UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

School of Informatics

Enhancing Textual Datasets with Large Language Model-based Data Augmentation for Neural Machine Translation

End of Degree Project

Bachelor's Degree in Informatics Engineering

AUTHOR: Puchol Salort, Blai

Tutor: Casacuberta Nolla, Francisco

External cotutor: CHATZITHEODOROU, KONSTANTINOS

ACADEMIC YEAR: 2022/2023

Resum

En aquest TFG, proposem un sistema d'augment de dades textuals paral·leles per a llenguatges o dominis de baixos recursos utilitzant diferents tecnologies com ara models de llenguatge emmascarat, grans models lingüístics i eines i recursos de processament de llenguatge natural. El sistema detecta parts del text objetiu que es poden reemplaçar per altres de noves amb la mateixa categoria gramatical a nivell de paraula. Després, la paraula reemplaçada es troba al text font utilitzant alineació de paraules entre frases per reemplaçar-la amb un sinònim adequat. El procés inclou passos iteratius per identificar i reemplaçar paraules fins que no es puguin produir més canvis. El nostre enfocament aborda la necessitat de conjunts de dades més extenses i d'alta qualitat en tasques de processament del llenguatge natural en situacions de baixos recursos. El procés proposat està dissenyat per admetre múltiples idiomes i diversos tipus de text. Es poden utilitzar grans models lingüístics per mantenir la qualitat de les dades augmentades respecte a les originals i facilitar diversos conjunts de dades textuals tant en casos monolingües com multilingües. La qualitat de les dades augmentades és avaluada per humans en funció de diversos criteris, com ara fluïdesa, coherència i rellevància, i es realitzen avaluacions automàtiques per comprovar la millora en el rendiment del model de llenguatge. Aquesta avaluació automatitzada utilitza mètriques de darrera generació com BLEU, TER i chrF. El sistema proposat té com a objectiu millorar la qualitat i quantitat de dades textuals que es poden utilitzar per a tasques de processament del llenguatge natural, com ara la traducció automàtica.

Paraules clau: augment de dades, grans models de llenguatge, grans models lingüístics, processament de llenguatge natural, traducció automàtica, generació de text

Resumen

En este TFG, proponemos un sistema de aumento de datos textuales paralelos para lenguajes o dominios de bajos recursos utilizando diferentes tecnologías como modelos de lenguaje enmascarado, grandes modelos de lenguaje y herramientas y recursos de procesamiento de lenguaje natural. El sistema detecta partes del texto objetivo que pueden reemplazarse por otras nuevas con la misma categoría gramatical a nivel de palabra. Luego, la palabra reemplazada se encuentra en el texto fuente utilizando alineación de palabras entre frases para reemplazarla con un sinónimo adecuado. El proceso incluye pasos iterativos para identificar y reemplazar palabras hasta que no puedan ocurrir más cambios. Nuestro enfoque aborda la necesidad de conjuntos de datos más extensos y de alta calidad en tareas de procesamiento del lenguaje natural en situaciones de bajos recursos. El proceso propuesto está diseñado para admitir múltiples idiomas y varios tipos de texto. Se pueden utilizar modelos de lenguaje grandes para mantener la calidad de los datos aumentados respecto a los originales y facilitar diversos conjuntos de datos textuales tanto en casos monolingües como multilingües. La calidad de los datos aumentados es evaluada por humanos en función de varios criterios, como fluidez, coherencia y relevancia, y se realizan evaluaciones automáticas para comprobar la mejora en el rendimiento del modelo de lenguaje. Esta evaluación automatizada emplea métricas de última generación como BLEU, TER y chrF. El sistema propuesto tiene como objetivo mejorar la calidad y cantidad de datos textuales que se pueden utilizar para tareas de procesamiento del lenguaje natural, como la traducción automática.

Palabras clave: aumento de datos, grandes modelos de lenguaje, procesamiento de lenguaje natural, traducción automática, generación de texto

Abstract

This work proposes a parallel textual data augmentation framework for low-resource languages or low-resource domains using different technologies like *Masked Language Models*, *Large Language Models*, and *Natural Language Processing* tools and resources. The framework detects parts of the target text that can be replaced with new ones with the same grammatical category at the word level. Then, the replaced word is found in the source text using word alignment to replace it with a suitable synonym. The process includes iterative steps of identifying and replacing words until no further changes can occur. Our approach addresses the need for more extensive high-quality datasets in natural language processing tasks for low-resource situations. The proposed process is designed to support multiple languages and various text types. Large Language Models can be used to maintain the augmented data's quality and facilitate diverse textual datasets in both monolingual and multilingual cases. The quality of the augmented data is evaluated by humans based on various criteria such as fluency, coherence, and relevance, and automatic evaluations are performed to check the improvement in the language model's performance. This automated evaluation employs state-of-the-art metrics such as BLEU, TER, and chrF. The proposed framework aims to improve the quality and quantity of textual data that can be used for natural language processing tasks, such as machine translation.

Keywords: data augmentation, large language models, natural language processing, machine translation, text generation

Contents

List of Figures

List of Tables

CHAPTER 1
Introduction

Machine translation (MT) is a research field that harnesses computer algorithms and artificial intelligence to seamlessly translate text or speech from one language to another [1]. It involves a complex interplay of linguistic and statistical techniques, breaking a source language sentence's intricate structures and meaning to craft an equivalent expression in the target language. The journey towards achieving accurate computer-based translations has been lengthy, spanning decades of dedicated research and development. Yet, it wasn't until the advent of neural networks that the field of MT witnessed a quantum leap in performance.

The emergence of *Neural Machine Translation* (NMT) in recent years has started a groundbreaking era in the domain of automated language translation [1]. These sophisticated models, powered by deep learning prowess, have showcased remarkable proficiency when applied to many language pairs. However, it's crucial to recognize that the efficacy of MT hinges significantly upon the accessibility and scale of high-quality training datasets. When faced with inadequate or limited data, NMT models may falter, leading to translation inaccuracies, less-than-fluid prose, and suboptimal linguistic conversions.

One formidable obstacle that looms over the effective training of NMT models is the absence of extensive, quality training datasets [1]. Creating and meticulously filtering parallel corpora for each conceivable language pairing is an arduous and time-intensive, often demanding substantial human resources and domain expertise. Consequently, researchers and practitioners encounter formidable constraints in their quest to develop robust NMT systems, especially when addressing low-resource languages or specialized domains offering scanty data reservoirs.

In summary, MT remains a dynamic and promising field that relies on cutting-edge technology, and NMT has unlocked remarkable potential. However, the enduring challenge of data scarcity underscores the intricate nature of this evolving domain. It emphasizes the need for innovative solutions to ensure that the benefits of automated translation are accessible to a broader range of languages and specialized domains, bridging linguistic gaps and fostering global communication.

1.1 Motivation

Since the performance of MT models is usually linked to the amount of data used to train them, there is a need to create high-quality datasets. It is an essential concern in MT providers, such as Pangeanic, where I have developed this project and then applied it to the working pipeline. I decided to go for this project because there is little research

related to this topic, which attracted me. In addition, I am passionate about the artificial intelligence field and recent state-of-the-art language models. This passion started mainly due to two courses taught during my degree at university: *Perception* during the 3rd year and *Machine Learning* during the 4th year. Therefore, I found the perfect opportunity to learn and understand how this technology works and make this a basis for my future work in this field.

1.2 Objectives

The primary objective of this DFP is to explore and evaluate the effectiveness of enhancing textual datasets for MT or other NLP tasks using large language models. We want to achieve that by creating new synthetic data for the datasets used for training MT models. After fine-tuning the models with the enhanced dataset with synthetic data, we will evaluate the overall performance to check the effectiveness of the proposed data augmentation technique for parallel texts.

1.3 Memory structure

This work is structured into 6 chapters, each addressing a specific aspect of the research. First, we will explain what NLP is, the leading use cases, and the problems it has to deal with. We will also explain how computers understand text and process it. The next chapter will review the state-of-the-art technology used in this field and how it has improved MT. This chapter will also explore the evolution of the different techniques used to generate synthetic text. The following chapter introduces our approach to enhancing parallel datasets and providing an easy data augmentation tool. Finally, we evaluate and analyze the results obtained with our solution using different metrics, ending with a conclusion and assessment of the work with proposals for future work.

CHAPTER 2
Natural Language Processing

Natural Language Processing (NLP) is a subfield of artificial intelligence and computational linguistics that focuses on the interaction between computers and human language [1, 2]. This research field had its roots in the 1950s when Alan Turing published the article *Computing Machinery and Intelligence* [3], where he introduced the *Turing test* as an intelligence criterion for the first time. The test included tasks on generating and understanding natural language by computers. In subsequent decades, this discipline studied linguistic aspects of human-human and human-machine communication, developing linguistic models that employed computational frameworks to implement processes with these models.

Contrary to what someone may think, computers cannot understand language directly, so this has to be converted to representations computers can understand and process. It involves developing and applying algorithms and techniques to enable computers to understand, interpret, and generate human language meaningfully and practically. However, researchers have to face many problems in this process due to the intrinsic complexity of human language. This research field is essential and encompasses various computational tasks and applications, such as MT, text understanding, sentiment analysis, question answering, and text generation.

2.1 Main NLP tasks

Since its birth, NLP has tried to automate many natural language tasks that people previously could only do [1, 2]. Thanks to state-of-the-art technologies, a good deal of work can now be automated using computers and algorithms [4]. The main tasks involving this field are:

- **Machine Translation (MT):** The first approach in this field was made in 1954 by IBM when the *Georgetown experiment* [5] achieved the fully automatic translation of more than sixty Russian sentences into English. Since then, many NLP techniques have been used to develop MT systems that automatically translate text or speech from one language to another. These systems leverage statistical models, neural networks, and language resources to bridge the linguistic and cultural gaps between different languages, as reviewed in section 3.1.

- **Natural-Language Understanding (NLU):** Since the early 60s, researchers have made efforts to create algorithms that aim to understand the meaning and context of written text, including tasks such as part-of-speech tagging, named entity recognition, and syntactic parsing. These techniques enable computers to extract struc-

tured information from unstructured text data, being very useful today in tasks such as *anonymization*.

- **Question Answering:** NLP systems can process natural language questions and provide relevant and accurate answers by extracting information from extensive collections of documents or knowledge bases. This is particularly useful for information retrieval and virtual assistants.

- **Text Generation:** State-of-the-art NLP techniques and large language models enable computers to generate human-like text, including automated article writing, chatbots, and even creative writing. These systems employ language modeling, text summarization, and dialogue generation to produce coherent and contextually appropriate text, as seen in section 3.5.

- **Sentiment Analysis:** NLP algorithms focus on determining the polarity of a text, whether it is positive, negative or neutral, but it can go beyond that, detecting specific feelings and emotions, urgency and even intentions. This analysis can be helpful for understanding public opinion, customer feedback, and social media trends, among other applications.

This project focuses on the field of MT but it is important to understand what some of the other tasks involved in NLP consist of, as they are mentioned throughout the project.

2.2 NLP problems

Given that computers work with binary language, the NLP field faces many challenges in its pursuit to enable computers to comprehend natural language [1]. These challenges stem from natural language's inherent complexity and richness, which exhibits ambiguity [6], syntactic intricacies, contextual nuances, and a lack of explicit semantics [7]. Understanding and processing natural language requires tackling problems such as resolving ambiguity, grasping the contextual meaning [8], handling grammatical variations, and inferring implicit information. Additionally, the scarcity of annotated data, the variability of language [9], and the influence of cultural and social factors further compound the difficulties NLP research face. In this section, we delve into these NLP problems:

- **Syntax and Grammar:** Natural language has complex syntactic and grammatical rules. Understanding sentence structure, verb tenses, subject-object relationships, and other grammatical aspects is essential for proper comprehension. Handling grammatical errors, colloquial language, and variations in linguistic patterns across different languages further complicates the task.

- **Ambiguity:** Natural language is often ambiguous, also for people, and the exact words or phrases can have multiple meanings depending on the context. For example, the sentence *"The lecturer said on Friday she would take a test"* can either mean the lecturer told the students about the test on Friday or that the test will be held on Friday. Resolving this ambiguity requires a deep understanding of the context, background knowledge, and world events. Another ambiguity example: *"Time flies like an arrow; fruit flies like a banana."*. In the first sentence, *flies* is the main verb, while in the second sentence, *flies* is the subject. Humans can rely on background knowledge, common sense, and world experience to fill in gaps, but these are challenging for machines to acquire and utilize effectively.

- **Contextual Understanding:** Interpreting natural language requires considering the broader context beyond individual words or sentences. For instance, in the sentence *"Academic writing can be intimidating, especially when they only get negative feedback."* the pronoun *they* do not specify to whom it refers. However, since it is about academic writing, we can assume it refers to students. Understanding pronoun references, implicit meaning, sarcasm, metaphor, and other nuances of language often requires a grasp of world knowledge, cultural context, and common sense reasoning.

- **Variability and Creativity:** As time goes by, natural language exhibits tremendous variability in terms of vocabulary, sentence structure, and expressions. This variability speed increases as society evolves faster, creating, modifying or creating new expressions. For instance, *"That's a piece of cake."* is an idiom that means something is easy. Nonetheless, it is hard for a computer to recognize and interpret such creative expressions and understand their figurative meanings. People often use creative language, slang, idioms, and figurative speech, which can be challenging for machines to interpret accurately.

- **Data Sparsity:** Obtaining large-scale quality datasets can be challenging, especially for specific domains or low-resource languages. This is one of the most significant issues of NLP since language models' performance usually relies on the amount of data they have been trained on and its quality. This project will address an approach, explained in Chapter 4.1, to enhance datasets, specifically for MT, and improve its results.

2.3 Text normalization

As explained above, natural language is complex, and making a good text representation for computers was very challenging in the early stages of this field. A computer's language processing and understanding depend on how text is represented. Capturing the meaning and characteristics of a word is complex, and this could only be achieved with the appearance of deep learning.

Starting with the most basic, computers represent characters, including words, using character encoding. Character encoding is a system that assigns numeric codes to each character in a character set, allowing computers to store, represent and manipulate text data. There are various character encoding, but the most commonly used today is The Unicode Standard [1], based on an ISO international standard. This standard defines many formats, but the most common is UTF-8 (Unicode Transformation Format 8-bit). This variable-length encoding scheme can represent the entire Unicode character set, which includes a vast range of characters from different languages, symbols, emojis, and special characters. Each character is represented in this encoding by a variable-length 8-bit bytes sequence. This encoding takes advantage of this and it is better than other encodings like ASCII [10], whose characters are represented using a single byte. This is why it is limited to 128 different characters, the Latin alphabet, numbers, and basic symbols. For instance:

- The character 'A' is represented as 41_{16}, or 01000001_2, in ASCII encoding and UTF-8 encoding.

- The character 'Ω', the Greek capital letter Omega, can only be represented using UTF-8 as the sequence of bytes CE_{16} $A9_{16}$, or 11001110_2 10101001_2.

[1] *The Unicode Consortium*. Consulted in https://www.unicode.org/standard/principles.html

Now we know that text is a sequence of these representations equivalent to a sequence of numbers. However, a text must be converted to a more convenient, standard form. This process is called text normalization [1], where *tokenization* plays an essential role [11]. Tokenization consists of splitting a text into smaller pieces called *tokens* which can be sentences, words, subwords, or characters. During word tokenization, words can be converted to lowercase and punctuation and symbols can be removed, depending on the process's technic and final purpose. However, we must keep punctuation for most NLP applications, like MT, since it can change the text's meaning. For instance, in English, whitespace often separates words, but whitespace is not the only separator. In the sentence, *"I'm happy."* there are 3 words as *I* and *am* are different words. In some texts like tweets, emoticons like :) or hashtags like `#NLP` have to be tokenized. Furthermore, in some languages, such as Chinese and Japanese, words are not separated by whitespace, which makes them more challenging to tokenize. In fact, for many Chinese NLP tasks, it works better to take characters rather than words as input. This occurs because characters are at a reasonable semantic lever for most applications, resulting in a vast vocabulary with a large number of rare words [21].

There is a third option to tokenize text, which does not tokenize words or characters but uses data to automatically tell what the tokens should be [1]. This is especially useful in dealing with unknown words, a fundamental problem in language processing. As we will see in the chapter 3, NLP algorithms learn characteristics about language from a training corpus and then use these characteristics to make decisions about a different corpus and its language. Let us say a training corpus contains the words *slow*, *fast*, *faster*, but not *slower*. Then, if the word *slower* appears during an inference computation, the system will not know what to do with it. To deal with the problem of unknown words, modern tokenizers often automatically induce tokens smaller than words, called subwords. Subwords can be arbitrary substrings or meaning-bearing units like the morphemes *-est* or *-er*, which are the smallest unit of a language with meaning. For example, the word *unhappiest* has the morphemes *un-*, *happy*, and *-est*. Thus, the unseen word *slower* can be represented by a sequence of learned subword units, such as *slow* and *-er*. These tokenization systems comprise 2 parts: a token learner, which induces a set of tokens from the training corpus, and a token segmenter, which takes a raw text during inference and segments it into the learned tokens. 3 algorithms are widely used: Byte-Pair Encoding [12], Unigram Language Modeling [13] and WordPiece [14].

Another common task in text normalization is *lemmatization* [1], which determines that 2 words have the same origin, despite their surface differences. For example, the terms *wrote*, *written* and *writes* are different verb conjugations of *write*. In this case, the term *write* is the *lemma* of these words, and they are mapped to *write*. This process is beneficial in morphologically complex languages like Arabic. A simpler version of this process is called *stemming* and mainly consists of striping suffixes from the end of the word.

After text normalization, we obtain a text representation that NLP systems can process to compute a satisfactory result. For instance, an MT system preprocesses the source sentence, and after the text normalization, it computes a sentence in the target language with an equivalent meaning. Yet, this is a lexical representation that lacks meaning. Representing the meaning of tokens is a challenging question still nowadays. We will review how this meaning is captured in numbers in section 2.4.

2.4 Vectorial representations

The standard way to represent the meaning of language in NLP is by using numerical vectors that help the model mitigate many of the language aspects explained in section 2.2. This process is called vector semantics and has its roots in the 1950s, when 2 ideas merged. The first one was representing the connotation of a word with multidimensional vectors. While the second consisted of defining the meaning by its distribution in language and exposed that words with similar meanings will occur in very similar distributions. The main idea is to represent text tokens as a point in a multidimensional semantic space derived from the distributions of the token neighbors in a text. The vectors used to represent the token's meaning are called *embeddings*, whose name comes from the mathematical sense of mapping between different spaces or structures [1].

