uc3m | Universidad Carlos III de Madrid

Dual Bachelor in Data Science and Engineering and Telecommunication Technologies Engineering.
2024-2025

Bachelor Thesis

Enhancing Spanish Language Learning: An AI-Driven Approach to Grammar Error Correction Using Deep Learning Techniques

---

Daniel Toribio Bruna

Pedro Manuel Moreno Marcos

Leganés, June 2025

SUMMARY

In a world where communication is increasingly digital and global, the ability to write clearly and correctly is more important than ever. Grammar errors can impede understanding, leading to potential misunderstandings and miscommunications in both personal and professional contexts. For non-native speakers, mastering grammar is an essential component of language proficiency, directly impacting their educational and career opportunities. Thus, there is a growing need for effective tools that can assist language learners in achieving grammatical accuracy, enhancing not only individual communication skills but also fostering better cross-cultural interactions.

This thesis presents the development and implementation of an artificial intelligence tool designed to detect and correct grammatical errors in Spanish language students' essays, using deep learning techniques and large language models. The project draws on the COWS-L2H dataset, which comprises university-level student essays and their corrections. This research explores various model architectures, including LSTM Encoder-Decoders and transformers like GPT-2 and newer models such as Llama 3.2 and Gemma 3, to evaluate and optimize their performance in grammar correction tasks through fine-tuning. The best results are obtained after fine-tuning Llama 3.2 and Gemma 3, but to obtain a balance between performance and inference time the final model is the Llama 3.2 3B.

The primary outcome of this project is an accessible, user-friendly AI tool that significantly aids educators by reducing the time required to provide grammatical feedback, and empowers students to independently refine their writing skills. This work underscores the potential of AI to enhance educational outcomes.

Ultimately, the project's findings aim to foster higher educational efficiency and cognitive development through improved writing capabilities, contributing positively to the educational technology landscape.

Keywords: Deep Learning, Natural Language Processing, Grammar Error Correction, Artificial Intelligence, Data Privacy, Spanish Language, Large Language Models

RESUMEN

En un mundo donde la comunicación es cada vez más digital y global, la capacidad de escribir con claridad y correctamente es más importante que nunca. Los errores gramaticales pueden dificultar la comprensión, lo que lleva a posibles malentendidos y fallos en la comunicación, tanto en contextos personales como profesionales. Para los hablantes no nativos, dominar la gramática es un componente esencial de la competencia lingüística, impactando directamente en sus oportunidades educativas y laborales. Por ello, existe una necesidad creciente de herramientas eficaces que puedan ayudar a los estudiantes de idiomas a lograr precisión gramatical, mejorando no solo sus habilidades comunicativas individuales, sino también fomentando mejores interacciones interculturales.

Esta tesis presenta el desarrollo e implementación de una herramienta de inteligencia artificial diseñada para detectar y corregir errores gramaticales en ensayos de estudiantes de español, utilizando técnicas de aprendizaje profundo y modelos de lenguaje de gran escala. El proyecto se basa en el conjunto de datos COWS-L2H, que comprende ensayos de estudiantes universitarios y sus respectivas correcciones. Esta investigación explora diversas arquitecturas de modelos, incluyendo "*Encoder-Decoders*" LSTM y "*transformers*" como GPT-2, así como modelos más recientes como Llama 3.2 y Gemma 3, para evaluar y optimizar su desempeño en tareas de corrección gramatical mediante fine-tuning. Los mejores resultados se obtienen tras ajustar Llama 3.2 y Gemma 3, pero para lograr un equilibrio entre rendimiento y tiempo de inferencia, el modelo final seleccionado es Llama 3.2 3B.

El principal resultado de este proyecto es crear una herramienta de IA accesible y fácil de usar, que ayude significativamente a los docentes a reducir el tiempo necesario para proporcionar retroalimentación gramatical, y que empodera a los estudiantes para mejorar sus habilidades de escritura de forma autónoma. Este trabajo destaca el potencial de la inteligencia artificial para mejorar los resultados educativos.

En última instancia, los hallazgos del proyecto buscan fomentar una mayor eficiencia educativa y desarrollo cognitivo mediante la mejora de las capacidades de redacción, contribuyendo de manera positiva al panorama de la tecnología educativa.

Keywords: Aprendizaje profundo, Procesamiento del lenguaje natural, Corrección de errores gramaticales, Inteligencia artificial, Privacidad de los datos, Lengua española, Modelos de lenguaje de gran escala.

DEDICATION

With this work, I bring to a close a very important and meaningful period of my life. Despite the many years of hard work and sacrifice, the happy moments stand out, enriched by the many people who have influenced me, and to whom I am deeply grateful.

This chapter of my life would not have been possible without the friends I've made along the way. I extend my heartfelt thanks to them for all their support and assistance. I also fondly remember the teachers and classmates who have left their mark on me, guiding me through the countless decisions I've made over the past five years.

I am especially thankful to my tutor, Pedro, for his dedication and for helping me complete this work. Most importantly, I am grateful to him for trusting me and giving me the opportunity to collaborate with him on one of his projects.

Finally, I owe my deepest gratitude to my family for their unwavering support throughout the years. To my parents, thank you for your love, encouragement, and advice—for celebrating my triumphs and for guiding me to where I am today.

Thank you all.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Motivation of the project

According to the Ministerio de Asuntos Exteriores, Unión Europea y Cooperación [1], Spanish is one of the most widely spoken languages in the world, about a 7,5 % of the worldwide population speaks it, with millions of non-native speakers striving to achieve proficiency for personal, educational, and professional purposes. Despite its popularity, resources for language learners, especially those that provide immediate and personalized feedback, remain limited.

Secondly, educators face significant challenges in providing timely and individualized feedback to a large number of students [2]. Automated tools can assist educators by handling routine error detection tasks, allowing them to focus more on content and higher-level language skills rather than grammar correction. This not only enhances the learning experience but also optimizes the teaching workflow.

The project is also motivated by the advancements in artificial intelligence and natural language processing [3] [4] [5]. Recent developments in these fields have opened up new possibilities for creating sophisticated models that can understand and process human language with remarkable accuracy. Leveraging these technologies can lead to innovative educational tools that enhance the learning experience and reduce language barriers.

Lastly, personal data must be processed with the utmost privacy and security [6]. Many tools available today utilize user-provided data for purposes beyond the user's original intent, leading to significant data privacy concerns [7]. Ensuring that personal information is handled responsibly and transparently is essential to maintaining user trust and compliance with privacy regulations.

In summary, this project is driven by a combination of the global importance of Spanish as a second language, the educational need for automated and secure feedback tools, and the exciting potential of AI and NLP technologies to facilitate language learning. Through this project, we hope to contribute positively to the language learning community and educational technology field.

1.2. Objectives

The main objective of this project is to develop an AI tool capable of detecting and correcting grammatical errors in essays written by students learning Spanish in the Corpus of Written Spanish of L2 and Heritage Speakers (COWS-L2H) dataset. To achieve this, the project has been structured into several key sub-objectives.

A. Obtain Processed Dataset

1. Data Exploration

- Explore the COWS-L2H dataset to understand its structure, features, and content.
- Identify any missing, inconsistent, or anomalous data that needs addressing.

2. Data Cleaning

- Address data quality issues such as missing values, duplicates, and outliers.
- Ensure consistency in data formatting.

3. Data Preparation

- Split the dataset into training, validation, and test sets to facilitate model development and evaluation.
- Transform the data so it can be used as input in the models.

B. Create a Modeling Environment

1. Infrastructure Setup

- Set up the necessary hardware and software environment, including any cloud services or local resources needed for model training.
- Install relevant libraries and frameworks for model development.

2. Benchmark Establishment

- Create a simple baseline model to serve as a benchmark for future model improvements.

3. Model Development

- Design and implement various model architectures, experimenting with different algorithms and structures.
- Conduct fine-tuning to improve model performance.

4. Evaluation Metric Selection

- Identify and select appropriate evaluation metrics that align with the primary objectives of the project.

C. Analyze Results

1. Performance Evaluation

- Calculate and compare the selected evaluation metrics across all developed models.
- Identify the best-performing models and understand their strengths and limitations.

2. Model Interpretation

- Conduct detailed analyses of the top models to interpret their behavior and output in specific cases.

3. Selection and Justification

- Select the most suitable model based on a thorough evaluation of performance metrics and other relevant considerations.
- Provide a rationale for the chosen model, highlighting its advantages for achieving the project's primary objective.

4. User Interface Development

- Design and implement a graphical user interface to enhance model accessibility, usability and privacy.
- Ensure that the interface is intuitive, user-friendly, and facilitates interaction with the model outputs.

1.3. Structure of the report

The list below describes the structure of this report to facilitate comprehension.

- **Chapter 1: Introduction**
  Chapter 1 outlines the goals and underlying motivation of the project. It also provides an overview of the document's structure.

- **Chapter 2: State of the art**
  Chapter 2 includes a discussion of the Grammatical Error Correction (GEC) task, and an analysis of related works, providing the foundation and context for the research conducted in this thesis. At the end of the chapter, the contributions of this project are emphasized.

- **Chapter 3: Methodology**
  Chapter 3 describes the knowledge and methodologies required to preprocess data, build models, evaluate models, analyze results, and build the AI tool. A theoretical understanding of these methods is essential for appreciating the work.

- **Chapter 4: Data processing**
  Chapter 4 explains the steps necessary to preprocess the COWS-L2H dataset and analyze it to understand the data and prepare it for model training.

- **Chapter 5: Modeling, training and fine-tuning**
  Chapter 5 delineates the procedure for devising an environment for implementing, training and fine-tuning deep learning models to solve the GEC task.

- **Chapter 6: Inference and results**
  Chapter 6 describes the process of making inferences and evaluating models, and it analyzes the performance of each model. The top performing models are subjected to a deeper analysis, in order to select the final model.

- **Chapter 7: Graphical User Interface**
  Chapter 7 explains how the Graphical User Interface (GUI) was developed.

- **Chapter 8: Conclusions**
  Chapter 8 reviews the project's objectives and findings, highlighting the significance of the results and discussing potential directions for future research.

2. STATE OF THE ART

In this section, we will delve into the subject central to this project: Grammar Error Correction (GEC). We will also discuss and compare previous studies that share certain characteristics with our work, to identify similarities and differences. Notably, these studies were chosen because of their relevance to GEC tasks and their contributions to the fields of Natural Language Processing and Deep Learning, which are foundational to our project.

Towards the chapter's conclusion, we will emphasize the unique contributions this project makes in comparison to other research in the same domain.

2.1. Introduction to Grammar Error Correction

Writing is a complex skill, and achieving grammatical and comprehensible text can be particularly challenging for non-native language users . While even native speakers make occasional minor mistakes with punctuation, spelling, or word choice, non-native writers often struggle with a wider range of grammatical infelicities. The field of Natural Language Processing (NLP) has a long history of addressing the problem of ill-formed input, dating back to the 1980s, as downstream tasks like parsing typically required grammatical input [8]. However, practical applications designed to significantly assist non-native writers only began to emerge in the 2000s, initially relying on rule-based systems that applied hand-coded mal-rules derived from robust parsers, like Microsoft's ESL Assistant [9].

According to Bryant [10], "Grammatical Error Correction (GEC) is the task of automatically identifying and correcting errors in text. While the term Grammatical is commonly used, it is somewhat a misnomer, as the task often extends beyond strictly grammatical issues to include errors like spelling and discourse-level mistakes; Language Error Correction might be a more accurate description". The definition of what constitutes an error and its correction is inherently complex. Errors can vary in scope and complexity, and there are often multiple valid ways to correct a single error. Factors such as the writer's communicative intention, the wider context of the text (e.g., sentence vs. document level), dialect, and genre can all influence whether something is perceived as an error or what the appropriate correction should be [10]. Most GEC research has historically framed the task at the isolated sentence level, which can fail to capture errors requiring broader context [11].

The GEC field saw a significant shift towards data-driven approaches starting in the mid-2000s, utilizing supervised machine learning models trained on annotated corpora of errorful text paired with corrections [12]. With the creation of test datasets, like the First Certificate in English (FCE) corpus a clear turn from rule-based methods to those

dependent on data occurred [13] [14].

As said by Olsson and Sahkgren [15], like most NLP tasks, modern GEC systems are fundamentally dependent on data. High-quality annotated data is paramount for training state-of-the-art neural models, which often require vast amounts of text. However, human annotation is a slow and labor-intensive process, making high-quality GEC data relatively scarce. This data sparsity has been a major driving force behind research into techniques for generating artificial, or synthetic, errorful data [16] [17]. Annotation guidelines are crucial for ensuring data quality, particularly in defining the distinction between minimal corrections, which make the fewest changes necessary for grammatical, and fluent corrections, which may involve more extensive rewrites for naturalness [10].

Several human-annotated datasets have become standard benchmarks for training and evaluating English GEC systems [18]:

- First Certificate in English (FCE): A public subset of the Cambridge Learner Corpus, consisting of essays by intermediate (B1-B2) non-native speakers.
- National University of Singapore Corpus of Learner English (NUCLE) / CoNLL test sets: Comprises essays by advanced (C1) non-native speakers [19].
- Johns Hopkins Fluency-Extended GUG corpus (JFLEG): A smaller dataset focusing specifically on fluent corrections, with sentences extracted from learner essays and annotated by crowdsourced workers [20].

While English GEC has received the most attention, corpora for other languages, such as Arabic, Chinese, Czech, German, Russian, and Ukrainian, are increasingly being developed and released, facilitating research into multilingual GEC [10]. COWS-L2H is a prominent publicly available Spanish dataset commonly used for research in the field of grammatical error correction [21].

Measuring the performance of GEC systems is a critical aspect of the field. This typically involves comparing the system's output to human-annotated corrections using reference-based evaluation metrics [22]. The most commonly used benchmark metrics include:

- MaxMatch (M2) scorer: An edit-based F0.5 score that weights precision higher than recall [23].
- ERRANT scorer: provides fine-grained analysis by classifying edits into detailed error types [24].
- GLEU: Inspired by the BLEU score from machine translation [25], GLEU does not require explicit edit annotations and is typically used for evaluating systems on the JFLEG dataset [26].

Despite these established metrics, robust evaluation remains an unsolved problem, with challenges in correlating metric scores with subjective human judgments and adequately accounting for multiple valid corrections or untokenized text [24].

The landscape of GEC approaches has evolved significantly. Early classifier-based systems targeted specific error types [27] [28] [29] [30]. This was followed by statistical machine translation (SMT) approaches, which viewed GEC as a monolingual translation task and could handle multiple error types simultaneously [12]. With advancements in deep learning, neural machine translation (NMT), particularly Transformer-based architectures [31], became the dominant state-of-the-art approach [32].

More recently, the advent of large pre-trained language models (LLMs) has influenced GEC [33]. Directly fine-tuning these large pre-trained models with GEC parallel data has been shown to achieve performance comparable to, and even state-of-the-art results [34]. These fine-tuned models are powerful components in modern GEC systems. While very large generative LLMs can potentially be used in zero-shot or few-shot settings as error correctors [35], particularly for multilingual GEC without explicit task training [36], formal benchmarking against standard GEC test sets was not widely published yet.

Bryant and Wang [10] [22] define the limitations, future work and challenges of GEC field with including improving domain generalization, personalizing systems for different learners, generating explanatory feedback, effectively incorporating document-level context, and further refining evaluation methods.

To compare this project with existing works and current research in the field, it is essential to review articles and commercial tools addressing similar topics.

For Grammar Error Correction (GEC) in Spanish texts, one of the most relevant tools is LanguageTool languagetool.org/es. This tool is capable of correcting not only grammatical errors but also spelling and stylistic issues. It offers a free version and a premium version that provides comprehensive correction of all errors in the submitted text. While most of its corrections are based on explicit rules, LanguageTool also incorporates AI models for its corrections. According to their privacy policy, the text entered into the application may be used to train their AI models.

Another tool with similar capabilities as LanguageTool is Grammarly grammarly.com . In addition to correcting grammar errors, Grammarly excels in enhancing text fluency and offers features such as plagiarism detection and identifying AI-generated text, among others. However, it is important to note that Grammarly currently supports only the English language.

While these tools are highly effective in practical applications, evaluating the performance of underlying grammatical error correction models requires standardized benchmarks and metrics. Table 2.1 presents models that achieve top performance on the JFLEG benchmark, evaluated using the GLEU score [20], [37]. These results are included because GLEU will also be used as the evaluation metric in this project, allowing for

meaningful performance comparisons with our models, even though a different dataset is used and their corrections were made just on sentences instead of whole texts.

TABLE 2.1. COMPARISON OF GLEU SCORES ACROSS DIFFERENT TOP PERFORMING SYSTEMS IN THE JFLEG DATASET.

System	GLEU
Coyne et al. (2023) [38]	65.02
Ge et al. (2018) [39]	62.42
Liu et al. (2021) [40]	61.61
Grundkiewicz and Junczys-Dowmunt (2018) [41]	61.50

The highest-performing models in terms of the GLEU score have been achieved using deep learning techniques and LLMs. Notably, the top system leveraged GPT-3.5 and GPT-4 for text correction [38], attaining the best GLEU scores, demonstrating the potential of LLMs for these type of tasks. In other leading cases, different approaches were utilized, employing neural network architectures such as Recurrent Neural Networks (RNNs) [39] [40], and transformers [41]. The use of RNNs with an Encoder-Decoder architecture are commonly used in the GEC tasks for sentence-level corrections motivated by the good results obtained by Google in a seq2seq task, more precisely, translation task [42].

2.2. Deep Learning in Natural Language Processing

2.2.1. General Overview of Deep Learning

Deep Learning (DL) is a subfield of machine learning that models high-level abstractions in data through the use of artificial neural networks with multiple layers. Inspired by the structure and function of the human brain, DL models are designed to automatically learn hierarchical representations of data. While the concept of neural networks dates back to the 1940s, DL gained substantial popularity in the 2010s due to increased computational power, availability of large datasets, and algorithmic advancements like better activation functions and optimization techniques [43].

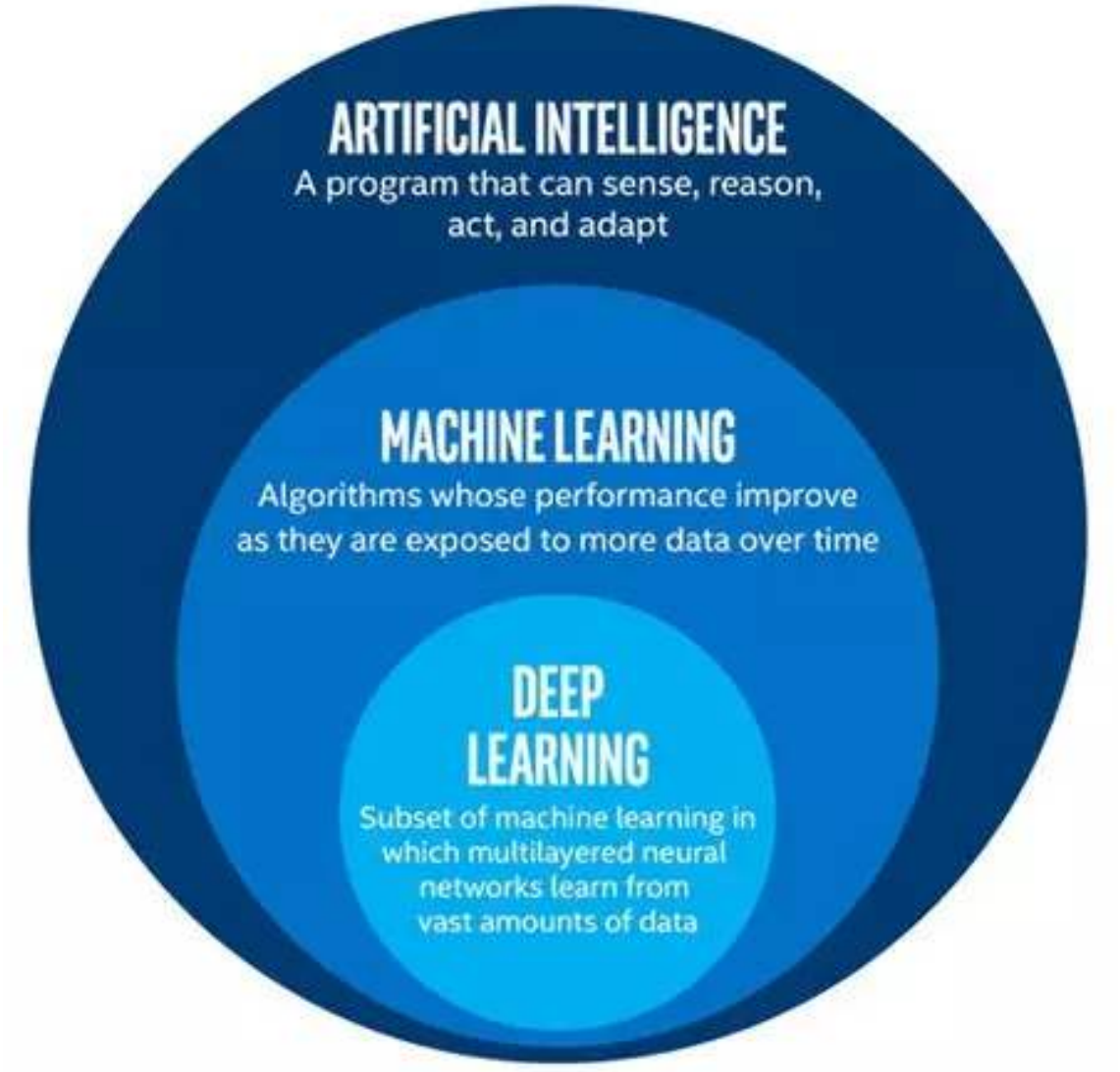

Fig. 2.1. Overview of AI fields [44].

