UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

School of Informatics

Text denormalization system based on neural models for Spanish and Catalan

End of Degree Project

Bachelor's Degree in Informatics Engineering

AUTHOR: Vicente Cantero, Violeta

Tutor: Segarra Soriano, Encarnación

Cotutor: Hurtado Oliver, Lluis Felip

Cotutor: Ahuir Esteve, Vicent

ACADEMIC YEAR: 2023/2024

Abstract

Automatic speech recognition systems usually generate a sequence of lowercase words without punctuation. To improve human readability and facilitate further processing of the output with NLP tools, it is usually desirable to fully restore the text to the standard orthographic conventions for the target language.

This task of denormalization, known as Inverse Text Normalization (ITN), is the process of converting the output of an automatic speech recognition system into its corresponding written form. This includes predicting punctuation and capitalization based on context, as well as making proper use of accents and spacing.

A corpus of journalistic news in Catalan and Spanish, DACSA, is available, providing a collection of (article, summary) pairs. The corpus contains news from different journalistic sources. This work proposes using neural models for the denormalization of articles and summaries from the DACSA corpus. To achieve this, a bilingual Spanish-Catalan language model is pre-trained, and a fine-tuning process for the denormalization task is performed.

To obtain a normalized version of the corpus, various automatic (self-objective) normalization and noise tasks are applied, such as: converting the text to uppercase, lowercase, or a combination of these, removing punctuation, or introducing spelling errors.

Keywords: text denormalization, journalistic texts, transformers, Catalan, Spanish

Resumen

Los sistemas de reconocimiento automático de voz suelen generar una secuencia de palabras en minúsculas y sin puntuación. Para mejorar la legibilidad humana y facilitar el procesamiento posterior de la salida con herramientas de Procesamiento de Lenguaje Natural (PLN), es generalmente deseable restaurar completamente el texto a las convenciones ortográficas estándar del idioma objetivo.

Esta tarea de desnormalización, conocida como Normalización Inversa del Texto (ITN), es el proceso de convertir la salida de un sistema de reconocimiento automático de voz en su forma escrita correspondiente. Esto incluye predecir la puntuación y la capitalización según el contexto, así como hacer un uso adecuado de los acentos y los espacios.

El corpus de noticias periodísticas en catalán y español llamado DACSA se ha usado para tal fin, proporcionando una colección de pares (artículo, resumen). El corpus contiene noticias de diferentes fuentes periodísticas. Este trabajo propone el uso de redes neuronales para la desnormalización de artículos y resúmenes del corpus DACSA. Para lograr esto, se preentrena un modelo de lenguaje bilingüe español-catalán y se realiza un proceso de ajuste fino (fine-tuning) para la tarea de desnormalización.

Para obtener una versión normalizada del corpus, se aplican diversas tareas automáticas de normalización y ruido, tales como: convertir el texto a mayúsculas, minúsculas o una combinación de ambas, eliminar la puntuación o introducir errores ortográficos.

Palabras clave: desnormalización del texto, textos periodísticos, transformers, catalán, español

Resum

Els sistemes de reconeixement automàtic de veu solen generar una seqüència de paraules en minúscules i sense puntuació. Per millorar la llegibilitat humana i facilitar el processament posterior de la sortida amb eines de Processament de Llenguatje Natural (PLN), generalment és desitjable restaurar completament el text a les convencions ortogràfiques estàndard de l'idioma objectiu.

Aquesta tasca de desnormalització, coneguda com a Normalització Inversa del Text (ITN), és el procés de convertir la sortida d'un sistema de reconeixement automàtic de veu en la seva forma escrita corresponent. Això inclou tant predir la puntuació i la capitalització segons el context, com fer un ús adequat dels accents i els espais.

EL corpus de notícies periodístiques en català i espanyol anomenat DACSA s'ha emprat per tal fi, proporcionant una col · lecció de parells (article, resum). El corpus conté notícies de diferents fonts periodístiques. Aquest treball proposa l'ús de models neuronals per a la desnormalització d'articles i resums del corpus DACSA. Per aconseguir això, es preentrena un model de llenguatge bilingüe espanyol-català i es realitza un procés d'ajust fi (fine-tuning) per a la tasca de desnormalització.

Per obtenir una versió normalitzada del corpus, s'apliquen diverses tasques automàtiques de normalització i soroll, com ara: convertir el text a majúscules, minúscules o una combinació d'aquestes, eliminar la puntuació o introduir errors ortogràfics.

Paraules clau: desnormalització del text, textos periodístics, transformadors, català, espanyol

Acknowledgments

A mi familia por acompañarme en estos duros años, animándome cuando pensaba que no podía seguir.

A mis hermanos por abrirme camino en la vida y servirme de ejemplo.

A los muchos amigos que han estado conmigo estos años; a Andrea, Lucía, María, Ronnie y Carles en especial.

A la gente maravillosa que he tenido la suerte de conocer fuera de Mara, Ewa y amigos del CERN.

A Jesse por creer en mí

y a la vida por darme tantas oportunidades.

Contents

List of Figures

List of Tables

1 Introduction

Contents

1.1 Motivation

Large language models (LLMs) have gained popularity recently as big companies have started developing their own and people have realized how much easier some tasks can become. Tedious tasks can now be done using Artificial Intelligence, which saves time and unnecessary effort. Artificial intelligence models are:

- efficient in handling repetitive and time-consuming tasks quickly and accurately, allowing individuals and organizations to focus on more strategic activities
- accessible to the general public now that many companies are releasing their models online
- customizable and versatile, as they can be fine-tuned to specific needs being used to
- continuously improving, ensuring that their capabilities are constantly expanding and providing users with increasingly sophisticated tools.

In particular, Natural Language Processing (NLP) is one of the most fascinating fields for human-machine interaction due to its potential and convenience when communicating with a machine. NLP enhances user experience by enabling more natural and intuitive interactions, such as voice commands and conversational agents. Given the widespread presence of text and language in human communication, NLP plays a crucial role in understanding, interpreting, and generating

human language. The abundance of textual data from social media, books, articles, websites, and customer feedback makes NLP invaluable for extracting meaningful information. Advances in deep learning have significantly improved NLP performance in tasks like language translation, sentiment analysis, and text generation. The widespread applications of NLP include powering chatbots and virtual assistants in customer service, and breaking down language barriers through machine translation and speech-to-text technologies, making content accessible to a broader audience.

1.2 Goals

The primary goal of this project is to develop a bilingual model capable of transforming denormalized text into coherent, human-like text. This involves correcting syntactic and orthographic errors, and adding appropriate punctuation, accents, capitalization, and other necessary modifications to ensure that the output is logically and grammatically sound. The specific goals of this project are as follows:

1. **Training a Bilingual Model:** Develop a model that supports both Catalan and Spanish to promote linguistic diversity and address the digital marginalization of minority languages like Catalan. The lack of resources for training models in Catalan perpetuates inequities between languages, and this project aims to bridge that gap by creating a robust, multilingual framework.

2. **Generating Grammatically Correct Text:** Utilize neural networks, in particular deep learning models, to produce grammatically correct text, properly punctuated, accurately capitalized, and appropriately accented. Ensure proper usage of language to help preserve linguistic heritage, particularly for minority languages. Create a normalizer tool that can correct texts lacking punctuation, capitalization, and accents, thereby improving the accuracy and quality of transcriptions from audio-to-text tasks.

3. **Employing Curriculum Learning:** Implement a curriculum learning approach to fine-tune the model by starting with simpler tasks and progressively increasing the complexity as the model learns. This method ensures efficient and effective training by avoiding overwhelming the model with complex tasks from the onset.

Additional Benefits:

A denormalizer tool can significantly enhance the quality of communication on digital platforms, where informal writing and errors are common. By providing correct versions of poorly written texts, it aids in clearer and more effective communication in both professional and casual contexts.

Furthermore, the denormalizer can serve as an educational tool, helping students and language learners understand common mistakes and learn the correct forms. Teachers and educators can use it to provide instant feedback on written assignments, thereby facilitating a better learning experience.

1.3 Thesis structure

This thesis is composed of five main chapters, each designed to provide a comprehensive understanding of the project's scope, methodology, results, and conclusions.

- **Chapter 1: Introduction.** This chapter aims to describe the context and justification for the project, outlining its objectives and the goals intended to be achieved.
- **Chapter 2: State of the Art.** The second chapter provides a theoretical framework essential for understanding the key concepts and developments in the field of natural language processing and text generation. It reviews the evolution of relevant technologies and recent advancements in neural network-based text normalization.
- **Chapter 3: Methodologies.** In this chapter, the datasets used throughout the project are described, followed by an explanation of the methodologies and techniques employed. It includes the detailed architecture of the BART model used for text denormalization, as well as the evaluation metrics and tools applied to assess the model's performance.
- **Chapter 4: Evaluation and Results.** The fourth chapter focuses on the experimental work conducted during the project. It details the procedures followed, challenges encountered, and decisions made throughout the process. The results obtained from the experiments are presented and discussed, providing insights into the model's effectiveness and areas for improvement.
- **Chapter 5: Conclusions.** The final chapter summarizes the key findings and achievements of the project. It reflects on the objectives met, and the challenges faced, and proposes potential directions for future research and development in this field.

1.4 Relevance to Undergraduate Studies

During my Computer Engineering Bachelor, I specialized in Computer Science. To develop this final thesis, I drew on valuable knowledge from the courses in my specialization, which include algorithms, machine learning, artificial intelligence, and optimization techniques, among others. Especially relevant were the programming languages courses such as *Programming Languages, Technologies and Paradigm* (11557) and *Data Structures and Algorithms* (11551), as they provided a strong foundation in programming principles and techniques. The course *Techniques, Tools, and Applications of Artificial Intelligence* was particularly important, as it covered neural networks, which are a critical component of this thesis. Additionally, *Information Storage and Retrieval Systems* and *Perception* were beneficial in understanding data management and processing, which are essential for handling the text data used in this project.

2

State of Art

Contents

2.1 Natural Language Processing

Natural language processing, or NLP, is an interdisciplinary subfield of computer science and linguistics that combines computational linguistics-rule-based modeling of human language with statistical and machine learning models. Its goal is to enable computers and digital devices to recognize, understand and generate text and speech. The technology should not only be able to understand the content, but also the contextual nuances of the language. It can then be able to accurately extract information and insights contained in documents, as well as categorize and organize the documents themselves. NLP is categorized in [1]:

- **Natural Language Understanding**, which aims to understand, extract meaning and discern context through syntactic and semantic analysis of text and speech. This process must

be able to find the particularly important information of a sentence to more appropriately capture its whole meaning, rather than understanding every word individually. Examples of NLU applications are sentiment analysis, speech recognition and spam filtering.