2.4.1. Term Frequency-Inverse Document Frequency

An early approach used in information retrieval, *Term Frequency-Inverse Document Frequency* (TF-IDF) [1], represented the relevance of a word by counting the nearby words. As its name says, it is the product of 2 terms. The first is the frequency of a word t in a document d, $\text{tf}_{t,d}$, defined as $\text{count}(t,d)$, squashed by using the $\log_{10}$ of the frequency. This squashing is done to avoid a word appearing 100 times more likely to be relevant than a word appearing 10 times. The second term gives higher weights to words that occur in a few documents. So it measures the rarity of a term across the entire document collection. The inverse document frequency idf_t is defined as N/df_t, where N is the number of documents, and df_t is the number of documents in which term t occurs. Since a large amount of documents can be found in many collections, this term is also squashed with a $\log_{10}$ function. The final function that defines TF-IDF results in a weighted value $w_{t,d}$ for the word t in a document d combining the previous terms:

$$w_{t,d} = \text{tf}_{t,d} \times \text{idf}_t = \log_{10}(\text{count}(t,d)+1) \times \log_{10}(\frac{N}{\text{df}_t})$$

The TF-IDF score can be calculated for each term in a document, creating co-occurrence matrices that capture the importance of each term. This vector representation can be used for various purposes, such as document similarity comparisons, keyword extraction, and information retrieval. However, it resulted in very long sparse vectors with mostly zeros and failed to capture words' semantic properties.

2.4.2. Positive pointwise mutual information

The following approach took TF-IDF as a base and focused on representing associations between 2 words, instead of between words and documents. *Pointwise mutual information* (PMI) [1] measures how often 2 events x and y occur together. In NLP, it counts how often 2 tokens co-occur in a corpus. Given a target word w and a context word c, its PMI can be defined as:

$$\text{PMI}(w,c) = \log_2 \frac{P(w,c)}{P(w)P(c)}$$

Where $P(w,c)$ is the probability of observing the 2 words together, and the denominator tells us the probability of the 2 words occurring by chance. This ratio helps us find words that have a strong association. Since this function returns values ranging from

negative to positive infinity, a positive version is often used. *Positive PMI* (PPMI) replaces all negative values with zero since negative values are unreliable unless our texts are enormous. So we can define the association of words x and c as $\max(\text{PMI}(w, c), 0)$. However, we will want to have a weighted association between all words in a text. To do this, we first create a co-occurrence matrix F with W rows and C columns, where each row is a word, each column is a context, and f_{ij} is the number of occurrences of the word w_i with the context c_j. Then, we can define the function PPMI_{ij} or $\text{PPMI}(w = i, c = j)$, which returns the PPMI value for the word w_i with context c_j:

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}, \qquad p_{i*} = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}, \qquad p_{*j} = \frac{\sum_{i=1}^{W} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

$$\text{PPMI}_{ij} = \max(\log_2 \frac{p_{ij}}{p_{i*}p_{*j}}, 0)$$

Let us see how it works in a small example. In Table 2.1, we have a co-occurrence matrix of a small sample of 4 words in 5 contexts, extracted from the Wikipedia corpus [2] and assuming the other words and contexts do not matter for the calculation.

	laptop	**data**	**result**	**pie**	**sugar**	**count(w)**
lemon	2	8	9	440	23	482
strawberry	0	0	1	50	90	141
digital	1651	1634	65	4	3	3357
information	3320	3971	370	5	15	7681
count(c)	4973	5613	445	499	131	11661

Table 2.1: Co-occurrence matrix extracted from the Wikipedia corpus for 4 words in 5 contexts, together with the marginals.

	laptop	**data**	**result**	**pie**	**sugar**
lemon	0	0	0	4,41	2,08
strawberry	0	0	0	3,05	5,83
digital	0,2	0,02	0	0	0
information	0,02	0,1	0,34	0	0

Table 2.2: Representing the association between words and contexts with a PPMI matrix.

Using this table, we can compute the association between words and context words using PPMI and get a new matrix, represented in Table 2.2. For example, we could compute the PPMI for the word *digital* and the context *laptop*.

$$P(w = \text{digital}, c = \text{laptop}) = \frac{1651}{11661} = 0,1415$$
$$P(w = \text{digital}) = \frac{3357}{11661} = 0,2878$$
$$P(c = \text{laptop}) = \frac{4973}{11661} = 0,4265$$
$$\text{PPMI}(w = \text{digital}, \text{laptop}) = \max(\log_2(\frac{0,1415}{0,2878 * 0,4265}), 0) = 0,2051$$

[2] Available at https://huggingface.co/datasets/wikipedia

After getting the whole matrix of PPMI values, we can observe that lemon and strawberry are strongly associated with pie and sugar, and digital and laptop are mildly associated too. We can find many zeros that were negative values, meaning that there is no relation between that word and the context. This system has limitations since infrequent words tend to have very high values, which creates a bias toward infrequent events. Furthermore, both techniques, TF-IDF and PPMI, resulted in very long sparse vectors with mostly zeros and failed to capture words' semantic properties. This issue is because the resulting vectors had a dimension as high as the number of documents, in TF-iDF, or known words, in PPMI.

2.4.3. Word2vec

To address the sparse vectors problem, researchers found that dense vectors performed better in capturing the semantic meaning of language [1]. Unlike the long sparse vectors seen above, these embeddings are short dense vectors with positive and negative real numbers. The dimension of these vectors ranges from 50 to 1000 dimensions, rather than the large number of terms in a vocabulary or documents in a collection.

It turned out that these vectors are better for NLP tasks since models have to learn fewer weights. Besides, they succeed in capturing synonymy and helping with generalization. For example, the dimensions for synonyms like the words *pet* and *fish* could be very unrelated and distinct in sparse vectors. Thus it would fail to catch the similarity of a term with *pet* as a neighbor and a term with *fish* as a neighbor. The idea of using dense vectors was first developed by Google in 2013 with *word2vec* [15], a software that combines 2 algorithms. These 2 algorithms are explained below.

- **Continuous Skip-gram Model:** This model predicts the context words given a target word. It operates by sliding a fixed-size window over a sentence and training the model to predict the surrounding words given a central target word within that window. It is fed with a one-hot encoded vector, a zeros vector of the vocabulary size, where there is only a 1 in the position corresponding to that word. The output of this model is a probability distribution over the vocabulary for the context words.
  The model's objective is to maximize the likelihood of the context words given the target word. It achieves this by learning to update the word vectors to increase the similarity between the target word's vector and the context word vectors. The training process involves iteratively adjusting the word vectors using gradient descent to minimize the difference between the predicted probabilities and the actual context words' one-hot encoded vectors.

- **Continuous Bag-of-Words Model (CBOW):** This model, on the other hand, focuses on predicting a target word based on its surrounding context words. It works by taking a window of context words and training the model to predict the central target word. It is fed with the one-hot encoded vectors of the context words and returns a probability distribution over the vocabulary for the target word.
  The objective of the CBOW model is similar to that of the skip-gram model: to maximize the likelihood of the target word given its context words. The model adjusts the word vectors to increase the similarity between the context word vectors and the target word's vector. Like the skip-gram model, the CBOW model employs gradient descent to iteratively update the word vectors to minimize the difference between predicted probabilities and actual target words' one-hot encoded vectors.

2.4.4. Embeddings from Language Models

Unlike the above word embeddings, *Embeddings from Language Models* (ELMo) [16] word representations are functions of the entire input sentence. ELMo is designed to capture the contextual meaning of words in a sentence, which is crucial for tasks like sentiment analysis, MT, and question answering. ELMo uses a bidirectional *Long Short-Term Memory* (LSTM) network to generate word embeddings, a type of *recurrent neural network*, see section 3.3. This network can capture dependencies and relationships between words in both directions within a sentence, forward and backward. ELMo uses a stacked, two-layer LSTM network. ELMo also incorporates character-level information to generate word embeddings. It creates a character-level representation for each word in the vocabulary using a character convolutional neural network, which allows the model to capture morphological and subword information.

The critical innovation of ELMo is that instead of generating a single, fixed vector representation for each word, ELMo produces multiple embeddings for each word, which are context-dependent. It does this by running the bidirectional LSTM over the sentence and obtaining hidden states at each time step for each word. These hidden states are vectors that capture the word's contextual information. To combine these contextual embeddings, ELMo uses a weighted sum. Each embedding is assigned a weight learned during the training process and these weights depend on the target task, making the model adaptable to different NLP tasks.

Then, ELMo embeddings can be integrated into downstream NLP models, concatenating or averaging ELMo embeddings with traditional word embeddings to enhance their performance in various tasks. Experiments confirmed that ELMo efficiently encodes different types of syntactic and semantic information about words' context, improving overall task performance. It was one of the groundbreaking models for contextual word embeddings. Nevertheless, later technologies outperformed ELMo representations, like BERT's architecture, based on the Transformer model, which is more sophisticated and powerful than ELMo in several ways. Transformer architecture is explained in section 3.4.

2.4.5. Data2vec

Data2vec [17] is a framework for either speech, NLP, or computer vision which uses the same learning method. It aims to predict latent representations of the entire input data based on a masked input in a self-distillation setup, which uses a standard Transformer architecture. The method combines masked prediction with the learning of latent target representations. Still, it generalizes this by using many network layers as targets, allowing it working across several data types.

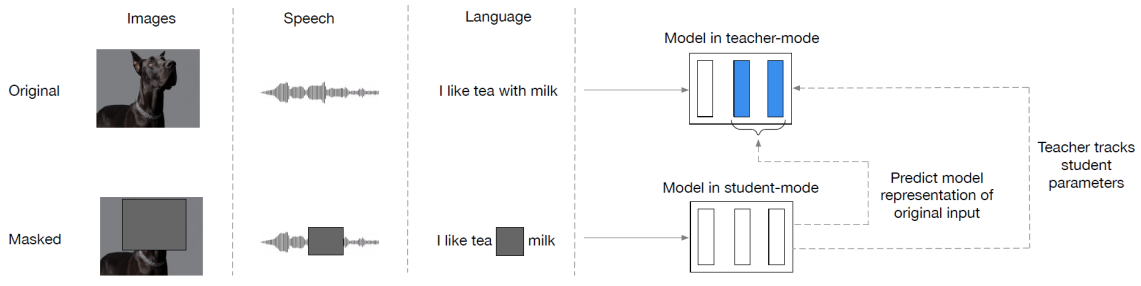

Figure 2.1: Data2vec learning process for different data types. Figure taken from *data2vec's research paper* [17].

The approach consists of training a Transformer network, which is used either in teacher or student mode, as seen in Figure 2.1. First, representations of the total input data are built to serve as targets in the learning task for the model in *teacher-mode*. Next, a masked version of the input sample is encoded in the model in *student-mode* to predict the full data representations. The weights of the teacher are an exponentially decaying average of the student. Since different modalities have vastly different inputs, each modality uses specific feature encoders and masking strategies from the literature.

Compared to prior NLP algorithms, data2vec predicts a continuous and contextualized representation instead of discrete linguistic tokens such as words, sub-words, or bytes. Thus, the model manages to adapt to a particular input example since the targets are not predefined, nor are their number limited. Besides, targets are contextualized, taking context information into account.

Experiments' results show that language models trained using data2vec outperform the previous techniques. Researchers successfully pre-trained the first NLP model which does not use discrete units as the training target, like words, subwords, characters, or bytes. Instead, the model predicts a contextualized latent representation emerging from self-attention over the entire unmasked text. This allows the model to compute targets with specific nuances of the current text instead of tokens that are generic to every text in which the particular discrete unit occurs. For this reason, the set of training targets is not a closed vocabulary, and the model can choose to define new targets as it sees fit.

2.5 Problem analysis

As mentioned, NLP algorithms rely on statistical models, machine learning techniques, and linguistic rules to process and analyze language data. Besides, they require large amounts of labeled training data to learn patterns and generalize from examples [1]. We will focus on the nature of the data used during the training of MT systems.

MT models are trained on a parallel corpus, a text that appears in 2 or more languages. There are several parallel corpora available with different origins. They can be governmental, like the *Europarl* corpus [18], which contains between 400 000 and 2 M sentences from 21 European languages extracted from the proceedings of the European Parliament. Other parallel corpora have been made crawling general web text, like *ParaCrawl* [20]. This large-scale corpus is automatically crawled from the internet periodically, covering various domains and multiple languages. Movie and TV subtitles have also been employed to create datasets, like the *OpenSubtitles* corpus [19]. This data for MT training come as aligned pairs of sentences. These sentence alignments must be created when crafting new corpora for low-resource languages or new domains, for example. The problem appears here since creating corpora for low-resource languages or specific domains is challenging. There are a lot of large parallel corpora with translations between English and other languages, but most of the world's languages have few or no parallel texts available. Moreover, the resource problem can be worst when translating low-resource domains, even for high-resource languages [1].

How to deal with this data sparsity is an essential ongoing research question, and several approaches have been developed to address it. These approaches are included in a general statistical technique called *data augmentation* (DA), which aims to generate new synthetic data from the current natural data available. The origins of DA can be traced back to computer vision and image classification tasks, where it has been used for decades. Researchers and practitioners found that applying simple geometric and color transformations to images could substantially improve the performance of image classification models. This practice became particularly important when working with

small or limited datasets. Data augmentation has since been extended to other domains, including NLP and speech recognition, where techniques like text paraphrasing and audio speed perturbation are used to create augmented datasets. Synthetic data helps the model learn invariant features and patterns, which is essential for better generalization. It exposes the model to a broader range of possible inputs.

We will review some approaches to augment text data in section 3.6 and present our solution for enhancing training datasets for MT models in section 4.1.

CHAPTER 3
State-of-the-art

Before presenting our solution to the problem, we must explain how the technologies that appear in this project work and the advances they have experienced over time to get to the point they are now. We will explore the underlying principles, methodologies, and recent advancements in MT and LLMs. Furthermore, we will review some approaches in related works in enhancing datasets for machine translation.

3.1 Machine Translation

Communication between different countries and cultures is essential, enabling information exchange and the continuity of our evolution. However, the sender and receiver of the information usually understand different languages, so this information must be translated into the receiver language to complete the communication. That is why translation, in its full generality, is now more critical than ever due to the great globalization that the world is suffering. Unfortunately, translation requires qualified people with excellent knowledge of the communication's source and target languages. The service of these translators is usually costly and even difficult to find for low-resource languages. For this reason, new ways of translating using technology began to be sought.

Machine translation (MT) [1], the process of automatically translating text from one language to another using computers, has been a big goal in the field of artificial intelligence for a long time. It has witnessed a remarkable evolution over the years, transforming how we bridge language barriers and facilitate global communication. MT has enabled the information access to anyone worldwide, no matter what language they speak. This fact has also helped to reduce the so-called *digital divide*, as much more information is available in English and languages spoken in wealthy countries. Speakers of lower-resourced languages can benefit from high-quality translation to receive information and knowledge.

From the early days of rule-based systems, which relied on manually crafted linguistic rules, to the statistical models that used the power of vast bilingual corpora, and finally, the breakthrough of neural networks with its deep learning architectures, MT has continuously evolved to deliver increasingly accurate and fluent translations. We will assess this evolution below.

3.1.1. Rule-Based Machine Translation

Rule-Based Machine Translation (RBMT) is an early approach in MT developed in the early 1970s, wherein translation is achieved through manually crafted linguistic rules and dic-

tionaries. RBMT systems operate by employing a set of predefined rules and syntactic structures specific to both the source and target languages [22].

The RBMT process encompasses several distinct stages. Initially, the source sentence undergoes analysis, examining its grammatical structure and extracting relevant linguistic information. This analysis involves parsing the sentence and identifying components such as elements of speech, verb conjugation, noun declension, and other linguistic features. Subsequently, the transfer stage is employed, wherein linguistic rules are applied to transform the analyzed source sentence into an intermediate representation that captures the intended meaning. This transformation involves various operations, including word order modifications, verb form adjustments, and structural transformations tailored to align the source sentence with the target language's requirements. Once the transfer stage is completed, the RBMT system proceeds to the generation phase, wherein the translated sentence is produced in the target language. This process entails mapping the intermediate representation to the target language's appropriate linguistic structures and vocabulary. Lexical and grammatical rules are employed to ensure accurate and fluent translations.

Critical technical aspects of RBMT enclose the utilization of linguistic rules, dictionaries, and knowledge engineering. Linguistic rules are manually constructed by linguists and domain experts, encompassing various linguistic aspects such as grammar, syntax, morphology, and semantics. These rules define patterns and transformations required to convert the source language structure into the corresponding target language structure. RBMT systems also rely on dictionaries that contain word lists, lexicons, and morphological information for both the source and target languages. These dictionaries provide the necessary vocabulary and semantic mappings crucial for accurate translations. Additionally, knowledge engineering plays a pivotal role in RBMT system development, involving extensive analysis of linguistic properties and collaboration between linguists and experts to refine rules and dictionaries based on linguistic knowledge and data analysis.

Nevertheless, RBMT also exhibits certain limitations. The development of RBMT systems needs comprehensive linguistic resources, which can be arduous and expensive to create. Furthermore, RBMT encounters challenges in handling natural language nuances, such as idiomatic expressions, ambiguity, and word sense disambiguation. Adaptation to new languages or domains also poses difficulties for RBMT, often requiring extensive manual rule development. While RBMT serves as a foundational approach in machine translation and contributes valuable insights into language structure and translation principles, its utilization has diminished in recent years.

3.1.2. Statistical Machine Translation

Statistical machine translation (SMT) is the following approach that appears to automatically translate text from one language to another, first introduced in 1988 [23]. It is based on statistical models that learn patterns and relationships between words, phrases, and sentences in a parallel corpus.

