70	0.70	0.21	1.24	0.73	0.73	0.00	3.53
	2	0.72	0.72	0.49	0.60	0.71	0.71	0.07	2.04	0.72	0.72	0.01	3.46
	3	0.71	0.71	0.40	0.60	0.72	0.72	0.06	2.13	0.73	0.73	0.00	3.31
	4	0.72	0.72	0.33	0.65	0.72	0.72	0.08	2.10	0.72	0.72	0.01	3.33
2.10 ⁻⁵	1	0.72	0.72	0.63	0.58	0.70	0.70	0.21	1.24	0.71	0.70	0.05	2.11
	2	0.72	0.72	0.49	0.60	0.71	0.71	0.07	2.04	0.72	0.72	0.05	2.61
	3	0.71	0.71	0.49	0.60	0.72	0.72	0.06	2.13	0.74	0.74	0.01	2.69
	4	0.71	0.71	0.43	0.65	0.72	0.72	0.08	2.10	0.74	0.74	0.02	2.74
3.10 ⁻⁵	1	0.72	0.72	0.63	0.58	0.70	0.70	0.21	1.24	0.70	0.70	0.21	1.48
	2	0.72	0.72	0.50	0.60	0.71	0.71	0.07	2.04	0.72	0.72	0.19	1.39
	3	0.71	0.71	0.37	0.68	0.72	0.72	0.06	2.13	0.72	0.72	0.09	1.96
	4	0.72	0.72	0.25	0.69	0.72	0.72	0.08	2.10	0.72	0.72	0.06	2.28
5.10 ⁻⁵	1	0.72	0.72	0.63	0.58	0.70	0.70	0.81	0.59	0.71	0.71	0.19	1.75
	2	0.72	0.72	0.49	0.60	0.72	0.72	0.52	0.57	0.72	0.72	0.14	1.70
	3	0.71	0.71	0.40	0.60	0.72	0.72	0.36	0.67	0.73	0.73	0.07	2.10
	4	0.72	0.72	0.33	0.65	0.73	0.73	0.24	0.75	0.73	0.73	0.05	2.22

XLNet with EXIST2023 English(4 epochs per training) without translation										
Batch/LR	4				8				16	
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Not available due to GPU's limitation	
1.10 ⁻⁴	1	0.73	0.73	0.00	3.53	0.54	0.67	0.70	0.70	
	2	0.72	0.72	0.01	3.46	0.54	0.71	0.69	0.70	
	3	0.73	0.73	0.00	3.31	0.66	0.66	0.69	0.69	
	4	0.72	0.72	0.01	3.33	0.66	0.66	0.69	0.69	
2.10 ⁻⁵	1	0.71	0.70	0.05	2.11	0.70	0.70	0.21	1.24	
	2	0.72	0.72	0.05	2.61	0.71	0.71	0.07	2.04	
	3	0.74	0.74	0.01	2.69	0.72	0.72	0.06	2.13	
	4	0.74	0.74	0.02	2.74	0.72	0.72	0.08	2.10	
3.10 ⁻⁵	1	0.70	0.70	0.21	1.48	0.70	0.70	0.21	1.24	
	2	0.72	0.72	0.19	1.39	0.71	0.71	0.07	2.04	
	3	0.72	0.72	0.09	1.96	0.72	0.72	0.06	2.13	
	4	0.72	0.72	0.06	2.28	0.72	0.72	0.08	2.10	
5.10 ⁻⁵	1	0.71	0.71	0.19	1.75	0.70	0.70	0.81	0.59	
	2	0.72	0.72	0.14	1.70	0.72	0.72	0.52	0.57	
	3	0.73	0.73	0.07	2.10	0.72	0.72	0.36	0.67	
	4	0.73	0.73	0.05	2.22	0.73	0.73	0.24	0.75	

HateBERT with EXIST2023 Spanish(4 epochs per per training) without translation													
Batch/LR	8				16				32				
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
1.10 ⁻⁵	1	0.53	0.69	0.21	0.78	0.53	0.69	0.63	1.53	0.49	0.38	0.21	0.69
	2	0.53	0.69	0.07	1.04	0.53	0.69	0.49	0.61	0.49	0.38	0.07	0.69
	3	0.53	0.69	0.06	1.09	0.53	0.69	0.61	1.02	0.53	0.69	0.06	0.69
	4	0.53	0.69	0.08	1.10	0.53	0.67	0.60	1.21	0.53	0.69	0.08	0.69
2.10 ⁻⁵	1	0.53	0.69	0.63	0.58	0.51	0.38	0.05	2.11	0.53	0.69	0.21	0.70
	2	0.53	0.69	0.49	0.60	0.51	0.38	0.05	2.61	0.53	0.69	0.07	0.69
	3	0.53	0.69	0.40	0.60	0.51	0.38	0.01	2.69	0.53	0.69	0.06	0.69
	4	0.53	0.69	0.33	0.65	0.51	0.38	0.02	2.74	0.53	0.69	0.08	0.69
3.10 ⁻⁵	1	0.53	0.69	0.21	1.48	0.49	0.38	0.21	0.69	0.53	0.69	0.56	0.69
	2	0.53	0.69	0.19	1.39	0.51	0.38	0.19	0.69	0.53	0.69	0.07	0.69
	3	0.53	0.69	0.09	1.96	0.53	0.69	0.09	0.96	0.53	0.69	0.06	0.69
	4	0.53	0.69	0.06	2.28	0.53	0.69	0.06	0.69	0.53	0.69	0.08	0.69
5.10 ⁻⁵	1	0.53	0.69	0.81	0.59	0.51	0.38	0.19	1.75	0.53	0.69	0.81	0.69
	2	0.53	0.69	0.52	0.57	0.51	0.38	0.14	1.70	0.53	0.69	0.52	0.69
	3	0.53	0.69	0.36	0.67	0.51	0.38	0.07	2.10	0.53	0.69	0.36	0.69
	4	0.53	0.69	0.24	0.75	0.51	0.38	0.05	2.10	0.53	0.69	0.24	0.67