- **Natural Language Generation**, which focuses on understanding and generating text in human language. Its difficulty lies in not only understanding but also deciding which content and in which form shall it be expressed. NLG is used for chatbots, voice assistants and summarization.

The evolution of **Natural Language Processing (NLP)** can be segmented into different periods: symbolic, statistical, and neural NLP.

The beginning of NLP dates back to the conceptual foundation of AI and NLP's goals presented in Alan Turing's paper of 1950 [2]. In that paper, Turing presented an "intelligence criteria", nowadays known as the Turing Test, to determine a machine's ability to show intelligent behaviour equivalent to that of a human. Turing insisted on the fact that the computer does not have to necessarily think like a human, but rather it must simulate intelligence so that it is indistinguishable from human intelligence.

The impact of Turing's paper and his proposed test for assessing machine intelligence catalyzed the mid-20th-century optimism, fostering the belief that the structure of language could be fully captured using sets of handcrafted rules. This optimism was also driven by the necessity of syntactic processing and by the fact that many NLP researchers established status in linguistic and language study, rather than computer science [1]. **Symbolic NLP** focused on emulating natural language understanding by applying a collection of rules to the data it confronted. The field was heavily dominated by symbolic or rule-based approaches, as it was thought that mimicking language could be achieved by dissecting and encoding the rules of language into computer programs. These systems were brittle, limited and difficultly scalable, as human language is inherently complex and variable.

As computational power increased and digital text became more available, the field shifted towards statistical methods. **Statistical NLP** focuses on learning from data, so algorithms were developed to automatically infer the rules of language from large corpora of text. This period saw the rise of machine learning techniques in NLP, with methods such as Hidden Markov Models (HMMs) and the development of machine translation systems, such as statistical machine translation (SMT), which relied on analyzing vast amounts of bilingual text data to learn how to translate

text from one language to another. It led to a significant advancement over the symbolic period because they could handle the ambiguity and variability of natural language more robustly.

The latest phase in the evolution of NLP, called **Neural NLP**, began with the advent of neural networks and deep learning. This period is characterized by the use of models inspired by the structure and function of the human brain, capable of learning complex patterns from data.

The breakthrough came with the development of word embeddings, such as Word2Vec [3] and GloVe [4], which provided a way to represent words as dense vectors in a continuous space, capturing semantic relationships between them. This was a significant departure from previous methods that treated words as isolated units without regard to their context.

The real game-changer was the introduction of architectures like Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) [5], and later, the Transformer model [6], which led to the development of models like BERT [7], GPT [8], and others. These models can process sequences of text and capture long-range dependencies, enabling a much deeper understanding of language context and nuances.

Neural NLP has revolutionized the field, enabling significant improvements in various applications such as machine translation, text summarization, question answering, and sentiment analysis. These models have outperformed traditional methods in numerous tasks, offering a more nuanced and context-aware interpretation of language.

Natural Language Processing, a branch of Artificial Intelligence, lies at the heart of applications and devices that handle summarization, question answering, speech recognition, sentiment analysis, text classification, machine translation... Nowadays, natural language processing is present in the form of digital assistants, voice-operated GPS systems, customer service chatbots, speech-to-text dictation software, and other consumer conveniences.

2.2 Deep learning architectures

Convolutional Neural Networks (CNNs)

Convolutional Neural Networks are a network architecture for deep learning primarily used for analysing spatial data such as images and videos.

Its architecture had a biological inspiration, as the idea was to capture the essence of visual processing in a computational form [9]. Mimicking the visual cortex's approach to feature detection, hierarchy and integration allowed the creation of CNN's most important features, which are the following:

- **Convolutional layers**, responsible for most of the computational work, serve as the cornerstone of a CNN. They are the main layers in charge of feature detection, and they use filters to accomplish this task. Each filter, of length NxN, is designed to be smaller than the original. Convolutional layers are applied to the input image to highlight various features in the image. It is thus called a filter because it systematically transverses the input matrix or image pixels to "filter through" the previous layer.

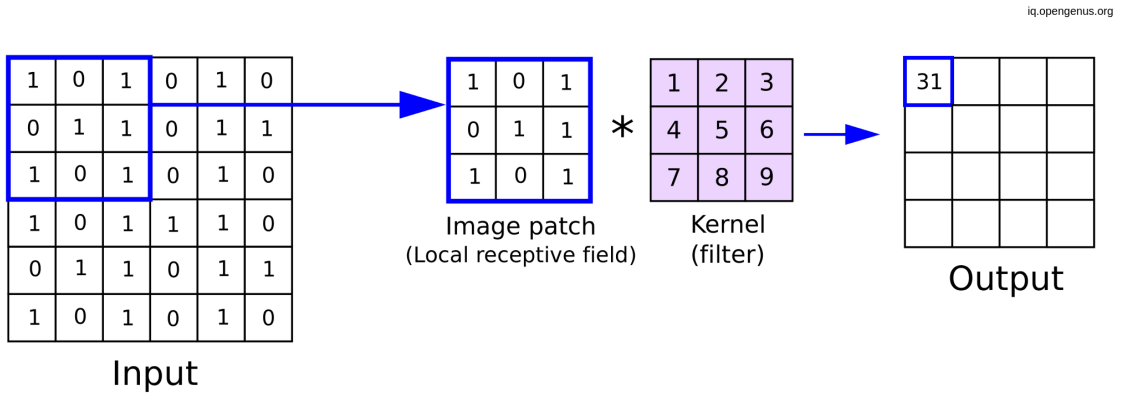

Figure 2.1: *Convolutional layer process*

Figure 2.1 shows how the dot product operation is applied at each position to extract localized features, effectively isolating and capturing specific aspects of the data. This selective operation results in a feature map that highlights the presence of detected attributes across the input space. CNNs learn hierarchies of features in a layer-wise manner, with lower layers capturing basic features like edges and textures, and higher layers identifying more complex features like objects and faces.

- **Pooling layers**, often used after convolutional layers, reduce the spatial dimensions (height and width) of the feature maps to reduce the spatial dimensions of the input (downsampling) and decrease the amount of computation. They are used to go from local information to a more global representation.

  Figure 2.3 shows two common pooling operations; max pooling, which takes the maximum value within a region, and average pooling, which computes the average.

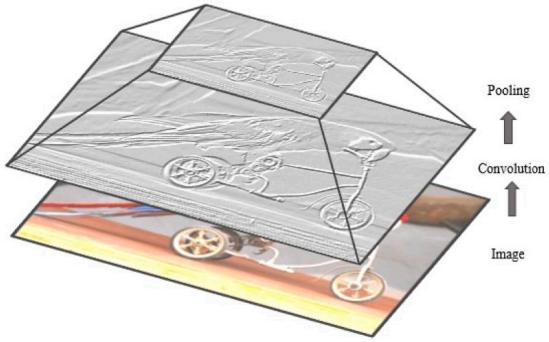

Figure 2.2: *Example of convolution and pooling operation*

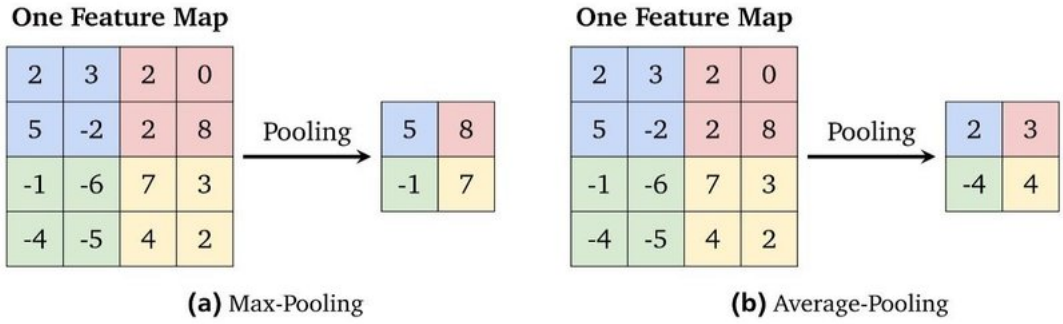

Figure 2.3: *Max and average pooling operations*

- **Fully connected layers** are crucial in neural networks, particularly after several convolutional and pooling layers, where they integrate all extracted features to perform high-level reasoning. These layers play a key role in identifying complex patterns across the entire input space, which is essential for tasks such as image classification. The final fully connected layer, matching the number of output classes, transforms the raw scores into probabilities, thereby making precise class predictions. This setup not only enhances the network's learning capabilities through depth and non-linearity introduced by activation functions like ReLU or sigmoid, but also ensures the versatility and adaptability of the model to various learning tasks.

Although initially proposed by Fukushima in 1988, CNN's widespread adoption was limited by the hardware available at the time for training such networks. In the early 1990s, LeNet, one of the earliest CNNs to use a back-propagation algorithm, was first proposed by Yann LeCun [9], and LeNet-5, a CNN consisting of 2 convolutional layers, 2 sub-sampling layers, 2 fully connected layers and an output layer, achieved state-of-the-art accuracies in the MNIST digit classification task, a handwritten digits dataset [10].

AlexNet, a deeper and wider CNN model compared to LeNet, drastically advanced the field of

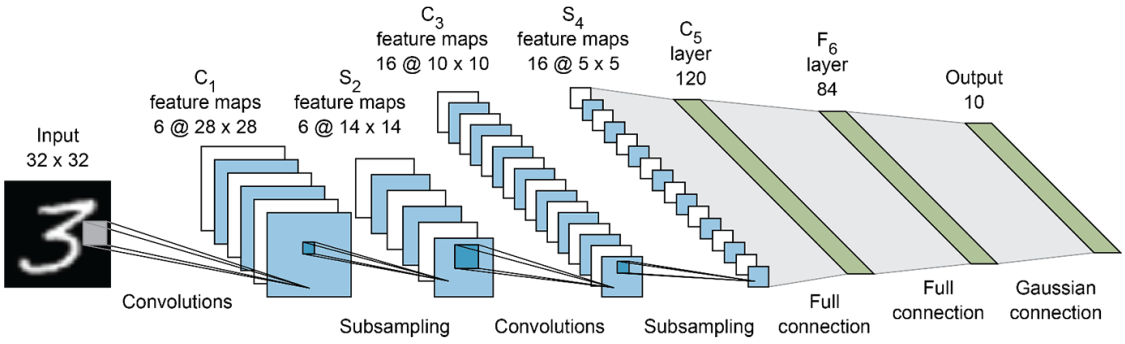

Figure 2.4: *Architecture of LeNet-5, a convolutional NN, here used for digits recognition*

deep learning by winning the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 [11]. Nowadays, 99% accuracy has been reached with CNN for the MNIST dataset [12].