The core idea behind SMT is to leverage the statistical properties observed in the training data to make translation decisions in data the model has not seen. In SMT, a text is translated according to the probability distribution $P(y|x)$ that the target language y is the translation of the source language x, given a set of model parameters θ [24]. When translating a new sentence, the goal is to find the translation $\widehat{y}$ with maximum probability, expressed as:

$$\widehat{y} = \arg\max_{y} P(y|x;\theta)$$

One approach that works well for computer implementation is applying *Bayes Theorem*, that is $P(y|x;\theta) = P(x|y;\theta_{tm})P(y;\theta_{lm})$, where the translation model $P(x|y;\theta_{tm})$ is the probability that x is the translation of y according to a set of parameters θ_{tm} learned during training, and the language model $P(y;\theta_{lm})$ is the probability of seeing y. A language model is a probability distribution over a sequence of words according to a set of parameters θ_{lm} generated with the training data. So the above equation can be re-written as:

$$\widehat{y} = \arg\max_{y} P(y|x;\theta) = \arg\max_{y} P(x|y;\theta_{tm})P(y;\theta_{lm})$$

The translation model in addition, is defined as a generative model, which is disintegrated via latent structures:

$$P(x|y;\theta_{tm}) = \sum_{z} P(x,y|z;\theta_{tm})$$

where z denotes these latent structures that can be word-based or phrase-based. In word-based systems, a technique called word alignment is used, which role is to find correspondence between words in the training parallel corpus. The problem with this approach is that generalizing is hard due to dependencies between sub-models. To solve this issue, in 2002 log-linear models were used to introduce knowledge sources:

$$P(y,x|\theta) = \frac{\sum_z \exp(\theta * \psi(x,y,z))}{\sum'_y \sum'_z \exp(\theta * \psi(x',y,z'))}$$

where $\psi(x,y,z)$ is a set of features defining the translation process and θ denotes the corresponding weights for each feature. Later in 2003, a new phrase-based translational model [25] was introduced, where source and target sentences are divided into smaller units called phrases. A phrase is a contiguous sequence of words that appears frequently in the training data. Each phrase pair, consisting of a source phrase and its corresponding target phrase, is assigned a weight in θ_{tm} that represents the likelihood of that translation. During the training phase, these weights are learned by examining the frequency of phrase pairs in the parallel corpus. During the translation process, the source sentence is segmented into phrases, and the system searches for the best translation of each source phrase based on the learned phrase pair weights. This search is guided by the language model, which estimates the probability of a target sentence based on the target language alone based on the parameters θ_{lm}. The translation output is selected by optimizing a scoring function that combines the phrase pair weights θ_{tm}, the language model probabilities θ_{lm}, and other factors like phrase reordering and distortion [24].

To train an SMT system, a large parallel corpus is required, consisting of aligned sentences in the source and target languages. The training process involves several steps, including extraction and alignment of the latent structures and model estimation. While SMT has been widely used and has achieved significant success in various translation tasks, it does have limitations [24]. These problems are data sparsity and feature engineering. The first one is due to the discrete symbol representation, which results in a weak model's effectiveness in capturing the features of the language. The last one is that SMT includes annotating hand-crafted attributes to capture local syntactic and semantic features. Designing general features is a challenge, since there can be millions of such features and mapping them between languages can be a cumbersome task.

3.1.3. Neural Machine Translation

Neural Machine Translation (NMT) has revolutionized the field of automated translation by employing deep learning techniques to generate high-quality translations. Unlike traditional SMT systems, NMT directly learns the mapping between a source language sentence and its target language translation without relying on explicit phrase-based or alignment models. They are are able to learn complex relationships among natural languages from the data without the need to manually hand features, which are hard to design [24]. The main idea remains the same, given a sentence $\mathbf{x} = x_1, \ldots, x_j, \ldots, x_J$, of length J in the source language, and a sentence $\mathbf{y} = y_1, \ldots, y_i, \ldots, y_I$ of length I in the target language, NMT tries to factor a sentence-level translation probability into sub-word translation probabilities:

$$P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}) = \prod_{i=1}^{I} P(y_i|\mathbf{x}, \mathbf{y}_{<i};\boldsymbol{\theta})$$

where $\mathbf{y}_{<i}$ is referred to as a partial translation of $\mathbf{y}$. This system has to face some issues, like the fact the context between the source sentence and the target sentence can be dispersed when the sentences become too long. To solve this issue, the seminal paper *Sequence to Sequence Learning with Neural Networks* [26], published in 2014, proposed an encoder-decoder network that could represent variable-length sentences into a vector representation of fixed length and use it to translate. This architecture is explained in section 3.2. However, before explaining the encoder-decoder architecture and its types, we must delve into how a neural network works.

A *neural network* is a network of small computing units, each taking a vector of input values and producing a single output value [1]. At its heart, a neural unit takes a weighted sum of its inputs, with one additional term, called a bias term, in the sum. So given a vector of real-valued numbers $\mathbf{x}$, a unit has a corresponding vector of weights $\boldsymbol{\theta}$ and a bias b that uses to compute a weighted sum z:

$$z = b + \sum_i \theta_i x_i = \boldsymbol{\theta} \cdot \mathbf{x} + b$$

The sum can be replaced by a dot product to use a more convenient vector notation. Then, instead of using z as output, resulting from a linear function of $\mathbf{x}$, a non-linear function f is applied to z. This non-linear function is called the *activation function*, and its result is the activation value for the neuronal unit, y. Since it is a neural network of only one unit, the result of the whole network would be $y = f(z)$. A diagram of a simple neural network with only one neuron can be seen in Figure 3.1, where the input is a three-dimensional vector $\mathbf{x}$.

Depending on the purpose of the neural network, an activation function that fits the task is chosen. There are many activation functions, but 3 are the most popular [1]:

- **Sigmoid:** This function maps the output into the range $(0, 1)$, which helps normalize concentrating the values between 0 and 1 and enables it to be used as a probability. This function is differentiable, crucial for gradient-based optimization algorithms like *backpropagation*. In practice, it is not commonly used as an activation function but fits in binary classification problems where the goal is to classify inputs into 2 classes.

$$y = \sigma(z) = \frac{1}{1 + e^{-z}}$$

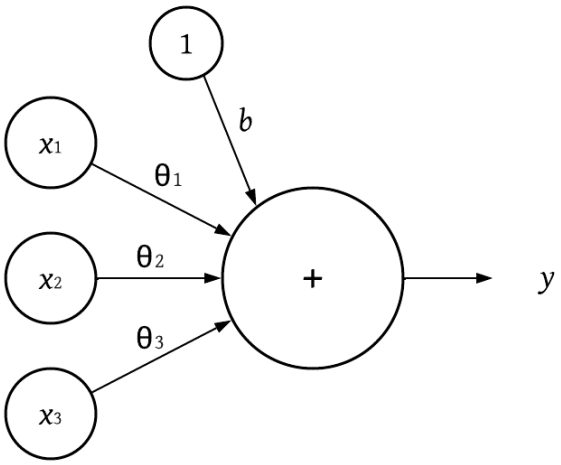

Figure 3.1: A single-neuron neuronal network, with a three-dimensional vector $\mathbf{x}$ as input, a weighted vector $\boldsymbol{\theta}$ and a bias b.

- **Hyperbolic Tangent (tanh):** This function is a variant of the sigmoid function and maps the output between the range $(-1, 1)$, so it is centered at 0 and provides stronger non-linearity. It maps negative inputs to negative values close to -1, positive inputs to positive values close to 1, and 0 input to 0. This is often used in hidden layers of neural networks, as it can handle both positive and negative inputs and it is beneficial for capturing complex relationships.

$$y = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

- **Rectified Linear Unit (ReLU):** This function is the simplest and possibly the most used. It is a simple thresholding function that sets negative values to 0 and leaves positive values unchanged. Besides, it introduces sparsity and non-linearity into the neural network, and it is computationally efficient and alleviates the *vanishing gradient* problem, which can occur with other activation functions. ReLU is widely used in deep learning models as it helps to mitigate the overfitting problem and accelerates training convergence.

$$y = \text{ReLU}(z) = \max(z, 0)$$

Knowing how a neuron works, we can now assume that a neural network is a set of connected units that compute the result. The training process in NMT uses parallel corpora to maximize the logarithmic likelihood of the source and target language and learn the optimal parameters $\widehat{\boldsymbol{\theta}}$. So we can define the objective function:

$$\widehat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{I} \log P(\mathbf{y}^{(i)} | \mathbf{x}^{(i)}; \boldsymbol{\theta})$$

After training, the learned parameters $\widehat{\boldsymbol{\theta}}$ are used to inference a translation $\widehat{\mathbf{y}}$:

$$\widehat{\mathbf{y}} = \arg\max_{\mathbf{y}} P(\mathbf{y} | \mathbf{x}; \widehat{\boldsymbol{\theta}})$$

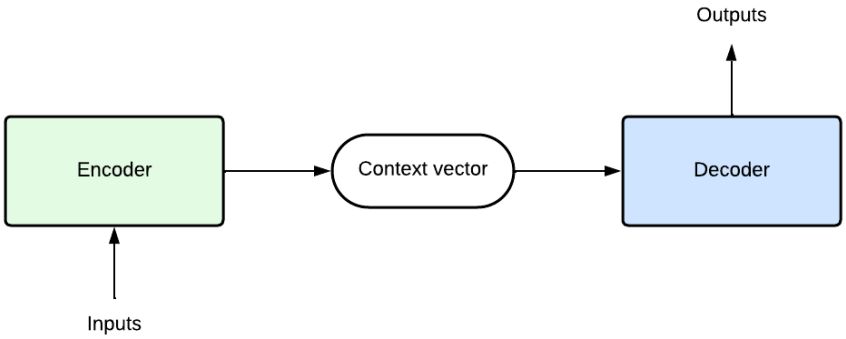

Figure 3.2: Encoder-Decoder Architecture.

3.2 Encode-Decoder Architecture

The *encoder-decoder architecture* is a fundamental framework in deep learning, first proposed in 2014 [26]. This architecture consists of a two-stage process, where the input is introduced into an encoder unit that maps its information into a numerical representation. This numerical representation is also called *context vector*, which is then fed into the decoder network to generate the output. We will focus on the MT encoder-decoder architecture and how these parts work, assuming the inputs and the outputs are sequences of text tokens. The architectural flow of a basic encoder-decoder based network can be seen in Figure 3.2.

- **Encoder:** During the encoding process, each token in the input sequence is fed into the encoder network sequentially. The encoder maintains an internal state that captures the information from the previously processed tokens. This internal state is updated at each step based on the current input token and the previous internal state. This allows the encoder to capture the contextual information and dependencies within the input sequence.

- **Context vector:** The encoder's final output is a condensed representation of the entire input sequence in a fixed-length numerical vector. It contains the accumulated features and context necessary to generate the output sequence. The model's performance depends on how well the decoder can represent the input's characteristics.

- **Decoder:** The decoder network takes the fixed-length context vector generated by the encoder and generates the output sequence token by token. At each step of the decoding process, the decoder receives an input token from the previously generated sequence. The decoder maintains its own internal state, which is initialized with the context vector from the encoder.

Since source and target sentences are usually of different lengths, these encoder and decoder units are usually based on recurrent neural networks (RNNs), explained in section 3.3. However, RNNs showed some limitations, like vanishing or exploding gradients, which led to the encoder-decoder transformer architecture, explained in section 3.4.

3.3 Recurrent Neural Networks

Language is an inherently temporal phenomenon. Spoken language is a sequence of acoustic events over time, and we comprehend and produce both spoken and written language as a continuous input stream. *Recurrent Neural Networks* (RNNs) [24] are a family of neural networks explicitly designed for sequential data processing, such as speech and text. The main idea behind RNNs is to benefit from this sequential data structure, and they are given this name since they work recurrently [1, 26]. Thus, any network with a cycle within its connections is recurrent, meaning that the value of some unit is directly or indirectly dependent on its own earlier outputs as an input. We will focus on encoder-decoder RNNs, also called sequence-to-sequence networks, which work well for MT. These models are instrumental in MT since they allow the source and target sentences to be of different lengths. Unlike SMT, this architecture does not maps tokens between the input and the output, which does not work for many languages. For example, in some languages, verbs are at the end; in others, the verb is at the end of the sentence.

As mentioned in section 3.2, an encoder-decoder architecture is based on a pair of RNNs. This type of langauge models try to predict the next word in a sequence given a preceding context. For example, if the proceeding context is *"I love"* and we want to know how likely the next word is *"you"*, we would compute $P(you \mid I\ love)$. Then, we can assign a probability to the whole sentence *"I love you"*, using the chain rule:

$$P(I\ love\ you) = P(I)\, P(love \mid I)\, P(you \mid I\ love)$$

The starting point to build the encoder and decoder units is the RNN language model $P(y)$, which stands for the probability of a sequence y of length n:

$$P(y_{1:n}) = \prod_{i=1}^{n} P(p_i \mid p_{<i}) = p(y_1)p(y_2 \mid y_1) \dots P(y_n \mid y_1, \dots, y_{n-1})$$

RNNs process one word of the sequence y at a time. Each word is represented as an embedding, explained in section 2.4. The encoder can be seen as a series of interconnected computational units, where each unit takes the current word's embedding and the computed state, also called the *hidden state*, of the previous unit as inputs. The encoder would have n computational units, with each unit corresponding to one word in the sentence. This allows the encoder to gradually incorporate information about the sentence by considering each word's context in relation to the previous words. Processing the words sequentially and updating the hidden state at each step captures the relevant information in the input sequence. The encoding process ends with the hidden state of the last word, which becomes the context vector. Then, the decoder unit uses the context vector as the starting point to generate the first word of the target sentence. This word is generated using the context vector and a unique token which determines that the source text has finished and the target text will start. The generated word is then fed into the next unit along with the current hidden state to generate the next word. Finally, the decoding process continues until a special token determines that the sentence has finished. More formally, the output of a unit at a particular time t can be defined as:

$$\mathbf{h}_t = g(\mathbf{h}_{t-1}, \mathbf{x}_t)$$
$$\mathbf{y}_t = f(\mathbf{h}_t)$$

Where g is an activation function, like tanh or ReLU, of the vector embedding representing the input word $\mathbf{x}_t$ and the hidden state of the previous unit $\mathbf{h}_{t-1}$. The final

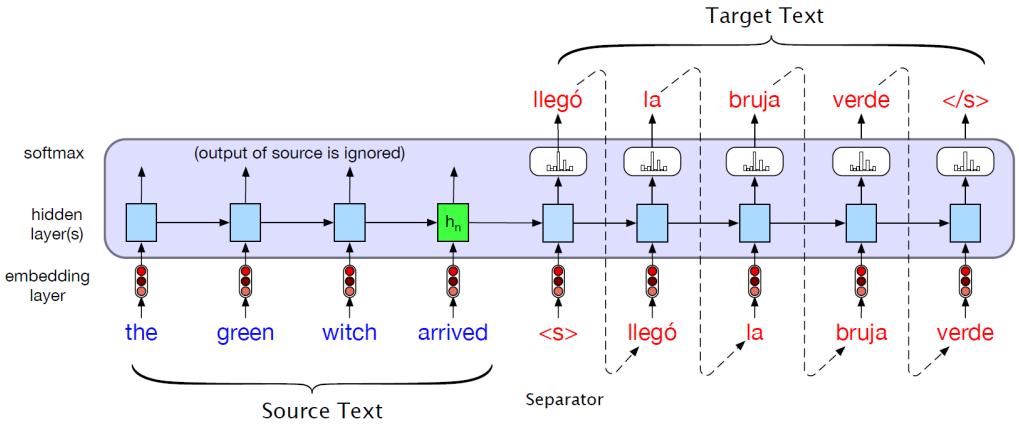

Figure 3.3: An inference example of a vanilla encoder-decoder with RNNs. Taken from *Speech and Language Processing* [1].

result at time t is a word embedding determined with a softmax function f of the hidden state. A softmax is a function that takes a vector of arbitrary values and maps them to a probability distribution, with each value ranging from 0 to 1 and all summing to 1. After applying the softmax function, the resulting probability distribution represents the model's belief about the likelihood of each known word in the model learned vocabulary. This process can be hard to imagine, so the figure 3.3 shows an example of an inference process, where the English sentence *"the green witch arrived"* is translated into Spanish. In this example, the token `<s>` separates the source and target sentences, and the token `</s>` ends the target sentence.

Encoder-decoder models with RNNs are trained with parallel corpus with sentences from the source and target languages. These sentences are fed into the model as one input. Thus, the source sentence is concatenated with the target sentence with a separator between them and a final unique token that determines the end of the sequence. Then a *self-supervision* algorithm is used to predict the next token at each time step t. It is called self-supervised because the data used is not labeled, and the model learns from the structure of the data. The model weights are learned by minimizing the error in predicting the next token in the training data, and they are adjusted using *cross-entropy* as a loss function. A loss function computes the difference between the correct distribution and the predicted probability distribution. So at time t, the cross-entropy loss L_{CE} is the negative log probability the model allocates to the next token w_{t+1} in the sequence.

$$L_{CE}(\hat{\mathbf{y}}_t, \mathbf{y}_t) = -\log \hat{\mathbf{y}}_t[w_{t+1}]$$

where $\hat{\mathbf{y}}_t$ is the predicted token and $\mathbf{y}_t$ is the correct token that comes from the training data. Thus, at each time step t, the model inputs the correct sequence of tokens $w_{1:t}$, and uses them to compute a probability distribution over possible following words to compute the model's loss for the next token w_{t+1}. Then we move to the next word, ignoring what the model predicted and instead, it uses the correct sequence of tokens to predict the word $wt + 2$. This technique is called *teacher forcing*, which gives each time step the correct sequence to predict the next token.