In traditional machine learning, feature engineering was a manual process that relied heavily on human expertise. By contrast, DL models are capable of learning features directly from raw data, reducing the need for task-specific engineering. These capabilities make DL particularly suitable for tasks involving unstructured data such as images, audio, and text.

In the context of text processing, DL has revolutionized the field of Natural Language Processing (NLP) by significantly improving performance in tasks such as machine translation, sentiment analysis, named entity recognition, and more recently, grammar error correction.

2.2.2. Foundational Deep Learning Architectures in NLP

Early applications of DL in NLP leveraged Recurrent Neural Networks (RNNs), which are suited for sequential data due to their internal memory. Variants such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRUs) addressed the vanishing gradient problem and allowed learning of long-term dependencies in text. These models became the foundation for early neural machine translation and text correction systems [10].

Later, Convolutional Neural Networks (CNNs)—traditionally used in computer vision—were adapted for NLP tasks. They demonstrated success in text classification and sentence modeling by capturing local features in text sequences [45].

The most significant leap in NLP came with the introduction of the Transformer architecture [46], which entirely replaced recurrence with a mechanism known as self-attention. Transformers allow parallel processing of input tokens, which are units into which text is broken down, such as words, subwords, or characters, and better capture long-range dependencies. This model architecture forms the basis for almost all modern language models, including GPT. The original architecture of transformers can be seen in Figure 2.2.

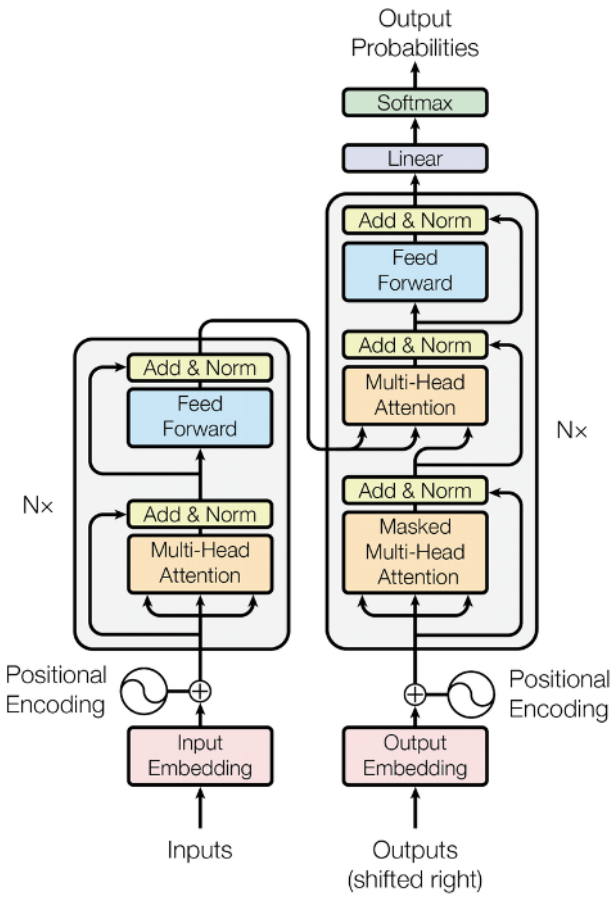

Fig. 2.2. Original architecture of a transformer [46].

2.3. Pre-trained Language Models and Efficient Fine-tuning

Pre-trained Language Models have become the foundation of modern NLP. These models are trained on massive text corpora to learn statistical representations of language. Once trained, PLMs can be fine-tuned on downstream tasks such as sentiment analysis, translation, and grammar error correction.

The key advantages of Pre-trained Language Models include:

- Transfer learning: knowledge acquired during pretraining is transferable to other tasks.
- Reduced data dependency: fine-tuning requires much less labeled data than training from scratch.
- Improved generalization: models encode a rich understanding of syntax, semantics, and world knowledge.

Fine-tuning refers to the process of adapting a pre-trained model to a specific task or domain. In the standard full fine-tuning approach, all weights of the pre-trained model are updated using labeled task-specific data. While this often yields strong performance, it is computationally expensive and memory-intensive, especially with large models containing billions of parameters [47].

Pre-training and fine-tuning these models rely on the principle of self-supervised learning. This technique enables models to learn from raw text without requiring manually annotated data. The two main strategies used are Causal Language Modeling (CLM) and Masked Language Modeling (MLM) [48].

In CLM, the model is trained to predict the next word in a sequence given the previous ones, maximizing the likelihood of the next token. The loss function that is minimize can be seen in equation 2.1. This encourages the model to learn the structure and flow of language over time. GPT models are based on this approach and they commonly only use the decoder part [47].

$$\mathcal{L}_{\text{CLM}} = -\sum_{t=1}^{T} \log P_\theta(x_t \mid x_{<t}) \tag{2.1}$$

$P_\theta(\cdot)$: Model's predicted probability distribution parameterized by θ

θ : Model parameters (weights)

T : Length of the token sequence

x_t : Token at position t

$x_{<t}$: All tokens before position t, i.e., $(x_1, x_2, \ldots, x_{t-1})$

In contrast, MLM involves randomly masking some words in a sentence and training the model to predict them. This helps the model learn bidirectional context, meaning it considers both the left and right sides of a masked token. In this case we minimize the loss in equation 2.2 BERT is a prominent example of a model trained using MLM [48].

$$\mathcal{L}_{\text{MLM}} = -\sum_{t \in \mathcal{M}} \log P_\theta(x_t \mid x_{\setminus\{x_t\}}) \tag{2.2}$$

$\mathcal{M} \subseteq \{1, \ldots, T\}$: Set of indices corresponding to masked tokens

$x_{\setminus\{x_t\}}$: Input sequence with token x_t masked (hidden)

$P_\theta(\cdot), x_t, T$: Same as above

As Chip Huyen said, while annotated data is not required for pre-training, curated datasets are still necessary during the fine-tuning phase to adapt the model to specific tasks because the model learn to respond in a certain way based on the input tokens 2.3.

$$\mathcal{L}_{\text{fine-tune}} = -\sum_{t=1}^{T} \log P_\theta(y_t \mid y_{<t}, x) \tag{2.3}$$

$y = (y_1, y_2, \ldots, y_T)$: Target output token sequence

y_t : Target token at position t

$y_{<t}$: Tokens before position t in the target sequence

x : Conditioning input or prompt (e.g., instruction or context)

$P_\theta(\cdot), \theta, T$: Same as above

The challenges of full fine-tuning include:

- High memory usage.
- Prolonged training times.
- The need to store a full copy of the fine-tuned model for each task.

To address these issues, the research community has proposed parameter-efficient fine-tuning (PEFT) techniques, including Adapter layers, Prefix-Tuning, LoRA, and most recently, QLoRA.

QLoRA (Quantized Low-Rank Adaptation) [49] is a recent and highly efficient fine-tuning method that enables the adaptation of large language models using modest computational resources.

QLoRA builds upon the original LoRA (Low-Rank Adaptation) method [50], which fine-tunes a model by inserting trainable low-rank matrices into each layer of the transformer, keeping the original weights frozen. A comparison between full fine-tuning, LoRA and QLoRA is shown in Figure 2.3.

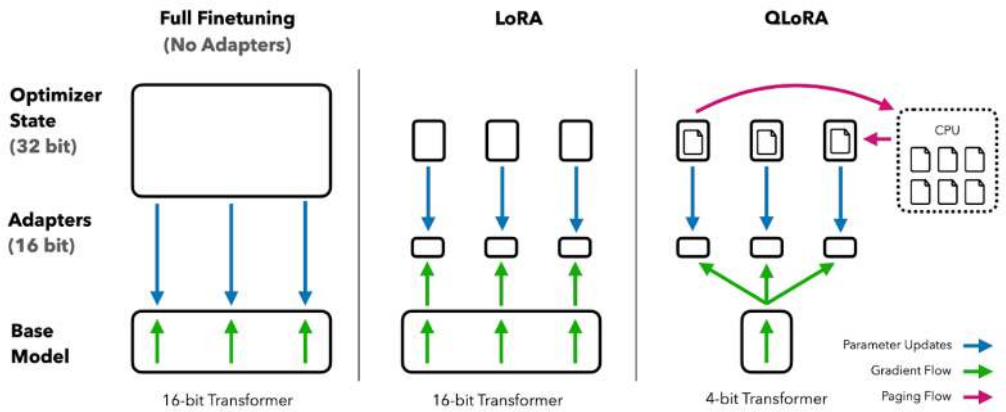

Fig. 2.3. Comparison of fine-tuning techniques [49].

QLoRA introduces three additional innovations:

- 4-bit quantization of the base model: The model is stored in a highly memory-efficient 4-bit format using Double Quantization (a nested quantization scheme that retains model quality).
- LoRA layers on top of quantized weights: Instead of updating the full model, only the inserted low-rank adapters are trained.
- Paging: a technique used to offload infrequently accessed data (like optimizer states) from GPU memory to CPU RAM, and load them back ("page in") when needed. This is critical when fine-tuning large language models on hardware with limited VRAM.

2.4. Contribution of the project

This project distinguishes itself from others through its specific objectives. Most research in the area of Grammar Error Correction has primarily concentrated on sentence-level corrections. However, this project takes a broader approach by addressing corrections at the document level, recognizing that the context provided by the entire text is crucial for accurately correcting certain types of errors, especially those that depend on the context of preceding sentences. By considering the full text, this project aims to enhance the overall accuracy and coherence of the corrections, ensuring that context-dependent errors are effectively identified and corrected.

Furthermore, whereas many existing systems, particularly those implemented for the JFLEG dataset, primarily focus on solving the Grammar Error Correction task for English texts, our project shifts the focus to Spanish texts. By doing so, it addresses a significant gap in the availability of robust GEC tools for Spanish. We employ the most advanced and promising approaches in the field to ensure that the corrections are as accurate and effective as possible.

While tools like LanguageTool offer grammar, spelling, and style corrections, they often require an Internet connection and may involve data privacy concerns, as user data can be used to train AI models. In contrast, our project prioritizes user data privacy by ensuring that all data remains under the user's control. This means that users do not have to worry about their text being used for external purposes, and they can benefit from a tool that provides all corrections at no cost.

Additionally, unlike other tools that necessitate an online connection, this project is designed to function offline. By enabling offline usage, it offers flexibility and accessibility in environments where Internet access may be limited or non-existent.

This comprehensive approach not only meets the needs of Spanish-speaking users but also reflects a commitment to delivering a user-centric, privacy-conscious, and technologically advanced solution in the realm of grammar error correction in entire texts.

3. METHODOLOGY

This chapter presents the fundamental concepts and techniques that form the basis for understanding the work carried out in this study. It includes in-depth explanations of the approaches used in data exploration, cleaning, and visualization, as well as the processes involved in designing, training, fine-tuning, and evaluating the models. A solid grasp of these technical elements is essential for following the analyses and discussions developed in the subsequent chapters. In Figure 3.1 is shown a general overview of the project workflow.

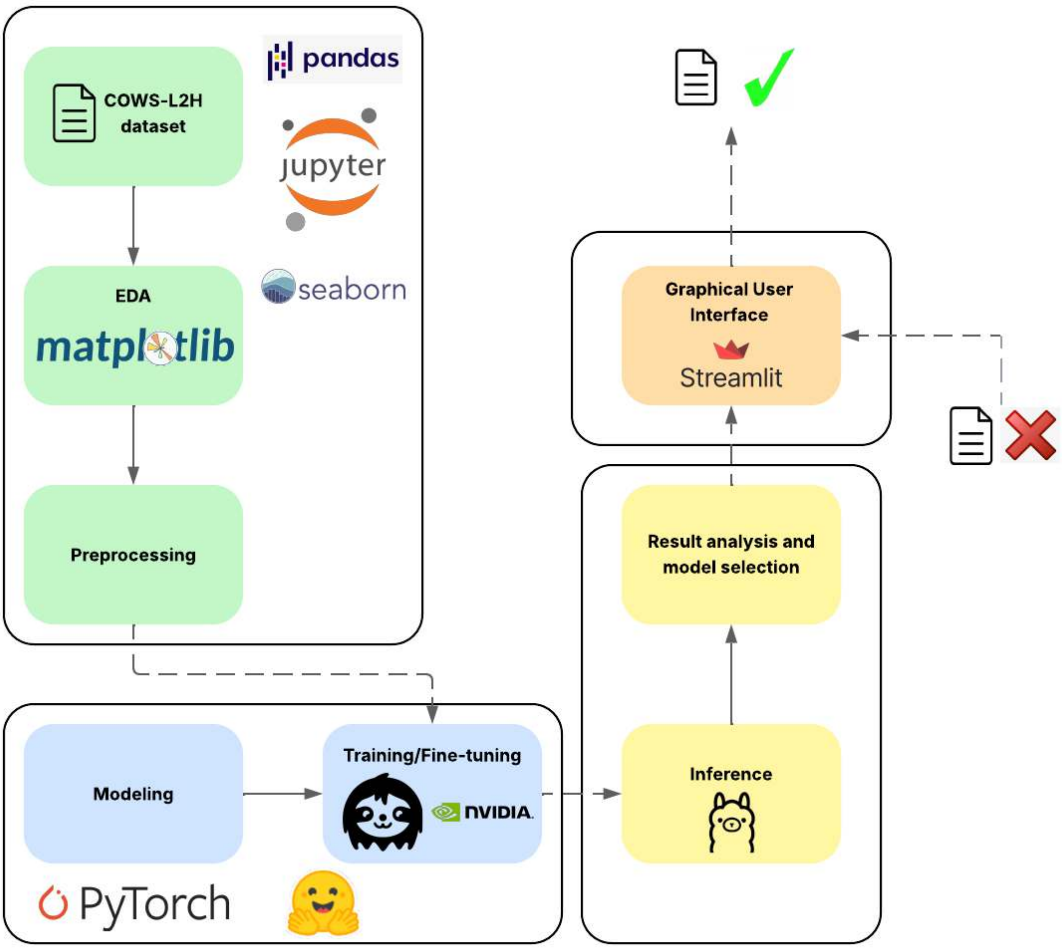

Fig. 3.1. Overview of the project workflow.

3.1. Data processing

The procedures for data manipulation are centered around the Corpus of Written Spanish of L2 and Heritage Speakers (COWS-L2H) and its specific data characteristics. For

the tasks of cleaning and analyzing this data, Python programming language and Python Notebooks are utilized which offer the essential tools needed for effective data processing.

3.1.1. Data and characteristics

The Corpus of Written Spanish of L2 and Heritage Speakers (COWS-L2H) [1] [21] is a unique anonymized dataset comprising short essays from students enrolled in Spanish courses in the University of California. What makes this dataset unique is that it not only includes student essays but also their corresponding corrections. This feature allows for training models to perform grammatical corrections in context, rather than in isolation. The dataset encompasses 5,382 essays contributed by 1,370 distinct students across 10 courses of varying levels. These essays, written on eight different topics, range in length from a minimum of 250 words to a maximum of 500 words.

Researchers can access the dataset for free via its GitHub repository ucdaviscl/-cowsl2h, where it is available for download as a folder containing 29 CSV files. Each file represents a collection of essays focused on the same topic during a particular quarter and includes information about the students and their essay corrections. The instructions given to the students to write essays in each of the topics are:

- famous: Write a text in Spanish about the following subject: "a famous person"
- vacation: Write a text in Spanish about the following subject: "your perfect vacation plan"
- special: Write a text in Spanish about the following subject: "a special person in your life"
- terrible: Write a text about the following subject: "a terrible story"
- yourself: Write a text describing yourself.
- beautiful: Write a text in Spanish about the following subject: "a beautiful story"
- place you dislike: Write a text in Spanish about the following subject: "a place you dislike"
- Chaplin: Watch a brief clip from Chaplin's 'The Kid'. Write a text in Spanish describing the scene.

All essays have the same number of features, 26 in total. The attributes related to the student are:

- id: this column contains a number that uniquely identifies the student.

[1] https://github.com/ucdaviscl/cowsl2h

- prompt: this column identifies the topic for which the essay was written. It can take a value from the ones above.
- quarter: this column informs about in which quarter and year the essay was written. The format is first a letter, F for Fall, W for Winter and S for Spring, and then the last to digits of a year.
- course: this column contains the course where the student that has written the essay is enrolled. The courses are SPA1-3 for introductory level, SPA21-22 for intermediate level, SPA23-24 for high level, SPA31-33 for Heritage Learner courses.
- age: it informs about the age of the student that has written the essay.
- gender: this columns determines the gender of the student who has written the essay between male or female (binary).
- l1 language: it gives information about the first language spoken by the student.
- other l1 language(s): it tells if the student has another first language and if it is the case it tells what language is.
- language(s) used at home: this column identifies which language uses the student at home
- language(s) studied: this column informs about the experience of the student learning other languages.
- listening comprehension: it contains a number in the range 1, minimum, to 5, maximum, about the self-perceived level in listening by the student.
- reading comprehension: it contains a number in the range 1, minimum, to 5, maximum, about the self-perceived level in reading by the student.
- speaking ability: it contains a number in the range 1, minimum, to 5, maximum, about the self-perceived level in speaking by the student.
- writing ability: it contains a number in the range 1, minimum, to 5, maximum, about the self-perceived level in writing by the student.
- study abroad: it has the answer to the question "Have you ever lived in a Spanish-speaking country?".

An essay could be corrected by two different annotators, that is the reason for adding a number at the end of the feature names that involve the corrections of the essay. These features are used to create training and evaluation data, and are:

- essay: it contains the plain text written by the student.

- a personal annotator1: this column has the essay corrected only on presence or absence of subject pronouns or articles.
- a personal annotator2: same type of corrections as previous column but made by another annotator.
- gender-number annotator1: this column has the essay corrected only on gender and number errors.
- gender-number annotator2: same type of corrections as previous column but made by another annotator.
- verbs annotation ann1: this column contains the essay corrected only on verb errors.
- verbs annotation ann2: same type of corrections as previous column but made by another annotator.
- updated annotation ann1: it contains the essay that has been revised according to various guidelines, ensuring that any remaining errors, not previously addressed, are now corrected.
- updated annotation ann2: same type of corrections as previous column but made by another annotator.
- corrected1: it contains the essay corrected in all errors.
- corrected2: same type of corrections as previous column but made by another annotator.

It is important to know that annotators were told to make corrections in a way that only evident spelling, grammatical, and lexical errors are corrected and they avoid changing the style even if the result was not completely natural for a native speaker.

3.1.2. Development tools

Python[2] is an interpreted, multi-paradigm, and dynamic programming language, meaning its code is executed directly without prior compilation. This characteristic facilitates rapid iteration, making Python particularly beneficial for data analysis and data science, which is why it was chosen for the development of this project. With its dynamic typing and automatic memory management, Python removes the need to explicitly define variable types or manually handle memory allocation, streamlining the coding process and reducing potential errors.

Another strength of Python is its extensive number of libraries that significantly enhance its capabilities. Notable libraries include NumPy, Pandas, Matplotlib, Scikit-learn,

[2] https://www.python.org/doc/

PyTorch, Unsloth, Streamlit and the Transformers library from Hugging Face[3]. These libraries facilitate data manipulation, the development, training and fine-tuning of machine learning and deep learning models, the creation of insightful data visualizations and building data apps. Each library contributes unique features, making Python an indispensable tool for data scientists.

To manage data in Python, the Pandas library was utilized. Pandas[4] is a free, open-source software library built on top of NumPy[5], making it highly effective for data manipulation and analysis. It offers fast, flexible, and powerful data structures designed to handle labeled and ordered data efficiently. The key data structure employed in this project is the Pandas DataFrame, a versatile two-dimensional structure that can handle columns of varying data types, facilitating complex data handling and analysis tasks.

Two development environments were utilized in this project: Visual Studio Code[6] and Colab Notebooks[7].

Visual Studio Code is a free, open-source code editor developed by Microsoft. It features syntax highlighting, version control integration, and a built-in terminal, making it ideal for running scripts locally, especially those requiring extended execution times.

Colab Notebooks are a free, cloud-based Jupyter notebook environment provided by Google. They enable users to write and execute Python code directly in the browser, with free access to powerful computing resources, including GPUs and TPUs, which are essential for training models.

Jupyter Notebooks[8] are essential for Exploratory Data Analysis (EDA) due to several key features:

- Interactive coding: The notebook structure allows for cells to be executed individually, with results displayed immediately, facilitating an exploratory coding style. This is enabled by the Jupyter Kernel, which executes code in each cell and returns the results, enabling dynamic and iterative analysis.

- Visualizations: Jupyter Notebooks provide access to data visualizations and other graphical outputs using libraries like Matplotlib. These visualizations are displayed inline, allowing for seamless integration of code and graphical representations, which helps in understanding data patterns and insights more clearly.

- Markdown support: Jupyter Notebooks support Markdown, enabling users to combine code, computation outputs, graphical visualizations, and explanatory text

[3] https://huggingface.co/docs/transformers/en/index
[4] https://pandas.pydata.org
[5] https://numpy.org
[6] https://code.visualstudio.com
[7] https://colab.google
[8] https://docs.jupyter.org/en/latest/start/index.html

within a single document. This allows for live code execution, documentation, and result presentation all in one place.

3.2. Exploratory data analysis techniques and tools

Exploratory Data Analysis (EDA) is crucial for understanding and extracting valuable insights from a dataset. Serving as a preliminary step in the research process, EDA facilitates the exploration of data to identify patterns, trends, and relationships.A variety of statistical and visualization techniques are applied in this section to obtain a more thorough understanding of the variables involved. By utilizing descriptive statistics, graphical representations, and data visualization tools, EDA provides a clear and comprehensive summary of the dataset's insights, helping inform subsequent analytical steps and decision-making.