Recurrent Neural Networks (RNNs)

Recurrent Neural Networks' foundational concepts date back to the early 1980s. The creation of these neural networks was driven by the need to overcome certain limitations observed in convolutional neural networks (CNNs). Some of these limitations are [13]:

- **Understanding Sequential Data:** CNNs struggle with sequential data, which requires processing inputs that depend on previous inputs. This limitation is particularly problematic in fields like text generation, machine translation, and stock market predictions, where the order and context significantly influence the output.
- **Flexibility in Input and Output Sizes:** CNNs typically handle fixed-size vectors, limiting their application in tasks where input and output sizes vary. This is a significant constraint in domains such as speech recognition or any dynamic text processing.
- **Dynamic Handling of Sequential Steps:** The architecture of CNNs sets a fixed number of processing steps determined by the number of layers, which is unsuitable for tasks that require adaptability to the lengths of input sequences.
- **Memory Capabilities:** CNNs, being predominantly used for pattern recognition in static data like images, lack an internal memory mechanism to retain information from previous inputs, a critical feature for processing sequences that contain interdependent information.

As a response to these limitations, RNNs were designed with specific characteristics:

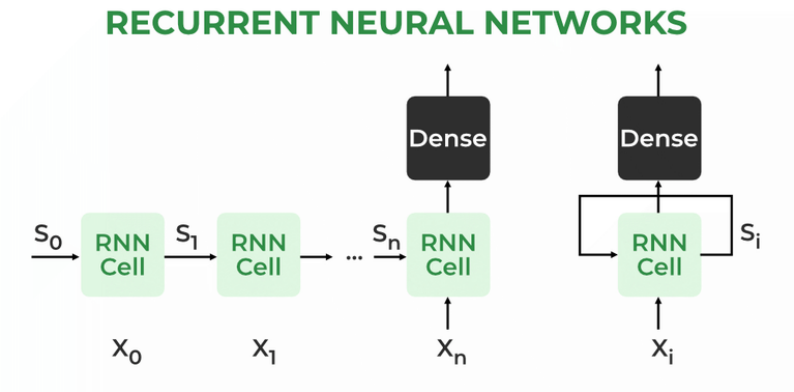

Figure 2.5: *RNN architecture*

- **Internal State or Memory:** Allows RNNs to retain state information from one input to the next, facilitating continuous data processing without resetting between inputs. At each time step, the RNN updates its "hidden state" which is a vector representing the memory of the network. This hidden state, represented by s_i in ***Figure 2.5***, captures information about all the previous inputs up to that point.

- **Flexible Architecture:** Unlike CNNs, RNNs can dynamically adjust to the length of input sequences, making them ideal for varied-length data in real-time processing. RNNs can be unrolled to adapt to the number of steps or to input as many time-series instances as desired. This unrolling characteristic can be seen in ***Figure 2.5***, where the output of a unit serves as the input of the following one, making it possible to add as many time steps as desired.

- **Sequential Processing Capability:** RNNs inherently process data sequentially, allowing each step to depend on the outcomes of previous steps, thereby handling tasks that require understanding of time and sequence effectively. These design features make RNNs particularly suited for applications where understanding the temporal dynamics of data is crucial.

The main problem with RNNs is that they suffer from the vanishing gradient problem, which renders them inefficient when being used with an excessively large number of cells.

Long Short-Term Memory (LSTM)

To mitigate long-term dependencies, Long Short-Term Memory neural networks were created. LSTMs have two different channels for long and short memory, represented with C_t and h_t in ***Figure 2.7.*** The cell state C_t serves as the long-term memory, using the forget and input gate to add or remove information. The hidden state h_t, represents the short-term memory,

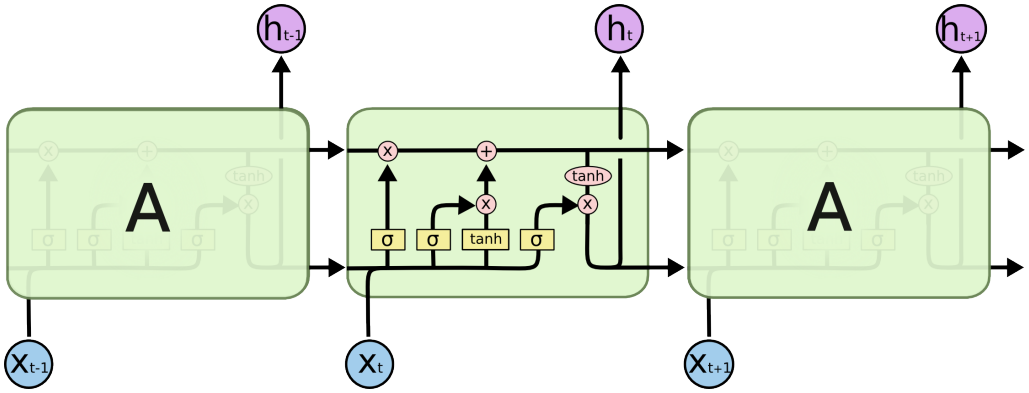

Figure 2.6: *LSTM architecture*

making use of the output gate to filter which parts of the cell state are used in the output at each time step. The input x_i is integrated into the network together with the previous hidden state $h\_t - 1$, influencing both the updates to the cell state C_t and the hidden state h_t. The flow of information is regulated by incorporating mechanisms called gates:

- **Input gate** to control the addition of new information into the cell state.
- **Forget gate** to discard irrelevant information. This gate decides what parts of the previous cell state (the long-term memory) are to be forgotten, which it does by applying a sigmoid function that outputs values between 0 and 1. A value of 0 means "completely forget this", while a value of 1 means "completely retain this".
- **Output gate** to decide what to transfer to the output.

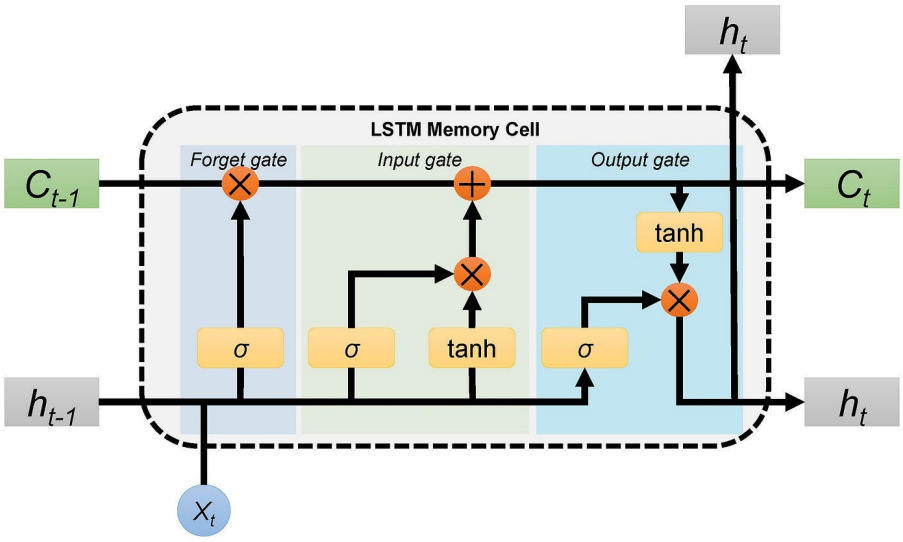

Figure 2.7: *LSTM cell*

RNNs and LSTMs were very advantageous for being able to theoretically process any length of input sequences and were naturally suited for the sequential nature of speech and text. However, their sequential processing nature limited parallelization during training and inference, which became a significant bottleneck as datasets and models grew in size.

Transformer Language Modeling

The Transformer was an architecture proposed by Google researchers in the paper *Attention is all you need* in 2017 [6]. The paper was groundbreaking and led to a rapid shift from the previously dominant sequence modelling architectures, like RNNs and LSTMs, towards attention-based models. The rapid transition was due to the fact that transformers allowed parallel processing of sequences and allowed handling long-range dependencies with ease, thanks to the self-attention mechanism.

The Transformer model simplifies the architecture needed for sequence-to-sequence tasks by eliminating recurrence and convolutions entirely. It relies solely on the attention mechanism to draw global dependencies between input and output, which makes the model less complex and easier to understand and implement. Self-attention is a very relevant mechanism that is used to give context. It refers to the strategy that the model uses to focus on parts of the input sequence to predict the output sequence. In a way, attention allows the model to create some connection/correlation between the input and the output. Self-attention works with the similarity of each word with all of the words in the sentence, including itself.

A transformer consists of two main parts: an encoder and a decoder, which can be used separately or together. While encoder-only models are useful for tasks that require understanding of the input, such as sentence classification and name entity recognition, decoder-only models are suitable for generative tasks, such as text generation. Encoder-decoder models, also known as sequence-to-sequence (seq2seq) models, are good for generative tasks that require an input, such as summarization or translation.

The encoder accepts inputs that represent text and converts them into numerical representations, which are called embeddings. These embeddings are crucial, as they serve as the input to the encoder's sub-layers, enabling the model to process and understand the text.