This approach has weaknesses when predicting the target sentence since as the output sequence is generated, the influence of the context vector will decrease. This issue can be solved by making the context vector available at each step in the decoding process by adding it as a parameter to the computation of the current hidden state. However, it still presents another problem. Since the decoder has the context vector as the only representation of the source sentence, it should capture all the meaning in the source

sentence. Regardless, in large sentences, the information at the beginning of the sentence can be less representative due to the recurrent computation of the context vector. The *attention mechanism* is a solution to the bottleneck created by the context vector. The idea of attention is focusing on a particular part of the source text relevant to the token the decoder is currently producing. The context vector is created as a weighted sum of all the encoder hidden states, depending on the token being decoded. To compute this dynamic vector $\mathbf{c}_i$, at each decoding step i, we focus on how relevant each encoding state $\mathbf{h}_j^e$ is for the previous decoding hidden state $\mathbf{h}_{i-1}^d$. The relevance is calculated as a score in many ways. The simplest way is to compute the *dot-product attention*. Then, these scores are normalized using a softmax function to create a vector of weights α_{ij}, which sets the relevance for each encoder hidden state j to the prior hidden decoder state $\mathbf{h}_{i-1}^d$.

$$\alpha_{ij} = \text{softmax}(\mathbf{h}_{i-1}^d \cdot \mathbf{h}_j^e) = \frac{\exp(\mathbf{h}_{i-1}^d \cdot \mathbf{h}_j^e)}{\sum_k \exp(\mathbf{h}_{i-1}^d \cdot \mathbf{h}_k^e)} \quad \forall j \in e$$

Finally, we get a context vector $\mathbf{c}_i$ for the current decoder state with a weighted average over the encoder hidden states $\mathbf{h}_j^e$. This process considers information from all the input tokens depending on the generated token.

$$\mathbf{c}_i = \sum_j \alpha_{ij} \mathbf{h}_j^e$$

The attention mechanism significantly improved the performance of the translation. However, NMT still had some changes to overcome. The large number of computations in RNNs and the need to maintain the hidden state at each step of training made the training highly inefficient and time-consuming. In 2017, *convolution networks* [31] were suggested to deal with this issue. The main advantage of this approach is that convolution operation does not depend on previously computed values and can be parallelized for multi-core training, making them faster for long sentences. Convolution networks can be stacked one after the other to substitute RNNs in the encoder-decoder architecture. While RNNs compute dependency among tokens in a sentence in $O(n)$, a convolution network can achieve the same in $O(\log_k n)$, where k is the size of a small matrix used in the convolution computation.

3.4 Transformers

In 2017 a new *Transformer* architecture was presented in the paper *Attention Is All You Need* [28], proposed by Google's researchers. This approach avoids recurrence and relies entirely on an attention mechanism to trace global dependencies between input and output. In the above networks, the required number of operations to create relations among 2 arbitrary tokens in the input or the output grows linearly in the distance between them. In the Transformer architecture, the number of operations is reduced to a constant number of operations since it does not use RNNs or convolution.

This architecture follows the encoder-decoder structure, where the encoder maps a text sequence $\mathbf{X} = \mathbf{x}_1, \ldots, \mathbf{x}_T$ to a sequence of continuous representation $\mathbf{H}^{enc} = \mathbf{h}_1, \ldots, \mathbf{h}_T$. Given $\mathbf{H}^{enc}$, the decoder generates the output sequence $\mathbf{Y} = \mathbf{y}_1, \ldots, \mathbf{y}_S$, token by token. At each step, the decoder is fed the previously generated tokens to generate the new ones, which is *auto-regressive*. Figure 3.4 represents the model's distribution and the components inside the encoder and the decoder. Both encoder and decoder transformers are

a stack with the same number of layers N. The input is fed into the encoder tokenized, representing each token as an embedding.

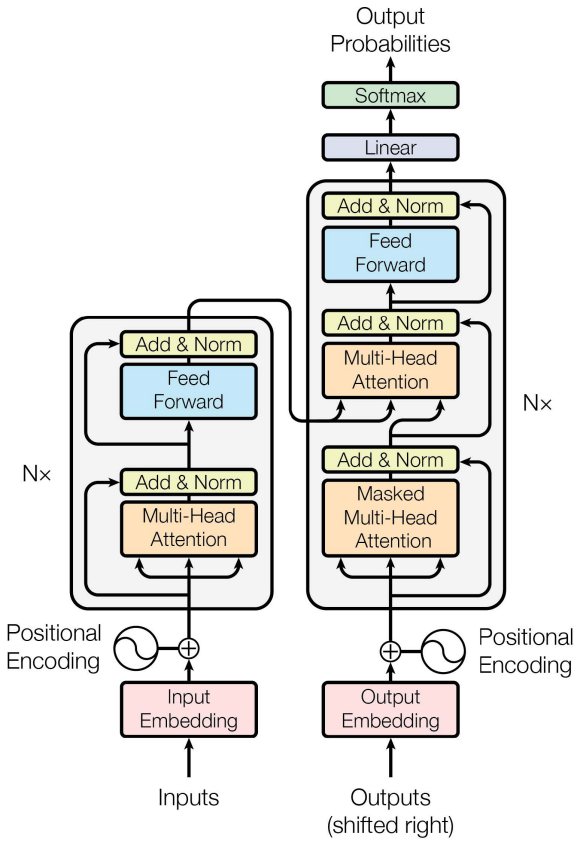

Figure 3.4: Encoder-decoder Tranformer architecture. Where N is the number of layers in the encoder and decoder units. Figure taken from *Attention Is All You Need* [28].

The encoder is a stack of 6 identical layers, each with 2 sub-layers. The first is a multi-head *self-attention mechanism*. A self-attention layer maps input sequences $\mathbf{x}$ with output sequences $\mathbf{y}$ of the same length. The unit only has access to the previous inputs $\mathbf{x}_{1:i}$, and itself, to process the value for the current token $\mathbf{x}_i$, but it has no access to the information of the tokens beyond it. In addition, these computations are independent, so they are easily parallelizable. Figure 3.5 shows how the relations in this layer would be for an input of 5 tokens. For instance, $\mathbf{y}_3$ is computed using $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$. Instead of the simple dot product comparison used in RNNs, transformers have a more sophisticated way of representing the contribution of each word in the representation of longer inputs. Each input embedding can play 3 different roles during the attention process: a role **query**, as the current focus of attention when being compared to all the preceding inputs, a role **key**, as a preceding input being compared to the current focus of attention, and as a **value** used to compute the output for the current focus of attention. 3 weight matrices, $\mathbf{W}^Q$, $\mathbf{W}^K$ and $\mathbf{W}^V$, are introduced to project each input $\mathbf{x}_i$ into a representation of its roles:

$$\mathbf{q}_i = \mathbf{W}^Q\mathbf{x}_i; \qquad \mathbf{k}_i = \mathbf{W}^K\mathbf{x}_i; \qquad \mathbf{v}_i = \mathbf{W}^V\mathbf{x}_i$$

All the inputs $\mathbf{x}$ and outputs $\mathbf{y}$, as well as the intermediate vectors, have the same dimensionality $1 \times d$. Thus, the dimensionalities for the transform matrices are $\mathbf{W}^Q \in \mathbb{R}^{d\times d}$, $\mathbf{W}^K \in \mathbb{R}^{d\times d}$, and $\mathbf{W}^V \in \mathbb{R}^{d\times d}$. Given these projections, the similarity score between the current focus of attention $\mathbf{x}_i$ and an element in the preceding context $\mathbf{x}_j$ is a product dot between its query vector $\mathbf{q}_i$ and the preceding tokens' key vectors $\mathbf{k}_j$. However, since a dot product can result in arbitrarily large values, it needs to be escalated by a square root

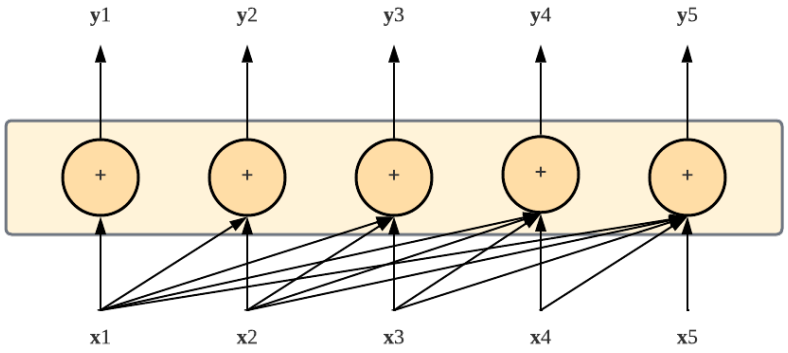

Figure 3.5: Self-attention layer. Example for an input $\mathbf{x}$ of 5 tokens where the $\mathbf{y}_i$ is computed using the $\mathbf{x}_{1:i}$

of the dimensionality of the vectors to avoid numerical issues and get effective loss of gradients. As in the RNNs, this score has to be normalized with a softmax function to create a vector of weights that sum 1:

$$\alpha_{ij} = \text{softmax}\left(\frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d}}\right) = \frac{\exp\left(\frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d}}\right)}{\sum_{k=1}^{i} \exp\left(\frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d}}\right)} \quad \forall j \leq i$$

The final output $\mathbf{y}_i$ is a weighted sum over the value vectors $\mathbf{v}$. Each output is calculated independently, so the entire process can be parallelized using matrix multiplication. Thus, the input embeddings of the sequence of length N can be packed into a matrix $\mathbf{X} \in \mathbb{R}^{N \times d}$, where each row is an input embedding. The above equations can then be re-written as matrices multiplications to produce $\mathbf{Q} \in \mathbb{R}^{N \times d}$, $\mathbf{K} \in \mathbb{R}^{N \times d}$, and $\mathbf{V} \in \mathbb{R}^{N \times d}$ that contain all key, query, and value vectors:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^Q; \qquad \mathbf{K} = \mathbf{X}\mathbf{W}^K; \qquad \mathbf{V} = \mathbf{X}\mathbf{W}^V$$

Given these matrices, the similarity score can be obtained multiplying $\mathbf{Q}$ and $\mathbf{K}^\mathbf{T}$, and after taking the softmax, multiply it by $\mathbf{V}$. The result of the multi-head attention layer is then a matrix of shape $N \times d$, that we can define as the function SelfAttention($\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$):

$$\text{SelfAttention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\mathbf{T}}{\sqrt{d}}\right)\mathbf{V}$$

Nonetheless, the calculation of the matrices $\mathbf{Q}$ and $\mathbf{K}^\mathbf{T}$ computes the values that follow the query, so the upper-triangular portion of the matrix is set to $-\infty$ to avoid any knowledge of the words that follow the sequence. Besides, instead of performing a single attention function with d dimensional keys, values and queries, the model linearly projects the queries, keys and values h times with different learned linear projections d_k, d_k and d_v, respectively. On each of these projections, the attention function is performed in parallel resulting in vectors of dimension d_v. These vectors are then concatenated and projected again, resulting in the final output.

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\mathbf{W}^O$$
$$\text{where head}_i = \text{SelfAttention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V)$$

where the projections are parameters matrices $\mathbf{W}_i^Q \in \mathbb{R}^{d \times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{d \times d_k}$, $\mathbf{W}_i^V \in \mathbb{R}^{d \times d_v}$ and $\mathbf{W}_i^O \in \mathbb{R}^{hd_v \times d}$. Thanks to this multi-head layer, the model can simultaneously compute how each sequence word relates to the others. For example, different syntactic, semantic and discourse relationships between verbs and their arguments in a sentence can be held.

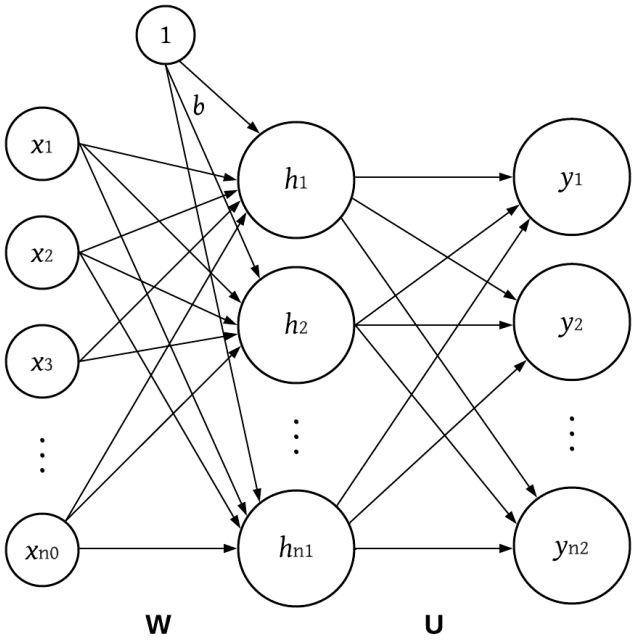

Figure 3.6: A simple feedforward network with an input **x**, a hidden layer **h** and an output layer **y**, given a matrix of weights **W** and a bias vector **b**.

Then, the result of the multi-head self-attention layer is fed into the feedforward layer. A feedforward network is a multi-layer network in which the units are connected without cycles. Thus, the outputs from the units in each layer are passed to the units in the next higher layer, and no result is passed back to the lower layers. Simple feedforward networks have 3 types of nodes: input units, hidden units and output units. The core of neural networks is the hidden layer **h** formed by hidden units h_i, each of which is a neural unit, as introduced in the section 3.1.3. These layers are usually fully-connected, meaning all units take all the outputs from all the units in the previous layer as input. Each hidden unit has a weight vector $\mathbf{w}_i$ and a bias b_i as parameters. Each weight vector and bias can be combined to form a matrix **W** and a vector **b** for each layer in the network. Thus, each element $\mathbf{W}_{ji}$ represents the weight of the connection between the input unit x_i to the hidden unit h_j. Using a weight matrix simplifies the computations and can be done efficiently with matrix operations. The output of the hidden layer **h** is computed by multiplying the weight matrix **W** by the input vector **x**, adding the bias vector **b**, and applying an activation function g to each element in the result, such as sigmoid, tanh or ReLU:

$$\mathbf{h} = g(\mathbf{W}\mathbf{x} + \mathbf{b})$$

Figure 3.6 shows an example of a feedforward network with an input $\mathbf{x} \in \mathbb{R}^{n_0}$, a hidden layer $\mathbf{h} \in \mathbb{R}^{n_1}$ and a vector $\mathbf{x} \in \mathbb{R}^{n_1}$. The above equation will compute the value of each h_i using the matrix $\mathbf{W} \in \mathbb{R}^{n_1 \times n_0}$. Then, the goal of the output layer is to compute a final vector $\mathbf{y} \in \mathbb{R}^{n_2}$ taking $\mathbf{h}$. Like the hidden layer, the output layer has a weight matrix $\mathbf{U} \in \mathbb{R}^{n_2 \times n_1}$, which is multiplied by $\mathbf{h}$ to produce the intermediate output $\mathbf{z} \in \mathbb{R}^{n_2}$. Similar to the matrix $\mathbf{W}$, each element $\mathbf{U}_{ij}$ is the weight value from the unit j to the output unit i. In some models, the bias is not added in this layer, so we will eliminate it to simplify. Finally, each value in $\mathbf{z}$ is normalized using a softmax function to compute final output $\mathbf{y}$:

$$\mathbf{z} = \mathbf{U}\mathbf{h}$$
$$\mathbf{y}_i = \text{softmax}(\mathbf{z}_i) = \frac{\exp(\mathbf{z}_i)}{\sum_{j=1}^{d} \exp(\mathbf{z}_j)} \quad 1 \leq i \leq n_2$$

This feedforward network is called a 2-layer network. However, depending on the task, these networks can have as many layers as desired. Feedforward networks with multiple hidden layers are called deep neural networks, which give birth to a new machine learning field, *deep learning*. In the case of the transformer architecture, the output of the self-attention layer $\mathbf{Z}$ is fed into this feedforward network to produce the output $\mathbf{Y}$. The network is applied to each position separately and identically, consisting in 2 linear transformations with a ReLU activation in between:

$$\text{FFN}(\mathbf{z}) = \max(0, \mathbf{z}\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2$$

These layers are inside a transformer block, which can be stacked to match the input and output dimensions. As shown in figure 3.4, a residual connection and a normalization layer follow the feedforward network and the self-attention layers. In deep networks, the residual connections pass information to a higher layer without going through an intermediate layer. This residual connection improves learning by allowing information from the activation to go forward, and the gradient to go backwards. In transformers, these connections are implemented by adding the layer input to its output before passing it on. Then these summed vectors are normalized using layer normalization. Simplifying, we can express the computation of each resulting vector $\mathbf{Y}$ of the transformer block as:

$$\mathbf{z} = \text{LayerNorm}(\mathbf{x} + \text{SelfAttention}(\mathbf{x}))$$
$$\mathbf{y} = \text{LayerNorm}(\mathbf{z} + \text{FFN}(\mathbf{z}))$$

Layer normalization is a variation of the standard scores of similarity seen above. The first step is to calculate the mean μ and the standard deviation σ over the elements over the elements of the vector $\mathbf{x}$. Given the dimensionality of the hidden layer d, these values are computed as:

$$\mu = \frac{1}{d}\sum_{i=1}^{d} x_i$$
$$\sigma = \sqrt{\frac{1}{d}\sum_{i=1}^{d}(x_i - \mu)^2}$$

Given these values, the vector components are normalized by subtracting the mean from each and dividing by the standard deviation. This computation results in a new vector whose mean is 0 and whose standard deviation is 1. Then, 2 learnable parameters are introduced, α and β, representing gain and offset values.

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - \mu}{\sigma}$$
$$\text{LayerNorm}(\mathbf{x}) = \alpha\hat{\mathbf{x}} + \beta$$

The output of the encoder blocks is represented as $\mathbf{H}^{enc}$, which is the context for the decoder unit representing the input. As well as the encoder, the decoder has a multi-head attention layer, followed by a normalization layer, a feedforward network, and another normalization layer. However, decoder blocks have an extra layer called the cross-attention layer. This layer is very similar to the multi-head self-attention layer, except the keys and values come from the output of the encoder. Thus, the encoder output is multiplied by the cross-attention layer's key weights $\mathbf{W}^K$ and $\mathbf{W}^V$ value weights. The query weights $\mathbf{W}^Q$, on the other hand, are multiplied by the output from the prior decoder layer $\mathbf{H}^{dec[i-1]}$.

$$\mathbf{Q} = \mathbf{W}^Q\mathbf{H}^{dec[i-1]}; \qquad \mathbf{K} = \mathbf{W}^K\mathbf{H}^{enc}; \qquad \mathbf{V} = \mathbf{W}^V\mathbf{H}^{enc}$$

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

As encoder-decoder models with RNNs, encoder-decoder transformers are trained with parallel corpus with sentences from the source and target languages. These sentences are fed into the model as one input. Thus, the source sentence is concatenated with the target sentence with a separator between them, and a final unique token determines the end of the sequence. Then, a *self-supervision* algorithm is used to predict the next word at each time step t and model weights are adjusted using *cross-entropy* as a loss function. So at time t, the cross-entropy loss L_{CE} is the negative logarithmic probability the model assigns to the next token w_{t+1} in the training sequence.

$$L_{CE}(\hat{\mathbf{y}}_t, \mathbf{y}_t) = -\log \hat{\mathbf{y}}_t[w_{t+1}]$$

where $\hat{\mathbf{y}}_t$ is the predicted token and $\mathbf{y}_t$ is the correct token that comes from the training data. Thus, at each time step t, the model inputs the correct sequence of tokens $w_{1:t}$, using *teacher forcing*.