3.2.1. Visualization tools

For the creation of graphs, the library Matlplotlib[9] and Seaborn[10] have been used. To effectively visualize data and achieve a better understanding of the dataset's nature, various types of plots were utilized. The primary graphs used in the project include:

- Bar plots: These are utilized to display the proportions of different variables, such as the proportion of essays for each theme.
- Scatter plots: These plots help illustrate the relationship between two numeric variables. For example, they are used to assess whether the length of a corrected essay differs significantly from the original essay.
- Box plots: They shows the distribution of quantitative data and are useful to identify outliers. In this project box plots are used to visualize the age distribution.
- Line graphs: They displays a sequence of data points that are joined with straight lines. In our case we will use them to visualize the training loss over epochs and steps.

3.3. Analytical methods

This section offers a comprehensive overview of the approaches and techniques used to build, train, fine-tune and evaluate the deep learning models in the project. This information serves as a crucial foundation for the research process by outlining the strategies employed to explore and investigate the research problem. Through a systematic and

[9] https://matplotlib.org
[10] https://seaborn.pydata.org

rigorous methodology, these analytical methods ensure the reliability, validity, and credibility of the findings. The section covers the selection of appropriate architectures and models, the application of training and fine-tuning tools and techniques, and the procedures used to analyze and interpret the results.

3.3.1. Deep learning models

Deep learning models have significantly advanced the field of natural language processing (NLP), enabling sophisticated analysis and understanding of text data. These models utilize complex neural network architectures to automatically learn intricate patterns and semantic relationships within large textual datasets, performing well on sequence-to-sequence (seq2seq) tasks such as GEC. For this project, we employ two primary types of models: LSTM-based Encoder-Decoder architectures and pre-trained Large Language Models (LLMs).

LSTM Encoder-Decoder

The LSTM Encoder-Decoder architecture is particularly effective for seq2seq tasks [51], making it well-suited for GEC. The encoder LSTM processes an input sequence, encoding the context into a fixed-size vector, while the decoder LSTM generates the corrected output sequence from this encoded context. This architecture is known for handling variable-length sequences effectively and maintaining long-range dependencies, crucial for accurately correcting grammar.

Large Language Models

Large Language Models, have recently revolutionized the field of NLP by leveraging transformer-based architectures capable of processing extensive text datasets. These models excel at understanding context and generating human-like text, enabling their application in tasks like grammar correction. The selection of LLMs for this project was based on several key criteria:

- Language: LLMs can be pre-trained in data in different languages in order to have a good performance in those languages. In this project the texts are in Spanish, meaning that the LLMs have to be pre-trained in Spanish texts or at least be a multilingual model which supports Spanish.
- Size: In LLMs the size of a model is given by the number of parameters. This feature is responsible of the computational efficiency and the performance. We opted to use relatively small models, such as those with around 4 billion parameters, to balance computational efficiency with performance. Smaller models require

less computational power and memory, facilitating faster experimentation and iteration during the research process. They are also more practical for deployment in environments with limited resources, making them well-suited for real-world applications where computational constraints are a consideration.

- Accessibility: Prioritizing open-source models was crucial to ensure transparency, reproducibility, and ease of customization. Open-source models can be more readily adapted to specific tasks, allowing us to fine-tune them effectively for grammar error correction without the limitations often associated with proprietary models.

- Context length: Since we are dealing with long essays, it is ideal to use models capable of processing the entire text simultaneously. This ensures that no contextual information is lost.

The selected language models for this project are GPT-2 [52] by OpenAI, LLaMA 3.2 [53] by Meta AI, and Gemma 3 [54] by Google DeepMind. Each of these models is open-source, and their size specifications are detailed in Table 3.1.

For comparison, the table also includes the LSTM Encoder-Decoder model. It is important to note that Q8_0 quantization is applied to the Llama 3.2 and Gemma 3 models, which compresses their weights into 8-bit unsigned fixed-point values, significantly reducing the memory required and allowing for their use in almost all devices with GPU access.

TABLE 3.1. SIZE CHARACTERISTICS OF THE SELECTED MODELS

Model	# of Parameters	Context Length
LSTM	69.64 million	592
GPT-2	124 million	1024
LLama 3.2	3 billion	128k
Gemma 3	4 billion	128k

GPT-2 Model Architecture

The GPT-2 model consists of several key components:

- **Transformer Blocks:** The architecture employs a series of transformer blocks, each comprising a multi-head self-attention mechanism followed by position-wise feedforward layers. These blocks enable the model to attend to different parts of the input sequence, effectively capturing long-range dependencies.

- **Self-Attention Mechanism:** GPT-2 uses a self-attention mechanism that allows the model to weigh the significance of different words in the input sequence relative to each other, facilitating context-aware predictions.

- **Layer Normalization and Residual Connections:** Each transformer block includes layer normalization and residual connections, which stabilize and improve model convergence during training.
- **Embedding Layers:** The model employs token and positional embeddings to convert input tokens into continuous vector representations. Positional embeddings are crucial as they provide information about the position of each token in the sequence, compensating for the lack of inherent ordering in transformer architectures.

The architecture can be seen in Figure 3.2.

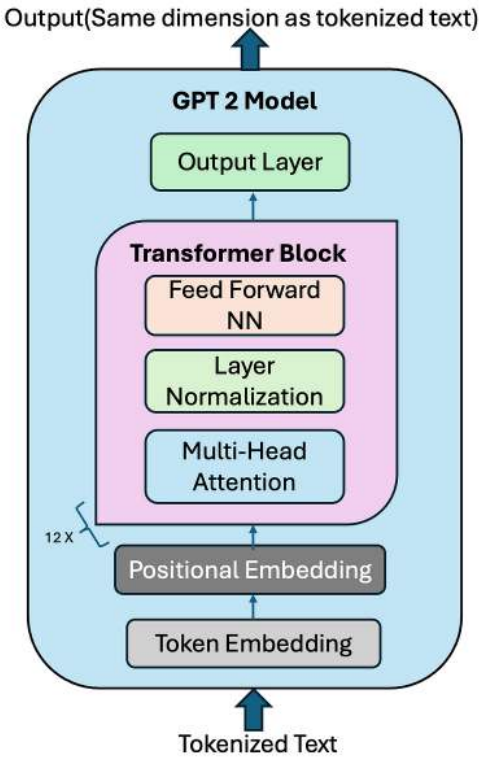

Fig. 3.2. GPT-2 Model Architecture [55].

Note that, in the case of GPT-2, we used the 124 million parameter version instead of the largest configuration with 1.5 billion parameters. This choice was made due to computational resource constraints, which limit the feasibility of performing instruction tuning on larger models.

3.3.2. Training techniques and tools

When it comes to specializing an AI model, there are three primary approaches: training, fine-tuning, and prompting. Training involves building a model from scratch or with a pre-existing architecture, feeding it large datasets to learn patterns and tasks specific to the desired outcome. Fine-tuning, on the other hand, starts with a pre-trained model and

adjusts its parameters by training on a smaller, task-specific dataset, enabling the model to adapt its generalized knowledge to a particular use-case with less data and computational effort than training from scratch. Lastly, prompting is a technique often used with language models where the model is given specific instructions or examples within the input to guide its responses towards a particular task or context. This method leverages the model's existing capabilities without the need to alter its underlying parameters, making it a quick and flexible way to specialize outputs for various applications.

For training the LSTM Encoder-Decoder model and performing instruction tuning of GPT-2, we used PyTorch[11]. It is important to note that GPT-2 was originally designed for text generation; therefore, it requires a specialized fine-tuning approach to enable it to follow explicit instructions, such as: "Correct the following text grammatically: text with errors." Instruction tuning involves providing the model with pairs of instructions and corresponding target outputs (i.e., the grammatically corrected text) during the fine-tuning process.

As we are working with pre-trained LLMs, we have to fine-tuned them with our data to obtain better results. In order to do so, the Unsloth[12] library is used. Unsloth make use of QLoRA to optimize the fine-tuning, achieving 2x faster with 50% less memory [56].

In terms of hardware, GPUs play a crucial role in accelerating both training and fine-tuning processes. For this project, given the reliance on the Unsloth library, it is essential that the GPUs are compatible with CUDA. Therefore, we utilize the NVIDIA T4 GPU[13], accessible via Google Colab, as it meets these requirements. The inference is done in local with Ollama[14] and making use of the M2 chip of a MacBook Pro.

3.3.3. Evaluation metrics

To assess the performance of the models, evaluation metrics are essential. This project tackles a sequence-to-sequence problem, specifically focusing on grammar correction. In the field of Grammatical Error Correction (GEC), the most commonly used evaluation metrics are the MaxMatch (M2) score [23], ERRANT [24], and GLEU [57]. However, given that our data only consists of the original essays and their corrections, both human and model-generated, we are limited to calculating only the GLEU metric for this project as the others requires a special annotation of the edits in the essays.

The underlying concept of GLEU is to offer a reward for hypothesis n-grams that align with the reference but diverge from the original text, while penalizing those aligning with the original text but not the reference. The GLEU metric is calculated as follows: Consider a set of original sentences $O = \{o_1, \ldots, o_k\}$ with their corresponding hypothesis $H = \{h_1, \ldots, h_k\}$ and reference sentences $R = \{r_1, \ldots, r_k\}$. Each original, hypothesis,

[11] https://pytorch.org
[12] https://docs.unsloth.ai
[13] https://www.nvidia.com/es-es/data-center/tesla-t4/
[14] https://ollama.com/

and reference sentence, denoted as o_i, h_i, and r_i, consists of sequences of n-grams with lengths $n = \{1, 2, \ldots, N\}$ (with $N = 4$ as the default for GLEU). These sequences are then utilized to determine a precision factor p_n (illustrated in Equation (3.1)), which considers the reward or penalty for n-gram matches.

$$p_n = \frac{\sum_{i=1}^{|H|} \left(\sum_{g \in \{h_i \cap r_i\}} \text{count}_{h_i,r_i}(g) - \sum_{g \in \{h_i \cap o_i\}} \max[0, \text{count}_{h_i,o_i}(g) - \text{count}_{h_i,r_i}(g)] \right)}{\sum_{i=1}^{|H|} \sum_{g \in \{h_i\}} \text{count}_{h_i}(g)} \tag{3.1}$$

$$\text{count}_a(g) = \#\text{ appearances of n-gram } g \text{ in } a$$
$$\text{count}_{a,b}(g) = \min(\#\text{ appearances of n-gram } g \text{ in } a, \#\text{ appearances of n-gram } g \text{ in } b)$$

GLEU includes a Brevity Penalty (BP) to penalize hypotheses that are short compared to the references (Equation (3.2)), where l_h represents the total token count in the hypothesis corpus and l_r signifies the overall token count in the sampled reference corpus.

$$\text{BP} = \begin{cases} 1 & \text{if } l_h > l_r \\ \exp(1 - l_r/l_h) & \text{if } l_h \leq l_r \end{cases} \tag{3.2}$$

GLEU is finally calculated as in Equation (3.3).

$$\text{GLEU}(O, H, R) = \text{BP} \cdot \exp\left(\frac{1}{N} \sum_{n=1}^{N} \log p_n \right) \tag{3.3}$$

GLEU scores range from 0 to 1. Scores close to 0 indicate poor performance in correcting grammatical errors, whereas scores near 1 signify excellent performance.

GLEU was computed using code from the GitHub repository cnap/gec-ranking created by its developers. The code was adapted to work with Python 3, as the original implementation was in Python 2.

Another metric considered for the performance of the models is the inference time, as longer times can affect the user experience and usability.

3.4. Graphical user interface

As one of the objectives of the project is to create an accessible AI tool that is easy to use, a Graphical User Interface (GUI) is needed. For the creation of the GUI the library Streamlit[15] is utilized. Streamlit provides the functionalities needed to create a GUI that is connected with the model.

[15] https://streamlit.io

In order to make the response of the model visually clearer, the library difflib[16] was used to obtain the changes between the input text and the output of the model. The changes have been highlighted, using red for the words in the original text and green for the corrections in the text.

[16] https://docs.python.org/es/3.13/library/difflib.html

4. DATA PROCESSING

The processes explained in this section are essential for understanding the subsequent work in this research. Additionally, important data analysis and pre-processing are conducted in this part.

This chapter explains how the text data is prepared, as it needs to be transformed in order to be completely prepare to be used for training and fine-tuning. It also includes a pre-visualization of the essay data. This step is done in order to get a clear vision of the initial data checking if something is out of what it was expected. Apart from that, it is explained how the CSVs are joined creating a clean dataset.

4.1. Creating the dataset

As a first step, 29 CSV files were read using the read_csv() function provided by Pandas. As all of the CSV files contain the same columns they were joined by using the concat() function of Pandas. As a result, the obtained dataset dimensions were 5382 rows and 26 columns.

Once the dataset was created, it is essential to clean it, looking at missing values and other issues in the data. In order to detect the missing values the function isna() from Pandas was used. A summary of the number of missing values in each column is displayed in Figure 4.1

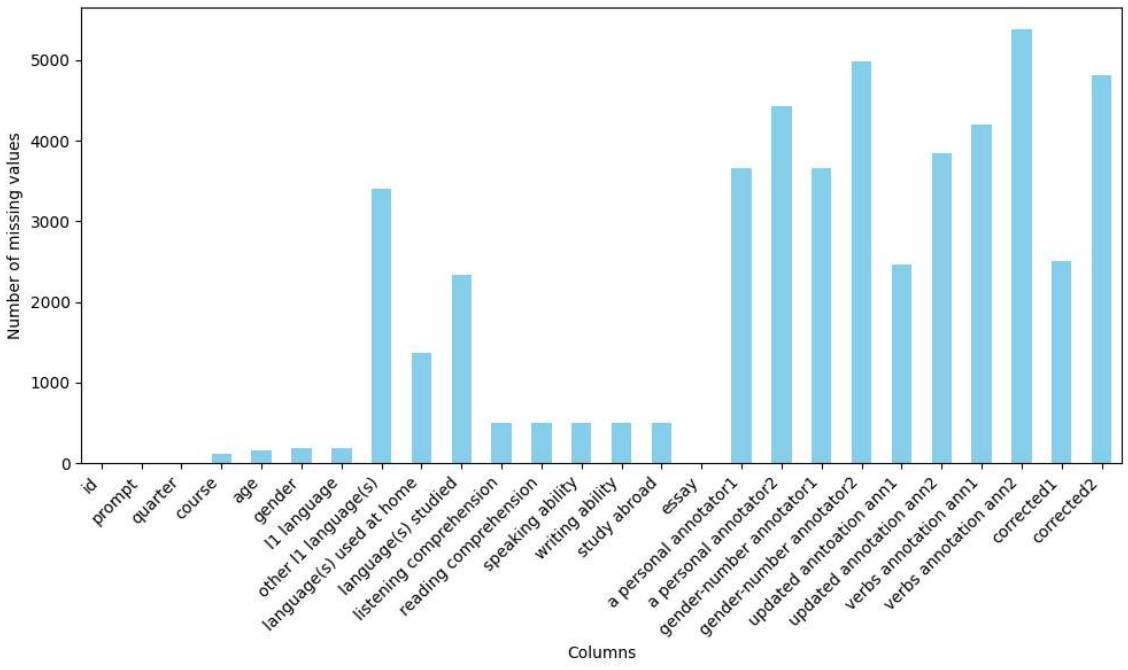

Fig. 4.1. Count of missing values per column.

There is a significant number of missing values, particularly in corrections that address only one type of error. For training, we will utilize the original essays alongside the versions corrected for all errors. This involves using the data from the columns "essay," "corrected1," and "corrected2." By filtering the dataset to include only the rows with values in "corrected1" or "corrected2" and creating a new row for each corrected text, we end up with 3896 essays with their corresponding corrections.

There were some annotation errors in the "age" and "gender" columns, leading to some rows containing incorrect data entries. For instance, a row might incorrectly list "Male" as the age and "20" as the gender. This error was corrected by looking at the data types in each row and swapping the values when a number was detected in the "gender" column.

Despite the documentation of the dataset were only ten courses were taken into account, the column "course" contains more courses which were not given any information about them.

4.2. Exploratory Data Analysis

This section presents the results and conclusions derived from the EDA performed on the cleaned dataset, offering a more comprehensive understanding of the data. Through this analysis, we have been able to identify key patterns, trends, and insights that enhance our knowledge of the dataset's structure and variables. By delving deeper into the data, we are better equipped to draw meaningful conclusions and make informed decisions based on the findings. This thorough examination lays the foundation for any further analysis or modeling efforts, ensuring that our approach is rooted in a solid understanding of the dataset's intricacies.

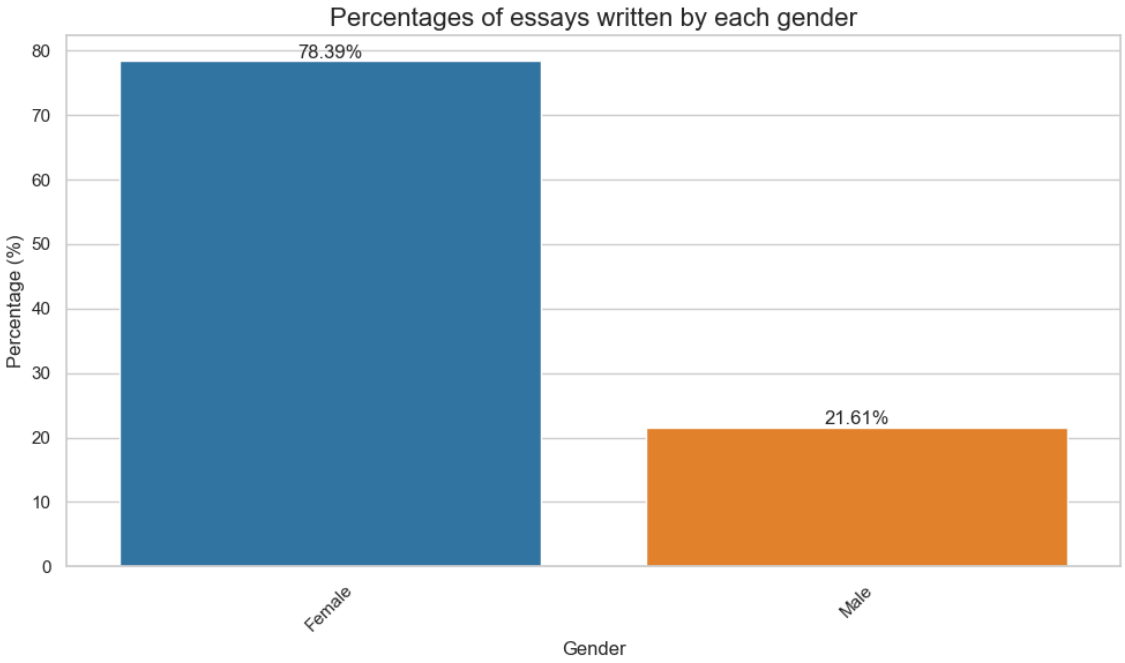

Fig. 4.2. Percentages of essays written by gender.

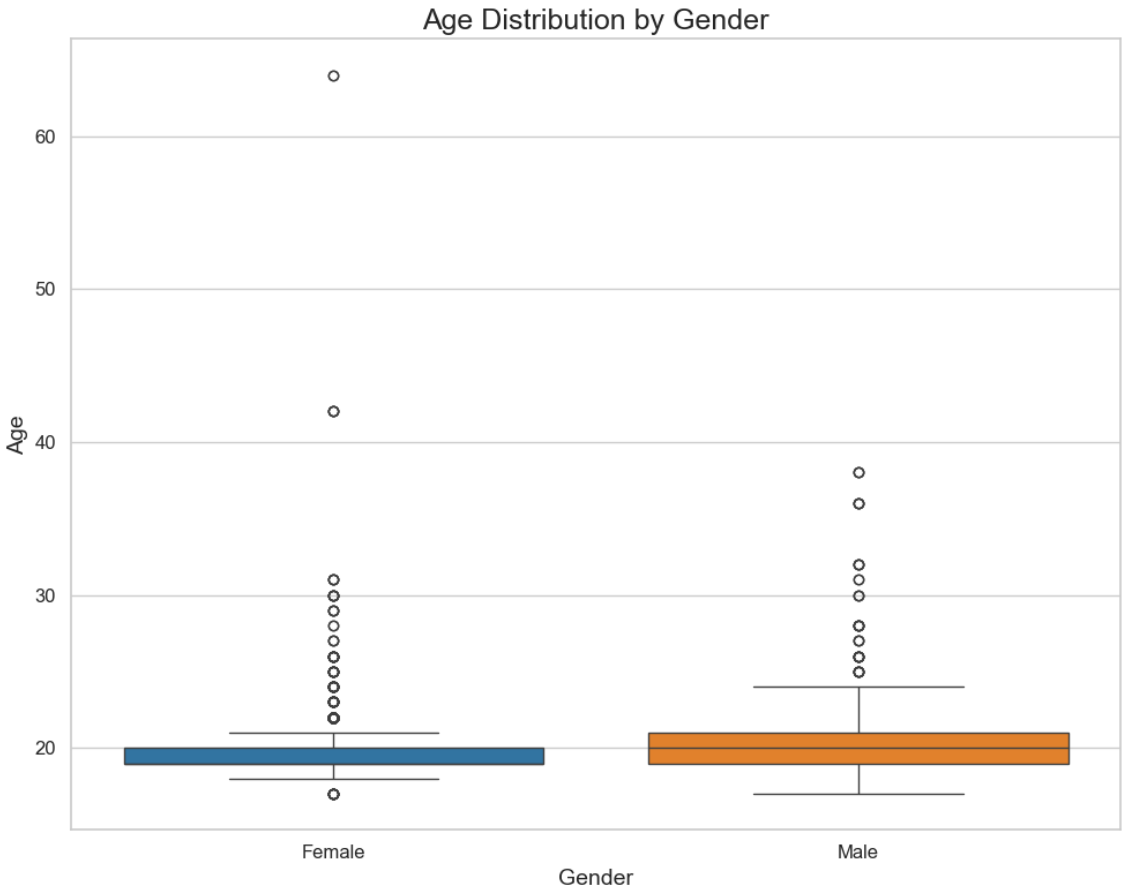

Fig. 4.3. Distribution of age by gender.