In **Figure 2.8**, the two encoder sub-layers are clearly visible:

1. **The multi-head attention layer**, which allows the encoder to consider other words in the

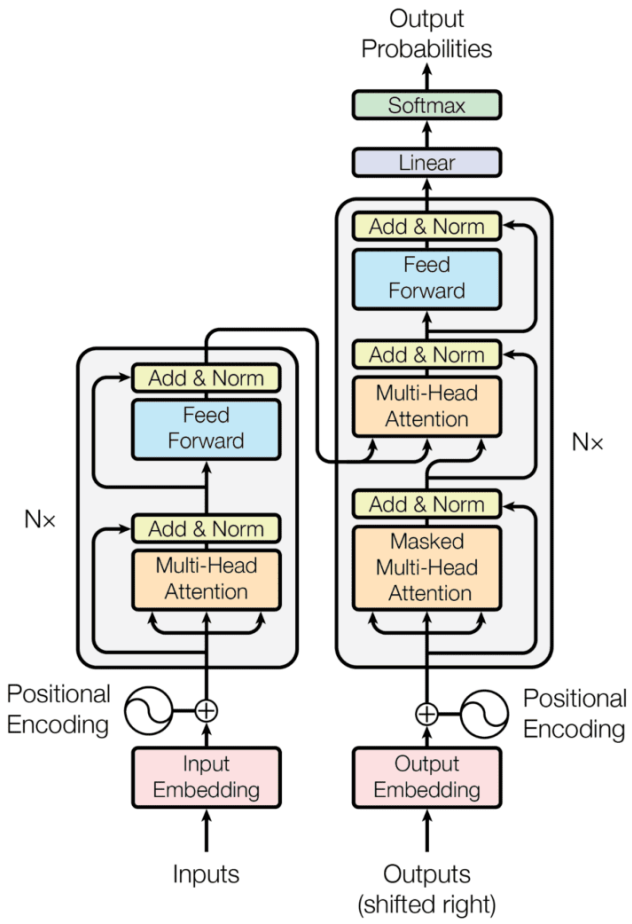

Figure 2.8: *Transformer's architecture, with an encoder (left) and a decoder(right)*

input sequence when encoding a specific word. Thanks to this layer, the model weights the importance of each word in the context of the others, improving the understanding of syntax and semantics. Being multi-head extends the self-attention mechanism by running through it several times in parallel. The independent attention outputs, called heads, focus on different parts of the input sequence, allowing the model to capture a range of relationships between words or features at different positions.

2. **The position-wise fully connected feed-forward network**, which is the second sub-layer, focuses on processing the signal at each position independently. The fully connected layers in the network ensure that each position of the input sequence is treated independently, but in the same manner. Furthermore, they contribute to the depth of the neural network, as well as adding non-linearity into the otherwise linear transformations of the attention mechanism, which helps the model learn more complex patterns.

This dual focus allows the Transformer to capture both positional and sequential aspects of the

data.

In contrast, the decoder generates output from the encoded data, although it also can take text as input, and predicts the next word in the sequence given the previous words. Decoders have three sub-layers:

1. **The masked multi-head self-attention mechanism** primary function is to prevent positions from attending to subsequent positions in the output sequence, ensuring that the predictions for a certain position can only depend on the known outputs at positions before it. The main idea of masking is to enforce causality in the model and to prevent the output of the decoder from being influenced by words that are supposed to come after it. Otherwise, it would be trivial to guess the next word if you already have access to it.

2. **The multi-head attention mechanism** over the encoder's output helps the model to focus on relevant parts of the input sequence, thus helping to align the output correctly with the input. This is crucial for tasks like translation, where the output needs to correspond to the input. Unlike the previous layer, the encoder-decoder attention can attend to all positions in the encoder's output, which serves as a guide to generate the next token in the sequence by focusing on inputs that are most relevant to the task at hand.

3. **The position-wise fully connected feed-forward network**, similar to the one in the encoder, applies further transformations to the decoder's output, introducing non-linearity into the model after the two attention mechanisms. By increasing the depth and complexity of the network, this layer adds another level of abstraction to the model's data representation, which enhances the learning capabilities of the model. The feed-forward network can be considered as a way to combine features within the same position, ensuring the model's understanding and generation of each token in a sequence are richly informed by a complex transformation of the inputs, refined by the attention mechanisms. The simplicity and effectiveness of this layer make the Transformer architecture highly adaptable, allowing it to excel in a variety of tasks, and helping the model make better predictions.

2.3 Datasets / Corpus

To train and fine-tune any kind of model, huge quantities of data are needed, especially when using deep learning. Initially, corpora were simple and small due to computational and storage

limitations, but the availability of the internet and advances in digital storage made it possible to compile much larger corpora nowadays.

Early corpora entailed a labour-intensive process of manually collecting, typing and digitizing texts. They also lacked standard formats, which made it difficult to use them across different systems or software. The significant variability in annotation quality and depth affected the reliability and reproducibility of research based on those datasets. Furthermore, access to some datasets was restricted to specific academic institutions or research groups, which limited wider community research and development.

Nowadays, with those aspects taken into account, there is a huge amount of datasets that can be used to train a wide number of different tasks. Specialized corpora for different languages and domains (e.g., legal, medical, social media) are relatively easy to find and allow training more accurate and context-aware systems.

In some cases, finding suitable datasets is still difficult, especially when addressing low-resource languages, which are those which have few digital available resources for language processing. No doubt that it is easier to find corpora in English, Chinese or Spanish than in Catalan, for instance. This is a clear example of a representation bias [14], which occurs when certain groups or types of data are underrepresented in the datasets used to train and develop NLP models. The lack of low-resource languages corpus causes that less models can be trained with such languages, thereby widening the existing inequitable gap between languages. There is an increasing focus on developing resources to foster more equitable technology development, and initiatives like Endangered Languages Project and Universal Dependencies aim to develop linguistic data and tools for under-resourced languages [15] [16].

The use of vast amounts of web data has a huge impact on AI systems, which may propagate biases of different kinds [14]:

- **Historical bias** arises when the data accurately reflects past prejudices or societal stereotypes. Even if the data collection and modelling are technically perfect, the resulting AI systems can perpetuate and amplify these biases.
- **Representation bias** occurs when the dataset does not adequately represent the target population or when the data collection methods systematically exclude certain groups. This leads to models performing well for majority groups, but poorly for minority groups.
- **Measurement bias** can occur during the labelling process, when the measurement tools or

methods are biased, or when the people annotating the data introduce their own subjective biases.

- **Aggregation bias** arises when a model is overly generalized and does not account for the differences and nuances among individual subgroups within the population. Assuming that one solution fits all leads to potentially leading to solutions that are optimal for the average but suboptimal for individuals.

- **Evaluation bias** happens when the evaluation metrics do not adequately capture the model's impact across all relevant user groups. It can lead to an incomplete or skewed understanding of the model's performance and effectiveness.

- **Learning bias** are the ones that the model learns from bias present in the training data. These biases, even if subtle, can be amplified by the algorithm used, which might learn to particularly perform poorly in the underrepresented group.

- **Deployment bias** occurs when the model is deployed in a real-world setting that differs significantly from the one it was trained and tested. It can lead to unexpected behaviour due to the different characteristics between real-world and training data.

It is important to highlight the need for careful consideration of how training data is collected and used, as well as the recommended use of mitigation strategies when representation fairness can be compromised. Some mitigation strategies are improving data representativeness, employing robust evaluation metrics, and involving diverse stakeholders in the development process to identify and address possible negative impacts early [14].

2.4 Text generation

Similar to NLP, text generation started with template-based and rule-based systems, which lacked flexibility and struggled with naturalness and variety. It then transitioned to statistical methods, like Marvok models, that predicted the probability of each subsequent word based on the previous words, without a deep understanding of language structure. N-grams models were also used to predict the next item in a sequence, based on the n previous items. N-gram models were foundational for statistical approaches in NLP and were used extensively in early machine learning translation and speech recognition.

ELIZA (1966) was one of the earliest examples of a text generation system. Created at MIT by Joseph Weizenbaum, it simulated conversation using pattern matching and substitution methodology to give users an illusion of understanding [17]. SHRDLU, created in the 1970s by Terry Winograd, was able to manipulate and interact with objects in a block world through natural language commands [18]. It was a ground-breaking program, which demonstrated a more sophisticated understanding and generation of language than ELIZA. They had some limitations; they lacked coherence and context, which could lead to nonsensical or irrelevant information, as well as scalability, making it difficult to adapt to new domains or large vocabularies without extensive reprogramming or additional tools.

Transformer's architecture revolutionized text generation. As mentioned before, its self-attention mechanism and its ability to use parallelization during the process made it highly effective for generating coherent and contextually relevant text. Transformers are used for multiple tasks, such as language translation or conversational AI. Generative Pre-Trained Transformers (GPTs), particularly the latest iterations like GPT-3 and beyond, represent the state-of-the-art in generating human-like text. These models have demonstrated an impressive ability to understand and produce text that is indistinguishable from human writing, making them invaluable in various natural language processing applications.

Modern text generation models often employ **transfer learning**, a technique used in machine learning that involves pre-training a model on a large corpus of text and then fine-tuning it for specific text generation tasks. Fine-tuning with datasets that are closely related to the task in hand enhances relevance and accuracy. This approach significantly improves the quality and efficiency of text generation, and that is the reason why we made use of it. Its name comes from transferring the weights that a model has learned from a previous task to another. That way, the model already has a baseline that allows, not only to more rapidly achieve better results, but also to reduce environmental impact. Training machine learning models requires significant power, especially training large models like GPT-3 [19]or BERT on hundreds of GPUs for weeks, which can emit as much carbon as several cars would in their entire lifetimes [20]. They also require extensive hardware infrastructure, which has its own environmental cost in terms of production, operation, and disposal.

Despite these advancements, it is important to mention that it is still challenging to handle long-term dependencies to generate content that remains coherent over extensive text and that there is an ongoing effort to ensure that the generated texts do not perpetuate harmful stereotypes or misinformation.

3

Methodologies

Contents

3.1 Corpus description: DACSA dataset

This section aims to accurately explain which dataset was used, as well as the modifications applied to it in order to train the model. The corpus used in this project is DACSA (Dataset for Automatic summarization of Catalan and Spanish newspaper Articles) [21], which consists of a set of document-summary pairs from different newspapers in both Spanish and Catalan.

As mentioned previously, it is important to highlight the difficulty of finding such a wide corpus in Catalan, a minority language. While it is increasingly easier to find available corpora in Spanish, it remains challenging to find sufficiently large corpora in Catalan to train robust models. The DACSA corpus fills this gap by providing a substantial and high-quality dataset for both languages.

The corpus has four partitions:

	Training	**Validation**	**TEST_I**	**TEST_NI**
Spanish	1,802,919	104,052	104,052	109,626
Catalan	636,596	35,376	35,376	17,836

Table 3.1: *Number of samples in each subset of the DACSA corpus for Spanish and Catalan languages*

	Training (%)	**Validation (%)**	**TEST_I (%)**	**TEST_NI (%)**
Spanish	85.0 %	4.9 %	4.9 %	5.2 %
Catalan	88.0 %	4.8 %	4.8 %	2.5 %

Table 3.2: *Proportion of each subset in percentage for Spanish and Catalan languages in the DACSA corpus*

- **Train**, used to teach the model to classify and summarize during training.
- **Validation**, used to fine-tune the model parameters and monitor performance, ensuring that the model does not overfit the training data.
- **Test.i**, includes sources that were part of the training set and gives a measure of how well the model performs on familiar data.
- **Test.ni**, conversely, consists of sources not present in the training process, allowing for evaluation of the model's generalization capabilities to unseen data.