3.5 Pre-trained Transformer models

The Transformer architecture significantly advanced the capabilities of LLMs. The self-attention mechanism allows the model to attend to different positions in the input sequence, capturing dependencies more effectively. The Transformer models demonstrated superior performance in various NLP tasks, including language modeling, MT, and text generation.

This outcome led to the birth of new transformer-based LLMs, eventually bringing us to today's revolution. Transformer-based models are trained in vast amounts of data and require much power to carry on the necessary training process. Big tech companies with

access to extensive computer clusters and data centers with substantial computational resources are the only ones capable of doing this process. It is vital to be aware of this process's environmental impact due to the significant energy consumed that can result in a substantial carbon footprint [32].

This booming technology continually evolves, and new models are presented almost daily. We will review some of the most popular and crucial models below. Yet, we should mention that there are a lot more with similar performance.

3.5.1. Generative Pre-trained Transformer (GPT)

Generative Pre-trained Transformer (GPT) represents a groundbreaking milestone in the development of large language models. It was the first pre-trained Transformer model that achieved significant success in fine-tuning for various NLP tasks, leading to state-of-the-art results. GPT, introduced by OpenAI in 2018 [33], leverages the transformer architecture and a massive amount of unsupervised text data for pre-training. Its training process consists of 2 stages. The first stage is an unsupervised pre-training on a large corpus of text. Given the corpus of tokens $\mathcal{X} = \{x_1, \ldots, x_n\}$, the objective is to maximize the following likelihood:

$$L_1(\mathcal{X}) = \sum_i \log P(x_i | x_{i-k}, \ldots, x_{i-1}; \mathbf{\Theta})$$

where k is the size of the context window, and the conditional probability P is modeled using a neural network with parameters $\mathbf{\Theta}$. These parameters are trained using stochastic gradient descent.

The GPT model uses a multi-layer Transformer decoder for the language model, applying a multi-headed self-attention operation over the input context tokens followed by position-wise feedforward layers to produce an output distribution over target tokens:

$$\begin{aligned} \mathbf{h}_0 &= \mathbf{U}\mathbf{W}_e + \mathbf{W}_p \\ \mathbf{h}_l &= \text{transformerBlock}(\mathbf{h}_{l-1}) \forall i \in [1, n] \\ P(x) &= \text{softmax}(\mathbf{h}_n \mathbf{W}_e^T) \end{aligned}$$

where $X = (x_{-k}, \ldots, x_{-1})$ is the context vector of tokens, n is the number of layers, $\mathbf{W}_e$ is the token embedding matrix, and $\mathbf{W}_p$ is the position embedding matrix. After the pre-training, the parameters are adapted to the supervised target task, which can be classification, entailment, similarity or multiple choice. Assuming a labeled dataset C, where each instance consists of a sequence of input tokens, $x^1, \ldots, x^m$, along with a label y, the inputs are passed through the pre-trained model to obtain the final transformer block's activation $\mathbf{h}_l^m$, which is then fed into an added linear output layer with parameters $\mathbf{W}_y$ to predict y:

$$P(y | x^1, \ldots, x^m) = \text{softmax}(\mathbf{h}_l^m \mathbf{W}_y)$$

This gives the objective L_2 to maximize, which is then combined with the previous one L_1 to make a weighted objective L_3, with weight λ:

$$\begin{aligned} L_2(\mathcal{C}) &= \sum_{(x,y)} \log P(y | x^1, \ldots, x^m) \\ L_3(\mathcal{C}) &= L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C}) \end{aligned}$$

For some tasks, like text classification, the model can be fine-tuned as described above. Other tasks, like question answering or textual entailment, have structured inputs such as ordered sentence pairs or triplets of documents, questions, and answers.

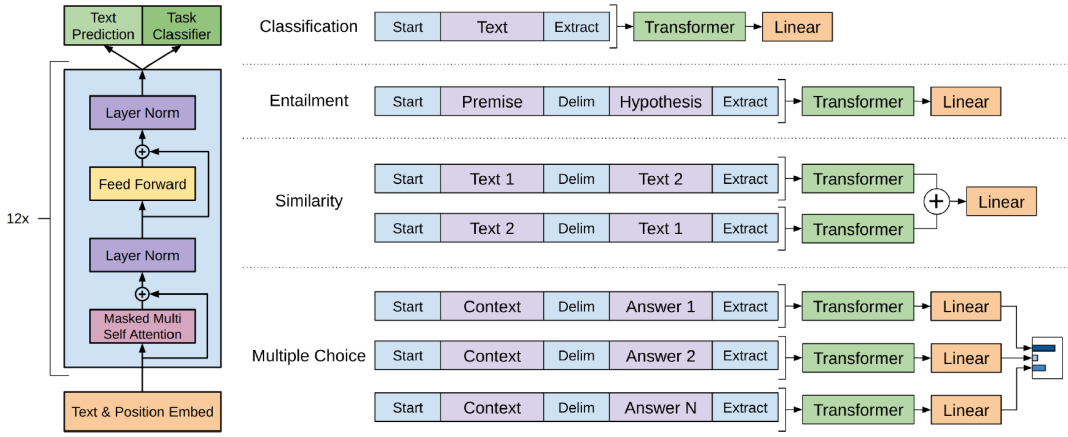

Figure 3.7: On the left is the Transformer architecture used and the training objectives. On the right are the input transformations for fine-tuning on different tasks. All inputs are converted to sequences of tokens processed by the pre-trained model, followed by a linear+softmax layer. Figure taken from *Improving language understanding by generative pre-training* [33].

Since the pre-trained model was trained on contiguous sequences of text, applying it to these tasks requires some modifications. These changes imply converting structured inputs into an ordered sequence that the pre-trained model can process. For example, the premise p and hypothesis h token sequences are concatenated for entailment tasks, with a delimiter token $\langle\$\rangle$ in between, a start token $\langle s\rangle$, and an ending token $\langle e\rangle$. These input transformations prevent extensive architecture changes across tasks but require independent processing, as shown in Figure 3.7.

The GPT model showcased its capacity for coherent text generation, text completion, and limited language understanding. While it could complete sentences, suggest word predictions, and even provide basic translations and summaries, its generated content sometimes lacked accuracy and coherence. Despite its capabilities, GPT exhibited limitations such as generating implausible responses, struggling with unfamiliar topics, and not consistently providing accurate answers to questions.

One year later, in 2019, *GPT-2* was presented [34], an evolved version of GPT-1 that brought several technical improvements and differences compared to its predecessor. GPT-2 is significantly larger, with models ranging from 117 M to 1,5 B parameters. The larger model sizes contribute to improved performance but also require more computational resources. It was trained on a more extensive and diverse dataset, encompassing a broader range of topics and writing styles, which increased scale and contributed to its improved language understanding and generation capabilities. GPT-2 showed some ability for few-shot and even zero-shot learning, performing tasks with just a few or even zero examples.

In 2020, OpenAI presented *GPT-3* [35], a massive model with 175 B parameters, making it substantially larger than any version of GPT-2. It demonstrated exceptional capabilities in language generation, understanding, and a wide range of tasks. GPT-3 could generate creative stories, answer questions accurately, provide translations, write code in various languages, and much more with minimal task-specific fine-tuning. OpenAI took advantage of this performance and launched a variant of the GPT-3.5 model, an improved version of GPT-3, fine-tuned specifically for conversational interactions, using labeled data for further training. The AI trainers play as users and AI assistants to build the answers based on prompts. Then, the answers with prompts are used as supervised data for further training of the pre-trained model. This model, called *chatGPT*, could generate human-like text responses in a chatbot or dialogue format. For this reason, OpenAI

made the power of this LLM available for everyone in a user-friendly chatbot interface. Its excellent performance in simulating a conversation with a human and its coherent responses led it to an unprecedented popularity in the field of AI, reaching ordinary people.

The research on GPT continued until, in 2023, *GPT-4* was presented [36]. It is a large multimodal model, that accepts image and text inputs to emit text outputs. It exhibits human-level performance on various professional and academic benchmarks to the point that it beats 90% of people who take the exam to become lawyers [37]. In fact, it scores in the top ranks for at least 34 different tests of ability in fields as diverse as macroeconomics, writing, math, and much more. This model shows astonishing performance and hints at a bright future for LLMs.

3.5.2. Bidirectional Encoder Representations from Transformers (BERT)

In 2018, Google AI presented the *Bidirectional Encoder Representations from Transformers* (BERT) [38], whose architecture is a multi-layer bidirectional Transformer encoder based on the original implementation described in section 3.4. BERT's Transformer uses bidirectional self-attention, while the GPT Transformer uses constrained self-attention where every token can only attend to the context to its left. It was released with different sizes, in which the number of layers and self-attention heads differed. The base model was as big as GPT-1 for comparison purposes. BERT uses WordPiece tokenization, which splits words into subwords, allowing the model to handle out-of-vocabulary words and improve generalization. The first token of every sequence is always `[CLS]`, and `[SEP]` is used to separate the target sentence and as the ending token. The final input representation E for a given token is the sum of the corresponding token embedding, the position embedding and the segment embedding, indicating whether the token belongs to sentence `A` or `B`.

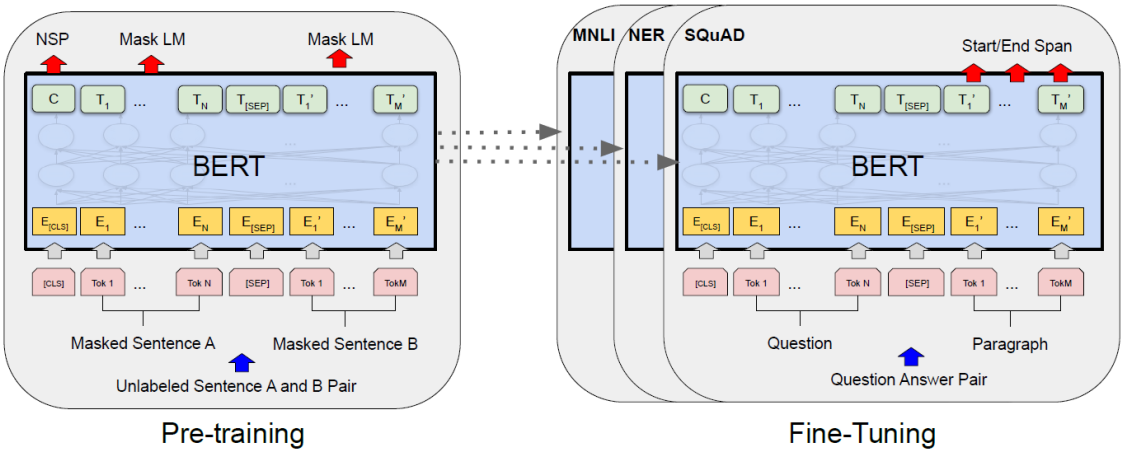

Figure 3.8: BERT pre-training and fine-tuning methods. The same architectures are used in pre-training and fine-tuning, except in the output layers. The same pre-trained parameters are used to initialize the models for different tasks. Figure taken from *BERT's research paper* [38].

As GPT, BERT is pre-trained on a large corpus of text data, learning to predict missing words in sentences. It does this by masking out a certain percentage of the words in each sentence with the token `[MASK]` and learning to predict them based on the context provided by the surrounding tokens on the left and the right. Additionally, BERT learns to predict whether 2 sentences in a pair follow each other in the original text or not. Specifically, when choosing the sentences `A` and `B` for each pre-training example, 50% of the time `B` is the actual next sentence that follows `A`, labeled as `IsNext`, and 50% of the

time it is a random sentence from the corpus, labeled as `NotNext`. This process helps the model learn contextual relationships between words and sentences.

After pre-training, BERT can be fine-tuned on specific NLP tasks such as text classification, named entity recognition, question-answering, and more. The specific inputs and outputs are fed into BERT for each task, and all the parameters end-to-end are fine-tuned, as seen in Figure 3.8. At the input, sentence `A` and sentence `B` from pre-training are analogous to:

1. **Paraphrase:** Sentence pairs.
2. **Entailment:** Hypothesis-premise pairs.
3. **Question answering:** Question-passage pairs.
4. **Text classification:** A degenerate text-∅ pair.

At the output, the token representations are fed into an output layer for token-level tasks, such as sequence tagging or question answering, and the `[CLS]` representation is fed into an output layer for classification, such as entailment or sentiment analysis. This fine-tuning enables BERT to adapt its knowledge to the nuances of the task. Compared to pre-training, fine-tuning is relatively inexpensive.

BERT's key innovation lies in its ability to capture complex contextual relationships in language, and it achieved state-of-the-art performance on a wide range of tasks without requiring extensive task-specific architecture modifications. This performance led to the research of new models based on BERT. Researchers extended BERT's architecture to develop models that perform well in specific languages, such as *BETO* for Spanish, developed by the Barcelona Supercomputing Center [39], *BERTje* for Dutch, *BERTurk* for Turkish, and more. *mBERT*, multilingual version of the model trained in 104 languages, was also released in the same year.

Facebook AI presented a *Robustly Optimized BERT Pre-training Approach* (RoBERTa) [40] in 2019, a model that builds upon BERT's architecture and training methodology. It optimized various hyperparameters and training techniques, improving performance on various NLP tasks. RoBERTa drops the *Next Sentence Prediction* objective and focuses solely on the *Mask Language Model* task, masking a larger percentage of words in each sentence and sampling different masked words for each training epoch. RoBERTa demonstrated that careful tuning of training strategies can lead to substantial performance gains.

Facebook AI also released *Bidirectional and Auto-Regressive Transformers* (BART) [41] in 2019, a BERT-based model combining bidirectional and auto-regressive training objectives. Like BERT, BART is trained to predict masked words in sentences, and also learns to reconstruct corrupted sentences. The denoising autoencoder reconstruction objective involves corrupting the input sentences and training the model to reconstruct the original sentences. This autoencoder-style training contributes to BART's auto-regressive generation capability. This combination makes BART suitable for tasks involving language understanding and text generation, such as summarization, translation, and more.

3.5.3. Text-To-Text Transfer Transformer (T5)

Google AI presented a model architecture called *Text-to-Text Transfer Transformer* (T5) in 2020 [42]. It is a variant of the Transformer architecture designed to be trained in all NLP tasks with a unified *text-to-text* format, where inputs and outputs are treated as text sequences. For example, if we want to translate *I love you.* from English to German, the model would be fed the sequence "*translate English to German: I love you.*" and it would

be trained to output "*Ich liebe dich.*" This approach allows T5 to handle a wide range of tasks, including classification, translation, summarization, and more, using a consistent framework.

T5 encoder-decoder implementation is roughly equivalent to the original Transformer architecture described in Section 3.4, except for removing the Layer Norm bias, placing the layer normalization outside the residual path, and using a different position embedding scheme. While the original Transformer used a sinusoidal position signal or learned position embeddings, T5 uses relative position embeddings. These simplified position embeddings are simply a scalar, representing the offset between the *key* and *query*, added to the corresponding logit for computing the attention weights. For efficiency, the position embedding parameters are shared across all layers in the model, though each attention head uses a different learned position embedding within a given layer.

The encoder and decoder consist of 12 blocks, each comprising self-attention, optional encoder-decoder attention, and a feed-forward network. The feed-forward networks in each block consist of a dense layer with an output dimensionality of $d_{ff} = 3\,072$, followed by a ReLU nonlinearity and another dense layer. The *key* and *query* matrices of all attention mechanisms have an inner dimensionality of $d_{kv} = 64$ and all attention mechanisms have 12 heads. All other sub-layers and embeddings have a dimensionality of $d_{model} = 768$. This results in a model with about 220 million parameters, almost twice the number of parameters of $\text{BERT}_{\text{BASE}}$.

T5 is pre-trained in a large unlabeled corpus made by the research team called the *Colossal Clean Crawled Corpus* (C4), a cleaned version of the Common Crawl dataset, a publicly-available web archive that provides "web extracted text" by non-text content from the scraped HTML files. The team cleaned this dataset following some heuristics to obtain 750 GB of natural language in English. A model for each task was pre-trained for $2^{19} = 524\,288$ steps on C4 before fine-tuning. An inverse square root learning rate schedule was used during pre-training:

$$\frac{1}{\sqrt{\max(n,k)}}$$

where n is the current training iteration and k is the number of warm-up steps, set to 10^4 in all the experiments. This formula sets a constant learning rate of $0,01$ for the first 104 steps, then exponentially decays the learning rate until pre-training is over. Then, models were fine-tuned for $2^{18} = 262\,144$ steps on all tasks.

T5's consistently strong performance across a broad spectrum of tasks established its reputation as a competent and versatile model in the field of NLP. Because of this, in 2021, Google Research launched *mT5* [43], a multilingual version of the T5 model. This model was pre-trained in similar conditions to T5 with a multilingual dataset called *mC4*, containing more than 100 different languages. Researchers demonstrated that the T5 recipe can be applied to the multilingual setting and achieved strong performance on diverse benchmarks.

3.5.4. BLOOM

In 2022, the *BLOOM* model was released [44], a multilingual Transformer model based on GPT-2 and developed collaboratively by over 1 000 researchers from more than 70 countries and more than 250 institutions. It is designed to address the issue of limited accessibility to powerful LLMs, which are typically only available to large industrial labs. The model utilizes a decoder-only architecture, which generates text in 46 natural languages and 13 programming languages based on previous tokens without direct access

to future tokens. BLOOM has 176 B parameters and comprises 70 layers with 112 attention heads each. This model represents the most extensive collaboration of AI researchers on a single project. It was trained transparently over a year, with the final training run lasting 117 days on the Jean Zay supercomputer in Paris, France. Competitive results can be achieved with zero-shot or few-shot learning, which can be improved after multitasking fine-tuning.

3.6 Data augmentation

Data augmentation (DA) is the process of artificially increasing datasets by creating new samples using the existing data. We can distinguish 2 results depending on the technique. The first approach involves minor modifications to the items in the dataset, resulting in augmented data. Nonetheless, recent methods rely on deep learning to create new synthetic data without the original database. DA can be used for many data types, such as images, audio, video, or text. This section will explain the main approaches of NLP data augmentation techniques.