Figure 4.2 reveals that the majority of the essays are authored by females. In addition,

Figure 4.3 illustrates that the age distribution for both genders is quite similar and centered around 20 years. Although there are some outliers present in the age data, they do not pose any issues, as this feature will not be utilized in our modeling processes.

We are working with essays written in response to specific prompts, ensuring that the context is focused on particular topics. As shown in Figure 4.4, the distribution of essays across various topics is relatively balanced, with the exceptions of "chaplin" and "place_you_dislike." Due to the limited number of essays for these topics, we can expect that the model's performance may be suboptimal for them.

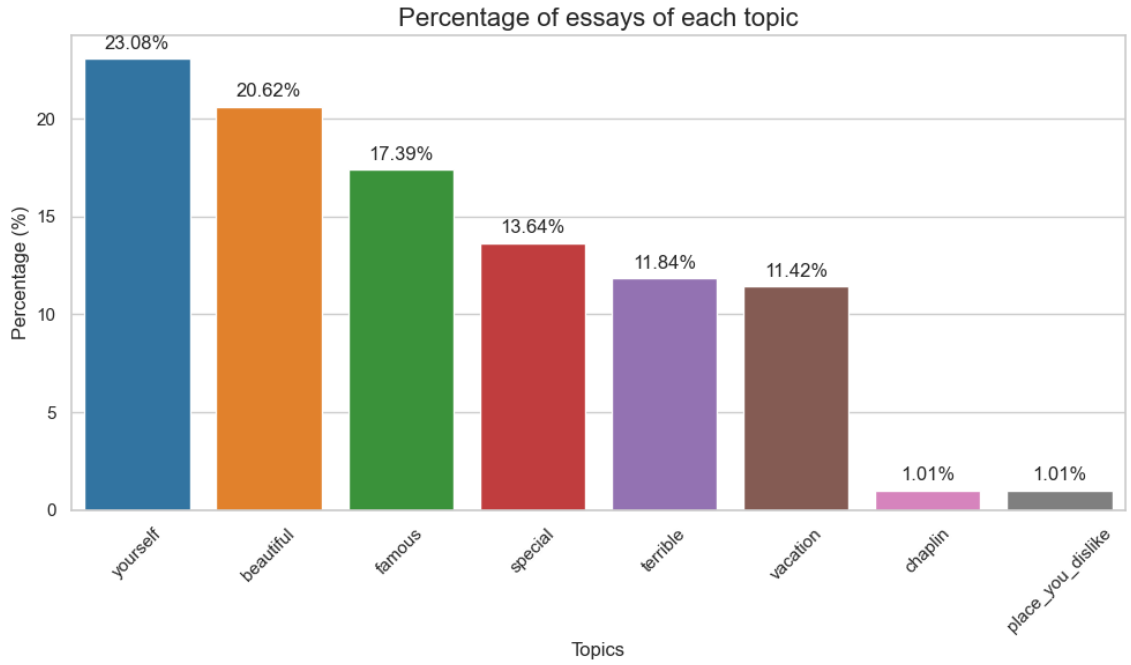

Fig. 4.4. Distribution of essays by topic.

One crucial factor in working with language models is understanding the anticipated number of input and output tokens, as this heavily influences both performance and inference time. To assess essay lengths, we counted the number of words, creating a new variable for this purpose. As depicted in Figure 4.5, the average essay length is 258.12 tokens, indicating that the essays are relatively short. However, with a maximum length of 592 tokens, the models must be equipped to process and generate at least this number of tokens.

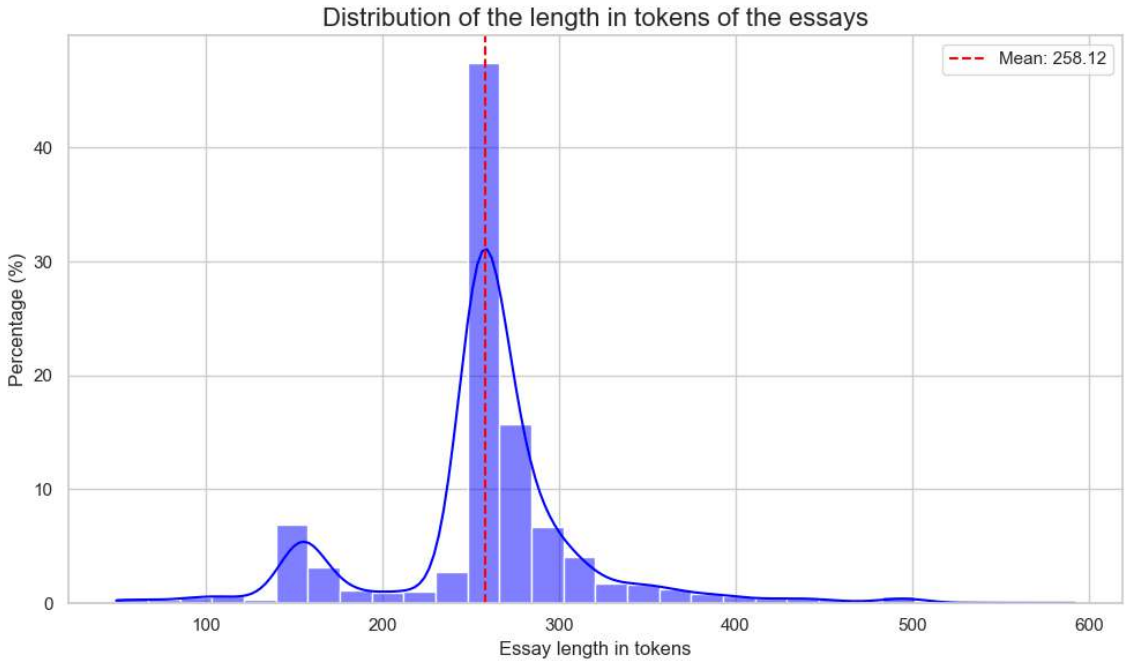

Fig. 4.5. Distribution of the length of the essays.

Lastly, it is essential to ensure that the lengths of the essays and their respective corrections do not differ significantly, as we expect our models to make corrections without introducing a substantial number of new words. Figure 4.6 illustrates that most essays and their corrections have similar lengths. For visualization purposes, a slope of 1 has been plotted, indicating that a correction with the same length as the original essay is represented by points close to this line, suggesting minimal addition or subtraction of words. However, we identified an outlier: an essay with a length of 280 tokens had a correction of fewer than 40 tokens. Upon analysis, we found the correction to be incomplete, so it was removed. The remaining cases that deviate from the slope of 1 were examined and they have no issues.

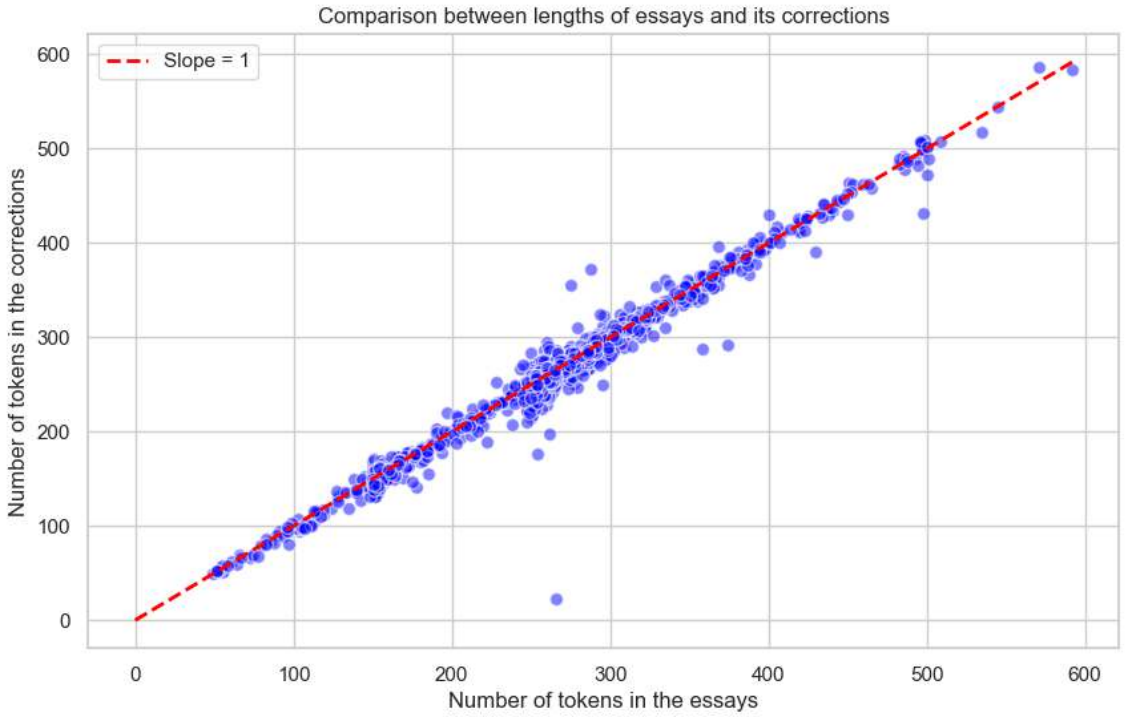

Fig. 4.6. Comparison between the lengths of the essays and its corrections.

4.2.1. Preparing text data for modeling

In this section the different ways text data was prepared to be used as input in each model are discussed.

LSTM Encoder-Decoder

For the LSTM network, the essays need to be divided into three components: an input for the encoder, an input for the decoder, and an output for the decoder. In the decoder input, the special tokens <start> and <end> are placed at the beginning and end of the essay, respectively. For the decoder output, only the <end> token is appended at the end of the essay. No special tokens are added to the encoder input. This setup helps the model differentiate between the start and end of sequences, facilitating accurate processing and prediction.

LLMs

For preparing text for Large Language Models (LLMs), we utilized the functionalities of the Unsloth library. Depending on the specific model, a distinct conversational style is required, necessitating the use of a tailored chat template to ensure proper model utilization.

For the GPT-2 model, the dataset needs to be transformed to include an instruction,

an input, and an output. The instruction is "Corrige gramaticalmente el siguiente texto," the input is the original essay, and the output is the grammatically corrected version of the essay. As illustrated in Figure 4.7, the data is formatted such that the final input to the model is a concatenation of the instruction, the input, and the word "Respuesta," followed by the output. This format provides a clear directive for the model to perform grammatical corrections on the given text.

{'instruction': 'Corrige gramaticalmente el siguiente texto.',
'input': 'Hola! Soy estudiante graduada de biología. Yo vivo en *CITY*, pero yo soy de *CITY*.
Tengo *AGE* años…',
'output': '¡Hola! Soy estudiante graduada de biología. Yo vivo en *CITY*, pero yo soy de *CITY*.
Tengo *AGE* años...'}

Fig. 4.7. Example of one observation of the dataset after the preparation for training and testing in GPT-2 model.

For Llama 3.2, the Llama 3.1 format is adopted, which uses specific tokens such as <|begin_of_text|>, <|start_header_id|>, <|end_header_id|>, and <|eot_id|>, along with a structure that distinguishes between role and content. In the system role of the prompt, instructions for correcting the text with minor changes were included, while the user's content focused on the input essays. The role of the assistant was employed for the target corrections. To transform the entire dataset effectively, we applied the standardize_sharegpt function from the Unsloth.chat_template library. In Figure 4.8 there is an example of how the dataset is transformed and Figure 4.9 shows how data is finally inputted to the model.

{'conversations': [{'content': 'Hola! Soy estudiante graduada de biología. Yo vivo en *CITY*, pero yo
soy de *CITY*. Tengo *AGE* años…',
'role': 'user'},
{'content': Hola! Soy estudiante graduada de biología. Yo vivo en *CITY*, pero yo soy de *CITY*.
Tengo *AGE* años…',
'role': 'assistant'}]}

Fig. 4.8. Example of one observation of the dataset after the preparation for training and testing in Llama 3.2 model.

<|begin_of_text|><|start_header_id|>system<|end_header_id|>

Cutting Knowledge Date: December 2023
Today Date: 4 March 2025

Eres un corrector de errores gramaticales. Devuelve el texto corregido sin dar explicaciones.<|eot_id|><|start_header_id|>user<|end_header_id|>

Hola! Soy estudiante graduada de biología. Yo vivo en *CITY*, pero yo soy de *CITY*. Tengo *AGE* años...<|eot_id|><|start_header_id|>assistant<|end_header_id|>

¡Hola! Soy estudiante graduada de biología. Yo vivo en *CITY*, pero yo soy de *CITY*. Tengo *AGE* años...
<|eot_id|>

Fig. 4.9. Example of one observation that is prepared to be the input of the Llama 3.2 model.

For Gemma 3, the chat template style differs, utilizing special tokens such as <bos>, <start_of_turn>, and <end_of_turn>. Despite these differences, the preparation of training and testing data remains consistent in terms of role and content structure. In this case, the standardize_data_formats function was employed to ensure the data is correctly formatted. The final conversation format used as input for Gemma 3 model can be seen in Figure 4.10.

<bos><start_of_turn>user
Eres un corrector de errores gramaticales. Devuelve el texto corregido sin dar explicaciones:
Hola! Soy estudiante graduada de biología. Yo vivo en *CITY*, pero yo soy de *CITY*. Tengo *AGE* años...<end_of_turn>
<start_of_turn>model
¡Hola! Soy estudiante graduada de biología. Yo vivo en *CITY*, pero yo soy de *CITY*. Tengo *AGE* años...<end_of_turn>

Fig. 4.10. Example of one observation that is prepared to be the input of the Gemma 3 model.

5. MODELING, TRAINING AND FINE-TUNING

In this chapter, we discuss the core aspects of building, training or fine-tune the models for the task of grammar error correction. This process involves selecting appropriate architectures, configuring models for optimal performance, and executing a training pipeline. The choice of models, including LSTM Encoder-Decoder, GPT-2, LLama 3.2, and Gemma 3, reflects a diverse set of strategies aimed at leveraging both traditional and cutting-edge advances in deep learning.

The objectives of this chapter are to outline the specific architectures employed, the training methodologies adopted, and discuss the processes of fine-tuning these models for improved performance and efficiency in grammar correction. A thorough understanding of these elements is essential, as the effectiveness of a grammar correction system is deeply rooted in how well its models are designed and trained.

Moreover, this chapter will address the experimental setup, including how data splits are structured to ensure robust training, and testing phases. Attention will also be given to the optimization techniques and computational resources that make the successful implementation of these models. In addition, highlighting the challenges faced during the modeling and training phases, as well as the strategies implemented to overcome them, are discussed in this section.

5.1. Model architecture and implementation

5.1.1. LSTM Encoder-Decoder

In this implementation, two distinct Spanish tokenizer instances from the PyTorch library are employed: one for the encoder input and the other for the decoder sequences. These tokenizers transform textual data into numerical sequences, where each word (or token) is represented by a unique integer identifier. The encoder's input tokenizer considers all tokens, while the decoder's tokenizer excludes punctuation to achieve cleaner output sequences. This differentiation is crucial for effective, context-aware transformations of the data into a machine-understandable format.

Encoder Architecture

For the encoder, the following layers are used:

- Embedding Layer: This layer maps input tokens to dense vectors of a fixed size, specifically 300 in this project. This vector size is commonly used in fields such as Word2Vec [58] and GloVe [59] embeddings because it has demonstrated an

adequate capacity to capture the semantic properties of words. The argument padding_idx is set to zero to ensure that padding entries do not contribute to the gradients during training. This layer is implemented using `nn.Embedding` from PyTorch.

- LSTM Layer: This layer processes the embedding sequences and calculates the outputs, the final hidden state, and the final cell state. In this architecture, only the final hidden state and the final cell state are passed to the decoder to initialize it at an informed point. This layer is implemented using `nn.LSTM` from PyTorch.

Decoder Architecture

The decoder utilizes another set of embedding and LSTM layers configured similarly to those in the encoder. In addition, a linear layer is introduced to produce the logits of each token in the vocabulary, which is crucial for predicting the next token in the sequence. This is implemented with `nn.Linear` from PyTorch.

Seq2Seq Class

To facilitate correct interaction between the encoder and the decoder, a new class named Seq2Seq is created. Within its forward method, the following operations are performed:

1. Output Tensor Initialization: A tensor is initialized to store the predicted token distributions for each sequence position in the output.
2. Encoding Phase: The source sequence is passed through the encoder, generating the hidden and cell states that encapsulate the input sequence's essential information.
3. Decoding Phase: The decoder uses the initial states from the encoder and the input sequence to compute the probabilities for each token, ultimately generating the output sequence.

An illustration of the architecture is shown in Figure 5.1.

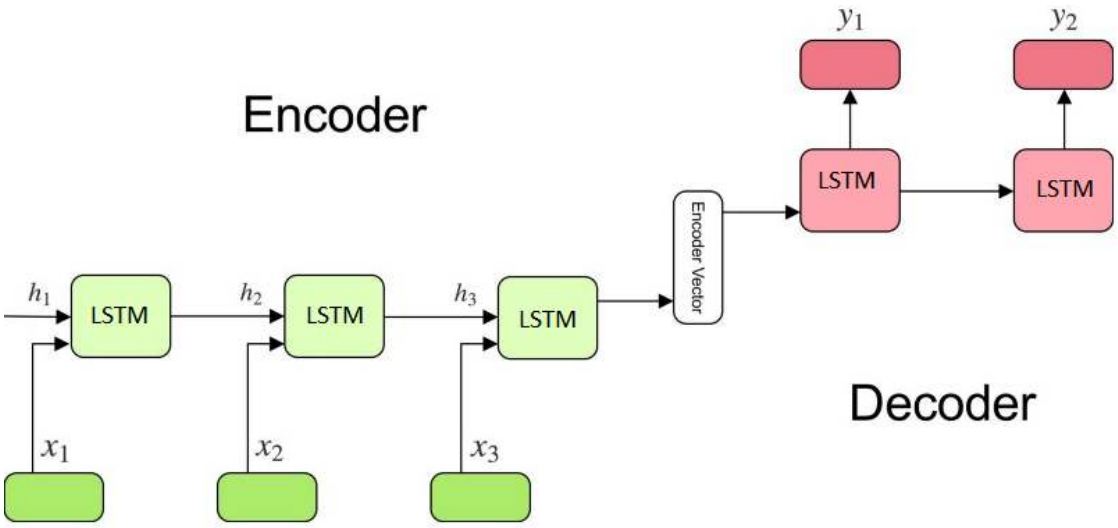

Fig. 5.1. LSTM Encoder-Decoder architecture [60].

5.1.2. GPT-2

The Generative Pre-trained Transformer 2 (GPT-2) architecture is employed in this implementation for text generation tasks. GPT-2, developed by OpenAI, is known for its ability to generate coherent and contextually relevant text continuations. It is a unidirectional transformer-based model that predicts the next word in a sequence given the previous context.

The original model from OpenAI was pre-trained only with English texts. In order to obtain coherent results, instead of loading the original model, another GPT-2 trained on Spanish texts is used. The model has been loaded from the Hugging Face repository DeepESP/gpt2-spanish, which has been trained on Spanish Wikipedia articles and books. The final architecture was implemented using 124 million parameters, embedding dimension of 768, a total of 12 transformer block layers and 12 attention heads.

Tokenizer

A tokenizer is used to preprocess the text data before inputting it into the GPT-2 model. In this implementation, a Byte Pair Encoding (BPE) tokenizer from the Hugging Face Transformers library is utilized. This tokenizer is responsible for:

- **Converting Text to Tokens**: The tokenizer splits the input text into smaller units or tokens based on subword units, capturing common patterns and morphemes. Each token is then mapped to a unique integer in the vocabulary.
- **Ensuring Consistency Across Texts**: By using a pretrained BPE tokenizer aligned with the GPT-2 model, the text inputs maintain consistency with the pretraining

corpus, enhancing model performance.

Seq2Seq Task Adaptation

Although GPT-2 is primarily designed for text generation, it can be instruction tuned meaning that it can be adapted to follow instructions, in this case, to correct grammatical errors in the essays.

5.1.3. Llama 3.2

In the case of Llama 3.2 there was no need of implementing the architecture neither the tokenizer as with FastLanguageModel.from_pretrained from Unsloth the model with its weights and the tokenizer can be loaded. The instruct version of the model is used, which is already prepared for receiving instructions. The complete architecture can be seen in Figure 5.2.