Table 3.1 shows the exact number of samples for each subset, while **Table 3.2** shows the splits' proportions for each language. Training data comprises the majority of the dataset, with 85% for Spanish and 88% for Catalan. Approximately 5% of the Spanish samples are allocated to each of the remaining subsets: validation, TEST_I, and TEST_NI, which together account for the remaining 15% of the dataset. In contrast, the distribution for Catalan is slightly different, with the TEST_NI subset, which includes sources not present in the training process, comprising 2.5% of the samples.

In **Table 3.3** the proportions of Spanish and Catalan are presented, showing that the Spanish samples outnumber the Catalan ones by at least 48% difference. Although the DACSA dataset was originally designed for summarization tasks, its rich and diverse text data makes it suitable for a wide range of text-to-text tasks in NLP. This includes tasks such as machine translation, paraphrasing, text generation, question answering, named entity recognition, etc. In this project, the

	Training	**Validation**	**TEST_I**	**TEST_NI**
Spanish	73.90%	74.63%	74.63%	86.02%
Catalan	26.10%	25.37%	25.37%	13.98%

Table 3.3: *Proportions of Spanish and Catalan samples in each subset*

DACSA dataset has been utilized to train a model focused on correcting grammatically incorrect inputs and generating the correct output, demonstrating the versatility and applicability of the DACSA corpus beyond its original purpose of summarization.

Study of the dataset

To understand how the fine-tuning noising tasks will affect the input data, an analysis of the distribution of various textual elements, such as capital letters, accents, and punctuation marks, was conducted. This analysis is crucial for tailoring the noising tasks to the dataset, ensuring that the modifications reflect the actual composition of the text. For instance, understanding the prevalence of certain elements helps in assessing the impact of tasks like removing exclamations or adding accents, ensuring that the applied transformations are meaningful.

Category	**Catalan (%)**	**Spanish (%)**
Capital letters	41.1	40.2
Accents	29.6	30.1
Dots	12.1	13.0
Commas	17.0	16.4
Question marks	0.2	0.2
Exclamation marks	0.2	0.2

Table 3.4: *Distribution of different punctuation and capitalization elements in Catalan and Spanish datasets*

Table 3.4 presents the distribution of various punctuation and capitalization elements in the Catalan and Spanish datasets. The percentages show how frequently each category appears relative to the total occurrences of these elements in the respective languages.

Both datasets show a high proportion of **capital letters**, with Catalan at 41.1% and Spanish at 40.2%. The use of **accents** is also quite similar, with 29.6% in Catalan and 30.1% in Spanish. This indicates that both languages heavily rely on accents to convey correct pronunciation and meaning, reflecting their phonetic richness.

The proportions of **periods** and **commas** are closely aligned between the two languages, suggesting similar sentence structures and complexity. Catalan has 12.1% dots and 17.0% commas, while Spanish has 13.0% dots and 16.4% commas. This indicates that both datasets contain a comparable mix of sentence-ending punctuation and intra-sentence clauses.

The presence of **question marks** and **exclamation marks** is infrequent, each accounting for only 0.2% of their respective datasets. This suggests that the datasets are composed primarily of

declarative and descriptive sentences rather than interrogative or exclamatory ones, which can be explained by the corpus's original summarization purpose.

Overall, the data indicates that both datasets share very similar patterns in terms of punctuation and capitalization usage. The slight variations observed can be attributed to the specific content of the datasets but generally reflect the shared linguistic characteristics and punctuation norms of these two related languages. These similarities facilitate the development of bilingual models that can effectively handle text normalization tasks across both languages.

3.2 Model architecture: BART

The architecture used for this text-generation task is BART (Bidirectional and Auto-Regressive Transformers), a sequence-to-sequence model from the Transformers family. It was introduced by researchers from Facebook AI in a paper titled *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension* in 2020 [22].

Since its introduction, BART has gained significant attention and has been widely adopted in various natural language processing tasks. Other researchers and organizations have used and fine-tuned BART models for specific applications, but the original BART model and its main variations (such as BART-base and BART-large) were initially published by Facebook AI. Thanks to platforms like Hugging Face, which make trained models available for public use, we could retrain the *facebook/bart-large* model and avoid starting from scratch, making it easier to achieve good results.

BART is designed for text generation tasks and has been shown to be highly effective in various natural language processing applications due to its ability to handle a wide range of noise types during training. This model is particularly advantageous because it combines the strengths of both bidirectional and auto-regressive architectures, allowing it to generate coherent and contextually relevant text.

BART's noise functions

BART was trained using a denoising autoencoder approach, where it learns to reconstruct original text from corrupted input. The different tasks or noise functions used for training BART include:

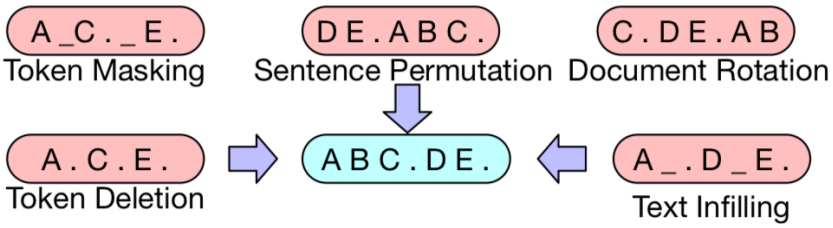

Figure 3.1: *BART noising transformations*

- **Token Masking**: Randomly selecting tokens in the input and replacing them with a special mask token. The model then learns to predict the original tokens from the masked input.
- **Token Deletion**: Randomly deleting tokens from the input text. The model must learn to infer the missing tokens and reconstruct the original text.
- **Text Infilling**: Removing a span of contiguous tokens and replacing them with a single mask token. The model is trained to predict the missing span.
- **Sentence Permutation**: Shuffling the order of sentences within a document. The model learns to reconstruct the original order of the sentences.
- **Document Rotation**: Selecting a token at random and rotating the document so that it begins with that token. The model must learn to understand the context and reorder the text correctly.

In this work, the noise function for training was inspired by the token masking function from the original BART paper. In our case, the noise function involved text infilling, where random spans of text were replaced with a mask token. This method challenges the model to generate text that fills in the gaps, thereby improving its ability to handle incomplete or noisy input.

Specifically, a custom function was implemented to mask 40% of tokens in the input texts, creating a challenging training scenario that encourages the model to learn robust text generation patterns. The choice of 40% masking was influenced by a study which found that this level of masking provides a balanced challenge that optimizes learning outcomes [23]. Additionally, the original BART paper demonstrated that text infilling with token masking shows the most consistently strong performance across various tasks, making it a robust choice for our application [22]. This approach aligns well with BART's design philosophy and leverages its strengths for effective text normalization in Catalan and Spanish.

Text representation in BART

For BART to understand the input text, it first has to be processed into a form that the model can interpret and work with effectively. This involves several key steps that convert raw text into numerical representations, enabling BART to perform its sophisticated text generation and normalization tasks. The main steps in this process are:

1. **Tokenization**: The text is first tokenized into subword units using a tokenizer, often based on Byte-Pair Encoding (BPE) or SentencePiece. This step breaks down the text into smaller, manageable pieces (tokens) that can include whole words, subwords, or even individual characters. For example, the sentence ”The quick brown fox” might be tokenized into [”The”, ”quick”, ”brow”, ”n”, ”fox”].

   For this project, the *facebook/bart-large* pretrained BPE tokenizer was used and further trained with the DACSA dataset to effectively tokenize Catalan and Spanish words. This adaptation is crucial as the original tokenizer was primarily trained on English text. By retraining with DACSA, the tokenizer learns the nuances of Catalan and Spanish, ensuring accurate tokenization and enhancing the model's performance in handling these languages.

2. **Embedding Layer**: Each token is then converted into a dense vector representation (embedding) using an embedding matrix. This matrix is learned during the training process and allows the model to capture the semantic meaning of each token. The embedding layer maps each token to a fixed-size vector, typically 512 or 1024 dimensions, depending on the model configuration.

   In our case, the *facebook/bart-large* model supports a sequence length of up to 1024 tokens, enabling it to process longer texts efficiently.

3. **Positional Encoding**: Since the Transformer architecture does not inherently capture the order of tokens, BART adds positional encodings to the token embeddings. These encodings provide information about the position of each token in the sequence. Positional encodings are added to the token embeddings, enabling the model to consider the order of words in the input text. This is done using sine and cosine functions of different frequencies.

4. **Encoder**: The token embeddings, augmented with positional encodings, are then passed through the encoder layers of the Transformer. BART uses a stack of encoder layers, where each layer consists of a multi-head self-attention mechanism and a feed-forward neural network. In the self-attention mechanism, each token attends to every other token in the se-

quence, capturing contextual relationships between words. The output of the encoder is a set of contextualized embeddings that represent the input text.

5. **Decoder**: In BART's architecture, the decoder also processes the text embeddings, but it operates differently. The decoder takes the encoder's output and the previous tokens generated (during training or inference) to generate the next token in the sequence. The decoder layers include masked multi-head self-attention mechanisms to ensure that predictions for a position can only depend on known outputs at positions before it, a multi-head attention mechanism over the encoder's output to focus on relevant parts of the input sequence, and a feed-forward neural network.

6. **Generation and Reconstruction**: During training, BART is trained as a denoising autoencoder. The input text is corrupted using various noise functions (such as token masking, token deletion, or sentence permutation), and the model learns to reconstruct the original text from the corrupted version. During inference, also called generation, the decoder generates text auto-regressively, predicting one token at a time until the entire sequence is generated. The output is then detokenized back into human-readable text.

3.3 Curriculum learning and tasks preparation

Curriculum learning is a training strategy inspired by the way humans and animals learn, where simpler concepts are introduced first and more complex ones gradually. The term was coined in the paper published by Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston, presented at the 26th International Conference on Machine Learning (ICML) in 2009 [24]. This methodology helps navigate non-convex optimization landscapes more effectively and it is applied to machine learning to improve training efficiency and effectiveness. It has been shown to:

- **Enhance training efficiency** by avoiding early exposure to highly challenging examples, which helps the model avoid getting stuck in poor local minima early in the training process

- **Improve generalization**, as the model starts learning with simpler, easy-to-understand examples, and progressively includes more complex and varied examples, gradually encountering the full diversity of the training data.