3.6.1. Easy Data Augmentation

Easy Data Augmentation (EDA) was one of the initial text augmenting techniques proposed in 2019 [45]. It randomly selected 4 simple operations: synonym replacement, random insertion, random deletion, and random swap. These operations aimed to create new examples by perturbing the original text while preserving its meaning to some extent.

- **Synonym Replacement:** Randomly select n words from the text that are rarely used. Subsequently, replace them by one of its synonyms randomly choosen. For example, given the first sentence, replacing 2 random words could result in the second sentence:

  *This **project** will focus on explaining the best how to **augment** a dataset.*

  *This **work** will focus on explaining the best how to **extend** a dataset.*

- **Random Insertion:** Randomly select n words from the text that are rarely used. Subsequently, add one of its synonyms at a random sentence position. For example, given the first sentence, inserting one word at a random position could result in the second sentence:

  *The kids were **playing** in the park.*

  *The kids were **enjoying** playing in the park.*

- **Random Swap:** Randomly select 2 words from the text that are rarely used and swap their positions. For example, given the first sentence, swapping 2 random could result in the second sentence:

  *The **quick** brown fox jumps over the **lazy** dog.*

  *The **lazy** brown fox jumps over the **quick** dog.*

- **Random Deletion:** Each word in the sentence is randomly removed with probability p. For example, given the first sentence, randomly removing each word could result in the second sentence:

  *The dos was **sitting** on the couch.*

  The dog was on the couch.

Since sentences can vary in length, changing too many words in short sentences can lead to modifying the meaning or losing coherence. To compensate the length difference, the number of words changed n is based on the sentence length with the formula $n = \alpha l$, where α is the percentage of words changed in the sentence, and l is the sentence's length. In the case of random deletion, p is equal to α. We should also mention that words were selected in the above examples to produce coherent sentences. However, in a normal process, words will be chosen randomly, and the resulting sentences may have incoherent meanings or nonsense. For this reason, EDA is a good technique for text classification but not for other NLP tasks like MT, where sentence quality is essential for the model performance.

3.6.2. Back-translation

Back-translation is the most common technique to augment datasets in MT. Parallel corpora can be challenging to find, and in the case of finding some bitext, it may be limited for some languages or domains. However, we often find a larger monolingual corpus that we can use and add to the parallel corpus.

Back-translation aims to improve MT models, given a small or limited parallel corpus and some monolingual data in the target language [1]. The first step is training an MT system in the reverse direction, from the source to the target language. Then, this model is used to translate the monolingual data in the target language to the source language. The resulting parallel corpus is finally added to the existing parallel corpus to train our objective MT model.

Many works and experiments have been done using this technique. For instance, in the ETSINF, this technique has been used to improve MT systems in a DFP, called *Using back-translation for machine translation based on transformer*, proposed by Vladyslav Mazurkevych [46]. This work aimed to enhance parallel corpus in 4 language pairs: English to French, English to German, German to English, and French to English. Then, he compared MT systems trained on the original parallel datasets with MT systems trained on the synthetic data added to the original. Despite being a limited experiment with low resources and relatively little training data, the results showed that the MT systems improved the performance.

Thus, in general, back-translation works surprisingly well. Some estimations say that an MT system trained on back-translated data can achieve about 2/3 of the gain as would training on the same amount of natural parallel data.

3.6.3. Masked Language Models

Contextual augmentation is another word-level data augmentation method that uses *Masked Language Models* (MLMs) such as BERT and RoBERTA to generate new text based on the context [47, 48]. It consists of inserting `<mask>` tokens in some text positions or replacing some words with `<mask>` tokens and then letting the MLM predict what words, different from the original, could be put in these masked positions. Since MLMs are pretrained on many texts, contextual augmentation can usually generate meaningful new texts.

This approach has some meaningful parameters, like selecting the percentage of words that will be masked to be changed. If the percentage is too low, the sentences will be almost identical, and the change will be minimal. On the other hand, if the percentage is too large, there is a danger that by changing too many words, the meaning or sentiment of the sentence will be lost. That behavior is not desired in text classification datasets or

sentiment analysis datasets. The most common range is from 20 % to 50 % replacement rate.

Another critical parameter is what words are selected for masking. The easiest way is to select them randomly, which can make undesired changes that lead to incoherent sentences. A more sophisticated approach is to select the words depending on their *part of speech* (POS) [47]. This can be done with some NLP packages, like *spaCy* [1], which uses trained models on a specific language, or multilingual, that can predict the POS tagging of each word. For instance, replacing nouns, adjectives, and adverbs carries less danger of losing coherence and meaning. Yet, it can make the sentence lose the sentiment and transform it into the opposite. Let's see an example:

*The movie I saw in the cinema looked pretty **interesting** to me.*

The movie I saw in the cinema looked pretty `<mask>` to me.

*The movie I saw in the cinema looked pretty **boring** to me.*

Moreover, these models can sometimes tag words incorrectly since the tagging is made individually without considering the context. For instance, the word *book* can be either a noun or a verb:

*This **book** is interesting.*

*I will **book** a table for two.*

Using MLMs can result in good data augmentation depending on the purpose of the dataset and the above parameters. In some tasks like sentiment analysis, it is common to lose the sentiment and break the dataset. In other tasks, like text classification or MT, suitable data can be obtained to improve the model's performance.

3.6.4. Large Language Models for Paraphrasing

Researchers found a new way to augment text using the state-of-the-art performance of *Large Language Models* (LLMs) fine-tuned for conversations [49]. We can highlight the chatGPT model or open-spurce models like Dolly-v2 [2], and StableVicuna [3]. The methodology is as simple as asking them to paraphrase the sentences to augment. Their language knowledge allows them to substitute not just individual words but an entire phrase or expression. These substitutions lead to a successful data augmentation since the sentence meaning and sentiment are almost identical or identical. As seen in Figure 3.9, this approach has been used recently to augment multilingual text datasets for text classification, sentiment analysis, or question answering. LLMs can even be used to augment complex datasets like:

- **XCOPA**, a multilingual Choice of Plausible Alternatives dataset. Each instance consists of a premise, a question, and 2 alternatives and the task is to predict the more plausible alternative.

- **XWinograd**, a multilingual dataset that consists of pronoun resolution problems aiming to evaluate the commonsense reasoning ability of a machine. Given a statement with 2 noun phrases and a pronoun, the challenge of the dataset is determining the referent of the pronoun, which can only be inferred from the context.

[1] Official GitHub page with source code and documentation: https://github.com/explosion/spaCy

[2] Official GitHub page with source code and documentation: https://github.com/databrickslabs/dolly

[3] Official page with documentation: https://lmsys.org/blog/2023-03-30-vicuna/

- **XStoryCloze**, a multilingual dataset in which each example consists of a four-sentence commonsense story, a correct ending, and a wrong ending.

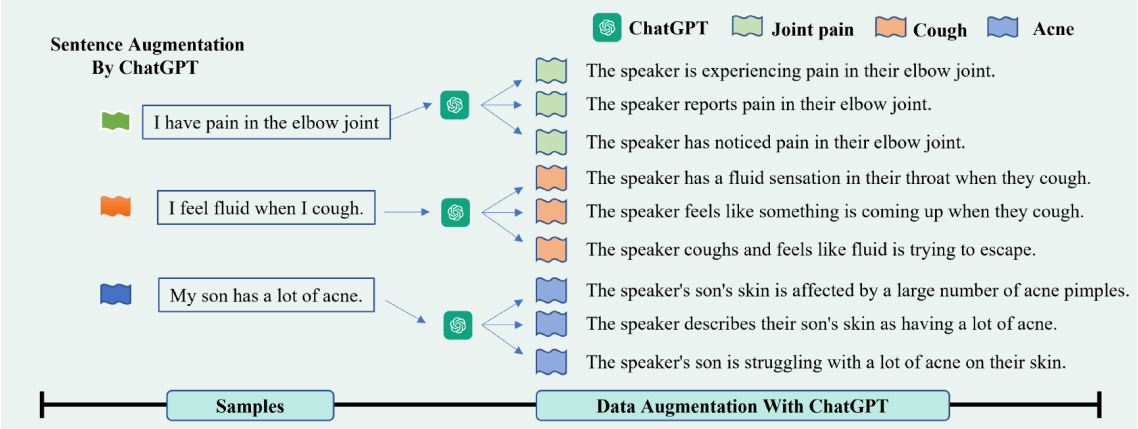

Figure 3.9: Sentence augmentation by chatGPT. An example of text augmentation in a text classification dataset using chatGPT. Figure taken from *AugGPT's research paper* [49].

LLMs can augment commonsense reasoning data and generate new samples with sense and coherence. Currently, this approach has limitations, but it is expected that in the future, the evolution of these models will improve its responses will improve.

Augment text using LLMs involves carefully designing and crafting prompts, called *prompt engineering*. Prompt engineering aims to guide the model's behavior and output to produce the desired results or responses. Sometimes, a few examples are required to show the model what the output should look like. Figure 3.10 shows the 2 options available to prompt chatGPT. The single-turn dialogue includes the instruction in each message, which will unnecessarily augment the number of tokens used. On the other hand, the multi-turn dialogue includes the instruction in a first message with a special role system. Then, each sentence to augment is sent in separate messages as the user and the assistant will answer the number of results requested. This solution avoids repeating the instruction in each message, resulting in less token usage. The number of tokens is essential since it can be translated into the used power or money, since these massive models are expensive.

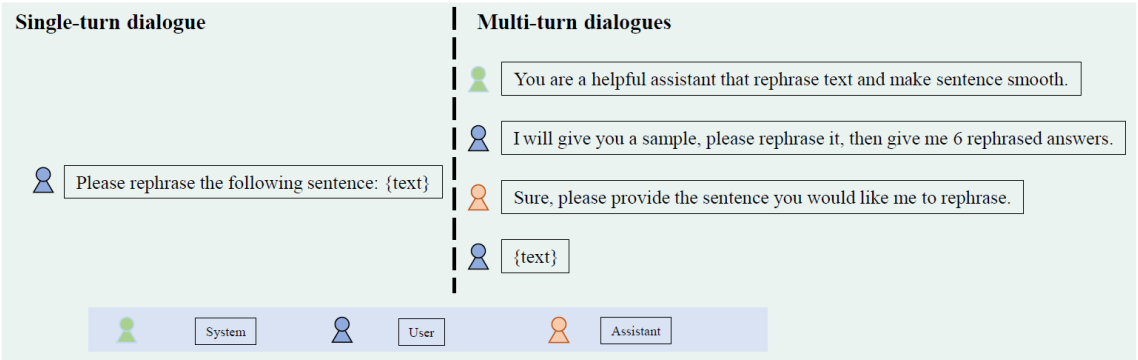

Figure 3.10: Prompt examples for DA in chatGPT. Single-turn dialogue and multi-turn-dialogue for DA in chatGPT. Figure taken from *AugGPT's research paper* [49].

Effective prompt engineering can help ensure that the model produces accurate and relevant responses. However, LLMs show some limitations, like recognizing and augmenting medical texts, since they can produce incorrect augmentation results due to the lack of domain knowledge. Moreover, these models perform less in low-resource lan-

guages since their vocabulary and knowledge are more limited than in high-resource languages. In general, unlike other methods, LLMs can expand the limited data at the semantic level to enhance data consistency and robustness, which results in a better performance than most of the current text data augmentation methods.

CHAPTER 4
Experimental framework

This chapter presents a comprehensive experimental framework to address the challenge of training MT systems in low-resource conditions by proposing an innovative solution for augmenting parallel datasets. The proposed solution unifies advanced techniques from NLP and ML, offering a promising avenue for enriching parallel datasets. The subsequent sections of this chapter delineate the critical components of this experimental framework, comprising the proposed solution, the corpora selected to augment, and the experiments done to check the solution's effectiveness.

While at the company as an intern, I made many experiments to augment text. After many experiments and trying different techniques to augment text data, I wrote a library in *Python* [1] called *DataAug* with different functions to augment corpora. This library is composed of 2 classes:

- **FastAug:** This class is designed to perform fast since sentences are not processed. It breaks text into tokens using a regular expression, and only words can be candidates to be masked. From the candidate tokens, it randomly selects n tokens to mask based on the sentence tokens count l and a replace rate λ. That means it will replace $n = \lfloor l * \lambda \rfloor$ if the masking is successful. Masking is successful if there is a word that is not equal to the original in the top_k predictions of the model. At least one word must have been substituted to save the augmented sentence. Forbidden words can be set not to be masked, and the model cannot return forbidden tokens if desired.

- **DeepAug:** This class is designed to get higher quality results but with a slower performance. It is slower than *FastAug* because each sentence is processed with *spaCy*, an NLP library in Python. This library is responsible for tokenizing the sentences, assigning a part-of-speech tag that indicates the word's grammatical category, and identifying named entities such as persons, organizations, locations, dates, and more within the text. This analysis uses a trained model in the text's language to predict these characteristics. This analysis allows specific word categories to be chosen as candidates for the masking process. For example, the default option only selects nouns, adjectives, and adverbs. As in *FastAug*, a replace rate λ can be set, so $n = \lfloor l * \lambda \rfloor$ words will be substituted, where l is the number of tokens in the sentence. At least one word must have been substituted to save the augmented sentence. Forbidden words can be set not to be masked, and the model cannot return forbidden tokens if desired. It also has the option of augmenting only the vocabulary. That means words can only be replaced with words not in the corpus.

[1] Official page with source code and documentation: https://www.python.org/

FastAug can only augment monolingual data, but *DeepAug* can augment both monolingual and parallel corpora. We will focus our experiments on parallel corpora augmentation, which will be explained below. The source code is not publicly available since it was developed for commercial and research purposes in the company. The experiments have been adapted to the available computational resources since the augmenting of the data and the training processes are done outside the company with personal resources.

4.1 Proposed solution

The proposed solution comprises a method integrated into the DeepAug library to augment parallel texts. To begin, let us review the parameters available in this method:

- **model (str):** The model's name in the Hugging Face that will be used for masking the selected words. *Hugging Face* [2] is a platform for sharing, discovering, and fine-tuning AI models. Its Python library, `transformers`, will download and load the model parameters and configuration files. This setting allows one to choose a specific language or domain model.
- **top_k (int):** The number of predictions the model will compute for each word. The computations are slower when this number is large, but each token has more chances to be substituted. On the other hand, if the number is small, the computations are faster, but a word has fewer chances to be changed.
- **data (pandas.DataFrame):** The data to augment is fed as a data frame using `pandas` [3], an open-source data analysis and manipulation tool coded in Python. This data frame comprises 2 columns whose names are its language 2-letter code, which must equal the following 2 parameters. The rows are the sentence pairs.
- **src_lang (str):** The 2-letter code of the source language.
- **tgt_lang (str):** The 2-letter code of the target language.
- **result (str):** The file name where the results are saved.
- **vocabulary (bool, optional):** Either to augment the vocabulary in the target language corpus. This option saves a dictionary with all the words in the target corpus, and only words not included in it will be accepted as replacements. The default value is `False` since this reduces the chances of replacing a word successfully.
- **allow (list, optional):** The list of allowed POS tags, based on the *Universal POS tags* [4]. Only words with these tags can be selected for masking. The default value is `["ADV", "ADJ", "NOUN"]`, which stands for adjectives, adverbs, and nouns.
- **replace_rate (float, optional):** The replace rate that defines the number of words to select for masking. The default value is 0, 3.
- **forbidden_words (list, optional):** The list of forbidden words that cannot be selected for masking or returned by the model. The default value is an empty list.
- **min_sentence_length (int, optional):** The minimum sentence word length to be selected for augmenting. The default value is 5.

[2] Official page: https://huggingface.co/
[3] Official page with source code and documentation: https://pandas.pydata.org/
[4] Official page with all POS tags: https://universaldependencies.org/u/pos/

- **max_sentence_length (int, optional):** The maximum length of a sentence to augment. The default value is 80 since sentences that are too long can last many seconds to be augmented.
- **verbose (bool, optional):** Choose to print in which step or sentence pair is the script. The default value is `False`.
- **cuda (bool, optional):** Choose if CUDA should be used and perform the model computations on GPU. The default value is `True`.

The first step of the process is downloading and loading the selected model and its configuration files, which is done automatically by the library `transformers`. This process is done with the `pipeline` method, whose input is the masked sentence in plain text, and the output is the decoded *top_k* tokens predicted by the model. This method compresses several steps into one, making the process more accessible.

Then, the correspondent spaCy trained model for each language is loaded. After loading, sentence pairs are saved into 2 spaCy *Corpus*, one per language. The trained model processes sentences before being saved into the corpus. After processing, sentences are tokenized, and words are POS-tagged. This process can be computed in parallel using multi-threading to speed up. In this step, if the *vocabulary* parameter is set to `True`, all unique words in the target sentences will be saved in a dictionary.

Once all sentences are processed and saved, the augmenting loop starts. First, it discards all the sentence pairs with fewer words than *min_sentence_length* or more words than *max_sentence_length*. Then, it is time for a crucial part of the process: the word alignment between sentences. The alignment is powered by the `awesome-align` library, which can extract word alignments from multilingual BERT [51]. This produces a mapping in *i-j* pairs. A pair *i-j* represents that the *i*th word of the source sentence is aligned with the *j*th word of the target sentence. However, this mapping needs an adaptation to the actual word positions since BERT has subword tokens. After many tests, it has been proven that the mapping works well in most high-resource languages. In some cases, some tokens do not have any pair, and in some cases, a token is related to more than one token in the other sentence.

Then, the candidate tokens for masking are selected in the target sentence. The requirements are that it has a pair in the source language, it has an allowed POS tag, it is a word, and it is not in the *forbidden_words* list. With the *replace_rate* as λ, if the number of candidate words is lower than $n = \lfloor l * \lambda \rfloor$, it will try substituting all the candidate words. If the number of candidate words is larger than n, the method will try substituting n random words of the candidate ones. When a word has been selected, it is substituted by the token `<mask>`, and the sentence is fed into the model. Then, it loops over the predictions to find a word different from the original, not in the forbidden words, and with the same POS tag as the original word. If the *vocabulary* parameter is set to `True`, the word cannot be in the vocabulary. If the prediction meets all these conditions, it tries to find a suitable synonym for the source sentence using the above mapping.