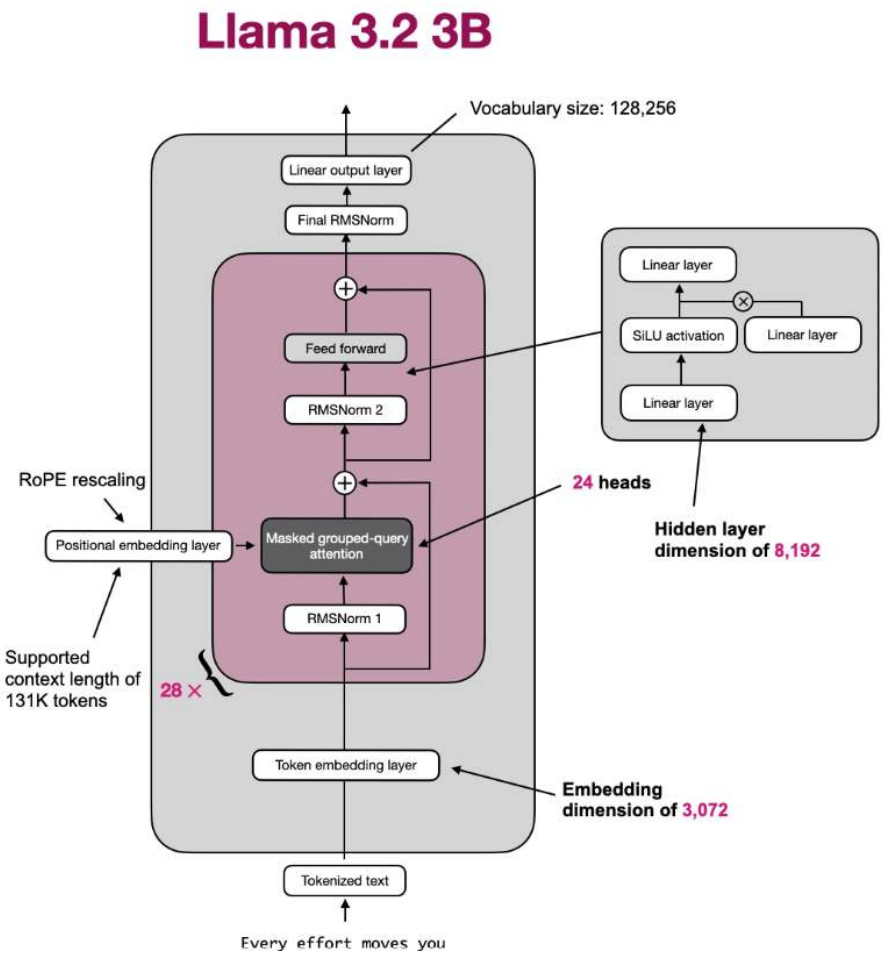

Fig. 5.2. LLama 3.2 3B architecture [47].

5.1.4. Gemma 3

In the case of Gemma 3 there was no need of implementing the architecture neither the tokenizer as with FastLanguageModel.from_pretrained from Unsloth the model with its weights and the tokenizer can be loaded. The instruct version of the model is used, which is already prepared for receiving instructions. Thecomplete architecture can be seen in Figure 5.3. In the 4 billions of parameters implementation the parameters are:

- Embedding size: 2560.
- Layers: 34.
- Feedforward hidden dimension: 20480.
- Number of heads: 8.
- Number of key value heads: 4.
- Query key value head size: 256.
- Vocabulary size: 262144.

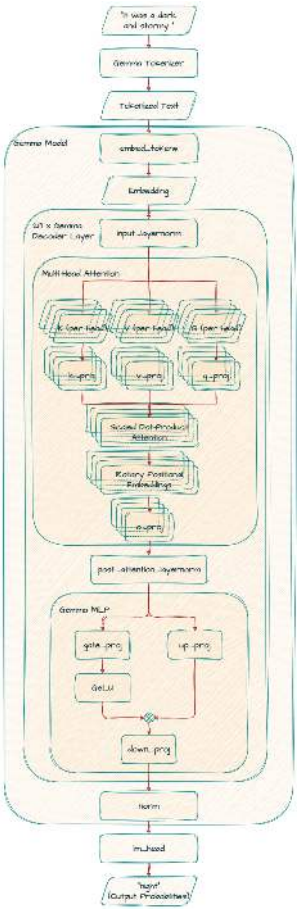

Fig. 5.3. Gemma 3 architecture [54].

5.2. Experimental setup

In this section, we detail the process of splitting the data into training, validation, and test sets. Additionally, we discuss the training configuration of each model.

For data splitting, we allocated 80% of the essays to the training set, 10% to the validation set, and 10% to the test set. Specifically, this corresponds to 3116 essays for training, 390 essays for validation, and 390 essays for testing.

The general training or fine-tuning configuration of each of the proposed models is shown in Table 5.1. In all cases the loss function used is a cross entropy implemented with nn.CrossEntropyLoss from PyTorch. All learning rates decrease during the training.

TABLE 5.1. TRAINING AND FINE-TUNING CONFIGURATIONS FOR EACH MODEL.

Model	Learning Rate	Batch Size	Number of Epochs	Optimizer
LSTM	0.001	64	20	Adam
GPT-2	0.0007	8	10	AdamW
LLama 3.2	0.0005	8	1	AdamW
Gemma 3	0.0005	8	1	AdamW

5.3. Training process

For the LSTM Encoder-Decoder model there are two different strategies, learn the embeddings from scratch or use pre-trained embeddings and just train the rest of parameters of the network. For both cases the training loop remains the same but in the second case we have to initialize the embedding layers with the pre-trained Word2Vec Spanish embeddings of size 300 from https://wikipedia2vec.github.io/wikipedia2vec/pretrained/.

The training loop is the usual one for training neural networks, this is, processing the input with the neural network, obtain the output, compute the loss, compute the gradients, update the weights of the network and set the gradients to zero. The only change is that teacher forcing is introduced to improve convergence, the teacher forcing ratio starts with a value of 1 and it is decreased each epoch until it reaches 0.5.

The evolution of the loss function over the epochs in the training and validation sets are illustrated in Figure 5.4 and Figure 5.5, the first one for the network without pre-trained embeddings and the second one with them. The training process uses early stopping when after 5 epochs the validation loss does not decrease, and the weights of the epoch that gave the lower validation loss are saved.

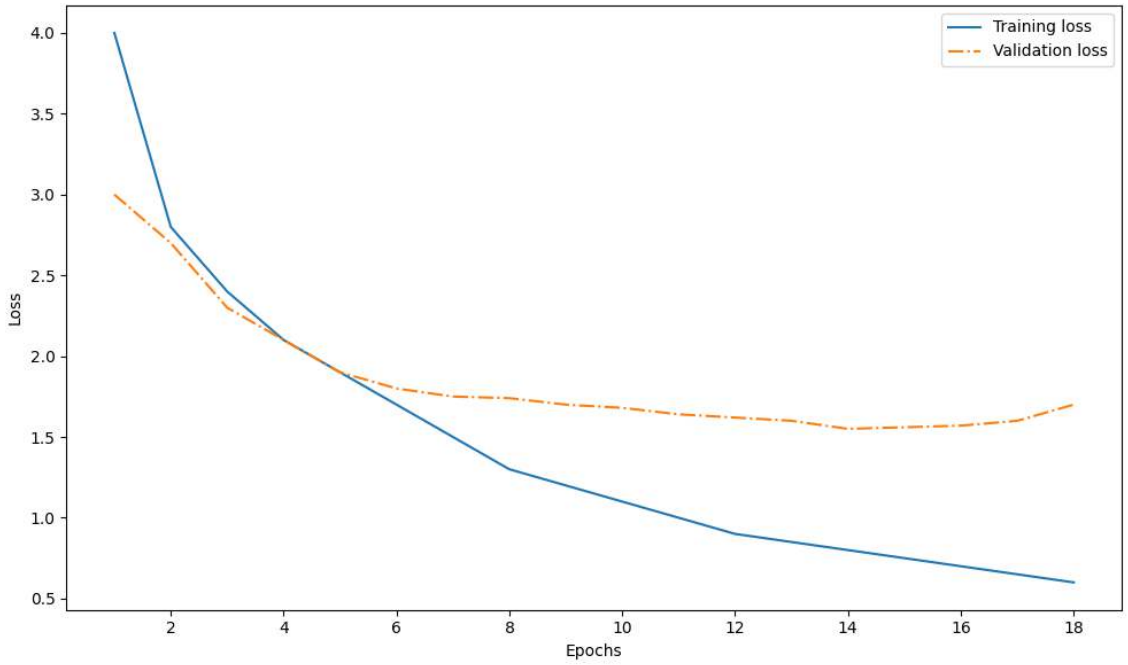

Fig. 5.4. Evolution of the loss function during training in the LSTM network without pre-trained embeddings.

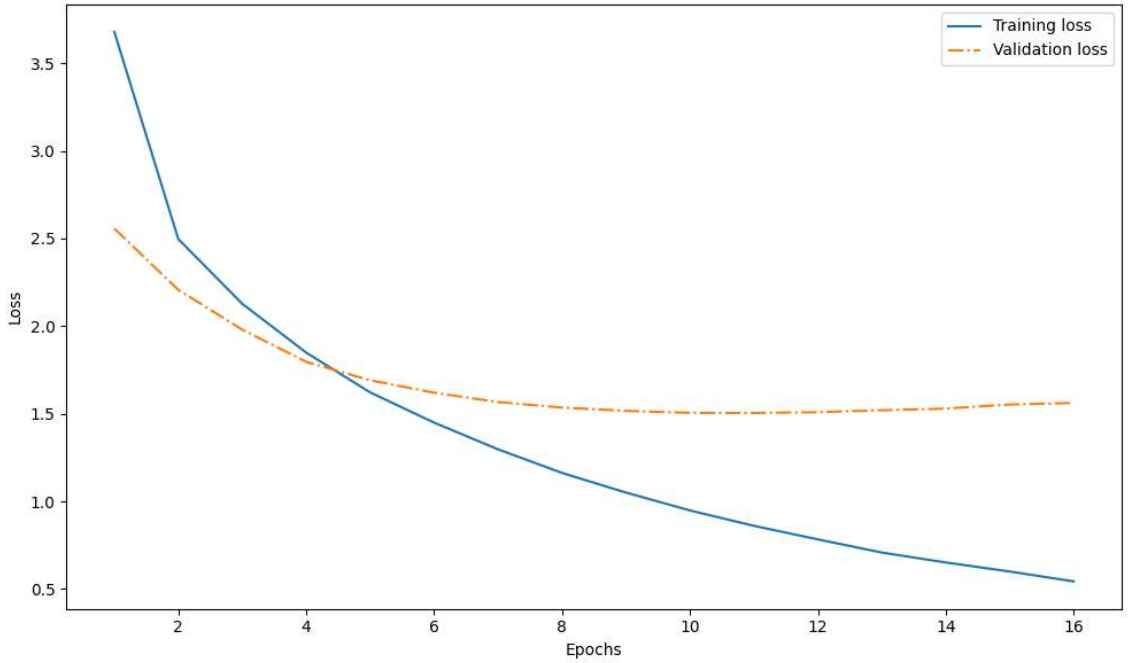

Fig. 5.5. Evolution of the loss function during training in the LSTM network with pre-trained embeddings.

As illustrated in Figure 5.4 and Figure 5.5, early stopping resulted in the final models ending training after 18 and 16 epochs, respectively, while the ultimate LSTM models were obtained after 13 and 11 epochs. This discrepancy occurs because the network without pre-trained embeddings begins with a less advantageous starting point and requires

more epochs to effectively learn from the data. Additionally, the validation loss is slightly lower when using pre-trained embeddings.

5.4. Fine-tuning process

The instruction tuning in the GPT-2 model is done to follow instructions, in our case, to correct grammar errors in the essays. The pipeline is the same as in the training of the LSTM network, but the optimizer changed to AdamW due to the increase in the complexity of the models.

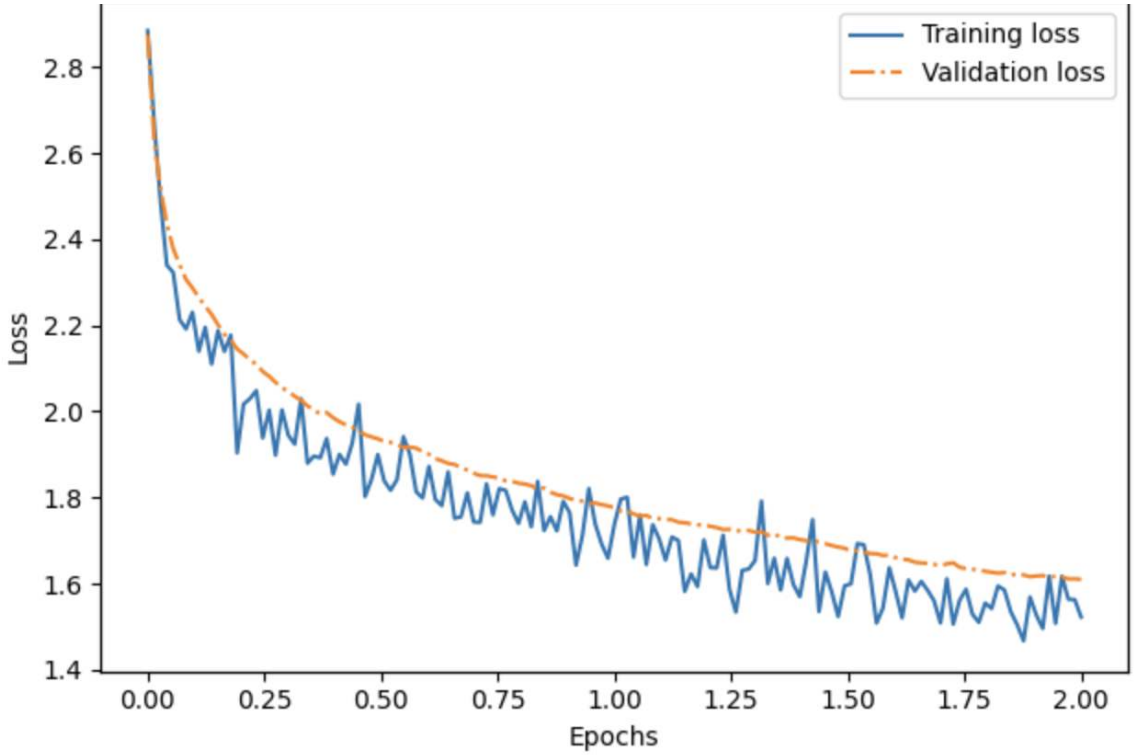

Fig. 5.6. Evolution of the loss function during instruction tuning in the Spanish GPT-2 model.

In Figure 5.6 we can see that there are fluctuations in the training loss, but the trend is down. This happens because as only two epochs are used for instruction tuning the steps of the training have also been plotted.

In the case of the models Llama 3.2 and Gemma 3 the use of the library Unsloth simplifies the code and makes the training faster and more efficient. In this case, the fine-tuning is done using QLoRA, with a rank and an alpha with a value of 16. With these changes instead of updating 3 and 4 billions of parameters, we only have to update 24 and 30 millions respectively. This models are only fine-tuned in a single epoch with 30 steps. These settings were chosen by the recommendation of the developers of the Unsloth library. The evolution of their losses can be seen in Figure 5.7 and Figure 5.8.

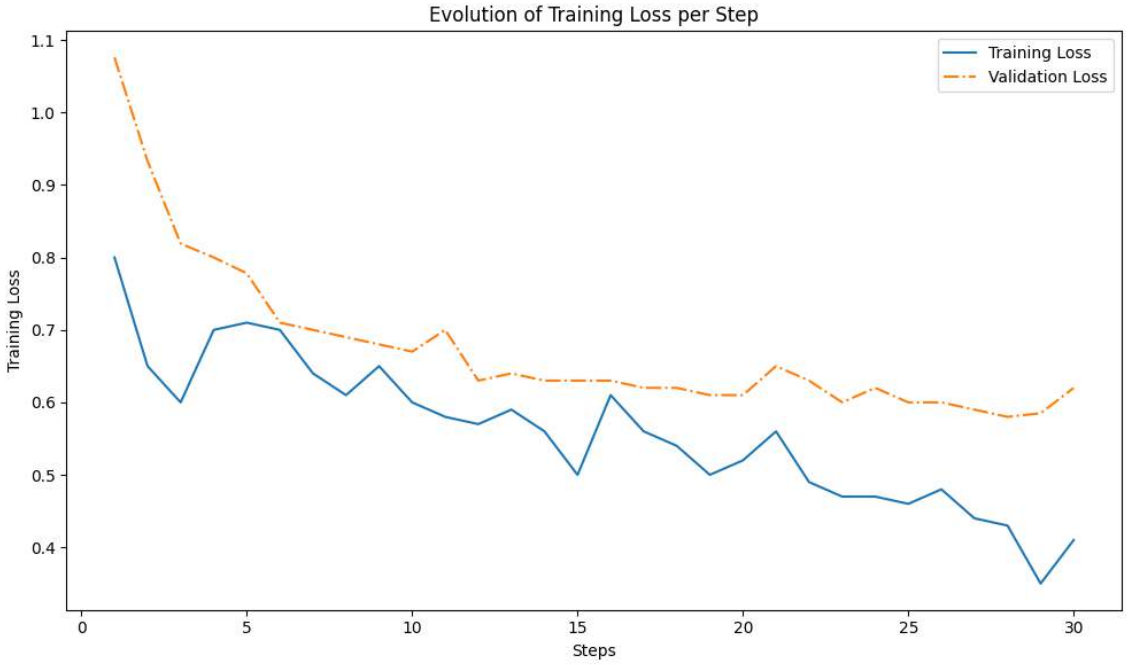

Fig. 5.7. Evolution of the loss function during fine-tuning in the Llama 3.2 model.

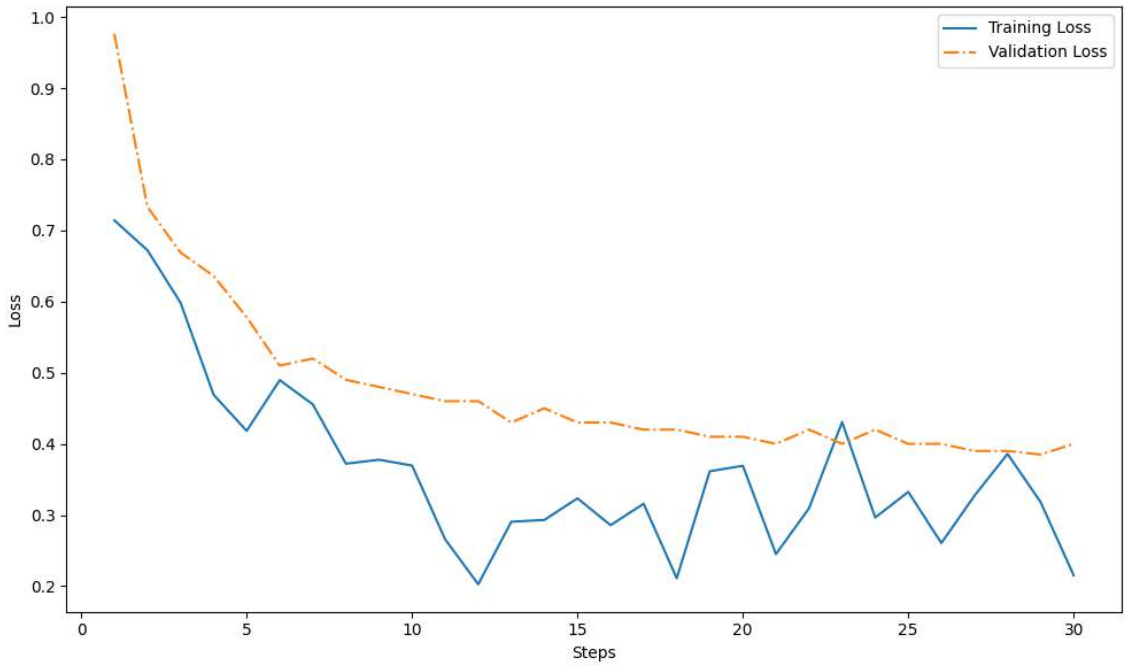

Fig. 5.8. Evolution of the loss function during fine-tuning in the Gemma 3 model.

The plots indicate that losses have decreased in both the validation and training sets, confirming the effectiveness of fine-tuning in both models. The observed instability in the training process is attributed to the plots being displayed on a step scale rather than an epoch scale. This choice of scale can lead to apparent fluctuations that might smooth out when viewed over whole epochs. The number of steps appears to be appropriate as the validation loss curve, after first steps, decreases slightly.

6. INFERENCE AND RESULTS

In this chapter, we provide a comprehensive overview of the inference process. We also discuss the results of each model in detail, with a particular focus on analyzing the language models that yielded strong performance. Ultimately, we select a final model for implementation in the GUI.

6.1. Result analysis

Inference was conducted locally by hosting the different models with Ollama, taking advantage of the MacBook Pro's M2 chip. Since Ollama is being utilized, the models need to be converted into the GGUF format to ensure seamless integration. The transformation of PyTorch models into GGUF format was accomplished using conversion scripts available from the GitHub repository ggml-org/llama.cpp. This process facilitates straightforward use of the models with Ollama.

After converting the models into GGUF format, the next step is to create a Modelfile. This file specifies the location of the GGUF file and defines various model parameters, including the prompt and temperature settings. The prompt used is consistent with the one employed during training. We chose a temperature value of 0.7 to achieve a balanced trade-off between creativity and precision.

With the models loaded in Ollama, we can leverage them in Python scripts using the chat and ChatResponse functions from the Ollama library. In the chat function we specify which model we want to use. After each essay is corrected, we save the original essay, the human-corrected version, and the model-corrected version in a JSON file. This file will be utilized to compute the GLEU metric, allowing us to analyze the model's performance effectively.

A general summary of the results is shown in Figure 6.1 and in Table 6.1. These results show a relatively good performance as the state-of-the-art results presented in section 2.1 are slightly higher but they were obtained on sentence-level.