DeHateBERT with EXIST2023 Spanish(4 epochs per per training) with translation													
Batch/LR	4				8				16				
Metrics	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	Acc	F1 score	Training loss	Valid. loss	
1.10 ⁻⁵	1	0.77	0.77	0.59	0.56	0.77	0.77	0.58	0.50	0.75	0.75	0.59	0.50
	2	0.74	0.74	0.51	0.66	0.77	0.76	0.50	0.51	0.77	0.77	0.50	0.49
	3	0.77	0.77	0.45	0.59	0.77	0.77	0.46	0.51	0.77	0.77	0.49	0.48
	4	0.77	0.77	0.40	0.64	0.78	0.78	0.42	0.51	0.77	0.77	0.47	0.49
2.10 ⁻⁵	1	0.77	0.77	0.59	0.56	0.75	0.75	0.59	0.50	0.72	0.72	0.59	0.50
	2	0.74	0.74	0.51	0.66	0.77	0.77	0.52	0.49	0.72	0.72	0.50	0.50
	3	0.77	0.77	0.45	0.59	0.77	0.77	0.49	0.48	0.73	0.73	0.49	0.48
	4	0.77	0.77	0.40	0.64	0.77	0.77	0.47	0.49	0.73	0.73	0.47	0.49
3.10 ⁻⁵	1	0.77	0.77	0.59	0.56	0.75	0.75	0.59	0.51	0.75	0.75	0.60	0.70
	2	0.74	0.74	0.51	0.66	0.77	0.77	0.53	0.49	0.77	0.77	0.60	0.79
	3	0.77	0.77	0.45	0.59	0.78	0.78	0.49	0.48	0.78	0.78	0.52	0.79
	4	0.77	0.77	0.40	0.64	0.77	0.77	0.47	0.49	0.78	0.78	0.47	0.86
5.10 ⁻⁵	1	0.75	0.75	0.60	0.55	0.76	0.75	0.59	0.51	0.75	0.75	0.59	0.50
	2	0.72	0.72	0.51	0.62	0.77	0.77	0.53	0.49	0.77	0.77	0.52	0.49
	3	0.77	0.77	0.46	0.58	0.77	0.77	0.46	0.58	0.77	0.77	0.49	0.48
	4	0.76	0.76	0.41	0.60	0.77	0.77	0.47	0.49	0.77	0.77	0.47	0.49

Bibliography

- [1] F. Rodríguez-Sánchez, J. Carrillo-de-Albornoz, L. Plaza, *et al.*, “Overview of exist 2021: Sexism identification in social networks”, *Procesamiento del Lenguaje Natural*, vol. 67, pp. 195–207, 2021.
- [2] D. o. E. UNITED NATIONS and S. Affairs, *Progress on the sustainable development goals: The gender snapshot 2022*, https://www.unwomen.org/sites/default/files/2022-09/Progress-on-the-sustainable-development-goals-the-gender-snapshot-2022-en_0.pdf, 2022.
- [3] A. Jha and R. Mamidi, “When does a compliment become sexist? analysis and classification of ambivalent sexism using twitter data”, in *Proceedings of the second workshop on NLP and computational social science*, 2017, pp. 7–16.
- [4] F. Rodríguez-Sánchez, J. Carrillo-de-Albornoz, and L. Plaza, “Automatic classification of sexism in social networks: An empirical study on twitter data”, *IEEE Access*, vol. 8, pp. 219 563–219 576, 2020.
- [5] UNED, *Exist: Sexism identification in social networks—first shared task at iberlef 2021*, <http://nlp.uned.es/exist2021/>, 2021.
- [6] UNED, *Exist: Sexism identification in social networks—third shared task at iberlef 2023*, <http://nlp.uned.es/exist2023/>, 2023.
- [7] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval: Tokenization*, <https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html>, Accessed: May 30, 2023, 2008.
- [8] *Stemming vs lemmatization in python*, <https://www.turing.com/kb/stemming-vs-lemmatization-in-python>, [Online; accessed March 30, 2023].
- [9] A. Mohta, A. Jain, A. Saluja, and S. Dahiya, “Pre-processing and emoji classification of whatsapp chats for sentiment analysis”, *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pp. 514–519, 2020.
- [10] J. Brownlee, *A gentle introduction to the bag-of-words model*, <https://machinelearningmastery.com/gentle-introduction-bag-words-model/>, Oct, 2017.
- [11] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space”, *arXiv preprint arXiv:1301.3781*, 2013.
- [12] M. Chablani, *Word2vec: Skip-gram model: Part 1*, <https://towardsdatascience.com/word2vec-skip-gram-model-part-1-intuition-78614e4d6e0b>, Jun, 2017.