The query for a synonym is made to the *Panlex API* [5], a large-scale linguistic database and resource that aims to connect and catalog the world's languages. One call can return many synonyms of the requested word in the source language, but only the first that equals the POS tag of the requested word is accepted. This requirement reduces the loss of coherence in the sentence. If the word is not found, it returns to the last sentence and selects a new word in the target sentence to mask. If the word is found and replaced successfully in the source sentence, the new pair is saved with the changes and returns

[5] Official page with API documentation: https://api.panlex.org/v2/

to the loop of selecting words. This loop ends when n correctly substituted words are reached, or no more candidate words are left. At least one word has to be changed to save the sentence pair as a new synthetic pair.

4.2 Corpora

The goal of this project is to improve datasets in low-resource cases. These cases can be either because it is a low-resource language or a specific domain with limited parallel data. The case chosen for this project is a low-resource domain like the medical. The dataset is called *EMEA* [6] and it is a parallel corpus of PDF documents from the *European Medicines Agency* in 22 languages, comprising 23 M sentence fragments or 311,65 M tokens. It was downloaded from *OPUS* [52], a free language resource of parallel corpora.

For this project, we have selected 3 familiar languages as the target language and English as the source language, resulting in the pairs: English-Spanish, English-French, and English-Greek. The original bitexts have almost 1,1 M sentence pairs, but many are too short or mainly composed of numbers. To obtain quality data, all sentence pairs with less than five words in any language are eliminated. This filter results in around 600 K pairs in each language pair. Detailed statistics of the sentence count can be seen in Table 4.1.

Language pair	Original dataset	Filtered dataset
English-Spanish	1 098 333	618 308
English-French	1 092 568	617 364
English-Greek	1 073 225	584 160

Table 4.1: EMEA dataset statistics. Number of sentence pairs before and after the filtering.

During the dataset construction, files were automatically converted from PDF to plain text using `pdftotext`, a method from the PDF rendering library *Poppler* [7], with the command line arguments `-layout -nopgbrk -eol unix`. There are some known problems with tables and multi-column layouts, but in general, the quality of the alignment is suitable.

4.3 Experiments

The purpose of the experiments is to check if the addition of the synthetic data to the original data can impact the model's performance. The experiment's selected model is the *mT5* model [43], explained in section 3.5.3. This model was selected because it was pre-trained with several languages, including those selected for this project. Its pre-trained knowledge from large-scale datasets allows it to capture a wide range of language patterns and nuances, which is crucial for accurate translation. Besides, its *text-to-text* format makes it easy to handle the translation task by taking source text as input and generating target text as output. mT5 has consistently demonstrated state-of-the-art performance across various NLP tasks, including MT. Since mT5 benefits from transfer learning, it can be fine-tuned on specific translation datasets, making it more effective in generating high-quality translations. Thus, a model for each language pair is fine-tuned with the original dataset, and the same model is fine-tuned with the original dataset plus the synthetic dataset.

[6] Data files for all language pairs are available at https://opus.nlpl.eu/EMEA.php
[7] Source code available at https://poppler.freedesktop.org/

The filtered dataset is split into 3 sets for the experiments: 70 % is used for training the models, 20 % is used for the evaluation during the training to tune hyperparameters and assess the model's performance, and 10 % is reserved for testing purposes. The count of sentence pairs obtained for each language pair is shown in Table 4.2.

	Filtered dataset	**Training set**	**Evaluation set**	**Test set**
Percentage	100 %	70 %	20 %	10 %
English-Spanish	618 308	432 815	123 662	61 831
English-French	617 364	432 154	123 473	61 737
English-Greek	584 160	408 912	116 832	58 416

Table 4.2: Dataset splits. Number of sentence pairs for training, evaluation and test subsets.

Then, the training set is fed to the augmenting pipeline to obtain synthetic data from the original sentence pairs. A replace rate of 25 % is used, and only nouns, adjectives, and adverbs are the allowed word categories to be masked. The model chosen to calculate the predictions for the masked words is *XLM-RoBERTa* [53], a multilingual version of RoBERTa trained in 100 different languages. Its advantages include determining the correct language from the input text and having the same implementation as RoBERTa. The number of predictions computed by the model is set to 5. These parameters generate each new sentence pair in around 5 seconds. This time limits the number of synthetic data generated with the available resources. Synthetic data has been generated using one computer with an *Intel i9-11900H*, 16 GB of RAM, and an *NVIDIA GeForce RTX 3060 Laptop GPU* with 6 GB dedicated memory. The model's predictions are computed with the GPU using CUDA 12 [8]. Even though it is suitable hardware, the data generation is slow and consumes many resources. After days of computations, the generated data equals almost a quarter of the training dataset. That means the original data has increased by 25 %. The detailed statistics for each language pair are in Table 4.3.

Language pair	**Training set**	**Synthetic set**	**Training set + Synthetic set**	**Increase**
English-Spanish	432 815	104 308	537 123	Δ 24, 1 %
English-French	432 154	105 014	537 168	Δ 24, 3 %
English-Greek	408 912	101 410	510 322	Δ 24, 8 %

Table 4.3: Synthetic data statistics. Number of sentence pairs generated with the proposed solution and the increase of their respective training set.

As mentioned above, the fine-tuning process is done with limited personal resources. This process cannot be done with the same hardware as data generation since it requires more computational power and memory. Thus, the fine-tuning is done in *Google Colaboratory* [9], a Google's cloud-based platform that allows users to run and collaborate on Python code, particularly for data analysis and machine learning tasks. It also provides access to free GPU resources, which makes it perfect for our project. The used environment has 13 GB of RAM and an *NVIDIA Tesla T4* with 15 GB of dedicated memory. This hardware allows us to fine-tune the smallest version of mT5, which has around 300 M parameters. So, 6 models will be fine-tuned: 3 with the original datasets and 3 with the original datasets plus the synthetic dataset to check if the new data impacts the model performance.

[8] Official page with software and documentation: https://developer.nvidia.com/cuda-toolkit
[9] Available at: https://colab.research.google.com/

T5 is usually trained using a prefix that explains the desired task. For instance, for translating, the source text would be: *translate English to Spanish: It is done*. However, mT5 was only pre-trained on mC4, excluding any supervised training. Since mT5 was pre-trained unsupervised, there is no actual advantage in using a task prefix for fine-tuning in a single task. The model learns that the task is translating from source to target language. The T5 tokenizer is used to tokenize the sentences since it has the same architecture as T5. All models are trained in one epoch with a learning rate of 10^{-3}, a batch size of 16, and a maximum sequence length of 128. *Adafactor* is used as the optimizer, a stochastic optimization method based on Adam that reduces memory usage while retaining the empirical benefits of adaptivity [54].

In order to evaluate the models' performance, we will use 3 state-of-the-art metrics employed to assess MT quality:

- **BLEU ↑ (Bilingual evaluation understudy) [55]:** This metric calculates the score of a translation by measuring the number of n-grams of varying lengths that occur within the set of references and in the model prediction. The BLEU metric ranges from 0 to 1, with 1 being the highest score. Thus, it is usually represented as a percentage.

- **chrF ↑ (Character n-gram F-score) [56]:** This metric uses the F-score based on character n-grams. *F-score* is a metric that combines the precision and the recall metrics with a coefficient β [57]:

  $$F_\beta = (1+\beta^2)\frac{PR}{\beta^2 P + R}$$

  where P is the *precision*, that represents the proportion of tokens the MT system returns that are accurately correct, and R is the *recall*, that indicates how many tokens that should have been found were found:

  $$P = \frac{|\text{true positives}|}{|\text{true positives}| + |\text{false positives}|}$$

  $$R = \frac{|\text{true positives}|}{|\text{true positives}| + |\text{false negatives}|}$$

  The general formula for the chrF score is:

  $$chrF_\beta = (1+\beta^2)\frac{chrP\, chrR}{\beta^2 chrP + chrR}$$

  where $chrP$ and $chrR$ stand for character n-gram precision and recall arithmetically averaged over all n-grams and β is a coefficient which assigns β times more importance to recall than to precision. Thus, if $\beta = 1$, they have the same importance. In contrast to related metrics, it is simple since it does not require any additional tools and knowledge sources and is independent of the language and the tokenization of the text. The chrF metric ranges from 0 to 1, with 1 indicating a perfect match at the character level. Thus, it is usually represented as a percentage.

- **TER ↓ (Translation Edit Rate) [58]:** This metric compares a translation against a set of reference translations and assigns a score to the similarity. Lower scores are given to hypotheses more similar to the references, so 0 means an exact match. When TER is used with multiple references, it does not combine them but scores the hypothesis against each reference individually. The reference against which the hypothesis has the fewest edits is deemed the closest. Possible edits include

the insertion, deletion, and substitution of single words as well as shifts of word sequences. Thus, it is defined as the minimum number of edits E needed to change a hypothesis to match one of the references, normalized by the average length of the references μ:

$$\text{TER} = \frac{E}{\mu}$$

The above metrics will be computed automatically using *sacreBLEU* [59], a Python library that aims to unify how these metrics are computed and make them easy to use.

CHAPTER 5
Experimental results

This chapter aims to present the experiments' results and assess whether synthetic data has positively impacted the models' performance. First, we will see some examples of synthetic data generated by the designed data augmentation pipeline. Then, the testing results for the original models will be compared with the models that added the synthetic data for training.

5.1 Synthetic data analysis

The proposed solution has been applied to 3 languages: Spanish, French, and Greek. At first, the purpose was to evaluate the synthetic data by human translators. However, this was not possible with the current workload in the company. For this reason, the data has been evaluated by people who know the language but are not experts. Random samples were chosen to check if the changes are coherent and if the target sentence is a suitable translation for the source sentence. During the experiments in the company, it was proven that the process works well in all these languages, but it has some limitations. We will review some samples for each language pair. We will examine both correct examples and examples that show some errors. The original words are highlighted in red, while the new words generated by the solution are highlighted in blue.

English	**Spanish**
Keep the pre-filled syringe in the outer carton adequate space in order to protect from light contamination.	Mantener la jeringa precargada en el embalaje exterior espacio adecuado para protegerla de la luz contaminación.
Therefore, in patients children with renal anaemia serious disease the medicinal product treatment has to be administered intravenously.	Por lo tanto, en los pacientes niños con anemia renal enfermedad grave, el producto tratamiento debe administrarse por vía intravenosa.
Keep the pre-filled fresh-filled syringe dough in the outer original carton in order to protect from light.	Mantener la jeringa precargada pasta fresca en el embalaje exterior original para protegerla de la luz.

Table 5.1: Synthetic English-Spanish correct pairs samples generated by the solution.

Table 5.1 shows some examples of generated sentences for the English-Spanish pair. These examples are successful cases where the translation is literal, and they have co-

herence. The substitution works without matter of the position of the words, so we can say the word alignment does an excellent job in many cases. We can observe that it even works well in many composed terms like *pre-filled*, substituted with *fresh-filled*. Yet, in the samples shown in Table 5.2, 2 words are replaced by words that do not fit the context. The word *consejos* is replaced by *councils*, which is a correct translation but does not fit in that context. The same goes for the word *cuadro*, which is replaced by *picture* in the English sentence.

English	**Spanish**
Effects Councils on ability to drive and use machines vehicles	Efectos consejos sobre la capacidad para conducir y utilizar máquinas vehículos
A physician´s evaluation of the individual patient´s clinical course hormonal situation and condition picture is necessary important.	Es necesario importante que el médico evalúe la evolución situación y el estado clínico cuadro hormonal del paciente.

Table 5.2: Synthetic English-Spanish incorrect pairs samples generated by the solution.

Table 5.3 shows some examples of generated sentences for the English-French pair. The solution has worked perfectly for the first 4 pairs since both source and target maintain coherence, and they are equivalent or almost equivalent. Here, the system manages to replace apostrophized words like *L'efficacité* by *L'activité*, which is very useful for languages like Catalan.

English	**French**
Increase in dosage rate of probenecid salt or sulfinpyrazone sugar may be necessary.	Une augmentation du dosage taux de probénécide sel ou de sulfinpyrazone sucre peut être nécessaire.
Respiratory, thoracic and mediastinal miscellaneous disorders:	Troubles respiratoires, thoraciques et médiastinaux diverses :
The efficacy activity of irbesartan is not influenced by age or gender weight.	L'efficacité L'activité d'Irbesartan Krka est indépendante de l'âge ou du sexe poids.
Within each frequency grouping table, undesirable effects equipment are presented in order of decreasing seriousness frequency.	Dans chaque groupe tableau de fréquence, les effets équipements indésirables sont présentés par ordre décroissant de gravité fréquence.

Table 5.3: Synthetic English-French correct pairs samples generated by the solution.

As seen in Table 5.4 There are also incorrect predictions, like the word *père*, replaced in English with *father*, making the sentence incoherent. Besides, there is one significant limitation here: the MLM sometimes changes the number of a word from singular to plural or vice versa, and it can even change the genre of a word. In romance languages, the articles, adjectives, and verbs that affect a word must also be changed. For instance, the word *effets* is replaced by its singular *effet*, but the preceding article *Les* is plural. On the other hand, English exhibits remarkable genre and number independence, with articles, adjectives, and verbs remaining unchanged when the words they modify shift in gender or number.

English	French
In rabbits father, abortion menstruation or early resorption menstruation was noted at doses children causing significant maternal toxicity, including mortality.	Chez le lapin père, des avortements règles ou des résorptions règles précoces ont été observés à des doses enfants entraînant des effets toxiques importants y compris létaux pour la mère.
The effects effect of CYP2C9 inducers such as rifampicin on the pharmacokinetic of irbesartan child have not never been evaluated.	Les effets effet des inducteurs du CYP2C9, tels tel que la rifampicine, sur la pharmacocinétique de l'irbésartan l'enfant n'ont pas jamais été évalués.

Table 5.4: Synthetic English-French incorrect pairs samples generated by the solution.

The same happens in the English-Greek dataset. Table 5.5 shows some examples of correct and incorrect generated pairs. The first 2 pairs are coherent and the meaning is very related or equal. Nonetheless, the last 2 pairs are incorrect. The word λεπτομέρειες is replaced by the word *particulars* in English, which is a synonym but does not fit the context. Something similar happens with the word έως, which is replaced by *until*, but the correct term should be *up to*, unfeasible with our solution.

English	Greek
Epoetins are growth factors that primarily no stimulate red blood cell production development.	Οι ερυθροποιητίνες είναι αυξητικοί παράγοντες που κυρίως δεν διεγείρουν την παραγωγή ανάπτυξη ερυθροκυττάρων.
Medicinal product object subject to medical prescription approval.	Φαρμακευτικό προϊόν αντικείμενο για το οποίο απαιτείται ιατρική συνταγή έγκριση.
If you want more information particulars about this medicine subject, please contact the local representative	λέ ιπ Εάν χρειάζεστε περισσότερες πληροφορίες λεπτομέρειες για το φάρμακο θέμα αυτό, παρακαλούμε επικοινωνήστε με τον
is approximately until 25% lower than the previous dose price.	επίπεδο περίπου έως 25% κάτω από την προηγούμενη δόση τιμή.

Table 5.5: Synthetic English-Greek pairs samples generated by the solution. The first 2 pairs are correct and the last 2 pairs are incorrect or inexact.

In general, the solution generates good results except for some cases. The reviewed limitations are incorrect synonym substitutions, wrong word substitutions in the source sentence, or the loss of coherence. Filtering nouns, adjectives, and adverbs prevents erroneous substitutions. In English, verbal conjugations, adjectives, nouns, and articles usually have the same form for all pronouns. In contrast, in romance languages, the same verbal conjugation, adjective, article, and noun changes depending on the grammatical person, the genre, and the number. This behaviour makes it hard to correctly match and replace parallel words in the sentences.

5.2 Testing results

Performance cannot be compared with any base model since it cannot translate without fine-tuning it. So, we will limit the assessment to the effect of the synthetic data compared with the original dataset. First, a mT5-small model per language pair has been fine-tuned with the mentioned configuration. The fine-tuning took 6 hours, and the evaluation took around 5 more hours for each model. The metrics results are presented in Table 5.6.

Data	Model	BLEU ↑	chrF ↑	TER ↓
Original data	English-Spanish	42,2	65	48,5
	English-French	38,1	61,4	56,6
	English-Greek	26,9	52,9	70,1

Table 5.6: MT evaluation results for the models fine-tuned with the original data. These results are suitable for an in-domain translation.

The best model is the English-Spanish with 42 points of BLEU, then the English-French model with 38 points of BLEU, and finally, the English-Greek model with 27 points of BLEU. These results make sense if we observe the percentage of training examples from each language during the pre-training of mT5. A correlation can be observed between the quality of the translations and their respective rates, as seen in the Figure 5.1. This correlation demonstrates that the translations benefit from the pre-trained knowledge of the model in each language. These results are suitable for an *in-domain* translation, given the number of samples in the training data and the size of the fine-tuned model.

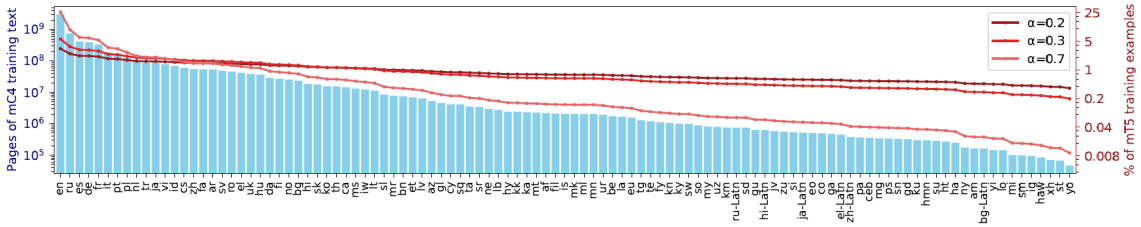

Figure 5.1: mT5 percentage of languages in the pre-training data. There is a correlation between these ratios and the evaluation results. Figure taken from *mT5's research paper* [43].

The next step is training 3 more mT5-small but including the generated synthetic data in the training set. Since the number of samples to learn is more significant, the training took around 2 more hours than the previous models. The metrics results are presented in Table 5.7.

Data	Model	BLEU ↑	chrF ↑	TER ↓
Original data	English-Spanish	45	67,3	46,8
+	English-French	40,2	62,8	55,6
Synthetic data	English-Greek	27,7	53,6	70,4

Table 5.7: MT evaluation results for the models fine-tuned with the original data plus the synthetic data.