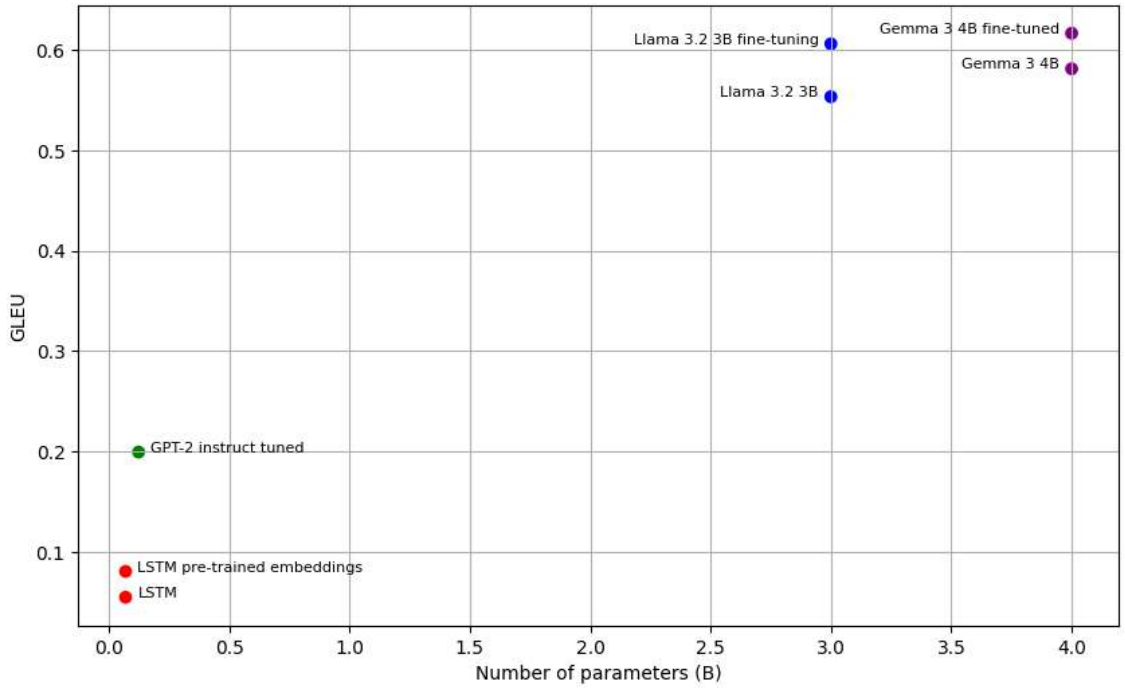

Fig. 6.1. Performance of the models considering GLEU score and the total number of parameters.

TABLE 6.1. GLEU SCORE AND INFERENCE TIME OF EACH MODEL.

Model	# of Parameters	GLEU	Inference time
LSTM	69.64 million	0.0549	5.8 seconds
LSTM with pre-trained embeddings	69.64 million	0.0806	5.8 seconds
GPT-2 instruction tuned	124 million	0.1994	7.8 seconds
LLama 3.2 3B	3 billion	0.5533	9.35 seconds
Gemma 3 4B	4 billion	0.5811	11.17 seconds
LLama 3.2 3B fine-tuned	3 billion	0.6059	9.35 seconds
Gemma 3 4B fine-tuned	4 billion	0.6165	11.17 seconds

To obtain good corrections, a minimum GLEU score of 0.5 is needed [26]. The results indicate that the Gemma 3 4B model, after fine-tuning, achieves the highest GLEU score, making it the best-performing model. However, it also has the highest number of parameters, leading to the longest inference times. These findings align with expectations, as Gemma 3 was released after Llama 3.2 and has a greater number of parameters, contributing to its improved performance but also increased computational demands.

The results also highlight the significant impact of fine-tuning large language models (LLMs). While fine-tuning led to an increase in the GLEU score across models, its effect was particularly pronounced in the Llama 3.2 model. This suggests that fine-tuning can be a powerful tool for enhancing model performance, especially in certain architectures like Llama 3.2.

For the LSTM Encoder-Decoder networks and GPT-2 model the results were not good. For the case of the LSTM models, this happens because training from scratch a relatively large network with a small dataset does not allow the network to learn at all. Also, the networks were implemented without any attention mechanism and essays were large resulting in missing corrections due to missing past information. For the case of the GPT-2 model what happens is that the original model was created for text generation and not for this specific task, and the dataset may not be large enough to give a good instruction tuning.

The results for the LSTM Encoder-Decoder networks and the GPT-2 model were less favorable. In the case of the LSTM models, the poor outcome can be attributed to the challenges of training a relatively large network from scratch with a small dataset, which hindered the network's learning capabilities. Additionally, these networks were implemented without an attention mechanism, and the length of the essays led to some corrections being missed due to lost past information.

For the GPT-2 model, the issue stems from its original design, which is optimized for text generation rather than the specific task at hand. Furthermore, the dataset used for fine-tuning may not have been sufficiently large to effectively adapt the model to this task, limiting its performance.

6.1.1. Type of error

Since the COWS-L2H dataset offers specific corrections for different types of errors, the GLEU metric can be calculated using these corrections to evaluate how effectively the top-performing models, Gemma 3 and Llama 3.2, address specific errors. These errors are categorized in columns such as "a personal annotator," "gender-number annotator," and "verbs annotation ann." By analyzing performance in these specific areas, we can gain deeper insights into each model's strengths and weaknesses in correcting particular types of errors, regarding on pronouns, articles, gender, number and verb errors.

It's important to note that the average of these scores does not equate to the overall GLEU score. This discrepancy arises because the general scores incorporate a broader set of essays, including those with fewer specific errors. Consequently, the general GLEU score reflects performance across a wider range of error types, while the specific error analyses focus only on particular categories. Results are shown in Figure 6.2.

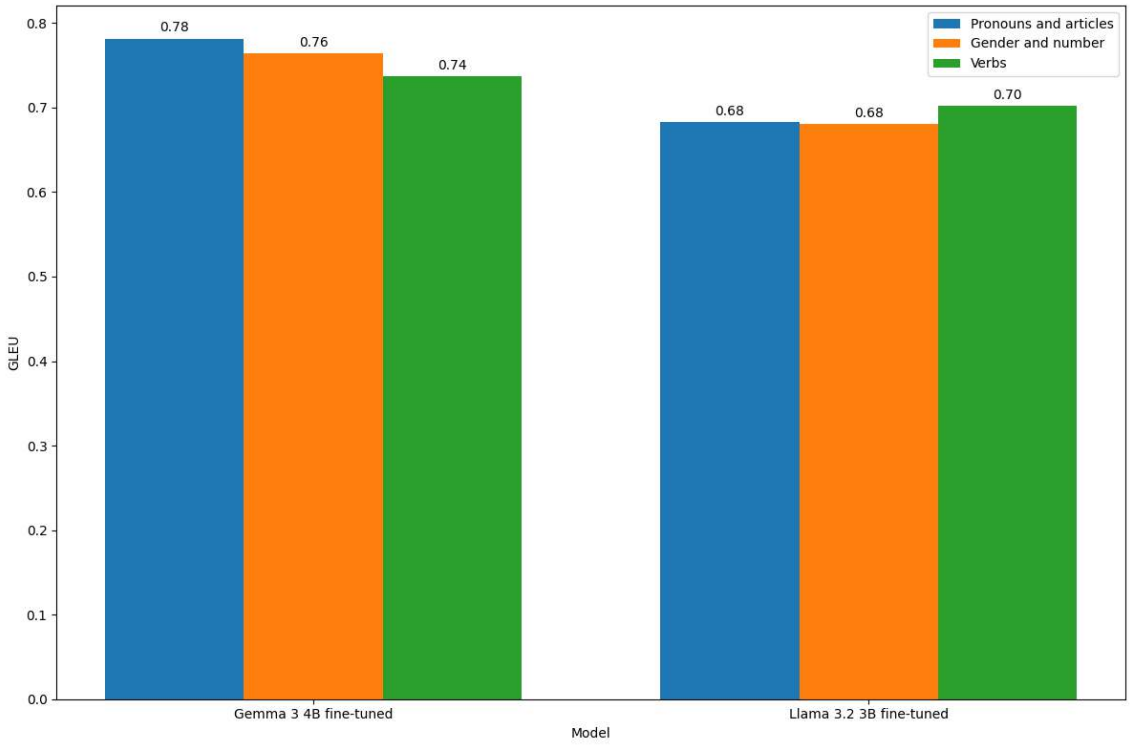

Fig. 6.2. Performance of the top-performing models in specific errors.

The results reveal that the fine-tuned Gemma 3 model outperforms the others across all specific error types, consistent with its superior overall performance, and a notable improvement in the correction of pronouns and articles. However, it is important to note that the analysis was conducted on a limited subset of essays due to incomplete data in the corrections for each error type: 137 essays for pronouns and articles, 135 for gender and number, and just 30 for verb corrections. This limited data set may influence the robustness of the findings for these specific categories [61].

6.1.2. Gender

An analysis was conducted to examine the models' performance in essay correction based on the author's gender. The goal was to determine whether the models are biased towards essays written by women, given the dataset imbalance, with 78.39% of the essays authored by women. Results of this analysis are presented in Figure 6.3.

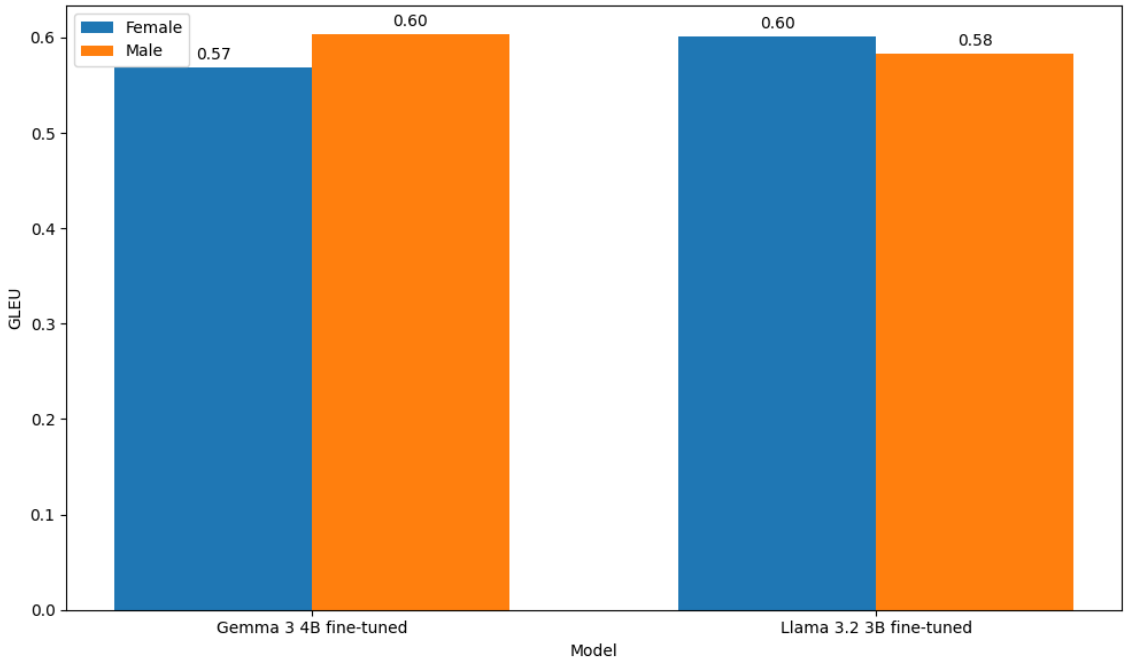

Fig. 6.3. Performance of the top-performing models depending on the author gender.

The results indicate that there is no significant difference in the model's performance on essays based on the author's gender, suggesting that the models, after fine-tuning, do not exhibit gender bias. However, similar to the previous experiment that focused on specific error types, the analysis of male-authored essays is based on a small sample size of only 54 essays. Therefore, these findings should be interpreted with caution, as the limited data may affect the reliability of the conclusions [61].

6.1.3. Topics

The last analysis conducted was to confirm if the models have been biased to the predominant topics, because each one has a more specific vocabulary. Results are shown in Figure 6.4.

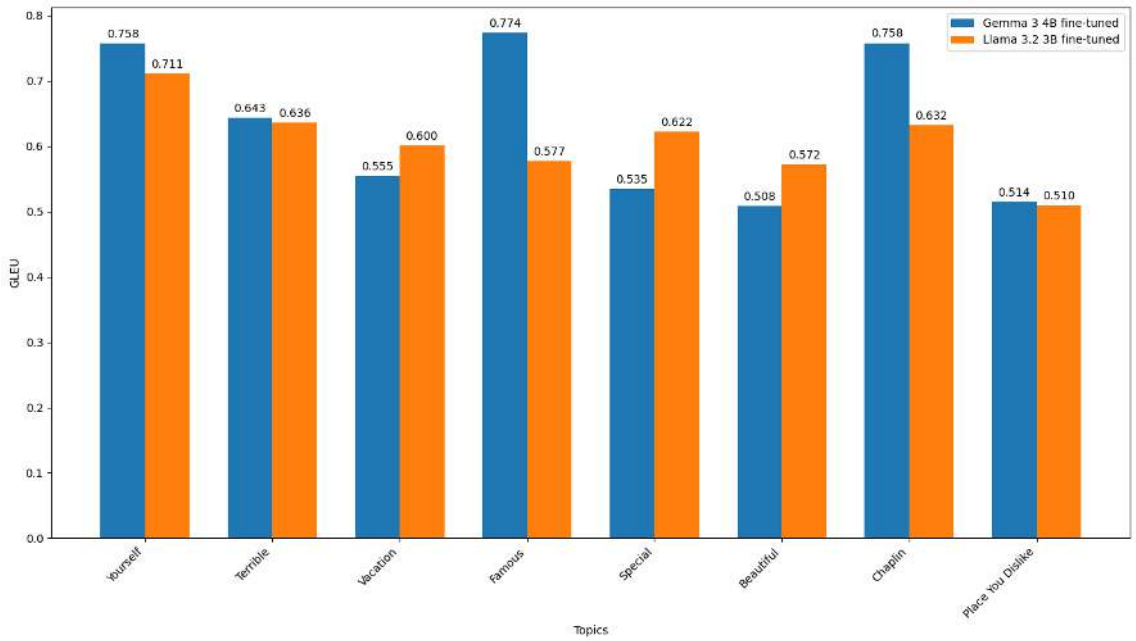

Fig. 6.4. Performance of the top-performing models depending on the topic.

Based on the results, we can confirm that the models tend to perform better on topics with a larger number of essays, such as "yourself" and "famous." Conversely, the performance is generally weaker on topics with limited representation in the dataset. For instance, the topic "Place You Dislike," which accounts for only 1% of the dataset's observations, shows the poorest performance. Interestingly, despite having a similar level of representation, the topic "Chaplin" demonstrates notably strong performance. This might be attributed to its specificity, as it involves students writing about their observations from a short video, potentially leading to more focused and uniform responses within that topic.

6.2. Final model selection

For the final AI tool implementation, we needed to select a model that strikes an optimal balance between performance and efficiency. After conducting an in-depth analysis across various domains, as well as evaluating overall performance metrics, we considered several factors, including GLEU performance, inference time, and the number of parameters each model possesses.

The Gemma 3 4B fine-tuned model emerged as the leader in terms of GLEU scores, demonstrating superior accuracy in corrections. However, the difference in performance compared to the Llama 3.2 3B fine-tuned model was not significantly large. Meanwhile, the Llama model has a notable advantage in terms of operational efficiency due to its smaller scale; it contains 1 billion fewer parameters, which translates to approximately 2 seconds less per inference. Note that the inference time of the models is important because users typically tolerate a response time of only 7 to 10 seconds before losing patience with a digital application [62].

This efficiency is a crucial consideration, particularly since the tool will be implemented locally. A smaller, more efficient model like the Llama 3.2 3B fine-tuned ensures broader accessibility, allowing more users to benefit from the tool without excessive computational demands. This makes it a more practical choice for widespread use, ensuring that the correction tool is not only effective but also readily available to a wider audience, thereby enhancing its usability and appeal. Consequently, the Llama 3.2 3B fine-tuned model was ultimately chosen for implementation in the application.

7. GRAPHICAL USER INTERFACE

In this chapter the details about the implementation of the graphical user interface are explained, dividing it in front-end and back-end. Note that in this project the focus was just on the local environment meaning this GUI is only accessible in local and only works if the local device has enough resources to run models in local.

7.1. Front-end

All the front-end of the GUI has been done with the functionalities of the library Streamlit. Firstly, when the user initializes the interface, it is only displayed the title, a box to introduce the essay and a button to allow the correction. These is done by using `st.title`, `st.text_area` and `st.button`. Figure 7.1 shows how the GUI looks when it is initialized.

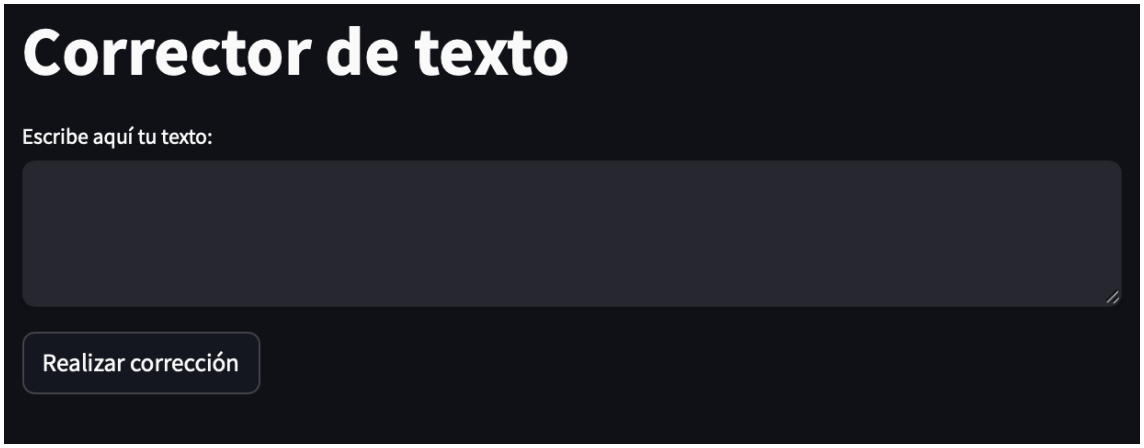

Fig. 7.1. GUI after processing an essay.

Once the text is introduced and the correction is done two columns are generated with the original and the corrected text, using `st.columns`, `st.header` and `st.write`. For the corrected text, the changes are shown in green and the errors in red. To do so the html code was modified, changing the attributes color and text-decoration. Figure 7.2 shows how it looks after correcting the essay.

Fig. 7.2. GUI after processing an essay.

7.2. Back-end

In order to process the essay with Llama 3.2 3B fine-tuned, Ollama is used. Creating a ChatResponse object allows to obtain the response of the model once it is given the essay as the input of the user. Once the corrected essay is obtained, Difflib is used to obtain the changes made by the model and use them for the highlighting.

In order to not allow inputs that are not essays a checking of the input is implemented. If the input is not text, only special or numeric characters, or it is empty, a message is displayed telling that only text is accepted. Cases were numbers or special characters are inside the text are allowed. In Figure 7.3 it is displayed the case when no text input is introduced.

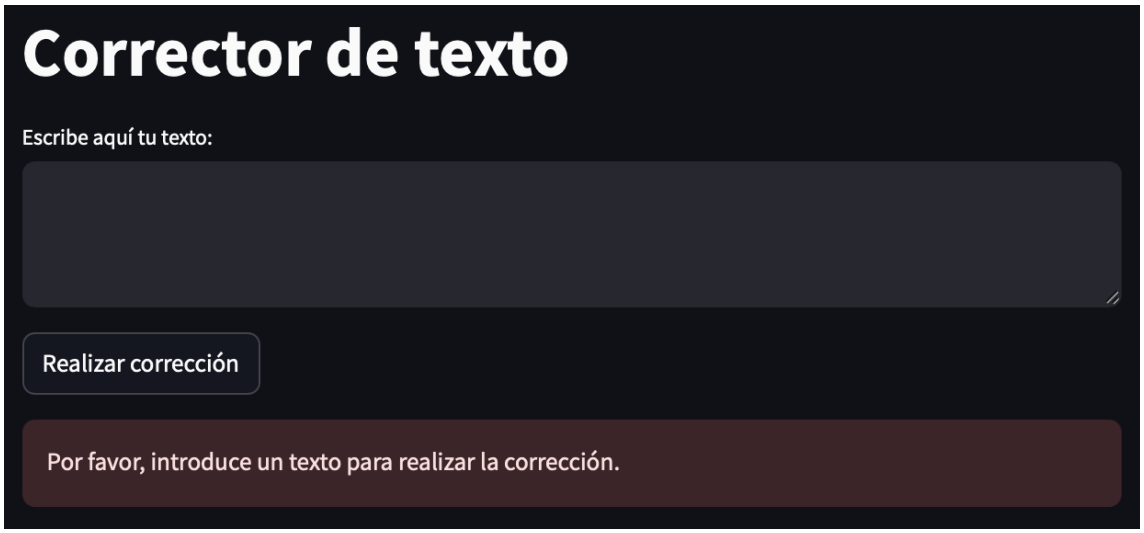

Fig. 7.3. GUI showing the error message.

As probably this application is going to be used by people who not necessarily know Spanish and the interface was in Spanish, an option to see the interface in English is given. This option allows the user to see the interface in Spanish or in English. It was implemented using the `st.selectbox` and a dictionary with the texts in both languages. Figure 7.4 shows the GUI in English. It also illustrates that the model can effectively correct short sentences and address spelling errors.