Fine-tuning tasks

To apply curriculum learning to the bilingual text generation model in hand, some noising tasks were created to be applied to the model's input. The following tasks were created:

- **lowercase**, completely converts all characters in the given texts to lowercase. This method was designed to help the model reconstruct texts that do not use capital letters, which might occur in informal writing or specific transcription errors.
- **remove_uppercase**, removes a specified percentage of uppercase letters from the given texts. It randomly selects which uppercase letters to convert to lowercase. This method addresses scenarios where capital letters for proper nouns or the beginnings of sentences are forgotten or mistyped.
- **join_sentences**, joins sentences by removing periods and ellipses from a specified percentage of locations in the given texts, converting subsequent uppercase letters to lowercase if needed. This addresses scenarios where people forget punctuation, ensuring the model can appropriately add punctuation to form compound and complex sentences, such as juxtaposed and subordinate clauses, very frequently used in Catalan and Spanish.
- **add_accents**, adds accents to a specified percentage of eligible vowels in the given texts. It identifies words without existing accents and randomly selects words to modify, adding accents to one of the vowels in each selected word. This method is intended to correct missing accents, addressing common errors where people might misspell words without the necessary accents.
- **change_accents**, changes the accents of a specified percentage of vowels that already have accents in the given texts. It randomly selects which accented vowels to change to a different accent, from acute to grave accent, and the other way around. This method handles scenarios where incorrect accents are used, particularly in languages like Catalan where the choice of accent can alter the meaning.
- **remove_accents**, removes the accents from a specified percentage of accented characters in the given texts. It randomly selects which accented characters to replace with their unaccented forms. It was intended to correct texts with missing accents.
- **remove_all_accents**, completely removes all accented characters from the given texts, replacing them with their unaccented counterparts. This is useful for reconstructing texts

that lack proper accentuation, such as those produced by some audio-to-text converters.

- **remove_spaces**, removes a specified percentage of spaces from the given texts. It calculates the number of spaces to remove based on a probability percentage and then randomly selects which spaces to remove. This method was designed for scenarios where audio-to-text converters mistakenly combine two words into one or when people mistype by omitting spaces due to fast typing.

- **remove_punctuation**, removes a specified percentage of punctuation marks from the given texts. It randomly selects which punctuation marks to remove, excluding those within numbers. This method helps the model learn how to appropriately punctuate sentences, correcting missing or misplaced punctuation.

- **normalize_space_tokenized**, tokenizes the given texts using NLTK's Spanish word tokenizer and then joins the tokens with spaces, effectively normalizing the spacing between words. This method ensures robust handling of spaces, correcting double spaces written by mistake.

- **normalize**, completely normalizes the given texts by removing accents, punctuation, and capitalization. It converts the text to lowercase and replaces accented characters with their unaccented equivalents. This method helps denormalize text generated by audio-to-text converters or serves as a denormalizer itself.

In the curriculum learning strategy implemented in this project, the *normalize* function was designed to be used individually due to its comprehensive nature. This function performs multiple transformations, including removing accents, punctuation, and capitalization, converting text to lowercase, and replacing accented characters with their unaccented equivalents. Because it encapsulates a wide range of transformations, applying the *normalize* function individually ensures that the model focuses entirely on this complex task without interference from other transformations.

In contrast, the remaining tasks, such as *lowercase*, *remove_uppercase*, and *remove_accents*, were designed to be applied in combination. These tasks target specific aspects of text normalization and can be combined to simulate various realistic text corruption scenarios. By allowing these tasks to be applied together, the model can learn to handle multiple types of errors and transformations simultaneously, enhancing its robustness and generalization capabilities.

Tasks order criteria

To order these tasks in a curriculum learning scheme with increasing difficulty, it is helpful to consider how each task affects the model's understanding and processing of language. Generally, tasks that alter more fundamental and basic aspects of the text, such as the structure of words and phrases, are considered simpler. In contrast, tasks requiring a finer knowledge of correct language usage are considered more advanced.

Table 3.5 shows the order in which tasks were introduced to the model. The column *difficulty increment* qualitatively describes the increase in difficulty from one task to the next. The labels "*Minimal*, *Moderate*, *Significant*, *High*, and *Very High* reflect the relative complexity jump between consecutive tasks and were used during the fine-tuning process to create the batch index from which tasks were allowed to be chosen.

In the curriculum learning strategy implemented for this project, tasks were introduced gradually, starting with simpler tasks and progressing to more complex ones. The difficulty increment does not refer to the intrinsic complexity of the tasks themselves, but rather to the challenge posed to the model in transitioning from one task to the next. It represents how tough it is for the model to adapt from performing one type of task to taking on a new, different task.

For example, transitioning from *lowercase* to *remove_uppercase* is marked as "Minimal" because the latter task is a slight variation of the former. However, moving from *remove_all_accents* to *remove_accents* is labelled as "Significant" because it requires the model to develop a more nuanced understanding of specific accents within the text.

This structured approach to task introduction allows the model to build on its previous knowledge and gradually adapt to more challenging tasks. By incrementally increasing the difficulty of transitions between tasks, the model can effectively learn and generalize across different types of text transformations. This method prevents the model from being overwhelmed by complex tasks too early in the training process, ensuring a smooth and efficient learning curve.

Order	Tasks	Task description	Difficulty increment
1	**lowercase**	This task is relatively simple as it only involves recognizing sentence beginnings and proper nouns to capitalize them correctly.	Minimal
2	**remove_uppercase**	Similar to the previous task but slightly more complex, as it requires understanding not only the beginning of sentences but also other grammatical uses of capital letters, such as in proper nouns and titles.	Moderate
3	**remove_all_accents**	This task focuses on the model's ability to understand correct accentuation in words based on context, which is crucial in languages like Spanish.	Moderate
4	**remove_accents**	This is more difficult as it requires the model to identify and correct specific accents that should not be present.	Significant
5	**add_accents**	This task is more advanced because it demands the model to have a detailed understanding of the correct use of accents in different contexts.	Significant
6	**change_accents**	This is a more advanced task that requires precise correction of accents, especially important for languages like Catalan where the choice of accent can alter the meaning.	High
7	**remove_spaces**	This task requires the model to have a good understanding of the lexical structure of the language to identify where spaces should be.	Moderate
8	**normalize_space_tokenized**	This task is complex because it alters the text structure in a way that requires the model to understand and correct incorrect space usage near punctuation.	High
9	**join_sentences**	Requires an advanced level of language comprehension, as the model must determine where sentences end and how paragraphs and dialogues are structured.	High
10	**remove_punctuation**	This involves a deep understanding of the syntactic structure and text rhythm, making it one of the most challenging tasks.	High
11	**normalize**	This is the most complex task as it combines multiple challenges, requiring the model to completely reconstruct the grammatical and stylistic structure of the original text.	Very High

Table 3.5: *Order of tasks according to difficulty*

4

Evaluation and results

Contents

4.1 Levenshtein Distance

The evaluation process for this study was primarily conducted using the **Levenshtein distance**, also known as the edit distance. This metric measures how different two strings are from one another by quantifying the minimum number of single-character edits required to transform one string into the other. These edits can be:

- **Insertions**: Adding a character.
- **Deletions**: Removing a character.
- **Substitutions**: Replacing one character with another.

For example, the Levenshtein distance between the words *kitten* and *sitting* is 3, given that for *kitten* to ”become” *sitting*, the following edits are needed:

1. Replace 'k' with 's': "kitten" -> "sitten"
2. Replace 'e' with 'i': "sitten" -> "sittin"
3. Insert 'g' at the end: "sittin" -> "sitting"

Applications of Levenshtein Distance

Levenshtein distance is a widely used metric in various fields due to its simplicity and effectiveness. Some common applications include spell checking, DNA sequencing, natural language processing, and plagiarism detection [25] [26].

Levenshtein distance is particularly useful in text normalization and evaluation for several reasons:

- **Granular Accuracy**: It provides a fine-grained measure of similarity, making it sensitive to even small differences between strings.
- **Simplicity and Interpretability**: The concept of insertions, deletions, and substitutions is intuitive, making the metric easy to understand and interpret.
- **Flexibility**: It can be applied to various types of text data, including words, sentences, and longer documents.

Levenshtein Distance in Our Evaluation

In our evaluation, the Levenshtein distance was used to measure the effectiveness of text normalization tasks. We calculated the average Levenshtein distance at both the character level and the token level for different tasks applied to the text. This dual-level analysis is important for the following reasons:

- **Character-Level Levenshtein Distance**: This metric provides a detailed view of the fine-grained edits needed to transform the normalized text into the target text. It helps in understanding the precision of the model at the most granular level, ensuring that individual character changes are accurately measured.
- **Token-Level Levenshtein Distance**: This metric aggregates the edits at the word or token level, which is crucial for understanding the model's performance in maintaining the

overall structure and meaning of the text. Token-level distance captures the broader changes needed, offering insights into the model's ability to handle word boundaries and larger text segments.

By analyzing both character-level and token-level Levenshtein distances, we gain a comprehensive understanding of the model's performance in text normalization. The character-level metric ensures accuracy in minute details, while the token-level metric ensures coherence and structural integrity.

Overall, the use of Levenshtein distance provided a comprehensive and detailed understanding of the model's performance in text normalization, highlighting areas of strength and potential improvement.

This introductory section sets the stage for a deeper dive into the detailed evaluation results presented in the subsequent sections.

4.2 Curriculum learning experimentation

As mentioned before, the *facebook/bart-large* model was initially trained using the token masking strategy on the bilingual DACSA corpus, specifically with the training split. This initial training phase was the most time-consuming, taking up to 15 days to complete.

Following this, a fine-tuning process was conducted utilizing the fine-tuning tasks and strategy presented in **Section 3.3**. Fine-tuning refers to the process of training the model on specific tasks that we want the model to be proficient in. This stage is crucial, as it tailors the model's capabilities to handle real-world applications more effectively.

The task creation process is vital to the success of the fine-tuning phase. During this process, different possible scenarios are considered to ensure that the tasks are comprehensive and cover all necessary aspects. For instance, if we want the BART model to correctly apply capitalization, we must consider the various contexts in which capital letters are used. These contexts include the beginnings of sentences, proper nouns, titles, and other grammatical structures. Therefore, the tasks should be designed to address these specific cases, ensuring that the model learns the rules governing the use of capital letters comprehensively.