The results show an improvement in almost all metrics per each language pair. For BLEU, the gain goes from almost 1 to 3 points. For chrF, the increase goes from nearly 1 point to 2 points. And finally, TER is inversed, so a decrease means an improvement in the quality, which is obtained in Spanish in almost 2 points and French in 1 point.

However, the TER for Greek shows a small decay. All differences in the metrics can be seen in Table 5.8. The difference is not very significant but offers a good trend.

Model	Δ BLEU ↑	Δ chrF ↑	Δ TER ↓
English-Spanish	2,8	2,3	1,7
English-French	2,1	1,4	1
English-Greek	0,8	0,7	−0,3

Table 5.8: Metrics improvement with the synthetic data.

The improvement is suitable given that the generated data only represents 20 % of the training set. In the case of Greek, the word alignment may have more fails than for French or Spanish.

CHAPTER 6
Conclusions

This chapter aims to extract some conclusions from the work done and the results exposed in the previous chapter. We also propose future work to continue studying this solution or improve it.

6.1 Conclusion

This project aimed to develop a DA method to enhance low-resource datasets for MT. For this purpose, we have used LLMs and grammatical resources to build a bitext augmenting pipeline. This pipeline also works for many languages in any direction, making it versatile and easy to use. It also allows specialization for a specific language or domain with the choice of the desired MLM.

The solution's effectiveness has been proven by fine-tuning an MT model with the original data and the same model fine-tuned by adding the synthetic data generated. Then, these models' quality was evaluated using state-of-the-art metrics in MT, and results showed a slight increase in performance. Given the limited resources and the little generated data, this approach marks a positive trend. Its disadvantages are the time it takes to generate new sentence pairs and the failure in word alignment in some cases, especially in low-resource languages.

6.2 Future work

As seen in the previous sections, the experiments showed some limitations. Future studies would test the solution with more languages, such as low-resource, and fine-tune different models. For instance, a bigger version of mT5, like mT5-base and mT5-large, or even other multilingual models. It is also interesting to generate the same number of synthetic data as the original data to fine-tune a model only with the synthetic data. This way, we can check how the designed solution affects the dataset's quality. Also, train a model with the original dataset plus the generated data, doubling the number of training samples.

Since it is slow to generate new sentence pairs, a new line of research would make the pipeline more efficient, making the process less expensive. For instance, using or training an MLM for the target language that computes the predictions faster and more efficiently. Another option would be removing the Panlex API step, which delays the entire process since it uses the network. That would require making a parallel glossary for the language pair to augment.

Finally, a good improvement will be using a conversational model to generate the synthetic pairs. This process would require designing an effective prompt so that the model understands explicitly the requested task. For instance, BLOOM has multilingual support and performed well in the tests, achieving zero-shot learning. Using an LLM will reduce the failure of the word alignments or the use of words unsuitable for the context. The LLM could even be used to evaluate the sentence pairs generated. Some studies have verified that LLMs are state-of-the-art MT evaluators [60].

Moreover, preliminary studies show the potential of ChatGPT as an evaluator of natural language generation, even in complex tasks [61]. Researchers conclude that shortly, chatGPT will exceed performance and become a reliable evaluator. However, the performance of the evaluation is sensitive to the format of the prompt, so the prompt should be carefully designed to obtain the expected results. LLMs models are even being used to compare the responses of different LLMs and assign a score on par with the latest evaluation methods [62].

Using chatGPT would be ideal because of its knowledge and understanding of natural language in many languages. However, for now, this option is still too costly to implement due to its price.

6.3 Acknowledgment

I want to thank Pangeanic for allowing me to intern in the machine translation team, especially Dr. Konstantinos Chatzitheodorou, for his help and mentoring. I have learned a lot during my stay in the company, including almost all the stuff in this project. I also want to thank Dr. Francisco Casacuberta for guiding me during this project and being my mentor.

Thank you all!

Bibliography

[1] JURAFSKY, Dan; MARTIN, James H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 2nd Edition, United Kingdom: Pearson Prentice Hall, 2008.

[2] MANARIS, Bill. Natural Language Processing: A Human-Computer Interaction Perspective In *Advances in Computers*, Elsevier, 1998, vol. 47, p. 1-66. ISBN 9780123744258. Available from: doi:10.1016/s0065-2458(08)x0003-8

[3] TURING, Alan M. I.—COMPUTING MACHINERY AND INTELLIGENCE, *Mind*, 1950, vol. LIX, no 236, p. 433–460. ISSN 1460-2113. ISSN 1460-2113. Available from: doi:10.1093/mind/LIX.236.433

[4] HIRSCHBERG, Julia; MANNING, Christopher D. Advances in natural language processing. *Science*, 2015, vol. 349, no 6245, p. 261-266. Available from: doi:10.1126/science.aaa8685

[5] HUTCHINS, John. The Georgetown-IBM Experiment Demonstrated in January 1954. In *Machine Translation: From Real Users to Research*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, p. 102–114. ISBN 9783540233008. Available from: doi:10.1007/978-3-540-30194-3_12

[6] NAVIGLI, Roberto, Word sense disambiguation: A survey. *ACM computing surveys (CSUR)*, 2009, vol. 41, no 2, p. 1-69. ISSN 1557-7341. Available from: doi:10.1145/1459352.1459355

[7] NIVRE, Joakim, et al. Universal Dependencies v1: A Multilingual Treebank Collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, 2016. p. 1659-1666. Portorož, Slovenia. European Language Resources Association (ELRA).

[8] DHINGRA, Bhuwan, et al. Neural models for reasoning over multiple mentions using coreference. *arXiv*, 2018. Available from: doi:10.48550/arXiv.1804.05922

[9] SCHWARTZ, Hansen, et al. Characterizing Geographic Variation in Well-Being Using Tweets. In *Proceedings of the International AAAI Conference on Web and Social Media*. 2013, p. 583-591. Available from: doi:10.1609/icwsm.v7i1.14442

[10] BRUEN, Aiden A.; FORCINITO, Mario A. ASCII. In *Cryptography, Information Theory, and Error-Correction*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2011, p. 445–446. ISBN 9781118033296. Available from: doi:10.1002/9781118033296.oth

[11] A. MULLEN, Lincoln, et al. Fast, consistent tokenization of natural language text. *Journal of Open Source Software*, 2018, vol. 3, no 23, p. 655. ISSN 2475-9066. Available from: doi:10.21105/joss.00655

[12] SENNRICH, Rico; HADDOW, Barry; BIRCH, Alexandra. Neural machine translation of rare words with subword units. *arXiv*, 2015. Available from: doi:10.48550/arXiv.1508.07909

[13] KUDO, Taku. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv*, 2018. Available from: doi:10.48550/arXiv.1804.10959

[14] SCHUSTER, Mike; NAKAJIMA, Kaisuke. Japanese and korean voice search. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2012, p. 5149-5152. ISBN 9781467300469. Available from: doi:10.1109/icassp.2012.6289079

[15] MIKOLOV, Tomas, et al. Efficient estimation of word representations in vector space. *arXiv*, 2013. Available from: doi:10.48550/arXiv.1301.3781

[16] PETERS, Matthew et al. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2018, vol. 1. Available from: doi:10.18653/v1/n18-1202

[17] BAEVSKI, Alexei, et al. Data2vec: A general framework for self-supervised learning in speech, vision and language. In *International Conference on Machine Learning*. PMLR, 2022, p. 1298-1312.

[18] KOEHN, Philipp. Europarl: A parallel corpus for statistical machine translation. *Proceedings of machine translation summit*, 2005, vol. 5, p. 79-86.

[19] LISON, Pierre; TIEDEMANN, Jörg, Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. European Language Resources Association (ELRA), 2016, p. 923–929.

[20] ESPLÀ-GOMIS, Miquel, et al. ParaCrawl: Web-scale parallel corpora for the languages of the EU. In *Proceedings of Machine Translation Summit XVII: Translator, Project and User Tracks*, 2019, p. 118-119.

[21] LI, Xiaoya, et al. Is word segmentation necessary for deep learning of Chinese representations?. *arXiv*, 2019. Available from: doi:10.48550/arXiv.1905.05526

[22] TATSUYA, Izuha; KUMANO, Akira; KURODA, Yuka, Toshiba Rule-Based Machine Translation System at NTCIR-7 PAT MT. In *Proceedings of NII Test Collection for IR Systems-7 Workshop Meeting*, Japan, 2008, p. 430-434.

[23] BROWN, Peter F., et al. A statistical approach to language translation. In *Coling Budapest 1988 Volume 1: International Conference on Computational Linguistics*, Morristown, NJ, USA: Association for Computational Linguistics, 1988. ISBN 9638431563. Available from: doi:10.3115/991635.991651

[24] SRIVASTAVA, Siddhant; SHUKLA, Anupam; TIWARI, Ritu. Machine translation: From statistical to modern deep-learning practices, *arXiv*, 2018. Available from: doi:10.48550/arXiv.1812.04238

[25] KOEHN, Philipp; OCH, Franz Josef; MARCU, Daniel, Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics*, Morristown, NJ, USA: Association for Computational Linguistics, 2003, p. 127-133. Available from: doi:10.3115/1073445.1073462

[26] SUTSKEVER, Ilya; VINYALS, Oriol; LE, Quoc V. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, Cambridge, MA, USA: MIT Press, 2014, vol. 2, p. 3104–3112.

[27] BAHDANAU, Dzmitry; CHO, Kyunghyun; BENGIO, Yoshua, Neural machine translation by jointly learning to align and translate. *arXiv*, 2014. Available from: doi:10.48550/arXiv.1409.0473

[28] VASWANI, Ashish, et al. Attention is all you need. In *Proceedings of Advances in Neural Information Processing Systems*, 2017, vol. 30.

[29] KOMBRINK, Stefan, et al. Recurrent Neural Network Based Language Modeling in Meeting Recognition. In *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2011, p. 2877-2880. Available from: doi:10.21437/Interspeech.2011-720

[30] GRAVES, Alex, Generating sequences with recurrent neural networks. *arXiv*, 2013. Available from: doi:10.48550/arXiv.1308.0850

[31] GEHRING, Jonas, et al. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*, 2017, vol. 70, p. 1243-1252.

[32] STRUBELL, Emma; GANESH, Ananya; MCCALLUM, Andrew, Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2019. Available from: doi:10.18653/v1/p19-1355

[33] RADFORD, Alec, et al. Improving language understanding by generative pre-training. *OpenAI blog*, 2018.

[34] RADFORD, Alec, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.

[35] BROWN, Tom, et al. Language models are few-shot learners. In *Proceedings of Advances in neural information processing systems*, 2020, vol. 33, p. 1877-1901.

[36] OpenAI, GPT-4 Technical Report. *arXiv*, 2023. Available from: doi:10.48550/arXiv.2303.08774

[37] KATZ, Daniel Martin, et al. GPT-4 passes the bar exam. In *SSRN*, 2023. ISSN 1556-5068. Available from: doi:10.2139/ssrn.4389233

[38] DEVLIN, Jacob, et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv*, 2018. Available from: doi:10.48550/arXiv.1810.04805

[39] CAÑETE, José, et al. Spanish pre-trained BERT model and evaluation data. *arXiv*, 2023. Available from: doi:10.48550/arXiv.2308.02976

[40] LIU, Yinhan, et al. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv*, 2019. Available from: doi:10.48550/arXiv.1907.11692

[41] LEWIS, Mike, et al. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension *arXiv*, 2019. Available from: doi:10.48550/arXiv.1910.13461

[42] RAFFEL, Colin, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 2020, vol. 21, no 1, p. 5485-5551.

[43] XUE, Linting, et al. mT5: A massively multilingual pre-trained text-to-text transformer. *arXiv*, 2020. Available from: doi:10.48550/arXiv.2010.11934

[44] SCAO, Teven Le, et al. BLOOM: A 176b-parameter open-access multilingual language model. *arXiv*, 2022. Available from: doi:10.48550/arXiv.2211.05100

[45] WEI, Jason; ZOU, Kai, EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* Stroudsburg, PA, USA: Association for Computational Linguistics, 2019. Available from: doi:10.18653/v1/d19-1670

[46] MAZURKEVYCH, Vladyslav, *Using back-translation for machine translation based on transformer*, 2021, Degree Final Work, Universitat Politècnica de València. Available from: http://hdl.handle.net/10251/173966

[47] GRYGORYEV, Nikita Teslenko, et al. English-Russian Data Augmentation for Neural Machine Translation. In *Proceedings of the 15th biennial conference of the Association for Machine Translation in the Americas (Workshop 2: Corpus Generation and Corpus Augmentation for Machine Translation)*, 2022, p. 1-10.

[48] KUMAR, Varun; CHOUDHARY, Ashutosh; CHO, Eunah. Data augmentation using pre-trained transformer models. *arXiv*, 2020. Available from: doi:10.48550/arXiv.2003.02245

[49] DAI, Haixing, et al. AugGPT: Leveraging chatGPT for Text Data Augmentation. *arXiv*, 2023. Available from: doi:10.48550/arXiv.2302.13007

[50] WHITEHOUSE, Chenxi; CHOUDHURY, Monojit; AJI, Alham Fikri. LLM-powered Data Augmentation for Enhanced Crosslingual Performance. *arXiv*, 2023. Available from: doi:10.48550/arXiv.2305.14288

[51] DOU, Zi-Yi; NEUBIG, Graham. Word alignment by fine-tuning embeddings on parallel corpora. *arXiv*, 2021. Available from: doi:10.48550/arXiv.2101.08231

[52] TIEDEMANN, Jörg. Parallel data, tools and interfaces in OPUS. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, 2012, p. 2214-2218.

[53] CONNEAU, Alexis, et al. Unsupervised cross-lingual representation learning at scale. *arXiv*, 2019. Available from: doi:10.48550/arXiv.1911.02116

[54] SHAZEER, Noam; STERN, Mitchell. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, PMLR, 2018, p. 4596-4604.

[55] PAPINENI, Kishore, et al. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, p. 311-318. Available from: doi:10.3115/1073083.1073135

[56] POPOVIĆ, Maja. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the tenth workshop on statistical machine translation*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2015, p. 392-395. Available from: doi:10.18653/v1/w15-3049

[57] DERCZYNSKI, Leon. Complementarity, F-score, and NLP Evaluation. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, 2016, p. 261-266.

[58] SNOVER, Matthew, et al. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, 2006, p. 223-231.

[59] POST, Matt. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2018. Available from: doi:10.18653/v1/w18-6319

[60] KOCMI, Tom; FEDERMANN, Christian. *Large Language Models Are State-of-the-Art Evaluators of Translation Quality*. In *Proceedings of the 24th Annual Conference of the European Association for Machine Translation*. European Association for Machine Translation, 2023, p. 193-203.

[61] WANG, Jiaan, et al. *Is chatGPT a Good NLG Evaluator? A Preliminary Study*. *arXiv*, 2023. Available from: doi:10.48550/arXiv.2303.04048

[62] ZHENG, Lianmin, et al. *Judging LLM-as-a-judge with MT-Bench and Chatbot Arena*. *arXiv*, 2023. Available from: doi:10.48550/arXiv.2306.05685

APPENDIX A

Sustainable Development Goals

Degree of relationship of the work with the Sustainable Development Goals (SDG).

Sustainable Development Goals	High	Medium	Low	Not Applicable
SDG 1. No poverty.				×
SDG 2. Zero hunger.				×
SDG 3. Good health and well-being.				×
SDG 4. Quality education.	×			
SDG 5. Gender equality.				×
SDG 6. Clean water and sanitation.				×
SDG 7. Affordable and clean energy.				×
SDG 8. Decent work and economic growth.				×
SDG 9. Industry, Innovation and Infrastructure.	×			
SDG 10. Reduced inequality.		×		
SDG 11. Sustainable cities and communities.				×
SDG 12. Responsible consumption and production.				×
SDG 13. Climate action.				×
SDG 14. Life below water.				×
SDG 15. Life on land.				×
SDG 16. Peace, justice and strong institutions.			×	
SDG 17. Partnership for the goals.		×		

Reflection on the relationship of the DFP with the most related SDGs.

This DFP constitutes a comprehensive effort to enhance the quality of model training datasets for machine translation, thereby making a small contribution to the broader field of translation. The goal of this endeavor can be aligned closely with several of the United Nations Sustainable Development Goals (SDGs), representing an approach to help address pressing global challenges.

The pivotal role that these machine translation models play in advancing quality education (SDG 4) cannot be overstated. They offer the potential to make educational content accessible in many languages, transcending geographical and linguistic barriers. Currently, the digital divide continues to affect millions of individuals living in regions where low-resource languages are spoken. The limited availability of educational content in these languages compounds the challenges these communities face. However, an effective automatic translation system can bridge this divide by ensuring that knowledge and educational materials are readily accessible to a broader audience. By facilitating the translation of educational content into diverse languages, this initiative aims to democratize access to education, thereby enhancing its availability and accessibility across diverse regions and cultures. In an era of increasing globalization, where multiculturalism is celebrated, accessing educational resources in one's native language can significantly elevate the quality of education. Moreover, addressing this issue directly contributes to reducing inequalities (SDG 10) between nations, as disparities often stem from unequal access to information and educational resources.

In addition to its role in education, machine translation holds promise in addressing another critical SDG: promoting peace and international cooperation (SDG 16). Effective communication is the base of diplomacy and international relations. In this context, automatic translation systems can be a crucial element, facilitating conflict resolution and fostering robust international relationships. It has the potential to bridge language gaps, break down communication barriers, and enable more effective dialogue between nations. Furthermore, machine translation can be instrumental in strengthening global collaboration and cooperation across various sectors. Governments, businesses, and academia can harness the power of this technology to form international alliances, collaborate on shared objectives, and collectively address global challenges (SDG 17).

Beyond its contributions to education and international relations, improving automatic translation models also aligns with the goal of driving industry and innovation (SDG 9). The advancement of natural language processing and the integration of efficient machine translation systems into various industries can lead to greater productivity, innovation, and economic growth. Industries ranging from e-commerce to customer support to content localization can benefit immensely from these technological advancements.

In summary, this DFP constitutes a multifaceted approach to address several key global challenges. By enhancing machine translation capabilities, it seeks to democratize access to education, bridge linguistic divides, facilitate diplomatic communication, and drive innovation across industries. Doing so contributes to the broader mission of sustainable development in our increasingly interconnected and diverse world.