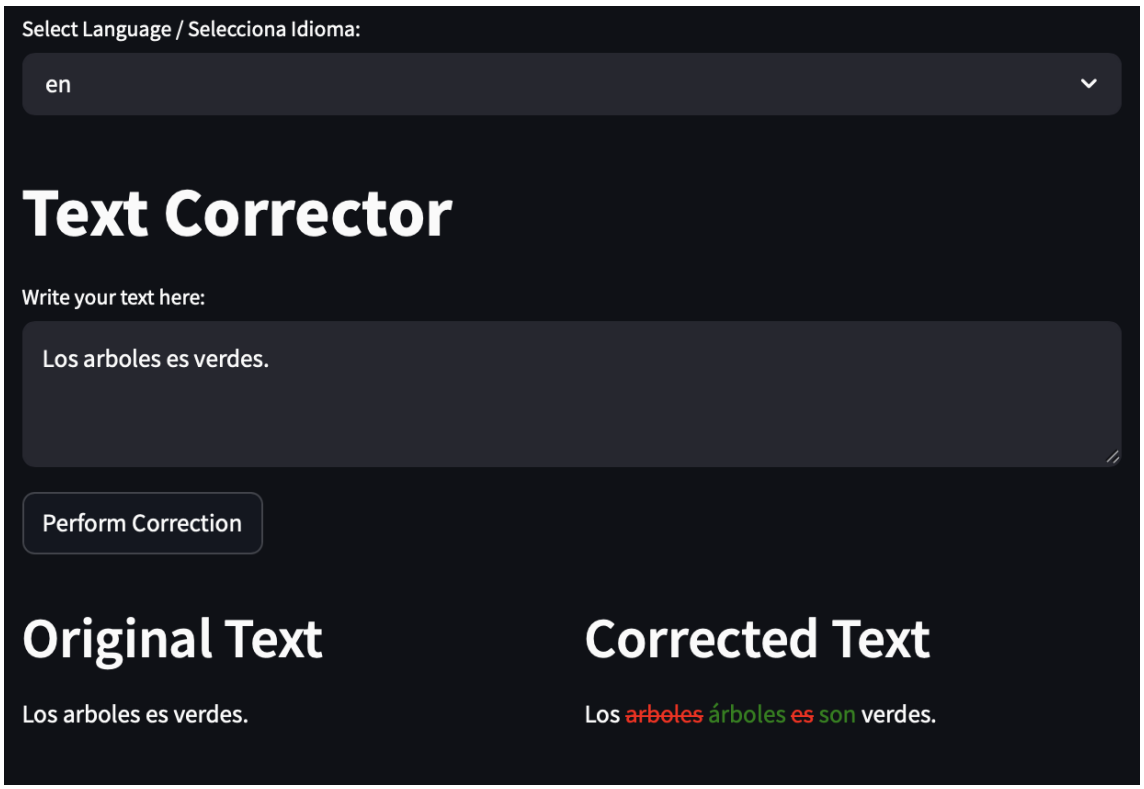

Fig. 7.4. GUI in English.

8. CONCLUSIONS

8.1. Summary of results

Artificial intelligence techniques are revolutionizing our understanding and optimization of complex systems. In the realm of Natural Language Processing, deep learning has introduced powerful tools, such as large language models, for enhancing writing. This study focused on developing language models to refine the essays of University of California students, utilizing data from the COWS-L2H dataset. Additionally, it analyzed how techniques like fine-tuning could enhance performance.

The main objectives of creating an AI tool for correcting the essays in the COWS-L2H dataset using deep learning techniques were accomplished throughout the course of this project.

Initially, the raw dataset was thoroughly explored and processed to identify and rectify any data quality issues, resulting in a cleaned dataset fit for analysis and training. Subsequently, an environment was established for building and evaluating models. A comprehensive process was devised for model development, encompassing training and evaluation using various models and architectures to identify the most effective techniques. To streamline and scale this process, scripts were developed, facilitating the efficient comparison of various models.

Finally, the models were tested using the GLEU metric, along with other metrics like inference time and the number of parameters. A comprehensive analysis was conducted to assess the performance of the top models in specific areas and to determine the presence of any gender or topic biases.

The conclusions that can be drawn from the results are:

- The models that achieved the highest GLEU scores were also the ones with the largest number of parameters. For instance, Llama 3.2, with approximately 3 billion parameters, outperformed smaller models like GPT-2, with 124 million parameters, and LSTM-based architectures, with around 70 million parameters, achieving a GLEU score of 0.6165, compared to 0.1994 and 0.0806, respectively. This suggests that the complexity of grammatical error correction tasks benefits from the representational capacity of large-scale models. However, such improvements come at the cost of increased computational demands. Therefore, it is essential to strike a balance between performance and efficiency, especially in real-time or resource-constrained applications, where smaller models with optimized training may offer a better trade-off.

- Fine-tuning pre-trained models with domain-specific data has proven to be highly

effective, boosting the GLEU scores of large language models by up to 9.5% in the best-case scenario. This enhancement underscores the value of tailoring general models to specific contexts to significantly improve their performance.

- Models that lacked proper pre-training performed poorly due to the limitations of a small dataset, making it impractical to train or instruction tune models from scratch. This highlights the importance of robust pre-training on extensive datasets to ensure effective model performance in subsequent fine-tuning stages.

- The best-performing models demonstrated a consistent ability to correct specific types of grammatical errors. There was no evidence to suggest that these models underperformed when correcting essays written by males, despite any data imbalance. However, a bias was observed in the models' corrections based on the essay topics. Nonetheless, due to the limited size of the test dataset, these results should be taken with caution.

Overall, this project highlights the substantial advantages of employing artificial intelligence tools to support teachers' work. Deep learning methods provide an automated approach to correcting student essays and enhancing their learning in Spanish writing. Offering students a secure AI tool for checking grammar errors in their personal essays can make learning more effective for teachers and students.

8.2. Application of the project

The primary application of this project is the use of the graphical user interface outlined in Chapter 5, which facilitates the detection and correction of grammatical errors. This tool offers several key benefits across educational contexts.

For teachers, it can significantly reduce the workload associated with correcting grammar errors, giving them time to focus on other essential and time-consuming tasks, such as individual student support. Moreover, teachers can employ this tool to identify common errors in student essays, enabling them to provide targeted feedback and instructional insights to the entire class, thus addressing common learning gaps effectively.

Beyond immediate classroom applications, the tool offers substantial advantages at the institutional level. Universities can leverage this tool to systematically track and evaluate students' writing performance across different courses and over time.

Students also benefit considerably from this application. Regardless of a teacher's availability, students can receive timely, automated feedback on their essays, allowing for continuous learning and improvement. The tool encourages students to engage in self-directed learning by helping them identify and understand their grammatical weaknesses, fostering their development as independent writers. Furthermore, such technology integration can enhance students' digital literacy and readiness for a tech-driven world,

equipping them with valuable skills for future academic and professional endeavors. As AI continues to evolve, the potential for these tools to adapt and expand in educational settings is immense, promising enhanced personalization and efficiency in learning processes.

Additionally, the performance results obtained in this work could be of significant relevance, particularly in the context of grammatical error correction for Spanish texts—a field with limited resources and benchmarks compared to English. By fine-tuning state-of-the-art language models on the COWS-L2H dataset, this project demonstrates that high levels of grammatical accuracy can be achieved even in low-resource settings. These results contribute to the ongoing development of more inclusive and effective language technologies, and may serve as a reference point for future research or applications targeting Spanish language learners and educators.

8.3. Limitations

While this work highlights the significant potential of artificial intelligence and deep learning models for essay correction, it also brings to light important limitations and challenges. As with any modeling process, outcomes are strongly influenced by the quality, representativeness, and size of the training data, as well as the architecture and parameters of the models used. These factors can introduce biases or lead to inconsistent performance. In the context of education—where these tools may directly affect student evaluation and learning—such limitations carry serious implications. Practitioners and institutions must approach the deployment of these systems with caution, ensuring transparency, fairness, and continuous monitoring. The most notable limitations observed in this project include:

- The limited size of the dataset constrains training, impedes instruction tuning, and affects the accuracy of result analysis. Additionally, the quality of the human corrections significantly influences model performance. Because the essays were corrected with minimal changes, the model tends to make only minor adjustments, resulting in texts that are grammatically correct but may sound stylistically unnatural to native speakers.
- The imbalance in the number of essays across different themes also affects the model's behavior, leading to poorer performance in the less-represented categories.
- The requirement for substantial computational resources, such as GPUs, makes the tool less accessible. Although the models can technically run on CPUs, the inference time is significantly longer, which can limit practical usability.

While promising, this work's impact was constrained by challenges related to data and computational resources, issues that are common in a developing field. As time progresses

and more advancements are made in large language models, some of these limitations may be addressed. With careful and responsible development, this field can provide valuable insights into grammatical error correction, thereby enhancing the learning process for students. However, further work is necessary to tackle issues of inconsistency, imbalance, scalability, and security before these techniques can be broadly implemented.

8.4. Future work

While the work demonstrates the significant potential of artificial intelligence and deep learning models for essay correction, further research is necessary to address limitations related to data quality, computational resources, and responsible implementation. These issues must be resolved before such solutions can be widely adopted.

- The application of data augmentation techniques, such as noise injection, back-translation, or round-trip translation, can be employed to generate synthetic data. This approach can help create a more balanced and larger dataset, potentially leading to improved performance and enabling the training of models from scratch for performance evaluation.
- In the training and fine-tuning process, experimenting with different hyperparameters from those used in this project could enhance the effectiveness of the models. By adjusting parameters such as number of epochs, or steps, learning rate or batch size it may be possible to optimize performance and achieve better results.
- Develop a more sophisticated model that not only performs corrections but also provides explanations for each correction made, offering users insight into the rationale behind changes. Alternatively, the model could suggest multiple possible corrections for each error, allowing users to choose the option that best suits their context or preference. This approach would enhance user engagement and understanding, making the tool more educational and user-friendly.
- Optimizing models for CPU usage by reducing computational resource requirements would significantly increase accessibility to the tool, allowing more people to run the models locally without the need for high-end hardware. This optimization could democratize access, making the tool available to a broader audience and enhancing its practical applicability in various settings.
- Applying more precise metrics, such as the MaxMatch score or ERRANT, could yield a more accurate evaluation of model performance in grammatical error correction. However, these metrics require essays to be specially annotated, which could pose a challenge in terms of resource and time investment. Utilizing these metrics, nonetheless, would provide deeper insights into how effectively the models address different types of errors.

Continued advancements in these future directions will allow grammar error correction and personalized support systems to realize their full potential, benefiting all students. However, it is crucial to proactively address privacy, ethics, and bias issues, incorporating these considerations as an integral part of the development process to ensure that the solutions are both effective and responsible.

BIBLIOGRAPHY

[1] Ministerio de Asuntos Exteriores, Unión Europea y Cooperación, *Spanish in the world*, https://www.exteriores.gob.es/en/PoliticaExterior/Paginas/ElEspanolEnElMundo.aspx, 2025.

[2] B. Paris, "Instructors' perspectives of challenges and barriers to providing effective feedback," vol. 10, Jan. 2022. DOI: 10.20343/teachlearninqu.10.3.

[3] A. Torfi, R. A. Shirvani, Y. Keneshloo, N. Tavaf, and E. A. Fox, *Natural language processing advancements by deep learning: A survey*, 2021. arXiv: 2003.01200 [cs.CL].

[4] Y. Chai, L. Jin, S. Feng, and Z. Xin, "Evolution and advancements in deep learning models for natural language processing," *Applied and Computational Engineering*, vol. 77, pp. 144–149, 2024.

[5] W. Khan, A. Daud, K. Khan, S. Muhammad, and R. Haq, "Exploring the frontiers of deep learning and natural language processing: A comprehensive overview of key challenges and emerging trends," *Natural Language Processing Journal*, vol. 4, p. 100 026, 2023.

[6] H. Kibriya, W. Z. Khan, A. Siddiqa, and M. K. Khan, "Privacy issues in large language models: A survey," *Computers and Electrical Engineering*, vol. 120, p. 109 698, 2024.

[7] X. Wu, R. Duan, and J. Ni, "Unveiling security, privacy, and ethical concerns of chatgpt," *Journal of Information and Intelligence*, vol. 2, no. 2, pp. 102–115, 2024.

[8] S. C. Kwasny and N. K. Sondheimer, "Relaxation techniques for parsing grammatically ill-formed input in natural language understanding systems," *American Journal of Computational Linguistics*, vol. 7, no. 2, pp. 99–108, 1981.

[9] C. Leacock, M. Gamon, and C. Brockett, "User input and interactions on Microsoft Research ESL assistant," in *Proceedings of the 4th Workshop on Innovative Use of NLP for Building Educational Applications*, Boulder, Colorado: Association for Computational Linguistics, Jun. 2009, pp. 73–81.

[10] C. Bryant, Z. Yuan, M. R. Qorib, H. Cao, H. T. Ng, and T. Briscoe, "Grammatical error correction: A survey of the state of the art," *Computational Linguistics*, pp. 1–59, Jul. 2023. DOI: 10.1162/coli_a_00478.

[11] S. Chollampatt, W. Wang, and H. T. Ng, "Cross-sentence grammatical error correction," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 435–445. DOI: 10.18653/v1/P19-1042.

[12] C. Brockett, W. B. Dolan, and M. Gamon, "Correcting ESL errors using phrasal SMT techniques," in *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, Sydney, Australia: Association for Computational Linguistics, Jul. 2006, pp. 249–256. DOI: 10.3115/1220175.1220207.

[13] R. Dale, I. Anisimoff, and G. Narroway, "HOO 2012: A report on the preposition and determiner error correction shared task," in *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, Montréal, Canada: Association for Computational Linguistics, Jun. 2012, pp. 54–62.

[14] C. Leacock, M. Chodorow, M. Gamon, and J. Tetreault, *Automated Grammatical Error Detection for Language Learners, Second Edition*. Jan. 2010, vol. 7. DOI: 10.2200/S00275ED1V01Y201006HLT009.

[15] F. Olsson and M. Sahlgren, *We need to talk about data: The importance of data readiness in natural language processing*, 2021. arXiv: 2110.05464 [cs.CL].

[16] A. Solyman *et al.*, "Optimizing the impact of data augmentation for low-resource grammatical error correction," *Journal of King Saud University-Computer and Information Sciences*, vol. 35, no. 6, p. 101 572, 2023.

[17] Z. Wan, X. Wan, and W. Wang, "Improving grammatical error correction with data augmentation by editing latent representation," in *Proceedings of the 28th International Conference on Computational Linguistics*, 2020, pp. 2202–2212.

[18] C. Napoles, M. Nădejde, and J. Tetreault, "Enabling robust grammatical error correction in new domains: Data sets, metrics, and analyses," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 551–566, 2019.

[19] E. F. Tjong Kim Sang and F. De Meulder, "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition," in *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, 2003, pp. 142–147.

[20] C. Napoles, K. Sakaguchi, and J. Tetreault, "Jfleg: A fluency corpus and benchmark for grammatical error correction," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, Valencia, Spain: Association for Computational Linguistics, 2017, pp. 229–234.

[21] A. Yamada, S. Davidson, P. Fernández-Mira, A. Carando, K. Sagae, and C. Sánchez-Gutiérrez, "Cows-l2h: A corpus of spanish learner writing," *Research in Corpus Linguistics*, vol. 8, no. 1, pp. 17–32, 2020.

[22] Y. Wang, Y. Wang, K. Dang, J. Liu, and Z. Liu, "A comprehensive survey of grammatical error correction," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 12, no. 5, pp. 1–51, 2021.

[23] D. Dahlmeier and H. T. Ng, "Better evaluation for grammatical error correction," in *Proceedings of the 2012 conference of the north american chapter of the association for computational linguistics: human language technologies*, 2012, pp. 568–572.

[24] C. Bryant, M. Felice, and E. Briscoe, "Automatic annotation and evaluation of error types for grammatical error correction," Association for Computational Linguistics, 2017.

[25] E. Reiter, "A structured review of the validity of BLEU," *Computational Linguistics*, vol. 44, no. 3, pp. 393–401, 2018.

[26] C. Napoles, K. Sakaguchi, M. Post, and J. Tetreault, *Gleu without tuning*, 2016. arXiv: 1605.02592 [cs.CL].

[27] N.-R. Han, M. Chodorow, and C. Leacock, "Detecting errors in english article usage by non-native speakers," *Natural Language Engineering*, vol. 12, pp. 115–129, Jun. 2006. DOI: 10.1017/S1351324906004190.

[28] M. Chodorow, J. Tetreault, and N.-R. Han, "Detection of grammatical errors involving prepositions," in *Proceedings of the 4th ACL-SIGSEM Workshop on Prepositions*, Prague, Czech Republic: Association for Computational Linguistics, Jun. 2007, pp. 25–30.

[29] G. Berend, V. Vincze, S. Zarrieß, and R. Farkas, "LFG-based features for noun number and article grammatical errors," in *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 62–67.

[30] J. Lee and S. Seneff, "Correcting misuse of verb forms," in *Proceedings of ACL-08: HLT*, Columbus, Ohio: Association for Computational Linguistics, Jun. 2008, pp. 174–182.

[31] D. Alikaniotis and V. Raheja, "The unreasonable effectiveness of transformer language models in grammatical error correction," *arXiv preprint arXiv:1906.01733*, 2019.

[32] S. Flachs, F. Stahlberg, and S. Kumar, "Data strategies for low-resource grammatical error correction," in *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, Online: Association for Computational Linguistics, Apr. 2021, pp. 117–122.

[33] S. Maity, A. Deroy, and S. Sarkar, *Exploring the capabilities of prompted large language models in educational and assessment applications*, 2024. arXiv: 2405.11579 [cs.CL].

[34] J. An *et al.*, "Evaluating performance of llama2 large language model enhanced by qlora fine-tuning for english grammatical error correction," in *Database and Expert Systems Applications*, Cham: Springer Nature Switzerland, 2024, pp. 194–206.

[35] A. Katinskaia and R. Yangarber, "GPT-3.5 for grammatical error correction," in *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, Torino, Italia: ELRA and ICCL, May 2024, pp. 7831–7843.

[36] G. Sutter Pessurno de Carvalho, "Multilingual grammatical error detection and its applications to prompt-based correction," M.S. thesis, University of Waterloo, 2024.

[37] M. Heilman, A. Cahill, N. Madnani, M. Lopez, M. Mulholland, and J. Tetreault, "Predicting grammaticality on an ordinal scale," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Baltimore, Maryland: Association for Computational Linguistics, 2014, pp. 174–180.

[38] S. Coyne, K. Sakaguchi, D. Galvan-Sosa, M. Zock, and K. Inui, *Analyzing the performance of gpt-3.5 and gpt-4 in grammatical error correction*, 2023. arXiv: 2303.14342 [cs.CL].

[39] T. Ge, F. Wei, and M. Zhou, *Reaching human-level performance in automatic grammatical error correction: An empirical study*, 2018. arXiv: 1807.01270 [cs.CL].

[40] Z. Liu, X. Yi, M. Sun, L. Yang, and T.-S. Chua, "Neural quality estimation with multiple hypotheses for grammatical error correction," in *Proceedings of NAACL*, 2021.

[41] R. Grundkiewicz and M. Junczys-Dowmunt, *Near human-level performance in grammatical error correction with hybrid machine translation*, 2018. arXiv: 1804.05945 [cs.CL].

[42] Y. Wu *et al.*, *Google's neural machine translation system: Bridging the gap between human and machine translation*, 2016. arXiv: 1609.08144 [cs.CL].

[43] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[44] R. Alonso. "Diferencias entre ia, machine learning y deep learning." (Jan. 2025), [Online]. Available: https://hardzone.es/tutoriales/rendimiento/diferencias-ia-deep-machine-learning/.

[45] W. Yin, K. Kann, M. Yu, and H. Schütze, *Comparative study of cnn and rnn for natural language processing*, 2017. arXiv: 1702.01923 [cs.CL].

[46] A. Vaswani *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[47] S. Raschka, *Build A Large Language Model (From Scratch)*. Manning, 2024.

[48] C. Huyen, *AI Engineering*. USA: O'Reilly Media, 2025.

[49] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, *Qlora: Efficient finetuning of quantized llms*, 2023. arXiv: 2305.14314 [cs.LG].

[50] E. J. Hu *et al.*, *Lora: Low-rank adaptation of large language models*, 2021. arXiv: 2106.09685 [cs.CL].

[51] Y. Zhang, "Encoder-decoder models in sequence-to-sequence learning: A survey of rnn and lstm approaches," *Applied and Computational Engineering*, vol. 22, pp. 218–226, 2023.

[52] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[53] Meta AI, *Llama 3.2: Revolutionizing edge ai and vision with open, customizable models*, https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/, 2024.

[54] G. Team *et al.*, *Gemma 3 technical report*, 2025. arXiv: 2503.19786 [cs.CL].

[55] V. Koti, *From theory to code: Step-by-step implementation and code breakdown of gpt-2 model*, *Medium*, 2023. [Online]. Available: https://medium.com/@vipul.koti333/from-theory-to-code-step-by-step-implementation-and-code-breakdown-of-gpt-2-model-7bde8d5cecda.

[56] D. Han, M. Han, and U. team, *Unsloth*, 2023. [Online]. Available: http://github.com/unslothai/unsloth.

[57] C. Napoles, K. Sakaguchi, M. Post, and J. Tetreault, "Ground truth for grammatical error correction metrics," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 588–593.

[58] K. W. Church, "Word2vec," *Natural Language Engineering*, vol. 23, no. 1, pp. 155–162, 2017.