To train the model to recognize and apply capital letters correctly, the input data must include examples where capital letters are missing or used incorrectly. This allows the model to learn

the intuition behind proper capitalization. The fine-tuning phase, which involved three epochs of inference and correction, took seven days to complete. By the end of this phase, the model was adequately trained to predict and evaluate its performance.

In this fine-tuning phase, specific noising tasks were created to simulate the kinds of errors that the model is expected to correct. These tasks degrade the input text in controlled ways, such as removing capitalization, punctuation, or accents, and then require the model to restore the text to its correct form. This approach ensures that the model learns to handle a wide range of text normalization challenges effectively.

Results

The evaluation of the model's performance was conducted in multiple ways. The model was assessed individually for Spanish and Catalan, to understand well how the model behaves for each language. This comprehensive evaluation helps determine the model's effectiveness in each language and have a sense of the overall performance in a bilingual context.

The Levenshtein distance was used to calculate the normalized difference between the predictions and the expected labels, as well as between the normalized text and the expected labels. The normalization was performed using the maximum length of the two texts being compared. To intuitively understand the model's performance, the proportion of correctly predicted characters or tokens was computed.

Firstly, the normalized Levenshtein distance between the predictions and the expected labels is divided by the normalized Levenshtein distance between the normalized text and the expected labels. Secondly, this ratio is subtracted from 1 and multiplied by 100 to convert it to a percentage. This percentage represents the portion of characters or tokens that the model has correctly predicted in the text. By using this method, we can intuitively grasp the model's performance in predicting the correct characters or tokens.

	Spanish	**Catalan**
% of characters correctly predicted	90.95	95.9
% of tokens correctly predicted	91.1	96.5

Table 4.1: *Individual language performance of the curriculum learning model*

Table 4.1, shows that **percentage of correctly predicted characters** is 5% higher for Catalan (95.9) than for Spanish (90.95). This means that on average, only 4 out of 100 characters would be wrongly predicted in Catalan, and 9 out of 100 in Spanish. To better understand the

expected overall performance, meaning how the model would behave for text in both languages, it is useful to refer to **Table 3.3**, where the dataset's language distribution is shown. Given that the dataset is not equally distributed between languages, the overall expected performance will not be as good as Catalan, but still better than the Spanish accuracy rate. Overall, the model seems to be able to correctly predict a fairly high percentage of the characters.

Similar to the character-level analysis, the percentage per token is 5.4% higher for Catalan (96.5) than for Spanish (91.1), meaning that the model is more efficient in normalizing Catalan than Spanish texts. Even if the model performs slightly better for Catalan than for Spanish, the overall percentages would be reasonably high, meaning that the resulting text should be of fairly good quality.

4.3 Average tasks performance

Given that different tasks were applied to the input texts, analysis per task was performed to see if some tasks had better performance than others. To achieve this, 7967 batches were categorized by the transformations applied to them to add noise to the input of the model, and then the previously mentioned metric was measured for each task group.

Table 4.2 shows the number of batches to which a certain task was applied and the percentage of batches that a specific task has been applied to. The significant difference in the number of

Tasks	Number of batches	% applied
normalize	1895	23.8
lowercase	4147	52
remove_uppercase	4182	52.5
remove_accents	4184	52.5
normalize_space_tokenized	4242	53.3
change_accents	4244	53,3
add_accents	4258	53.5
remove_all_accents	4271	53.6
join_sentences	4271	53.6
remove_punctuation	4279	53.7
remove_spaces	4298	53.95

Table 4.2: *Number of batches per task, in increasing order*

batches for the *normalize* task was anticipated. This task was the last to be introduced during the fine-tuning process, given that it was considered the hardest one and was applied individually. Some batches only had the *normalize* task applied, while others had a combination of the rest of the tasks. This explains why the percentages applied for the rest of the tasks cannot be normalized,

because they overlap.

While tasks were introduced to the model progressively, their application was random. The random selection strategy likely leads to a near-uniform distribution of tasks once they are introduced. This randomness causes tasks introduced later to be applied almost as frequently as earlier tasks during their respective periods. The small 2% variation between the rest of the tasks, indicates that, once introduced, tasks were frequently selected.

Tasks	**Spanish %**	**Catalan %**
change_accents	90.86	95.78
join_sentences	90.87	95.91
remove_punctuation	90.92	95.83
remove_accents	90.92	95.70
remove_uppercase	90.91	95.92
remove_spaces	90.98	95.88
remove_all_accents	90.96	95.89
lowercase	90.97	95.65
normalize_space_tokenized	90.94	95.78
add_accents	91.00	96.01
normalize	91.05	95.92

Table 4.3: *Curriculum learning model performance per task (chars)*

Presented in **Table 4.3** are the model's character performance per task in Spanish and Catalan, listed in increasing order. There is a 0.19% variation in Spanish, and a 0.36% variation in Catalan, from the lowest to the highest task's performance. This suggests that the model handles these transformations well without significant degradation in character-level accuracy, especially in Catalan, where only 4 out of the 100 characters would, on average, be wrong. In Spanish, the number ascends to 9 out of 100 mistaken characters, resulting in an good overall performance.

Tasks	**Spanish %**	**Catalan %**
lowercase	91.10	96.10
join_sentences	91.10	96.30
remove_punctuation	91.10	96.30
change_accents	91.10	96.30
remove_spaces	91.10	96.30
remove_all_accents	91.20	96.20
remove_accents	91.10	96.20
normalize_space_tokenized	91.20	96.20
remove_uppercase	91.10	96.30
normalize	91.10	96.40
add_accents	91.20	96.30

Table 4.4: *Curriculum learning model performance per task (tokens)*

Table 4.4 contains the individual token accuracy for each task. Similar to the character accuracy, variation across tasks is insignificant, with only 0.3% in Catalan and 0.1% in Spanish.

In Spanish, the tasks that performed the best were *remove_accents*, *normalize_space_tokenized*, and *add_accents*, with 91.2%. In Catalan, the *normalize* task performed the best, with 96.4%. *Normalize* was the highest performance, with 96.4% in Catalan, meaning that, on average, only 4 mistakes out of 100 tokens would occur.

Tasks	**% applied Spanish**	**% applied Catalan**
normalize	23.35	24.57
lowercase	52.42	51.84
remove_uppercase	53.11	51.84
remove_accents	52.57	53.09
normalize_space_tokenized	53.79	54.44
change_accents	53.42	52.38
add_accents	53.43	51.93
remove_all_accents	53.93	51.66
join_sentences	54.33	52.02
remove_punctuation	54.26	53.36
remove_spaces	53.82	54.08

Table 4.5: *Percentage of applying each task in Spanish and Catalan*

Table 4.5 shows the percentage of task application in Catalan and Spanish. The percentages range from 23.4-24.6% for the *normalize* task and 52-54% for the rest of the tasks. Tasks being applied at similar rates across the dataset results in each task having a comparable influence on the overall normalization process. This uniform distribution means no single task disproportionately affects the model's performance, leading to similar Levenshtein distances. Because tasks were applied uniformly, the model learns to handle each transformation in a balanced manner.

It is important to note that since multiple tasks are often applied in combination, the cumulative effect of these tasks can average out the differences in individual task performance. This combination of uniform application and the averaging effect prevents overfitting to any specific task and promotes generalization, resulting in similar Levenshtein distances for different tasks.

4.4 Equal probability model results

To evaluate the effectiveness of the curriculum learning strategy, a comparison was made with a model trained using an equal probability approach for task application from the start. This second model, referred to as the *equal probability model*, applied each of the previously mentioned tasks with equal likelihood from the beginning of the training process. That means that, as opposed to the curriculum learning model, this one confronted the most difficult tasks at the beginning. The results are presented in **Table 4.6**, and are directly comparable to **Table 4.1**, in which a

significant reduction can be observed. At a character level, the curriculum learning model shows consistent improvements, increasing performance from 86.97 to 90.95 in Spanish and 95.71 to 95.9 in Catalan. It exhibits similar behaviour at a token level, where the curriculum learning model increases performance from 87.8 to 91.1 in Spanish and 96.2 to 96.5 in Catalan. The most notable improvements are observed in Spanish, which will proportionally benefit the overall performance due to the larger Spanish dataset size 3.3. The curriculum learning model consistently outperforms

	Spanish	**Catalan**
% of characters correctly predicted	86.97	95.71
% of tokens correctly predicted	87.8	96.2

Table 4.6: *Individual and overall language performance of the equal probability model*

the equal probability model across both languages and metrics. Notably, the improvements are more significant in Spanish, though both languages show clear benefits from the curriculum learning approach. This indicates that the curriculum learning model is more effective at handling and correcting text, leading to lower Levenshtein distances and thus higher accuracy in text generation tasks.

4.5 Example of use

Below there is an example of the labeled text, input and output of the curriculum learning model. The label text was compared to the output text in order to calculate the Levenshtein distance. The input to the model was the result of applying the normalize function, which eliminates any capital letter, accent, punctuation mark, etc., to the label text. This section provides an example of the label text, input, and output of the curriculum learning model. The label text serves as the ground truth against which the output text is compared to calculate the Levenshtein distance. The input to the model is the result of applying the normalize function to the label text, which removes all capital letters, accents, punctuation marks, etc.

Original text: Los datos del Instituto Nacional de Estadística (INE) revelan que entre 2018 y 2019 aumentó el número de ocupados que trabajaron en su domicilio particular. Con todo, el porcentaje de empleados que teletrabajan es superior entre los hombres que entre las mujeres. La submuestra de la EPA también indica que un 66,1% de los trabajadores de Baleares hace jornada continua, con un porcentaje mayor entre las mujeres. Asimismo, los hombres siguen teniendo mayor presencia en los puestos directivos, tanto en el rango de encargado, mandos intermedios y directores de pequeñas o grandes empresas. Además, hay más mujeres que hombres que manifiestan que les

gustaría trabajar más horas en su empleo actual, o en un empleo diferente con mayor horario. Por otro lado, en 2019, en Baleares, unas 3.000 mujeres en inactividad buscaban empleo después de haber dejado un empleo anterior por cuidado de niños o de adultos dependientes. En el caso de los hombres, son solamente alrededor de 300 en la misma situación. Además, otras 1.500 mujeres en situación de inactividad habían dejado su empleo anterior por otras obligaciones familiares o personales.