- [13] J. Brownlee, *Encoder-decoder recurrent neural network models for neural machine translation*, <https://machinelearningmastery.com/encoder-decoder-recurrent-neural-network-models-neural-machine-translation/>, 2019.
- [14] J. Zhang, *Attention mechanism in deep learning (2017 edition)*, <https://blog.csdn.net/malefactor/article/details/78767781>, 2017.
- [15] J. Alammar, *The illustrated transformer*, <https://jalammar.github.io/illustrated-transformer/>, 2020.
- [16] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need", *Advances in neural information processing systems*, vol. 30, 2017.
- [17] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding", *arXiv preprint arXiv:1810.04805*, 2018.
- [18] J. Alammar, *The illustrated bert, elmo, and co. (how nlp cracked transfer learning)*, <http://jalammar.github.io/illustrated-bert/>, Jul, 2020.
- [19] P. Fan, *Tweet sentiment extraction: Why roberta and not bert?*, <https://www.kaggle.com/c/tweet-sentiment-extraction/discuss>, 2020.
- [20] Y. Liu, M. Ott, N. Goyal, *et al.*, "Roberta: A robustly optimized bert pretraining approach", *arXiv preprint arXiv:1907.11692*, 2019.
- [21] @pawangfg, *Overview of roberta model*, <https://www.geeksforgeeks.org/overview-of-roberta-model/>, Jan, 2023.
- [22] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding", *Advances in neural information processing systems*, vol. 32, 2019.
- [23] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context", *arXiv preprint arXiv:1901.02860*, 2019.
- [24] R. Horev, *Transformer-xl explained: Combining transformers and rnns into a state-of-the-art language model*, <https://towardsdatascience.com/transformer-xl-explained-combining-transformers-and-rnns-into-a-state-of-the-art-language-model-c0cfe9e5a924>, Jan. 2019.
- [25] T. Caselli, V. Basile, J. Mitrović, and M. Granitzer, "HateBERT: Retraining BERT for abusive language detection in English", in *Proceedings of the 5th Workshop on Online Abuse and Harms (WOAH 2021)*, Online: Association for Computational Linguistics, Aug. 2021. DOI: 10.18653/v1/2021.woah-1.3. [Online]. Available: <https://aclanthology.org/2021.woah-1.3>.
- [26] S. S. Aluru, B. Mathew, P. Saha, and A. Mukherjee, "Deep learning models for multilingual hate speech detection", *arXiv preprint arXiv:2004.06465*, 2020.
- [27] V. Basile, C. Bosco, E. Fersini, *et al.*, "Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter", in *Proceedings of the 13th International Workshop on Semantic Evaluation*, 2019, pp. 54–63.
- [28] J. C. Pereira-Kohatsu, L. Quijano-Sánchez, F. Liberatore, and J. Camacho-Collados, "Detecting and monitoring hate speech in twitter", *Sensors (Basel, Switzerland)*, vol. 19, no. 21, 2019.

- [29] H. N. B, *Confusion matrix, accuracy, precision, recall, f1 score*, <https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd>, 2019.
- [30] A. Luashchuk, *8 reasons why python is good for artificial intelligence and machine learning*, <https://towardsdatascience.com/8-reasons-why-python-is-good-for-artificial-intelligence-and-machine-learning-4a23f6bed2e6>, 2019.
- [31] M. I. González Sánchez, "Research and analysis of hate and other emotions in social media", 2022.
- [32] *BERT*, https://huggingface.co/docs/transformers/model_doc/bert.
- [33] *RoBERTa*, https://huggingface.co/docs/transformers/model_doc/roberta.
- [34] *XLNet*, https://huggingface.co/docs/transformers/model_doc/xlnet.
- [35] N. Kishore, *How to solve cuda out of memory error*, <https://medium.com/@snk.nitin/how-to-solve-cuda-out-of-memory-error-850bb247cfb2>, 2022.
- [36] *Hate-speech-CNERG/dehatebert-mono-spanish*, <https://huggingface.co/Hate-speech-CNERG/dehatebert-mono-spanish>.
- [37] *Exist2023 task 1 results leaderboard*, https://docs.google.com/spreadsheets/d/1ezHbWh9_Z15RotWdABJfs1IMM10W069b/edit?usp=sharing&oid=104632477484864942307&rtpof=true&sd=true, 2023.