[59] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[60] P. Dhote. "Seq2seq encoder-decoder lstm model." (2020), [Online]. Available: https://pradeep-dhote9.medium.com/seq2seq-encoder-decoder-lstm-model-1a1c9a43bbac.

[61] K. Ercikan, "Limitations in sample-to-population generalizing," in *Generalizing from educational research*, Routledge, 2009, pp. 221–244.

[62] I. Arapakis, S. Park, and M. Pielot, "Impact of response latency on user behaviour in mobile web search," in *Proceedings of the 2021 Conference on Human Information Interaction and Retrieval*, ser. CHIIR '21, ACM, Mar. 2021, pp. 279–283. DOI: 10.1145/3406522.3446038.

[63] M. Dodman and G. Barbiero, "Creative commons attribution 4.0 international (cc by sa 4.0)," 2013.

[64] A. Sinclair, “License profile: Apache license, version 2.0,” *IFOSS L. Rev.*, vol. 2, p. 107, 2010.

[65] *Regulation (EU) 2016/679 of the European Parliament and of the Council, General Data Protection Regulation (GDPR)*, Official Journal of the European Union, L 119, 4 May 2016, pp. 1–88, 2016.

A. PLANNING

A.1. Task description

The project was planned and done according different tasks.

- Task 1-Setting the objective: The first thing to do is to set a topic and its objective.
  - Task 1.1-Planning a problem: Planning a problem and how to solve it.
  - Task 1.2-Research of solutions: Searching for possible processes to develop the goal of the project.
  - Task 1.3-Getting familiar with Deep Learning and Large Language Models.

- Task 2: Processing the data: For the proper development of the project it is important to understand the dataset, cleaning it and preparing it for the models.
  - Task 2.1-Getting familiar with the data: Understand what kind of information has, and extract the relevant data for the project.
  - Task 2.2-Analysis of the data: Make the Exploratory Data Analysis, making visualizations of the data, trying to understand how it behaves.
  - Task 2.3-Cleaning the data: Dealing with missing values and other issues in the dataset.
  - Task 2.4-Preparing the data to be used as input: Formatting the essays so they can be fitted as the input of different models.

- Task 3: Modeling, training and fine-tuning: Select the models, implement them and training or fine-tune them.
  - Task 3.1: Selection of the models: Research and select the models and architectures that fit the most for the grammar error correction task.
  - Task 3.2-Implementation of the models: Implement the architectures of the models or load a pre-trained version of them.
  - Task 3.3-Training and fine-tuning: Depending on the model, train or fine-tune them using different approaches, tracking the process to control any issue.

- Task 4: Inference and evaluation of the models.
  - Task 4.1-Inference setup: Preparing the environment for an efficient inference to obtain the test results.

  - Task 4.2-Evaluation of the models: Use the selected metrics to measure the performance of the models, in a general and in a specific way.
  - Task 4.3-Selection of the final model.

- Task 5: Graphical User Interface
  - Task 5.1-Front-end development: Visualization of the corrections and intuitive application interface.
  - Task 5.2-Back-end development: Connection between the application and the model and the functionalities of the interface.

- Task 6: Preparation of the final report: compile the work into a concluding report and prepare the project defense.
  - Task 6.1: Structuring and planning the document: Designing the structure and content of the final report.
  - Task 6.2: Drafting the report: Writing the initial full draft of the final report.
  - Task 6.3: Reviewing and revising the report: Refining, editing, and polishing the report to yield the final version. It also includes obtaining feedback from the advisor.
  - Task 6.4: Preparing the defense: Creating a presentation to summarize and defend the key parts of the work. It also involves practicing the delivery and preparing for potential questions.

A.2. Task sequencing

The task that has been explained above, are now indicated how much time does it took to develop them and which were the other task that depend in order to be done.

TABLE A.1. TASKS DURATIONS AND DEPENDENCIES.

Task	Duration	Dependencies
Task 1-Setting the objective		
Task 1.1-Planning a problem	1 week	
Task 1.2-Research of solutions	3 weeks	1.1
Task 1.3-Getting familiar with DL and LLMs	2 weeks	1.1, 1.2
Task 2: Processing the data		
Task 2.1-Getting familiar with the data	1 week	1.2
Task 2.2-Analysis of the data	1 week	2.1
Task 2.3-Cleaning the data	2 week	2.2
Task 2.4-Preparing the data to be used as input	1 week	2.2, 3.1
Task 3: Modeling, training and fine-tuning		
Task 3.1: Selection of the models	1 week	1.2, 1.3
Task 3.2-Implementation of the models	5 weeks	3.1
Task 3.3-Training and fine-tuning	5 weeks	3.2
Task 4: Inference and evaluation of the models		
Task 4.1-Inference setup	2 week	3.2
Task 4.2-Evaluation of the models	5 weeks	4.1, 3.3
Task 4.3-Selection of the final model	1 week	4.2
Task 5: Graphical User Interface		
Task 5.1-Front-end development	1 week	4.3
Task 5.2-Back-end development	1 week	4.3
Task 6: Preparation of the final report		
Task 6.1: Structuring and planning the document	1 week	5.1, 5.2
Task 6.2: Drafting the report	6 weeks	6.1
Task 6.3: Reviewing and revising the report	3 week	6.2
Task 6.4: Preparing the defense	1 week	6.3

As shown in Table A.1, the tasks that required the most time were those involving code development and those dependent on training and inference durations. Code creation often necessitates meticulous debugging and fine-tuning to ensure functionality, which can be time-consuming. Similarly, tasks reliant on training and inference times are inherently lengthy due to the computational demands of processing large datasets and fine-tuning models. These combined factors contributed significantly to the overall time expenditure in the project.

A.3. Gantt chart

For a detailed overview of task distribution and the project's total duration, a Gantt chart can illustrate the structure and dependencies of various tasks. Note that task 3.1, which

is model selection is given a duration of one week because most of the models were selected in that period of time. In reality, it was a task that depends on the publication of new open-source models, like Gemma 3 which released on March 10, 2025.

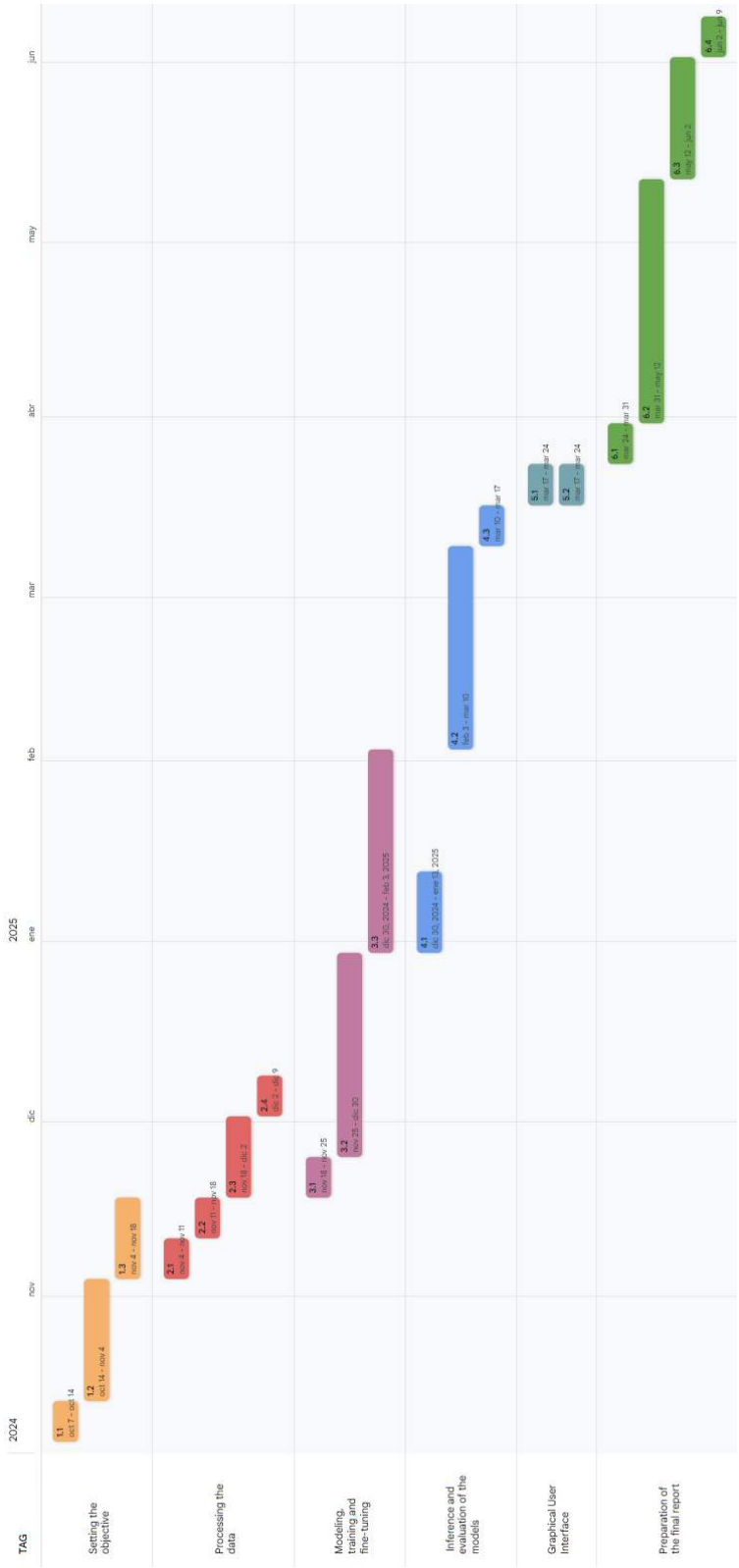

Fig. A.1. Gantt chart.

B. BUDGET

The undertaking of this initiative has necessitated the allocation of fiscal resources, which are enumerated below. They have been divided into two broad categories: expenditures related to human capital and outlays associated with material assets.

B.1. Human resources

This project was developed by a university student pursuing a Dual Bachelor in Data Science and Engineering and Telecommunication Technologies Engineering at Universidad Carlos III. The student's work was supervised by his advisor, Pedro Manuel Moreno Marcos. To accurately allocate funds for the human resources involved, the hours contributed by both individuals must be considered.

The project spanned approximately a total of 35 weeks. During this period, the student and his advisor held 17 meetings, averaging 1 hour per meeting, totaling approximately 17 hours. Additionally, the advisor dedicated extra time outside these meetings to review drafts and provide feedback via email—amounting to 1 extra hour per month and an additional 10 hours in the final month, totaling 18 hours. Furthermore, the advisor spent an extra 12 hours assisting with the report of the project, defense and related activities. Overall, the advisor contributed 47 hours of guidance to the project.

For financial estimation, the hourly pay rate for an experienced engineer or technical advisor in this field is approximately 24€ in Madrid according to Glassdoor. Assuming a 40-hour work week over 4 weeks, this equates to a monthly salary of around 3,840€. Therefore, the advisor's 47 hours of work amount to an estimated cost of 1,128€.

For a student engineer, an annual base salary of 30,000€ is assumed based on Glassdoor, Glassdoor, translating to 2,500€ per month or 15.60€ per hour (based on a 160-hour work month). For cost calculations, this rate is rounded up to €16 per hour.

The student was solely responsible for the project workload, estimated at about 10 hours per week for 35 weeks, resulting in a total of 350 hours. Consequently, the estimated cost of human resources attributable to the student is 5,600€.

B.2. Hardware and software costs

In terms of hardware, the project has been carried out with an MacBook Pro with M2 and 32 GB RAM, which costs 1,795€. The depreciation cost over the duration of the project is estimated to be 170.95€, considering a lifetime of 7 years.

$$\text{Amortization} = \left(\frac{\text{cost} \times \text{time of use}}{\text{lifetime}}\right) = \left(\frac{1795 \times \frac{8}{12}}{7}\right) = 170.95\,€$$

With respect to the software, the following open source programs have been used, therefore they do not involve an additional cost:

- Google Colab: A free to use online development environment for Python software.
- Google Sheets: A free online spreadsheet software.
- Google Drive: Free online storage and file sharing platform.
- Overleaf: A free online LaTeX editor for writing and collaborating on technical documents.

All libraries and models were open-source so they do not contribute to the costs, but the estimated cost of consumable materials such as access to the internet, light and other supplies would represent an additional 10% of the total cost.

B.3. Total cost

In Table B.1 the final costs of the project can be seen, adding up all of the human resources needed and their underlying costs, the hardware cost and the additional cost of supplies.

TABLE B.1. TOTAL BUDGET.

Type	Description	Cost
Human resources	Tutor	1,128€
Human resources	Student	5,600€
Hardware costs	Laptop	170.95€
Cost of consumables	Other (10%)	766.55€
Total costs		7,665.5€

C. REGULATORY FRAMEWORK

Multiple regulations and numerous laws have substantially shaped the development and execution of this project. Two of the most significant regulatory considerations were data privacy and copyright licensing. Data privacy regulations, such as the GDPR and similar laws, impose strict requirements on the collection, storage, and use of personal information. To comply with these laws is crucial to ensure robust data privacy protections, including obtaining proper consent, anonymizing data, securely managing sensitive information, and adhering to data retention and deletion policies. This appendix addresses two of the most significant regulatory factors: data privacy and Creative Commons licensing.

The Creative Commons CC-BY (Attribution) license [63] facilitates the reuse and redistribution of these open educational resources. Under this license, others may remix, adapt, and build upon the data and materials in this project, even for commercial purposes, as long as proper attribution is given to the original source. This unrestricted license maximizes the impact and utility of this work for the educational community. Any images included in the project carry Creative Commons licenses that permit reuse with proper attribution.

Regarding software elements or datasets that fall under an Apache License 2.0 [64], this open-source license allows users to freely use, modify, and distribute the software, requiring only that they include a copy of the license and note any modifications. This provides flexibility for further development and innovation while protecting the rights of contributors. The COWS-L2H dataset is shared under this license, permitting its use for research intentions.

The COWS-L2H dataset from the University of California is governed by stringent policies designed to ensure privacy, ethics, and the responsible use of student data. These data protection and privacy policies specify that student data will be collected and utilized exclusively for educational research and the enhancement of teaching, all while ensuring that personal information remains private and secure. The anonymization process adopted by the University of California involves replacing or modifying information in the essays that could be used to identify the authors. Students provided informed consent, allowing their anonymized data to be used for research purposes [21]. This approach safeguards individual privacy while enabling valuable research that can contribute to educational advancements.

The General Data Protection Regulation (GDPR) [65] does not apply to this project because the data obtained was already anonymized. According to GDPR guidelines, once data is truly anonymized and cannot be traced back to individual identities, it falls outside the scope of these regulations. This ensures that the privacy of individuals is protected while allowing the project to proceed without the constraints imposed by GDPR compliance.

All technical tools and software utilized in this project are governed by open-source licenses and are freely accessible. This includes programming languages such as Python, development environments like Jupyter Notebooks, and code editors such as Google Colab. By leveraging open and freely available tools, the project minimizes costs and lowers barriers to access, allowing other researchers and educators to replicate or expand upon this work if they choose to do so. This approach promotes collaboration and innovation within the community, fostering further advancements in the field.

D. SOCIO-ECONOMIC FRAMEWORK

The work done in this project presents several socio-economic benefits for students, professors, and educational institutions. Below, the beneficial aspects for each of these groups are explained.

The AI tool allows for the rapid detection and correction of grammatical errors, which reduces the workload for teachers and enables them to focus on more complex or time-consuming tasks such as content planning, personalized instruction, or student mentoring. From an economic perspective, this translates to greater efficiency in the allocation of human resources, reducing the need for manual error correction and repetitive feedback, which can be especially valuable in large classrooms or online courses. In the long term, this can lead to reduced operational costs for educational institutions, particularly when scaling up programs or implementing distance learning platforms.

Teachers can also use this tool to identify frequent errors across students' essays, allowing them to provide targeted feedback more efficiently. This aggregated data could support data-driven texts improvement, reducing the need for remedial classes and boosting overall learning outcomes with fewer resources.

Universities and educational institutions benefit economically from this tool by improving academic performance metrics, which are often tied to funding, reputation, and student retention rates. The ability to track writing performance across courses and over time enables better academic planning and assessment. Institutions investing in AI-enhanced tools like this one may also improve their competitiveness in attracting students and securing research or innovation funding related to digital transformation in education.

Students also stand to benefit significantly. Beyond receiving continuous feedback regardless of the teacher's availability, they can improve their writing faster, which may lead to better academic outcomes and enhanced career readiness. Improved communication skills are directly linked to greater employability and earning potential, especially in fields where writing is key, such as law, journalism, or international business.

On a broader scale, a society that prioritizes and achieves proficiency in writing with minimal grammatical errors stands to benefit across multiple dimensions. Clear and correct writing enhances effective communication, facilitating understanding and minimizing misunderstandings among individuals, businesses, and communities. This clarity is crucial in education, where students can better grasp complex concepts, and in the workplace, where precise directives can improve productivity and reduce costly errors. Furthermore, polished writing can elevate a society's cultural and intellectual discourse, as ideas are more aptly articulated and disseminated, fostering a more enlightened and engaging exchange of thoughts.

Moreover, when individuals and institutions communicate with grammatical accuracy,

it reflects professionalism and credibility, enhancing their reputation both domestically and internationally. This aspect of clear communication is especially critical in our globally interconnected world, where international relations and commerce thrive on mutual comprehension. Additionally, a society that values well-written communication implicitly values education, potentially leading to a more informed, literate, and economically productive populace.

All of the aforementioned advantages align with the Sustainable Development Goals, specifically with Goal 4: Quality Education, by promoting digital transformations that enhance learning outcomes. Furthermore, the economic efficiency gained from implementing such tools supports Goal 8: Decent Work and Economic Growth, as it contributes to higher productivity, skill development, and better employment prospects.

uc3m | Universidad Carlos III de Madrid

DECLARATION OF USE OF GENERATIVE IA IN BACHELOR THESIS (TFG)

I have used Generative AI in this work

Check all that apply:

YES	NO

If you have ticked YES, please complete the following 3 parts of this document:

Part 1: Reflection on ethical and responsible behaviour

Please be aware that the use of Generative AI carries some risks and may generate a series of consequences that affect the moral integrity of your performance with it. Therefore, we ask you to answer the following questions honestly (*please tick all that apply*):

Question		
1. In my interaction with Generative AI tools, I have submitted **sensitive data** with the consent of the data subjects.		
YES, I have used this data with permission	NO, I have used this data without authorisation	NO, I have not used sensitive data
2. In my interaction with Generative AI tools, I have submitted **copyrighted materials** with the permission of those concerned.		
YES, I have used these materials with permission	NO, I have used these materials without permission	NO, I have not used protected materials
3. In my interaction with Generative AI tools, I have submitted **personal data** with the consent of the data subjects.		
YES, I have used this data with permission	NO, I have used this data without authorisation	NO, I have not used personal data

4. My use of the Generative AI tool has **respected its terms of use**, as well as the essential ethical principles, not being maliciously oriented to obtain an inappropriate result for the work presented, that is to say, one that produces an impression or knowledge contrary to the reality of the results obtained, that supplants my own work or that could harm people.	
YES	NO

If you **did NOT** have the permission of those concerned in any of questions 1, 2 or 3, briefly explain why *(e.g. "the materials were protected but permitted use for this purpose" or "the terms of use, which can be found at this address (...), prevent the use I have made, but it was essential given the nature of the work".*

Part 2: Declaration of technical use

Use the following model statement as many times as necessary, in order to reflect all types of iteration you have had with Generative AI tools. Include one example for each type of use where indicated: *[Add an example].*

I declare that I have made use of the Generative AI system ChatGPT (GPT-4o)

Documentation and drafting:

- *Supporting reflection in relation to the development of the work: iterative process of analysing alternatives and approaches using AI*

- *Revision or rewriting of previously drafted paragraphs*

I have used it for the rewriting of some parographs to I have requested the rewriting of paragraphs to shorten and refine the conclusions and explanations while preserving the original ideas already written.

- *Search for information or answers to specific questions*

The questions asked were aimed at obtaining clear definitions.

- *Bibliography search*
- *Summary of bibliography consulted*
- *Translation of texts consulted*

Develop specific content

Generative AI has been used as a support tool for the development of the specific content of the dissertation, including:

- *Assistance in the development of lines of code (programming)*

I asked for some parts of my own code to be optimized in order to speed up inference or reduce memory usage.

- *Generation of diagrams, images, audios or videos*

- *Optimisation processes*

I have used it for optimizing some parts of the code.

- *Data processing: collection, analysis, cross-checking of data...*

- *Inspiration of ideas in the creative process*

I have used it when I have my own ideas about how to do something and I want another point of view.

- *Other uses linked to the generation of specific points of the specific development of the work*

If you think it is necessary to include it, add it in each case:

using the *prompt* ...
(write the request made to the IAG)
having as interaction ...
(describe what interaction you made with the IAG after the prompt's response)

Example: ***I declare that*** *I have made use of the Generative AI system* ***Chat GPT 3.5*** *to* ***search for information*** *using* ***the prompt: "Tell me a concrete example that illustrates the application of the urban planning proposal of the 15-minute City in Spain"****, having as interaction* ***the proposition of the concrete example of the Chamartín district that has been reflected in the report****.*

Part 3: Reflection on utility

Please provide a personal assessment (free format) of the strengths and weaknesses you have identified in the use of Generative AI tools in the development of your work. Mention if it has helped you in the learning process, or in the development or drawing conclusions from your work.

Generative AI has the advantage of performing very well in areas such as writing and code generation. However, many of the responses it provided were not directly applicable, and a deep understanding of what is expected from these models is necessary. It can support the learning process by addressing specific doubts and offering alternative perspectives, but caution is needed, as the answers are not always correct. Critical thinking is essential when using these tools.