Normalized input text: los datos del instituto nacional de estadistica ine revelan que entre 2018 y 2019 aumento el numero de ocupados que trabajaron en su domicilio particular...

Model output: Los datos del Instituto Nacional de Estadística (INE) revelan que entre 2018 y 2019 aumentó el número de ocupados que trabajaron en su domicilio particular. Con todo, el porcentaje de empleados que teletrabajan és superior entre los hombres que entre las mujeres. La submuestra de la EPA también indica que un 66,1% de los trabajadores de Baleares hace jornada contínua, con un porcentaje mayor entre les mujeres. Asimismo, los hombres siguen teniendo mayor presencia en los puestos directivos, tanto en el rango de encargado, mandos intermedios y directores de pequeñas o grandes empresas. Además, hay más mujeres que hombres que manifiestan que les gustaría trabajar más horas en su empleo actual o en un empleo diferente, con mayor horario, por otro lado. En 2019, en Baleares, unas 3.000 mujeres en inactividad buscaban empleo, después de haber dejado un empleo anterior, por cuidado de niños o de adultos dependientes. En el caso de los hombres, són solamente alrededor de 300 en la misma situación. Además otras 1.500 mujeres en situación de inactividad habían dejado su empleo anterior por otras obligaciones familiares o personales.

5

Conclusions

Contents

In conclusion, the initial training of the *facebook/bart-large* model on the bilingual *DACSA* corpus took 15 days, followed by a 7-day fine-tuning phase with specific tasks, making the model proficient in handling real-world applications.

Task creation was crucial for the fine-tuning phase, covering various contexts such as capitalization, punctuation, and accent normalization. The model was evaluated separately for Spanish and Catalan, using normalized Levenshtein distance to intuitively understand the model's performance by computing the proportion of correctly predicted characters and tokens. Overall, 4 to 9 incorrectly predicted character are expected over 100 chatacters.

Analysis across tasks was performed and no significant relationship between better performance and specific tasks was found. The random and uniform application of tasks ensured balanced training, preventing overfitting and promoting generalization. Tasks were applied uniformly across the dataset, leading to similar Levenshtein distances and high overall performance. This uniformity can be attributed to the similar rates of task application, which ensure that each task is applied frequently enough to have a comparable impact on the model's learning process.

As a result, the model is able to handle each transformation in a balanced manner, maintaining consistent performance across different tasks. This balanced exposure to the tasks leads to similar

Levenshtein distances for each task, indicating that the model maintains consistent performance regardless of the specific transformations applied.

Additionally, it is worth noting that the model performed better in Catalan than in Spanish, with 95.9% of characters and 96.5% of tokens correctly predicted in Catalan, compared to 90.95% of characters and 91.1% of tokens in Spanish. This 5% improvement could be due to the order in which the datasets were loaded and processed. Specifically, Catalan samples were among the last ones the model saw during training. The datasets were combined in such a way that Spanish and Catalan samples were shuffled and concatenated, possibly giving the model more recent exposure to Catalan data. This recent exposure might have provided the model with a fresher memory of Catalan samples, contributing to its slightly better performance in this language.

The curriculum learning model showed consistent improvements over the equal probability model, proving the initial hypothesis that using a curriculum learning strategy would result in better performance. Character-level performance improved by 4% in Spanish and 0.2% in Catalan. Token-level performance improved by 3% in Spanish and 0.3% in Catalan. These findings highlight the robustness of the curriculum learning approach in training the model to handle a wide range of text normalization challenges effectively, with balanced task application contributing to its high performance across both languages.

5.1 Potential Improvements

Controlled Introduction Strategy

One potential improvement for enhancing the model's performance is to implement a more controlled introduction strategy for tasks within the curriculum learning framework. Currently, tasks are introduced progressively but applied randomly, which can lead to a near-uniform distribution of tasks. To create distinct stages in the curriculum learning process, tasks could be introduced one at a time with clear thresholds. This strategy would ensure that more challenging tasks are applied less frequently at the beginning, allowing the model to gradually adapt and build its capability to handle these transformations.

By carefully controlling the introduction and frequency of tasks, the model could better learn and internalize each transformation before moving on to more complex tasks, ultimately improving overall performance.

Shuffling Datasets

Another improvement involves shuffling the Catalan and Spanish datasets more thoroughly before training. This approach aims to enhance the model's performance by ensuring that it does not develop a bias towards the order in which the data is presented.

In the current setup, the order of dataset loading may influence the model's performance, potentially giving an unintended advantage to the language that appears later in the sequence. By shuffling the datasets, the model would encounter a more diverse and randomized sample of both languages throughout the training process, promoting balanced learning and improving generalization across both Catalan and Spanish.

Additional Tasks for Inverse Text Normalization (ITN)

Expanding the range of tasks within the curriculum to include more complex and varied transformations, such as Inverse Text Normalization (ITN), could further enhance the model's capabilities. ITN involves converting spoken language transcriptions back into their written form, addressing challenges such as handling numbers, dates, and abbreviations.

By adding tasks that simulate ITN, the model can learn to manage a wider array of text normalization scenarios. This addition would not only improve the model's versatility, but also its robustness in handling diverse textual inputs. Integrating ITN tasks would provide the model with a more comprehensive training experience, preparing it to tackle a broader spectrum of normalization challenges in real-world applications.

A

Sustainable Development Goals (SDGs) in the Final Degree Projects

The Sustainable Development Goals (SDGs), established by the United Nations in 2015, are a set of 17 interlinked global goals designed to be a "blueprint to achieve a better and more sustainable future for all". These goals address the global challenges we face, including poverty, inequality, climate change, environmental degradation, peace, and justice.

In line with these global efforts, this final degree project aligns with several of these goals. **Table A.1** summarizes how each SDG is addressed within the scope of this work: Among the previously mentioned Sustainable Development Goals, this project is connected with:

1. **Quality education**: The model could potentially be used for educational purposes, helping students learn languages more effectively by providing grammatically correct and contextually appropriate text generation.

2. **Industry, innovation, and infrastructure**: This project employs advanced neural network techniques to create a bilingual text generation model, demonstrating significant innovation in the field of natural language processing.

3. **Reduced inequalities**: By creating a bilingual model that includes Catalan, a minority

Sustainable Development Goals	**High**	**Medium**	**Low**	**Not Applicable**
End poverty				X
Zero hunger				X
Good health and well-being				X
Quality education			X	
Gender equality				X
Clean water and sanitation				X
Affordable and clean energy				X
Decent work and economic growth.				X
Industry, innovation, and infrastructure		X		
Reduced inequalities	X			
Sustainable cities and communities				X
Responsible consumption and production		X		
Climate action.				X
Life below water				X
Life on land				X
Peace, justice, and strong institutions				X
Partnerships for the goals				X

Table A.1: Degree of relationship of the work with the Sustainable Development Goals (SDGs).

language, this project contributes to making digital tools more accessible to speakers of less widely spoken languages, thus reducing linguistic inequalities.

4. **Responsible consumption and production**: The project leverages a pre-trained model, thus reducing the environmental impact associated with training large language models from scratch. This mindful approach aligns with the goal of promoting sustainable production practices.

Bibliography

[1] S. Fanni, M. Febi, G. Aghakhanyan, and E. Neri, "Natural language processing," in *Handbook of Artificial Intelligence and Robotics*. 2023, pp. 87–99, ISBN: 978-3-031-25927-2. DOI: 10.1007/978-3-031-25928-9_5.

[2] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. 59, no. 236, pp. 433–460, 1950.

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv*, vol. abs/1301.3781, 2013.

[4] 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.

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 5998–6008. [Online]. Available: https://arxiv.org/abs/1706.03762.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv*, vol. abs/1810.04805, 2019.

[8] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," *arXiv*, vol. abs/1810.04805, 2018.

[9] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," in *The Handbook of Brain Theory and Neural Networks*. 1995.

[10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. DOI: 10.1109/5.726791.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[12] A. R. Choudhuri, B. G. Thakurata, and B. Debnath, "Mnist image classification using convolutional neural networks," in *Modeling, Simulation and Optimization*, Springer, 2022, ch. 19, pp. 255–266.

[13] M. Z. Alom, T. M. Taha, C. Yakopcic, *et al.*, "The history began from alexnet: A comprehensive survey on deep learning approaches," *arXiv*, vol. abs/1803.01164, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:3680335.

[14] H. Suresh and J. Guttag, "A framework for understanding sources of harm throughout the machine learning life cycle," in *Proceedings of the 2021 ACM Conference on Equity and Access in Algorithms, Mechanisms, and Optimization (EAAMO '21)*, Oct. 2021. DOI: 10.1145/3465416.3483305. [Online]. Available: http://dx.doi.org/10.1145/3465416.3483305.

[15] Endangered Languages Project, *Endangered languages*, 2024. [Online]. Available: https://www.endangeredlanguages.com.

[16] Universal Dependencies, *Universal dependencies*, 2024. [Online]. Available: https://universaldependencies.org.

[17] J. Weizenbaum, "Eliza – a computer program for the study of natural language communication between man and machine," *Communications of the ACM*, vol. 9, no. 1, pp. 36–45, 1966. DOI: 10.1145/365153.365168.

[18] T. Winograd, "Procedures as a representation for data in a computer program for understanding natural language," in *Proceedings of the Fall Joint Computer Conference*, 1971, pp. 29–45.

[19] T. B. Brown, B. Mann, N. Ryder, *et al.*, "Language models are few-shot learners," *arXiv*, vol. abs/2005.14165, 2020.

[20] E. Strubell, A. Ganesh, and A. McCallum, *Energy and policy considerations for deep learning in nlp*, 2019. arXiv: 1906.02243 [cs.CL].

[21] E. Segarra Soriano, V. Ahuir, L.-F. Hurtado, and J. González, "Dacsa: A large-scale dataset for automatic summarization of Catalan and Spanish newspaper articles," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2022.

[22] M. Lewis, Y. Liu, N. Goyal, *et al.*, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *arXiv*, vol. abs/1910.13461, 2020.

[23] A. Wettig, T. Gao, Z. Zhong, and D. Chen, "Should you mask 15 in masked language modeling?" *arXiv*, vol. abs/2202.08005, 2023.

[24] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th International Conference on Machine Learning*, ACM, 2009, pp. 41–48.

[25] D. Jurafsky and J. H. Martin, *Speech and Language Processing*. Prentice Hall, 2009.

[26] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.